НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ КОРАБЛЕБУДУВАННЯ

імені адмірала Макарова

Навчально-науковий інститут автоматики і електротехніки
(повне найменування інституту, назва факультету)

Кафедра програмованої електроніки, електротехніки і телекомунікацій
(повна назва кафедри)

«Допущений до захисту»
Завідувач кафедри _____ В. М. Рябенький
д.т.н., професор
« 16 » грудня 2024 р.

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття ступеня вищої освіти «магістр»

на тему: Ідентифікація суден на супутникових знімках із використанням технології AIC

Виконав: студент 6 курсу, групи 6321М
спеціальності 171
«Електроніка»
шифр і назва спеціальності)

ОПП «Електронні системи»

Вишневський В.В.
(прізвище та ініціали студента)

Керівник Дьяконов О.С.
(прізвище та ініціали)

Рецензент Іхсанов Ш.М.
(прізвище та ініціали)

м. Миколаїв – 2024 року

Національний університет кораблебудування імені адмірала Макарова

ННІ	автоматики та електротехніки
Кафедра	програмованої електроніки, електротехніки і телекомунікацій
Рівень вищої освіти	другий (магістерський)
Спеціальність	171 «Електроніка»
ОПП	Електронні системи

ЗАТВЕРДЖУЮ
Завідувач кафедри _____ В. М. Рябенький
д.т.н., професор
« 16 » жовтня 2024 р.

ЗАВДАННЯ
НА МАГІСТЕРСЬКУ АТЕСТАЦІЙНУ РОБОТУ СТУДЕНТУ

Вишневському Віктору Володимировичу
(прізвище, ім'я, по батькові)

1. Тема роботи Ідентифікація суден на супутникових знімках із використанням технології AIC

керівник роботи Дьяконов Олексій Сергійович, к.т.н., доцент
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом вищого навчального закладу
від «14» жовтня 2024 року № 1075-уч.

2. Строк подання студентом роботи Грудень 2024 р.

3. Вихідні дані до роботи колекція даних з сайту Marine Traffic, супутникові знімки sentinelhub

4. Мета дослідження Ідентифікувати судна на супутникових знімках. Створити нейромережу детекції. Дослідити можливість оцінки трафіку порту.

5. Зміст розрахунково-пояснювальної записки
(перелік питань, які потрібно розробити)
1. Аналіз джерел супутникових даних. Автоматизація завантаження даних
2. Розробка нейромережі для ідентифікації суден на супутникових знімках
3. Розробка алгоритмів оцінки трафіку порту
4. Охорона праці

6. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)
Презентація в PowerPoint, Приклад завантажених супутникових знімків, Дані з сайту Marine Traffic, Результати навчання на даних анотованих вручну, Візуалізація даних AIC на супутникових знімках, Результат навчання на втоматизовано розмічених даних, Візуальний моніторинг трафіку порту

7. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Охорона праці	Тубальцев А.М.		

8. Дата видачі завдання « 16 » жовтня 2024 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломної роботи	Строк виконання етапів роботи	Примітка
1.	Автоматизація роботи з супутниковими	03.10.2024	Вик.
	даними	26.10.2024	
2.	Обробка супутникових знімків за допомогою	26.10.2024	Вик.
	нейромережі	11.11.2024	
3.	Оцінка трафіку порту за даними AIS та	11.11.2024	Вик.
	супутниковими знімками	27.11.2024	
4.	Охорона праці	27.11.2024	Вик.
		02.11.2024	

Студент _____ Вишневський В.В.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Дьяконов О.С.
(підпис) (прізвище та ініціали)

Анотація

У дипломній роботі проведено аналіз джерел супутникових даних. Представлено підхід до моніторингу портового трафіку, який інтегрує супутникові знімки, дані автоматичної ідентифікаційної системи (AIC) та сучасні алгоритми машинного навчання. Розглянуто процес навчання нейронної мережі YOLOv8 для ідентифікації суден на супутникових зображеннях та можливість створення навчального датасету на базі даних AIC. Запропоновано два програмних рішення для моніторингу портового трафіку, що мають потенціал для спостереження за судноплавством.

Ключові слова: ідентифікація суден, супутникові знімки, нейронні мережі, моніторинг портового трафіку.

Abstract

The thesis analyzes sources of satellite data and presents an approach to port traffic monitoring that integrates satellite imagery, Automatic Identification System (AIS) data, and modern machine learning algorithms. The study explores the process of training the YOLOv8 neural network for vessel identification on satellite images and the potential to create a training dataset based on AIS data. Two software solutions for port traffic monitoring are proposed, demonstrating potential for vessel tracking and maritime observation.

Keywords: Vessel identification, Satellite imagery, Neural networks, Port traffic monitoring.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ ТА ТЕРМІНІВ

AIS – Automatic Identification System;

API – Application Programming Interface;

BBox – Bounding box;

CNN – Convolutional Neural Network;

COCO – Common Objects in Context;

CPU – Central Processing Unit;

CUDA – Compute Unified Device Architecture;

CVAT – Computer Vision Annotation Tool;

DWT – Deadweight;

ESA, ЄКА – Європейське космічне агентство;

GEE – Google Earth Engine;

GPU – Graphics Processing Unit;

HTML – HyperText Markup Language;

HTTP – Hypertext Transfer Protocol;

ID – Identification;

IoU – Intersection over Union;

JSON – JavaScript Object Notation;

mAP – Mean Average Precision;

NASA – National Aeronautics and Space Administration;

PR – Precision-Recall;

RGB – Red, Green, Blue;

RPN – Region Proposal Networks;

SAR – Synthetic Aperture Radar;

SSD – Single Shot Multibox Detector;

SSO – Sun-synchronous orbit;

TPU – Tensor Processing Unit;

TXT – Trusted Execution Technology;

URL – Uniform Resource Locator;

USGS – United States Geological Survey;

VH – Vertical-Horizontal;

VV – Vertical- Vertical;

WGS84 – World Geodetic System 1984;

XML – eXtensible Markup Language;

YOLO – You Only Look Once;

ЄС – Європейський союз.

ВСТУП

У сучасному світі розвиток технологій дистанційного зондування Землі та автоматизація обробки великих обсягів даних відкривають нові горизонти для моніторингу та аналізу різних процесів. Одним із перспективних напрямів є використання супутникових знімків для задач спостереження за судноплавством, що має ключове значення для транспортної логістики, безпеки мореплавства та екологічного моніторингу.

У роботі розглядається можливість комплексного підходу до моніторингу портового трафіку, який поєднує кілька джерел даних та інструментів. Зокрема, розглянуто доступні безкоштовні джерела супутникових знімків і розроблено скрипт для їх автоматичного завантаження. На основі отриманих зображень проведено навчання нейронної мережі YOLOv8 для ідентифікації суден, що дозволяє автоматизувати аналіз супутникових даних.

В роботі описано метод завантаження даних системи AIC з використанням парсингу веб-ресурсу MarineTraffic. Для практичного застосування результатів запропоновано два програмні рішення для моніторингу портового трафіку: перше базується на аналізі даних AIC, а друге використовує супутникові знімки у поєднанні з нейромережевими методами.

Розроблений підхід є можливим інструментом для моніторингу судноплавства та має потенціал для подальшого розвитку й адаптації до інших завдань у сфері аналізу просторових даних.

					171.6321М.КР.ПЗ.Р1		
Змн.	*Арк.*	*№ документу*	*Підпис*	*Дата*			
Розробник		Вишневський В.В.			Автоматизація роботи з супутниковими даними	*Літ.*	*Арк.* 9 / *Аркушів*
Консультант							
Н. контр.		Добрінова Л. П.				НУК імені адмірала Макарова	
Керівник		Дьяконов О.С.					
Зав. каф.		Рябенький В. М.					

РОЗДІЛ 1. АВТОМАТИЗАЦІЯ РОБОТИ З СУПУТНИКОВИМИ ДАНИМИ

1.1. Супутникові знімки: види супутникових знімків

Супутникові знімки – це джерело безцінної інформації про зміни нашої планети, що відкривають широкі можливості для досліджень та моніторингу земної поверхні. Супутникові знімки можна розділити за типом випромінювання, яке фіксує супутник. Супутники бувають [1] активні та пасивні.

- Активні: створюють власне випромінювання і фіксують його відбиття (радарні, лідарні).
- Пасивні: фіксують відбите або випромінюване об'єктами випромінювання (оптичні (видимі, інфрачервоні), гіперспектральні).

Видимі – це супутникові зображення, що реєструють видимі світлові хвилі. Ці зображення дають візуальне уявлення про поверхню Землі та хмарність. Цей тип зображень широко використовується в метеорології, авіації, моніторингу стихійних лих та дослідженнях навколишнього середовища. Проте їх недолік полягає в тому, що вони доступні лише в день.

Інфрачервоні супутникові знімки – це своєрідні "теплові фотографії" Землі, зроблені з космосу. На відміну від звичайних фотографій, які фіксують видиме світло, інфрачервоні датчики реєструють тепло, яке випромінюють об'єкти. Це дозволяє отримати унікальну інформацію, недоступну для видимих знімків.

Гіперспектральні знімки – це особливий вид зображень, який дозволяє отримати детальну інформацію про об'єкти на Землі, аналізуючи їх спектральні характеристики. На відміну від звичайних кольорових знімків, які фіксують лише три основні кольори (червоний, зелений, синій), гіперспектральні знімки збирають інформацію про сотні або навіть тисячі вузьких спектральних смуг.

Радарні SAR (Synthetic Aperture Radar) - це тип супутникової зйомки, що використовує хвилі радіолокації для дослідження поверхні землі. Радар

випромінює електромагнітні хвилі, які досягають поверхні і відбиваються назад до датчика. На основі характеристик відбитих хвиль можна визначити тип поверхні об'єкта та відстань між датчиком та об'єктом. Великою перевагою таких знімків є те, що їх можна отримувати в будь-яку погоду в день та в ночі

Лідар (Light Detection and Ranging) – це технологія, яка використовує лазерні імпульси для вимірювання відстаней. Супутник, обладнаний лідарною системою, випромінює лазерні промені в напрямку земної поверхні. Частина цих променів відбивається від різних об’єктів (земля, дерева, будівлі тощо) і повертається на супутник. За вимірами часу, за який лазерний імпульс повертається, можна визначити відстань до об’єкта. На основі цього можна отримати детальну тривимірну інформацію про поверхню Землі. На відміну від звичайних фотографій, які фіксують лише зовнішній вигляд об’єктів, лідар створює своєрідну «карту висот», дозволяючи побачити рельєф місцевості з надзвичайною точністю.

1.2. Джерела супутникових даних з відкритим доступом

Існує багато джерел супутникових даних [2], які можна використовувати залежно від потреб. Деякі платформи дозволять переглядати дані у браузері, деякі пропонують варіанти завантаження даних для їх подальшої обробки, що відкриває простір для дослідників. Слід зауважити, що більшість супутникових даних, доступних безкоштовно, або застаріли (від кількох місяців до кількох років), або мають низьку роздільну здатність (дозволяють виявити будівлі, але не машини або людей). Супутникові дані з високою роздільною здатністю, близькі до реального часу, зазвичай знаходяться в платному сегменті.

До найпоширеніших джерел супутникових знімків можна віднести такі ресурси як Google Earth Engine (GEE), Sentinel Hub, USGS (United States Geological Survey), Copernicus Data Space Ecosystem, NASA Earthdata. Але

якщо необхідно отримувати свіжі супутникові знімки то слід обрати Copernicus Data Space Ecosystem.

Copernicus Data Space Ecosystem – це новий портал Європейського космічного агентства (ESA) створений для спрощеного доступу до супутникових даних Copernicus та API, Sentinel Hub. Екосистема була запущена на початку 2023 року [3].

За допомогою Copernicus Browser (рис.1.1) екосистема надає миттєвий доступ до наборів даних, що включають повний архів і свіжі зібрані зображення з Sentinel-1, Sentinel-2, Sentinel-3 (над сушею та прибережними районами) та Sentinel-5P. Крім того доступні додаткові набори даних, такі як Copernicus Digital Elevation Model та безхмарні мозаїки WorldCover.

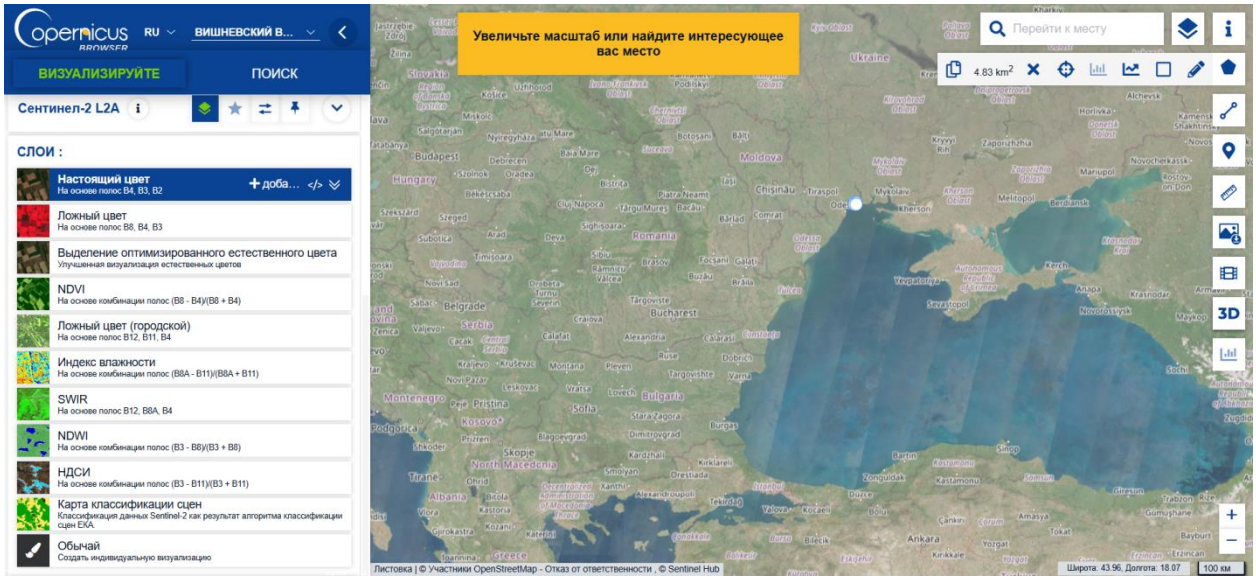


Рисунок 1.1 – Зовнішній вигляд Copernicus Browser

Супутники Sentinel-2. Місія Sentinel-2 складається з двох ідентичних супутників (рис.1.2), Sentinel-2A та Sentinel-2B, [4] які були запущені за допомогою європейської ракети-носія VEGA. Кожен із цих супутників важить близько 1,2 тонни. Sentinel-2A був успішно запущений 23 червня 2015 року, а Sentinel-2B – 7 березня 2017 року. Обидва супутники працюють на сонячно-синхронній орбіті (SSO) на висоті близько 786 кілометрів.

Роздільна здатність супутників Sentinel-2 варіюється від 10 до 60 метрів залежно від спектрального каналу. Вони обладнані мультиспектральними камерами, що працюють у видимому та інфрачервоному спектральних діапазонах (від 443 до 2190 нм), включно з каналами для короткочасного спостереження та високої роздільної здатності.

Особливістю Sentinel-2 є здатність надавати дані високої роздільної здатності для моніторингу сільського господарства, лісового господарства, гідрології, екології та інших програм, що потребують регулярного та детального спостереження за земною поверхнею.

Розробниками супутників Sentinel-2 є компанії Airbus Defence and Space (для Sentinel-2A) та Thales Alenia Space (для Sentinel-2B), які працюють під контрактами з ESA у рамках програми Copernicus.

Рисунок 1.2 – Зовнішній вигляд супутників Sentinel 2

Місія Sentinel-2 забезпечує систематичне покриття наступних територій (рис.1.3):

- весь суходіл континентів (разом з внутрішніми водоймами) між 56-ю південною паралеллю та 82,8-ю північною паралеллю;
- усі прибережні води до 20 км від берега;
- всі острови площею понад 100 км^2;
- всі острови ЄС;
- Середземне море;
- всі закриті моря (наприклад, Каспійське море).

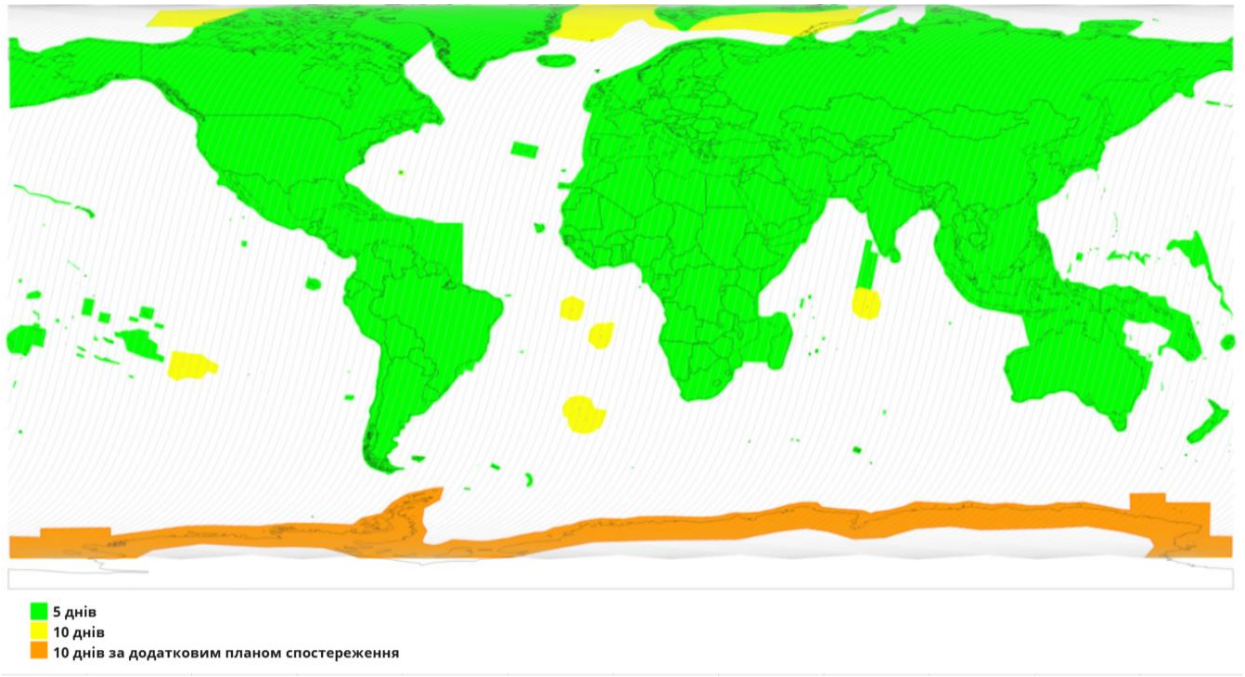


Рисунок 1.3 - Покриття та періодичність зйомки Sentinel-2

Завдяки двом супутникам усі регіони, зазначені вище, за однакових умов спостереження будуть відвідуватися кожні п'ять днів. Але за рахунок перекриття суміжних смуг спостереження частота повторного відвідування супутника буде збільшена (рис.1.4).

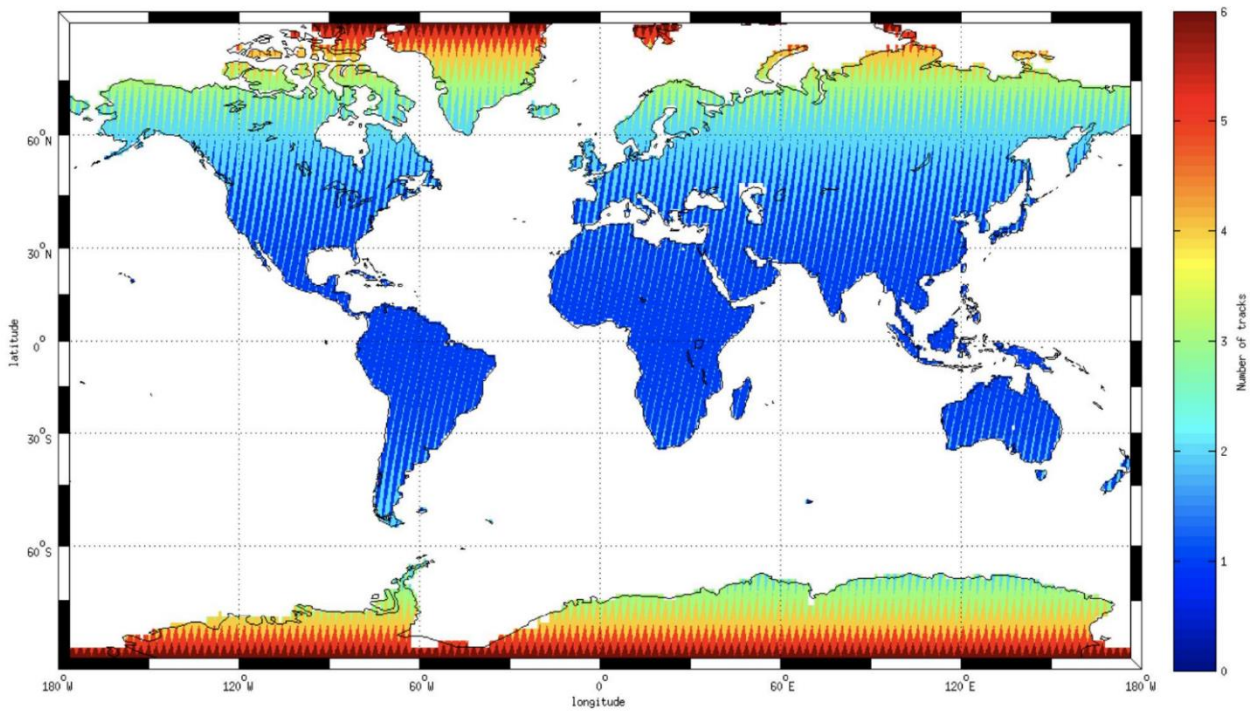


Рисунок 1.4 - Геометрична частота повторних відвідувань регіону враховуючі перекриття сусідніх орбіт супутників

Супутники Sentinel-1. Місія Sentinel-1 [5] складається з двох сонячно-синхронних супутників на полярній орбіті, які ділять одну й ту ж орбітальну площину з різницею фази орбіти 180° (рис. 1.5).

Sentinel-1A було запущено 3 квітня 2014 року. Sentinel-1B було запущено 25 квітня 2016 року. Але на космічному апараті Sentinel-1B виникла аномалія, пов'язана з живленням електроніки приладів, через що він не зміг передавати радіолокаційні дані з 23 грудня 2021 року. В результаті ЄКА та ЄС оголосили про завершення місії Sentinel-1B 3 серпня 2022 року.

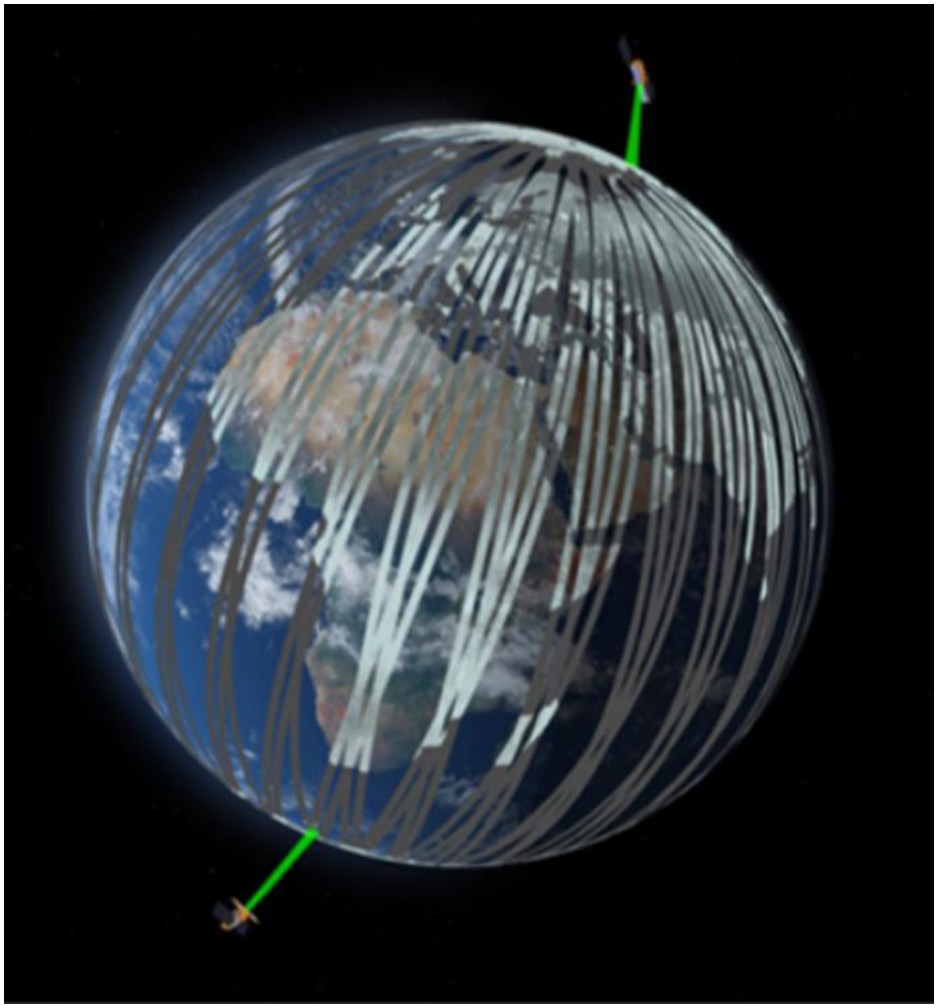


Рисунок 1.5 – Група супутників Sentinel-1

Супутники Sentinel-1 оснащені радаром із синтезованою апертурою С-діапазону, який забезпечує всепогодний збір даних вдень та вночі. Цей прилад має просторову роздільну здатність до 5 м і смугу дії до 410 км. Sentinel-1 знаходиться на навколополярній сонячно-синхронній орбіті, що забезпечує 12-денний цикл повторних відвідувань регіону (рис.1.6) та 175 обертів за цикл для одного супутника. Використання двох супутників збільшує цикл повторних відвідувань до шести днів.

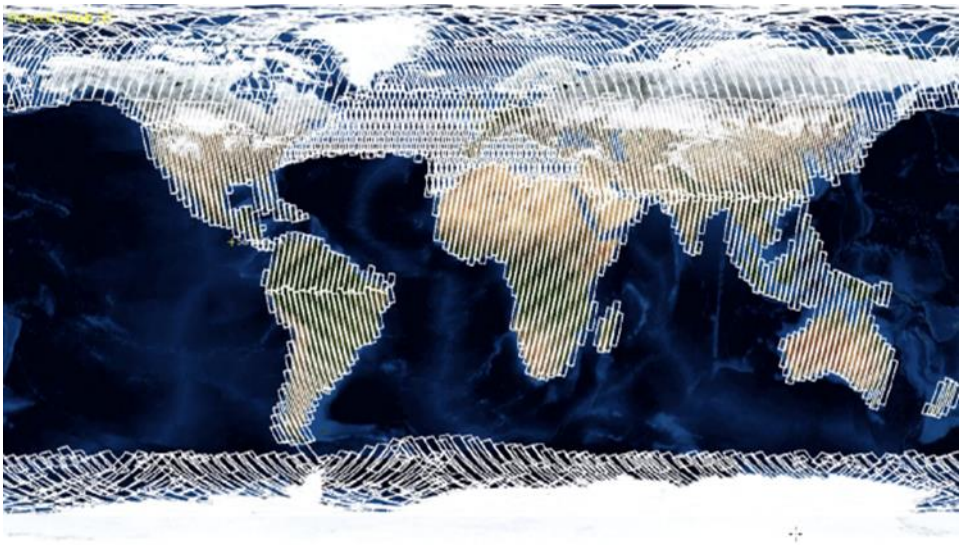


Рисунок 1.6 – Потенційне охоплення світу супутником Sentinel-1 IW

Щоб перевірити доступність супутникових знімків для конкретної ділянки в оптичному і SAR діапазоні можна скористатися Copernicus Browser. Для цього необхідно сфокусуватися на потрібному місці мапи обрати супутник Sentinel- 1 для SAR або Sentinel- 2 для оптичного діапазону і за допомогою календарю можна обирати дні за які є знімки і переглядати їх онлайн (рис. 1.7 – 1.8).

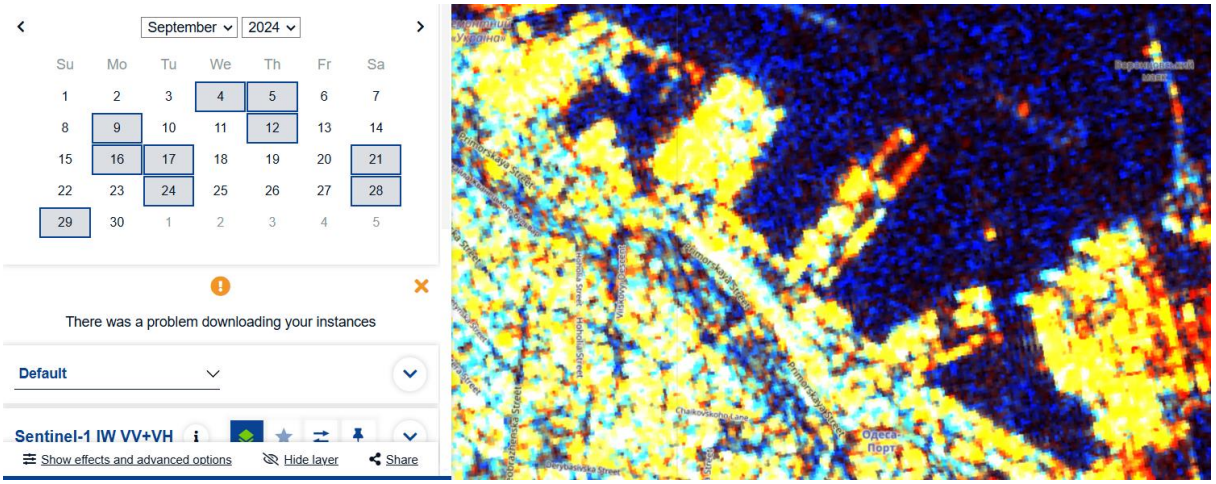


Рисунок 1.7 – Кількість доступних SAR знімків на прикладі порту Одеси за вересень 2024 року

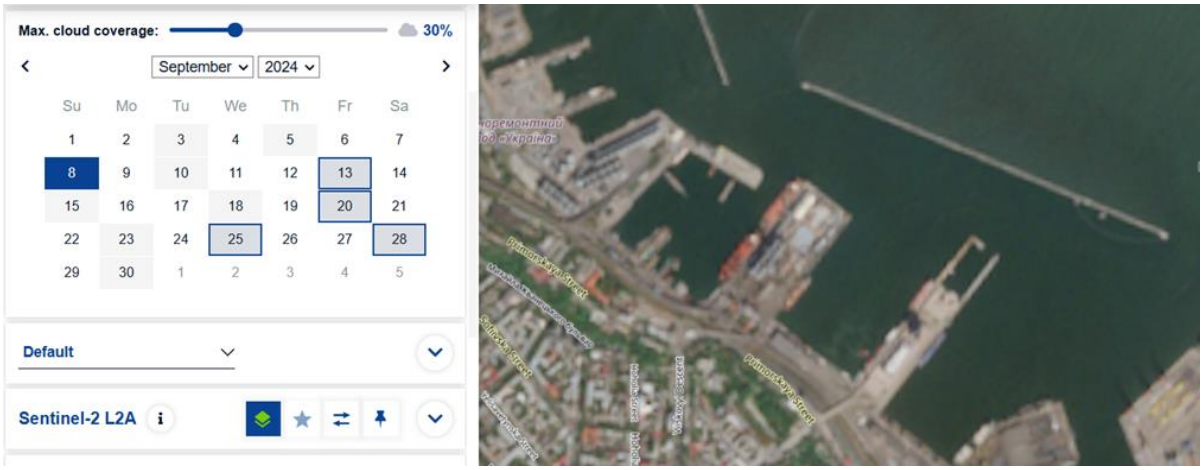


Рисунок 1.8 – Кількість доступних знімків порту Одеси в оптичному діапазоні за вересень 2024 року

Хоч в оптичному діапазоні супутник відвідував область інтересу 12 днів але через погодні умови і сильну хмарність адекватними для обробки залишаються 5 днів. Сумарно в SAR і оптичному діапазоні можна переглянути 15 супутникових знімків.

1.3. Автоматизоване завантаження супутникових знімків

Для проведення ефективних наукових досліджень, які базуються на аналізі супутникових знімків, необхідний постійний доступ до актуальних даних. Ручне завантаження великих обсягів інформації є дуже трудомістким. Автоматизація цього завдання дозволяє не тільки значно пришвидшити роботу дослідників, а й забезпечити високу точність та повторюваність результатів. Для автоматизації завантаження супутникових знімків використовується мова програмування Python та бібліотека sentinelhub (Додаток А). Перш за все необхідно зареєструватися та створити обліковий запис в Copernicus Data Space Ecosystem. Безкоштовний план цієї платформи дозволяє робити 30000 запитів та отримувати 30000 одиниць даних для обробки в місяць. Після реєстрації стають доступними ID клієнту, секретний ключ та ідентифікатор облікового запису, які необхідні для доступу до API

Sentinel Hub. Необхідно створити файл конфігурації (рис.1.9) для цього треба замінити відповідні поля своїми реєстраційними даними.

```
from sentinelhub import SHConfig
config = SHConfig()
config.sh_client_id = "Enter your SentinelHub client id"
config.sh_client_secret = "Enter your SentinelHub client secret"
config.sh_token_url = \
"https://identity.dataspace.copernicus.eu/auth/realms/CDSE/protocol/openid-connect/token"
config.sh_base_url = "https://sh.dataspace.copernicus.eu"
config.save("cdse")
```

Рисунок 1.9 – Створення конфігураційного файлу Sentinel Hub Python

Наступним кроком є підключення необхідних бібліотек (рис. 1.10). Бібліотека Matplotlib.pyplot [6] - це потужна бібліотека, яка широко використовується для створення різноманітних графіків, діаграм та візуалізацій даних.

```
1  import matplotlib.pyplot as plt
2  import numpy as np
3  from sentinelhub import (
4      SHConfig,
5      DataCollection,
6      SentinelHubCatalog,
7      SentinelHubRequest,
8      BBox,
9      bbox_to_dimensions,
10     CRS,
11     MimeType,
12 )
13 from datetime import datetime
```

Рисунок 1.10 – Підключення необхідних бібліотек

NumPy - це open-source [7] модуль для Python, який надає загальні математичні та числові операції у вигляді пре-компільованих функцій. Вони поєднуються у високорівневі пакети, що забезпечує функціонал, який можна порівняти із функціоналом MatLab. NumPy також надає базові методи для маніпуляції з великими масивами та матрицями. Для зручної роботи з датами та часом використовується модуль datetime.

Основною бібліотекою є SentinelHub – це потужна Python-бібліотека, спеціально розроблена для взаємодії з сервісами Sentinel Hub [8]. Вона дозволяє завантажувати, обробляти та аналізувати супутникові дані.

Для автоматизованого завантаження супутникових знімків необхідно виділити область інтересу, тобто конкретну ділянку земної поверхні. Найпоширеніший спосіб, що використовується в SentinelHub це використання BBox (Bounding Box) він задається за допомогою чотирьох координат: нижнього лівого і верхнього правого кута прямокутника, що охоплює область інтересу. Координати зазвичай задаються в системі координат WGS84 (широта, довгота). Один з варіантів звідки можна отримати координати Bounding Box – це скористатися сайтом bboxfinder.com (рис.1.11).

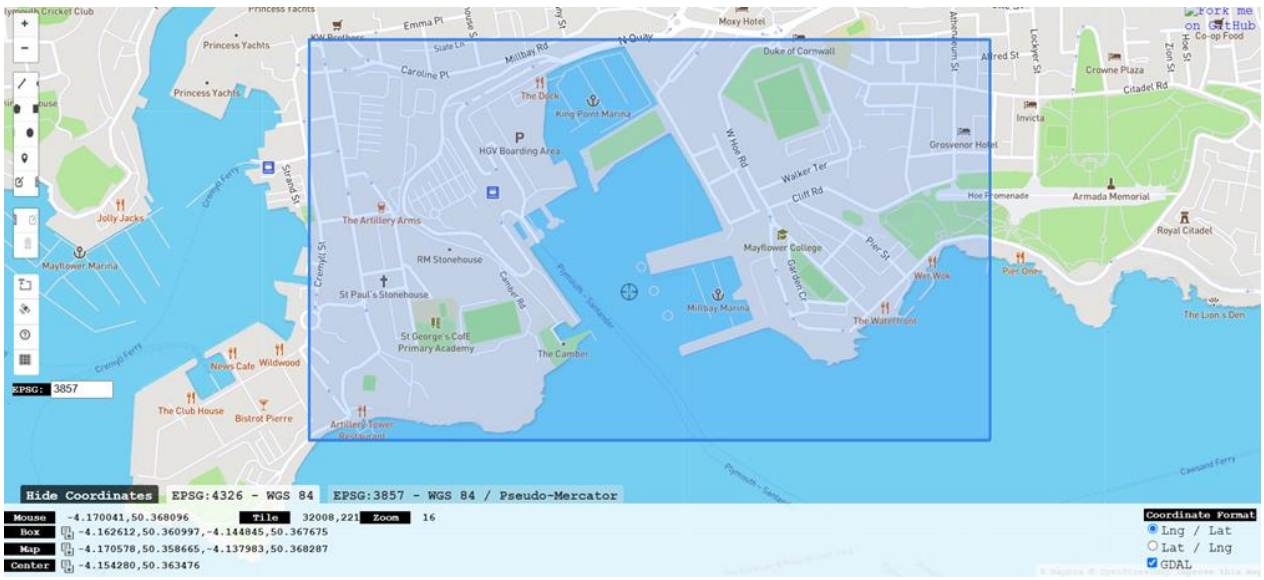


Рисунок 1.11 – Зовнішній вигляд сайту bboxfinder

Сайт дозволяє візуально на мапі виділяти потрібну область і одразу отримувати координати боксу. Інший варіант – це написання функції яка автоматично виділяє BBox (рис. 1.12).

```
30  def get_square_bbox(lat, lon, size_km):
31      km_per_deg_lat = 111
32      km_per_deg_lon = 85
33
34      lat_shift = size_km / (2 * km_per_deg_lat)
35      lon_shift = size_km / (2 * km_per_deg_lon)
36
37      return [lon - lon_shift, lat - lat_shift, lon + lon_shift, lat + lat_shift]
38
39  config = SHConfig('cdse')
40  print(config)
41
42  lat, lon = 46.48507,30.75807
43  square_size_km = 2
44
```

Рисунок 1.12 – Функція для автоматичного створення BBox

Функція приймає значення довготи та широти для точки навколо якої буде створена область інтересу, також параметр розміру області в кілометрах. Функція визначає, на скільки градусів потрібно зміститися від центральної точки вгору, вниз, вліво і вправо, щоб отримати кути квадрата заданого розміру. Константи (111 км/град для широти та 85 км/град для довготи) є приблизними значеннями, які часто використовуються в геопросторових обчисленнях для швидкого перетворення між географічними координатами (широта, довгота) та відстанню на земній поверхні в кілометрах. Також важливими параметрами для завантаження знімків є їх роздільна здатність (цей параметр залежить від характеристик супутника та розміру області інтересу) і часовий інтервал для завдання якого використовується модуль datetime (рис. 1.13).

```
lat, lon = 46.48507,30.75807
square_size_km = 2
aoi_coords_wgs84 = get_square_bbox(lat, lon, square_size_km)
resolution = 10
aoi_bbox = BBox(bbox=aoi_coords_wgs84, crs=CRS.WGS84)
aoi_size = bbox_to_dimensions(aoi_bbox, resolution=resolution)

print(f"Image shape at {resolution} m resolution: {aoi_size} pixels")

catalog = SentinelHubCatalog(config=config)

start_date = datetime( year: 2024,  month: 9,  day: 1)
end_date = datetime( year: 2024,  month: 10,  day: 1)
time_interval = (start_date.strftime('%Y-%m-%d'), end_date.strftime('%Y-%m-%d'))

```

Рисунок 1.13 – Завдання області інтересу

Коли область інтересу визначена, шукаємо всі доступні знімки за вказаний період. В даному прикладі пошук виконується в колекції даних SENTINEL2_L2A, також параметром fields можна визначити, які параметри треба включити в результати пошуку (рис. 1.14). В прикладі це ідентифікатор знімка та дата його зйомки.

```
search_iterator = catalog.search(
    DataCollection.SENTINEL2_L2A,
    bbox=aoi_bbox,
    time=time_interval,
    fields={"include": ["id", "properties.datetime"], "exclude": []},
)

results = list(search_iterator)
print("Total number of results:", len(results))

```

Рисунок 1.14 – Пошук доступних даних

Також важливою частиною запиту до SentinelHub є evalscript (або «Скрипт користувача») - це фрагмент коду Javascript (рис. 1.15), який визначає, як супутникові дані будуть оброблятися Sentinel Hub і які значення повинна повертати служба.

```
evalscript_true_color = """
    //VERSION=3

    function setup() {
        return {
            input: ["B02", "B03", "B04"],
            output: { bands: 3 }
        };
    }

    function evaluatePixel(sample) {
        return [sample.B04, sample.B03, sample.B02];
    }
"""

```

Рисунок 1.15 – Приклад evalscript для оптичного діапазону знімків

input: ["B02", "B03", "B04"]: Вказує, які спектральні смуги зображення будуть використані як вхідні дані. В даному випадку це смуги B02, B03 і B04, які зазвичай відповідають зеленому, червоному та синьому кольорам у видимій частині спектра.

output: { bands: 3 }: Вказує, що на виході ми отримаємо зображення з трьома каналами (RGB), що візуально сприйматиметься як природне кольорове зображення.

Наступний фрагмент коду (рис. 1.16) завантажує всі знайдені знімки і робить окремий запит для кожного з них з необхідними параметрами.

```python
for idx, result in enumerate(results):
    capture_date = result['properties']['datetime']
    time_interval_single = (capture_date, capture_date)
    print(f"Завантаження знімка за {capture_date}...")

    request_true_color = SentinelHubRequest(
        evalscript=evalscript_true_color,
        input_data=[
            SentinelHubRequest.input_data(
                data_collection=DataCollection.SENTINEL2_L2A.define_from(
                    name="s2l2a", service_url="https://sh.dataspace.copernicus.eu"
                ),
                time_interval=time_interval_single,
                other_args={"dataFilter": {"mosaickingOrder": "leastCC"}},
            )
        ],
        responses=[SentinelHubRequest.output_response( identifier: "default", MimeType.PNG)],
        bbox=aoi_bbox,
        size=aoi_size,
        config=config,
    )
    true_color_imgs = request_true_color.get_data()
    image = true_color_imgs[0]
```

Рисунок 1.16 - Цикл завантажує зображення Sentinel-2 і обробляє їх у оптичному діапазоні.

Також запит має додаткові параметри other_args={"dataFilter": {"mosaickingOrder": "leastCC"}}: ці аргументи вказують на сортування знімків за найменшою хмарністю.

Останнім етапом автоматизованого завантаження знімків є їхнє збереження (рис. 1.17). Кожен знімок зберігається з доданою до його назви датою зйомки, що дозволяє в подальшому формувати датасети. Також для підвищення яскравості знімків зображення помножується на числовий коефіцієнт 3.5.

```
    true_color_imgs = request_true_color.get_data()
    image = true_color_imgs[0]

    image_brightened = image * 3.5
    image_brightened = np.clip(image_brightened, a_min: 0, a_max: 255).astype(np.uint8)

    image_filename = f"KLAIPEDA_{capture_date[:10].replace('-', '_')}.png"
    plt.imsave(image_filename, image_brightened)
    print(f"Збережено знімок: {image_filename}")

print("Завантаження всіх знімків завершено.")
```

Рисунок 1.17 – Збереження знайдених знімків за датою

Для завантаження знімків з синтетичною апертурою необхідно замінити колекцію пошуку на SENTINEL1_IW (Додаток Б) та змінити evalscript (рис. 1.18).

```
evalscript_sar = """
    function setup() {
        return {
            input: ["VV", "VH", "dataMask"],
            output: { bands: 3, sampleType: "FLOAT32" }
        };
    }

    function toDb(linear) {
        return 10 * Math.log10(linear);
    }

    function evaluatePixel(sample) {
        var vv_db = toDb(sample.VV);
        var vh_db = toDb(sample.VH);
        var mask = sample.dataMask;
        return [vv_db * mask, vh_db * mask, mask];
    }
"""

```

Рисунок 1.18 - Приклад evalscript для знімків SAR

Функція setup визначає, які дані будуть на вході і які будуть на виході.

input:

- "VV" - дані вертикальної поляризації (Vertical-Vertical);
- "VH" - дані вертикальної переданої і горизонтальної прийнятої поляризації (Vertical-Horizontal);
- "dataMask" - маска даних, яка визначає, чи доступний піксель для аналізу (зазвичай 1 — доступний, 0 — недоступний).

output:

- bands: 3 - на виході будуть три канали (або "смуги");
- sampleType: "FLOAT32" - вихідні дані будуть представлені у форматі з плаваючою точкою (32-бітова точність).

Функція toDb перетворює SAR-дані з лінійної шкали у логарифмічну (децибели) так як лінійна шкала не завжди зручна для інтерпретації.

```
request_sar = SentinelHubRequest(
    evalscript=evalscript_sar,
    input_data=[
        SentinelHubRequest.input_data(
            data_collection=DataCollection.SENTINEL1_IW.define_from("s1iw", service_url=config.sh_base_url),
            time_interval=time_interval_single,
            other_args={
                "dataFilter": {
                    "resolution": "HIGH",
                    "mosaickingOrder": "mostRecent",
                    "orbitDirection": "ASCENDING",
                },
                "processing": {
                    "backCoeff": "SIGMA0_ELLIPSOID",
                    "orthorectify": True,
                    "demInstance": "COPERNICUS",
                    "speckleFilter": {
                        "type": "LEE",
                        "windowSizeX": 3,
                        "windowSizeY": 3,
                    },
                },
            },
        )
    ],
    responses=[SentinelHubRequest.output_response( identifier: "default", MimeType.TIFF)],
    bbox=aoi_bbox,
    size=aoi_size,
```

Рисунок 1.19 – Запит і обробка знімків Sentinel-1

В запиті вказуються окрім основних параметрів додаткові аргументи, такі як напрямок орбіти, роздільна здатність зображення, визначається тип коефіцієнта зворотного розсіювання, вказується джерело цифрової моделі висот та фільтри шумів.

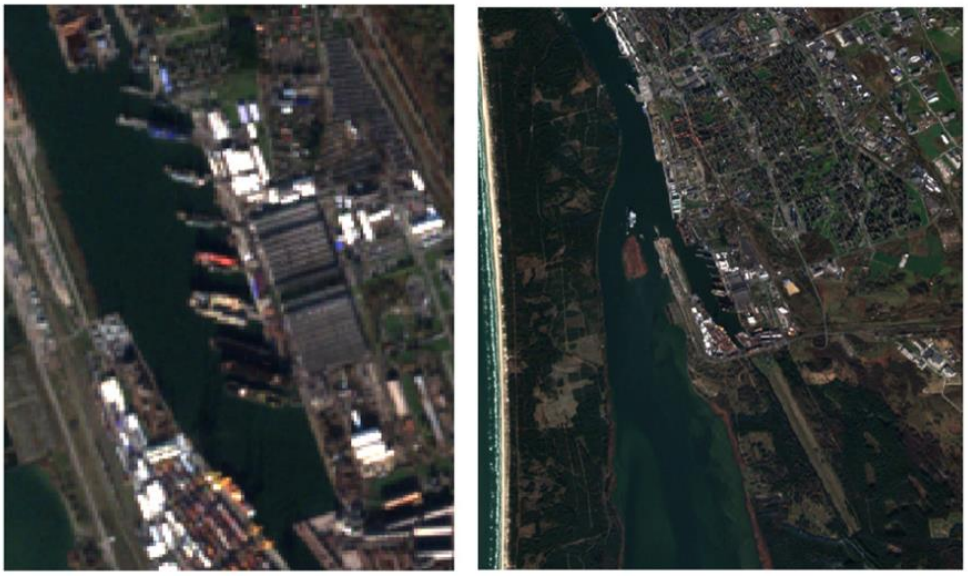
Заключним етапом автоматичного завантаження SAR-знімків є виділення каналів вертикальної та вертикально-горизонтальної поляризації та їх об'єднання шляхом розрахунку їхнього середнього арифметичного. Після чого результат множиться на маску даних, що дозволяє відфільтрувати невалідні пікселі. Потім знімки як і в випадку з оптичним діапазоном зберігаються з датою зйомки в імені файлу (рис. 1.20).

```
127     sar_imgs = request_sar.get_data()
128     sar_image_vv = sar_imgs[0][:, :, 0]  # Канал VV
129     sar_image_vh = sar_imgs[0][:, :, 1]  # Канал VH
130     data_mask = sar_imgs[0][:, :, 2]  # Маска даних
131
132     if np.sum(data_mask) == 0:
133         print(f"Знімок за {capture_date} не містить даних.")
134         continue
135
136     combined_sar_image = (sar_image_vv + sar_image_vh) / 2 * data_mask
137
138     image_filename = f"ODESSA_2_{capture_date[:10].replace('-', '_')}_{idx + 1}.png"
139     save_image(combined_sar_image, image_filename)
140     print(f"Збережено знімок: {image_filename}")
141
142 print("Завантаження всіх SAR знімків завершено.")
```

Рисунок 1.20 – Збереження SAR-знімків

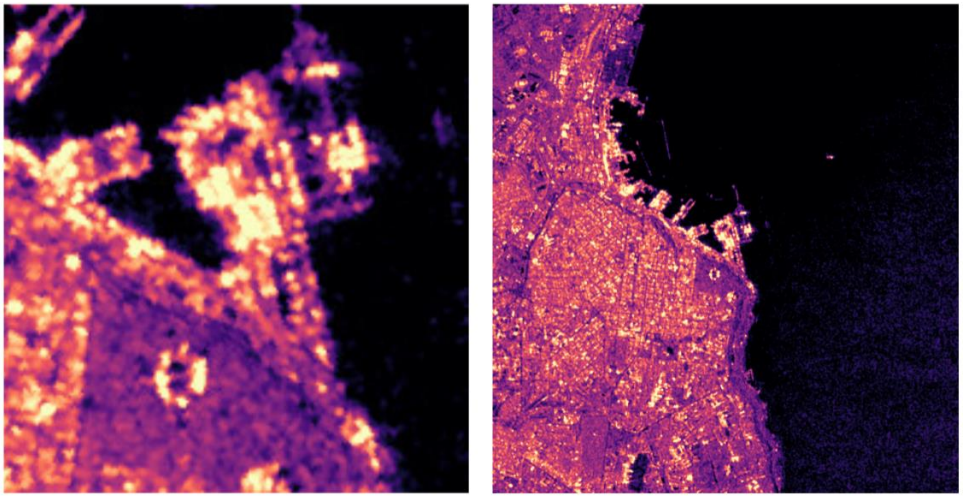
Результат роботи python-скрипту для автоматизації завантаження супутникових знімків в оптичному діапазоні розглянемо на прикладі суднобудівного заводу міста Клайпеда (рис.1.21 а, б).

Результати роботи скрипту для SAR-діапазону перевіримо на прикладі порту міста Одеси (рис.1.22 а, б).



А) Б)

Рисунок 1.21 – Приклад супутникового знімку:
(а) – для квадрату в 2 км, (б) – для квадрату 10 км



А) Б)

Рисунок 1.22 – Приклад SAR знімку:
(а) - для квадрату в 2 км, (б) – для квадрату 10 км

Також слід порівняти кількість доступних знімків оптичного та SAR-діапазону в Copernicus Browser з кількістю знімків автоматично завантажених за допомогою python-коду (рис. 1.23 - 1.24).

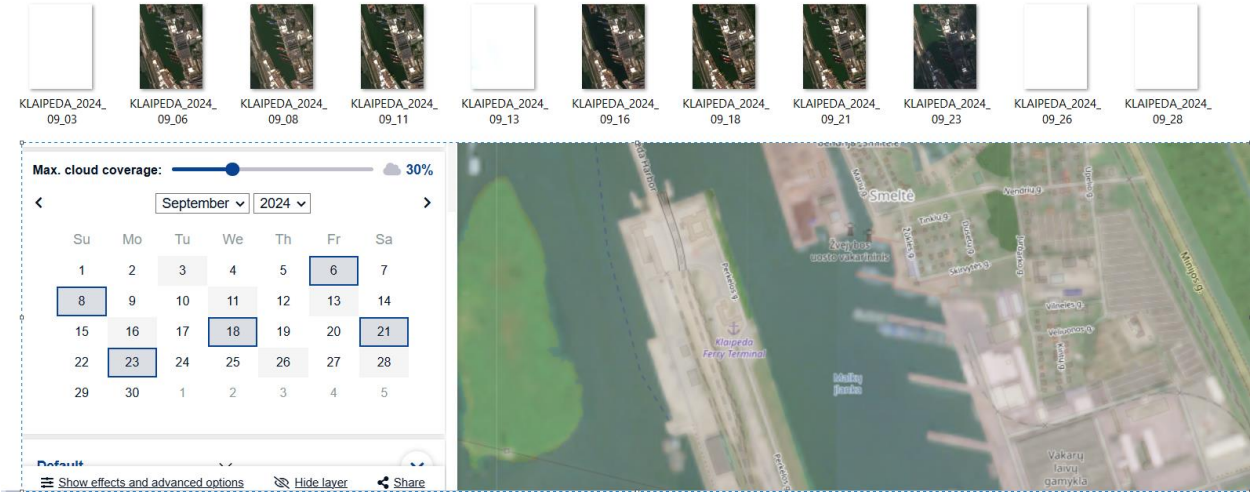


Рисунок 1.23 – Порівняння доступності супутникових даних оптичного діапазону

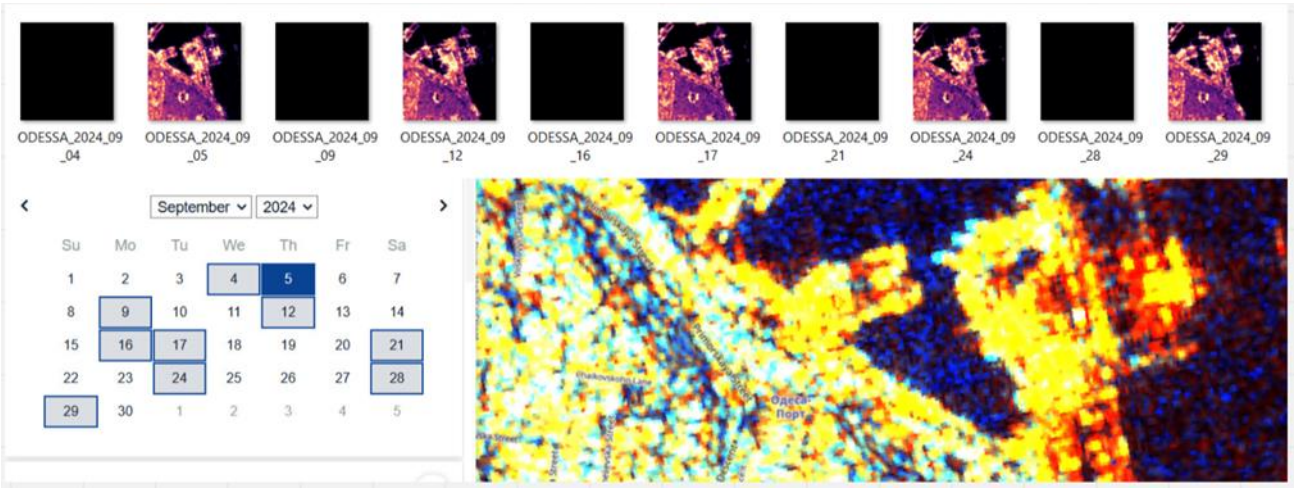


Рисунок 1.24 – Порівняння доступності знімків з синтетичною апертурою

З цього видно, що написаний скрипт завантажує абсолютно всі доступні знімки для обраної області інтересу в оптичному діапазоні. Але при завантаженні SAR-знімків завантажується половина з доступних в Copernicus Browser, друга половина несе зображення з невалідними пікселями.

Висновок

У розділі було розглянуто основні типи супутникових знімків, такі як: SAR, оптичні, гіперспектральні та інфрачервоні. Було наведено популярні джерела доступу до супутникових даних. Основну увагу приділено Copernicus Data Space Ecosystem та супутникам Sentinel-1 та Sentinel- 2 як основному джерелу SAR та оптичних супутникових даних. SAR-знімки дозволяють отримувати дані незалежно від погодних умов, оптичні знімки забезпечують деталізоване зображення в видимій частині спектра. Ці програми забезпечують регулярне оновлення супутникових знімків, що дозволяє проводити моніторинг земної поверхні в динаміці. Наведено приклади алгоритмів для автоматизованого завантаження знімків, що значно спрощує процес їх обробки та аналізу.

					171.6321М.КР.ПЗ.Р2		
Змн.	*Арк.*	*№ документу*	*Підпис*	*Дата*			
Розробник		Вишневський В.В.			Обробка супутникових знімків за допомогою нейромежі	*Літ.*	*Арк.* 31 / *Аркушів*
Консультант							
Н. контр.		Добрінова Л. П.				НУК імені адмірала Макарова	
Керівник		Дьяконов О.С.					
Зав. каф.		Рябенький В. М.					

РОЗДІЛ 2. ОБРОБКА СУПУТНИКОВИХ ЗНІМКІВ ЗА ДОПОМОГОЮ НЕЙРОМЕРЕЖІ

2.1. Види нейронних мереж детектування об’єктів

Завантаження супутникових знімків – це лише перший крок у процесі аналізу супутникових даних. Набагато складнішим завданням є вилучення з цих знімків корисної інформації. Одним із варіантів вирішення цієї задачі є використання нейронних мереж. Сучасні нейронні мережі надають потужні інструменти для автоматичного детектування, класифікації, та сегментації об'єктів на супутникових знімках [9].

Детектування об’єктів – це фундаментальна задача комп’ютерного зору, яка дозволяє алгоритмам автоматично знаходити та ідентифікувати різноманітні об’єкти на зображеннях або відео. Сучасні методи, використовують глибоке навчання, та згорткові нейронні мережі (CNN), які забезпечують високу точність у вирішенні завдання детектування. Методи виявлення об'єктів можна розділити на два основні типи: одноступеневі та двоступеневі детектори об'єктів (рис.2.1).

Загалом, детектори об'єктів на основі глибокого навчання отримують ознаки з вхідного зображення або відеокадру і вирішують дві послідовні задачі:

- Завдання №1: Знайти довільну кількість об'єктів (можливо, навіть нуль).
- Завдання №2: Класифікувати кожен окремий об'єкт та виділити його розмір за допомогою обмежувальної рамки.

Щоб спростити процес двоступеневі детектори ділять ці завдання на два етапи. Спочатку вони генерують регіони, які можуть містити об'єкти (регіональні пропозиції), а потім класифікують ці регіони та уточнюють їхні межі. Перевагами двоступеневих детекторів є точність вони зазвичай демонструють кращі результати особливо для складних сцен і маленьких об'єктів. Прикладом двоступеневого детектора є Faster R-CNN. В той час як

одноступеневі детектори поєднують обидва завдання в один етап та за один прохід по зображенню вони одночасно прогнозують як класи об'єктів, так і їхні координати в обмежувальних рамках. Перевагами такого підходу є швидкість, що досягається завдяки простій архітектурі, та ефективність, так як вони добре підходять для ситуацій, де швидкість обробки є критично важливою. Прикладами одноступеневих детекторів є: YOLO (You Only Look Once) та SSD (Single Shot MultiBox Detector).

Рисунок 2.1 – Види детекторів

Faster R-CNN - це двоступенева модель детектування об'єктів, яка використовує Регіональну Пропозиційну Мережу RPN (Region Proporsal Networks), яка була розроблена замість Selective Searc [10]. В основі RPN лежить система якорів. Архітектура Faster R-CNN утворена так, що зображення подається на вхід згорткової нейронної мережі, яка формує карту ознак. Карта ознак обробляється шаром RPN, де рухоме вікно проходить по карті ознак. Центр рухомого вікна пов'язаний із центром якорів. Якоря - це області, що мають різні співвідношення сторін та різні розміри. Існують 3 співвідношення сторін та 3 розміри. На основі метрики IoF (intersection-over-union) ступеня перетину якорів та дійсних розмічених прямокутників, виноситься рішення про поточний регіон – є в ньому об'єкт чи ні. Далі використовується алгоритм FastCNN, коли карта ознак з отриманими об'єктами передаються шару RoI з подальшою обробкою повнозв'язкових шарів і класифікацією, а також з визначенням зміщення регіонів потенційних об'єктів.

SSD (Single Shot MultiBox Detector) - архітектура SSD, що базується на попередньо навченій загортковій нейронній мережі (CNN). Зазвичай використовуються мережа типу VGG16. Відмінною рисою SSD є використання багатомасштабних карт ознак [11]. Ці карти збирають інформацію з різною роздільною здатністю, що дозволяє SSD ефективно виявляти об'єкти різних розмірів. Карти ознак з більш високою роздільною здатністю підходять для виявлення більш дрібних об'єктів, в той час як карти з більш низькою роздільною здатністю обробляють більші об'єкти. SSD використовує техніку, яка називається default boxs у кожному місці на картах об'єктів. Ці boxs мають різні співвідношення сторін і масштаби, надаючи різноманітний набір потенційних об'єктів. Кожен default box пов'язаний із двома наборами прогнозів:

- Оцінки класу, що вказують на ймовірність приналежності об'єкта до певного класу.

- Зміщення обмежуючої рамки: ці зміщення уточнюють default box, щоб він краще відповідав фактичному місцезнаходженню об'єкта.

YOLO (You Only Look Once) - алгоритм виявлення об'єктів в реальному часі, розроблений Джозефом Редмоном і Алі Фархаді в 2015 році. Це одноступеневий детектор об'єктів, який використовує згорткову нейронну мережу (CNN) для прогнозування обмежувальних рамок і ймовірних класів об'єктів на вхідних зображеннях. Перший YOLO був реалізований за допомогою фреймворка Darknet [12].

Алгоритм YOLO поділяє вхідне зображення на сітку, і для кожної комірки він передбачає ймовірність присутності об'єкта та координати обмежувальної рамки. Він також прогнозує клас об’єкту. На відміну від двоступеневих детекторів, таких як R-CNN і йому подібних, YOLO обробляє все зображення за один прохід, що робить його більш швидким і ефективним.

Процес декції YOLO можна розбити на кілька етапів:

1. Вхідне зображення пропускається через згорткову нейронну мережу для вилучення ознак із зображення.
2. Потім ознаки пропускаються через ряд повнозв’язних шарів, які передбачають вірогідність класів і координати обмежувальної рамки.
3. Зображення розділено на сітку, де кожна з комірок відповідає за прогнозування набору обмежувальних рамок і ймовірностей класів.
4. На виході мережа отримує набір обмежувальних рамок і ймовірностей кожного класу для кожної комірки.
5. Потім обмежуючі рамки фільтруються за допомогою алгоритму постобробки, який видаляє рамки, які перекривають одна одну і залишає ту, яка найбільш точно виділяє об’єкт.
6. Кінцевим результатом є набір прогнозованих обмежувальних рамок і класів для кожного об’єкта на зображенні.

Порівняння алгоритмів детектування

Для оцінки ефективності моделей виявлення об'єктів зазвичай використовується кілька показників:

1. Середня точність за всіма класами (mAP) – це метрика, яка використовується для оцінки якості моделей виявлення об'єктів. Вона обчислюється як середнє значення середніх точностей для кожного класу об'єктів. Середня точність для одного класу визначається як площа під кривою Precision-Recall (PR), яка відображає залежність між точністю та повнотою моделі при різних порогах класифікації. Точність показує, яка частка виявлених об'єктів є правильними, а повнота – яка частка всіх релевантних об'єктів була виявлена.
2. IoU, або перетин над об'єднанням, є метрикою, яка оцінює наскільки добре модель локалізує об'єкти на зображенні. Вона визначається як відношення площі перетину передбаченої та справжньої області до площі їх об'єднання (рис.2.2).
3. Recall: Показує, яка частка всіх релевантних об'єктів була виявлена моделлю. Це міра того, наскільки добре модель знаходить усі об'єкти, які насправді є на зображенні.
4. Precision: Показує, яка частка виявлених об'єктів є справді релевантними. Іншими словами, це міра того, наскільки часто модель не помиляється, коли стверджує, що знайшла об'єкт.

Рисунок 2.2 – Візуалізація IoU

Нижче наведено діаграму (рис.2.3) швидкості та точності основних методів детектування об'єктів (R-CNN, Fast R-CNN, Faster R-CNN, YOLO та SSD300).

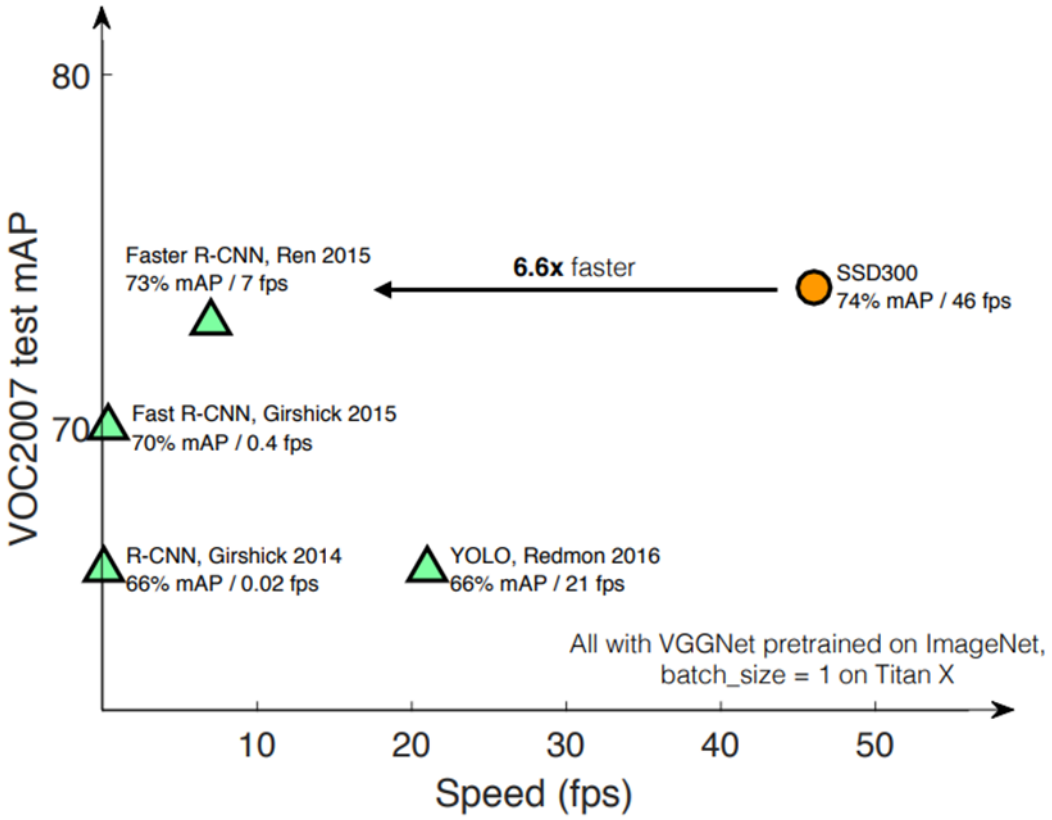


Рисунок 2.3 – Порівняльна діаграма алгоритмів детектування

2.2. Нейромережа YOLOv8

YOLOv8 - це остання розробка в лінійці алгоритмів YOLO, призначених для виявлення об'єктів у реальному часі [13]. Вона базується на досягненнях попередніх версій та пропонує нові функції та оптимізації, що значно покращують точність та швидкість роботи (рис.2.4). Завдяки своїм характеристикам YOLOv8 є ідеальним вибором для широкого спектру застосунків, що вимагають швидкого та точного виявлення об'єктів.

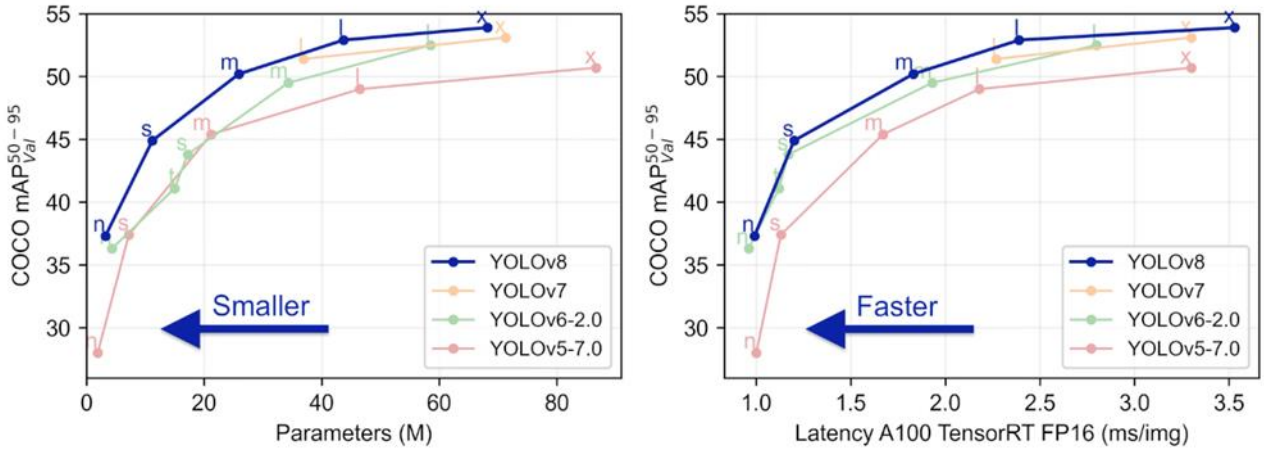


Рисунок 2.4 – Порівняння Версій YOLO

Основні характеристики:

- Завдяки новій, більш ефективній будові основних компонентів мережі, YOLOv8 краще вловлює деталі на зображеннях та точно визначає, що саме на них зображено.
- YOLOv8 використовує інноваційний підхід до виявлення об'єктів, який дозволяє йому працювати більш точно та ефективно, ніж методи, що покладаються на попередньо задані параметри якорів.
- Нейромережа ідеально балансує між точністю результатів та швидкістю обробки, що робить її ідеальним вибором для завдання обробки відео в реальному часі.
- YOLOv8 пропонує широкий вибір моделей, які вже навчені на великій кількості даних. Що дозволяє легко підібрати модель, яка найкраще підходить для конкретної задачі.

Серія моделей YOLOv8 пропонує широкий спектр рішень для різноманітних завдань у галузі комп'ютерного зору. Кожна модель спеціалізується на конкретній задачі, починаючи від простого виявлення об'єктів (рис. 2.6) на зображенні і закінчуючи складнішими завданнями, такими як сегментація (рис. 2.7), визначення положення ключових точок об'єктів, та класифікація зображень. У кожній з цих категорії є п'ять моделей.

YOLOv8 Nano - найшвидша і найменша, у той час як YOLOv8 Extra Large (YOLOv8x) - найточніша, але й найповільніша серед них (рис. 2.5).

Рисунок 2.5 – Моделі, що доступні в YOLOv8

Модель	Розмір (пікселів)	mAPval 50-95	Шввидкість CPU ONNX (мс)	Шввидкість A100 TensorRT (мс)	params (м)	FLOPs (B)
YOLOv8n	640	37.3	80.4	0.99	3.2	8.7
YOLOv8s	640	44.9	128.4	1.20	11.2	28.6
YOLOv8m	640	50.2	234.7	1.83	25.9	78.9
YOLOv8l	640	52.9	375.2	2.39	43.7	165.2
YOLOv8x	640	53.9	479.1	3.53	68.2	257.8

Рисунок 2.6 – Показники продуктивності детектування (COCO)

Модель	Розмір (пікселів)	mAPbox 50-95	mAPmask 50-95	Шввидкість CPU ONNX (мс)	Шввидкість A100 TensorRT (мс)	params (м)	FLOPs (B)
YOLOv8n-seg	640	36.7	30.5	96.1	1.21	3.4	12.6
YOLOv8s-seg	640	44.6	36.8	155.7	1.47	11.8	42.6
YOLOv8m-seg	640	49.9	40.8	317.0	2.18	27.3	110.2
YOLOv8l-seg	640	52.3	42.6	572.4	2.79	46.0	220.5
YOLOv8x-seg	640	53.4	43.4	712.1	4.02	71.8	344.1

Рисунок 2.7 – Показники продуктивності сегментації (COCO)

З технічної точки зору YOLOv8 - це група моделей згорткових нейронних мереж, створених та навчених з використанням фреймворку PyTorch.

PyTorch - це фреймворк глибокого навчання з відкритим вихідним кодом, відомий своєю гнучкістю та простотою використання. Це досягається

за рахунок його сумісності з популярною мовою програмування високого рівня Python.

Дві основні особливості PyTorch:

- Тензорні обчислення (подібні до NumPy) з потужною підтримкою прискорення GPU (графічного процесора).
- Автоматична диференціація для створення та навчання глибоких нейронних мереж.

2.3. Навчання YOLOv8 для детектування суден на супутникових знімках

Процес навчання моделі комп'ютерного зору полягає в оптимізації її внутрішніх параметрів для мінімізації розбіжностей між прогнозами моделі та фактичними значеннями. Спочатку модель отримує великий набір позначених зображень. Вона робить прогнози щодо того, що знаходиться на цих зображеннях, потім ці прогнози порівнюються з фактичними мітками, щоб визначити помилки. Ці помилки показують, наскільки далекі прогнози моделі від справжніх значень.

У процесі навчання модель ітеративно робить передбачення, обчислює помилки та оновлює свої параметри за допомогою процесу, що називається зворотним розповсюдженням помилки (backpropagation) (рис. 2.8). Під час цього процесу модель коригує свої внутрішні параметри (weights and biases), щоб зменшити значення помилки. Повторюючи цей цикл багато разів, модель поступово підвищує свою точність. Згодом вона вчиться розпізнавати складні моделі, такі як форми, кольори та текстури.

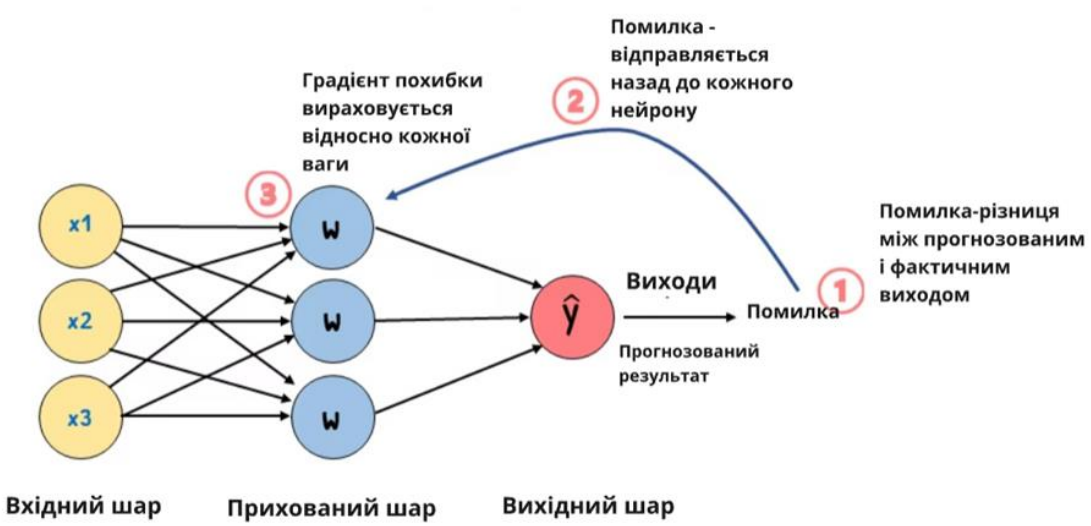


Рисунок 2.8 – Зворотне розповсюдження помилки

Підготовка даних для навчання

Підготовка даних є критичним етапом у процесі навчання будь-якої нейромережі. Якість і кількість даних значною мірою впливають на точність і надійність моделі. Першим етапом підготовки є збір даних. Для цього завантажуємо супутникові знімки в оптичному і SAR-діапазоні, набір даних має містити зображення, зроблені за різних умов освітлення, з різними кутами огляду та в різних середовищах. Тому супутникові знімки завантажувалися з різних портів і частин моря та з різним рівнем хмарності (рис. 2.9 – 2.10).

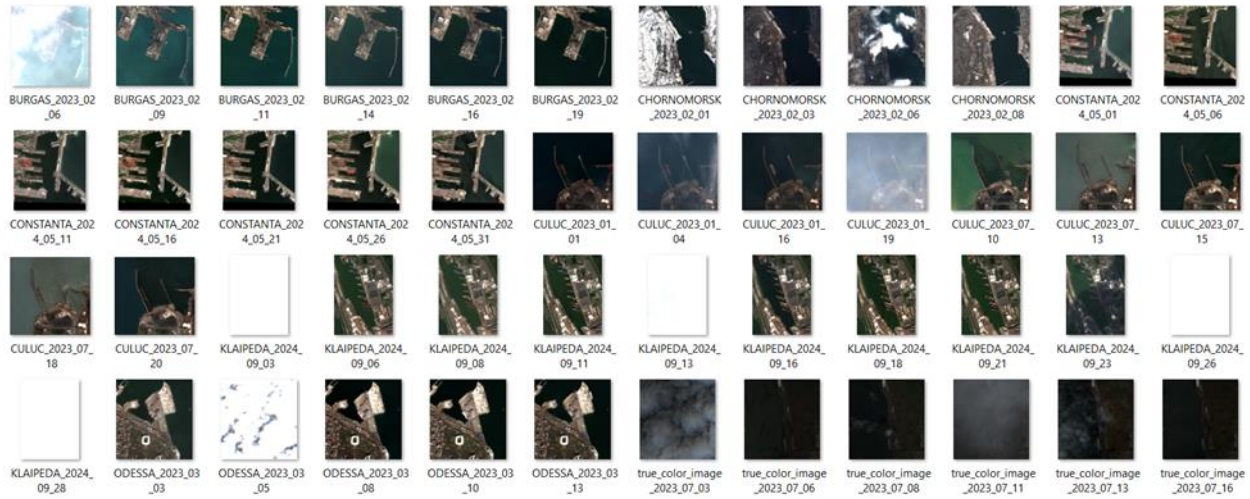


Рисунок 2.9 – Приклад оптичних супутникових даних

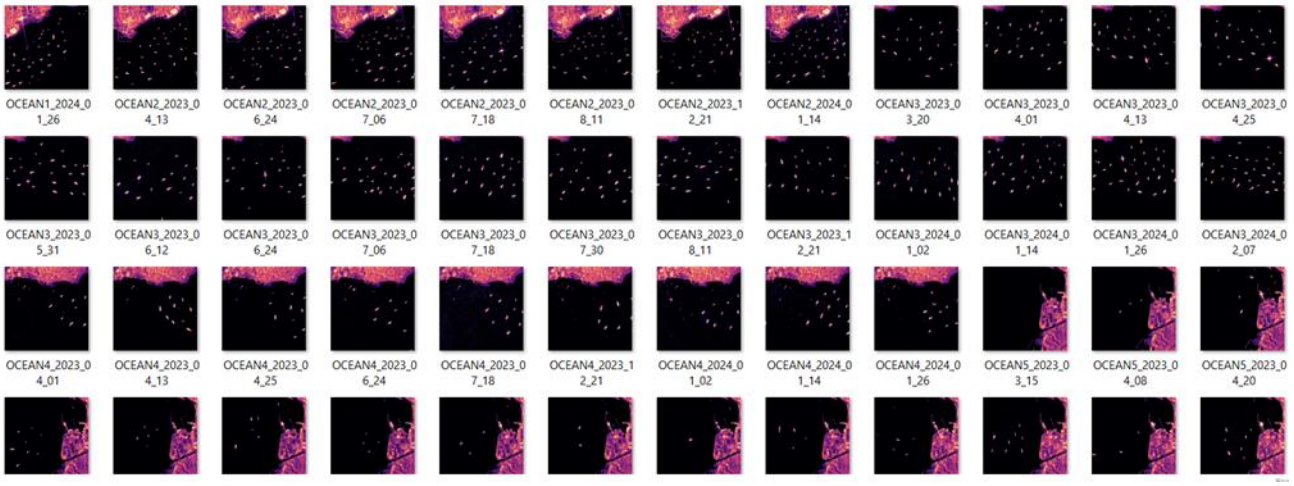


Рисунок 2.10 – Приклад SAR-даних

Наступним кроком є анотація даних, тобто на кожному зображенні повинні бути позначені всі об'єкти, які модель повинна виявляти. Анотації зображень можуть мати різні формати, навіть якщо вони містять одну й ту саму інформацію. Серед різних існуючих форматів два користуються найбільшою популярністю - це формат COCO JSON і формат YOLOv5 PyTorch TXT.

Нейромережа YOLOv8 використовує формат YOLOv5 PyTorch TXT де кожне зображення має один файл txt з одним рядком для кожної

обмежувальної рамки. Формат кожного рядка: class_id center_x center_y width height (рис.2.11).

```
1 0.617 0.3594420600858369 0.114 0.17381974248927037
1 0.094 0.38626609442060084 0.156 0.23605150214592274
1 0.295 0.3959227467811159 0.13 0.19527896995708155
1 0.785 0.398068669527897 0.07 0.14377682403433475
```

Рисунок 2.11 – Приклад анотаційного файлу YOLOv8

Існує багато відкритих інструментів для анотації даних, які значно полегшують процес підготовки датасету. Популярними рішеннями є [14]:

- Label Studio - це інструмент маркування даних з відкритим вихідним кодом. Він дозволяє маркувати різні типи даних, такі як аудіо, текст, зображення та відео. Він має простий і зручний інтерфейс і дозволяє експортувати дані в різні формати моделей. Його можна використовувати для підготовки необроблених даних або для поліпшення існуючих навчальних даних.
- Computer Vision Annotation Tool (CVAT) – це інструмент з відкритим вихідним кодом для розмітки цифрових зображень та відео. Основним його завданням є надання користувачеві зручних та ефективних засобів розмітки наборів даних. Також у CVAT є web-додаток, що працює в браузері. Він підтримує різні сценарії роботи, як індивідуальну так і командну роботу над розміткою.
- Labelme - це графічний інструмент для анотування зображень, створений за мотивами сайту labelme.csail.mit.edu. Він написаний на Python і використовує бібліотеку Qt для графічного інтерфейсу.
- Roboflow - це потужна платформа, розроблена для спрощення роботи з даними. Вона пропонує широкий спектр інструментів, які допомагають

створювати, керувати та покращувати набори даних для навчання моделей машинного зору [15].

Основні функціональні можливості Roboflow:

- Анотація зображень: простий і інтуїтивно зрозумілий інтерфейс для створення точних анотацій об'єктів на зображеннях.
- Управління даними: ефективне управління великими наборами даних, включаючи організацію, фільтрацію та пошук.
- Аугментація даних: широкий спектр методів аугментації для збільшення різноманітності набору даних і поліпшення узагальнення моделі.
- Інтеграція з моделями: легка інтеграція з популярними моделями машинного зору, такими як YOLO, Faster R-CNN та іншими.
- Деплоймент: можливість розгортання готових моделей на різних платформах.

Для навчання нейромережі детектування суден на знімках з супутників було обрано саме платформу Roboflow, так як вона максимально підходить для створення датасету в форматі yolo. Також платформа працює в браузері і надає можливість розмічати дані одразу декільком анотувальникам одночасно, що значно пришвидшує процес. Першим кроком є завантаження нерозмічених супутникових знімків на сайт та їх розподіл по команді, безкоштовний план платформи Roboflow передбачає одночасне анотування даних трьома людьми. Процес анотування даних полягає в виділенні потрібного об’єкту, в даному випадку судна обмежувальною рамкою (рис.2.12).

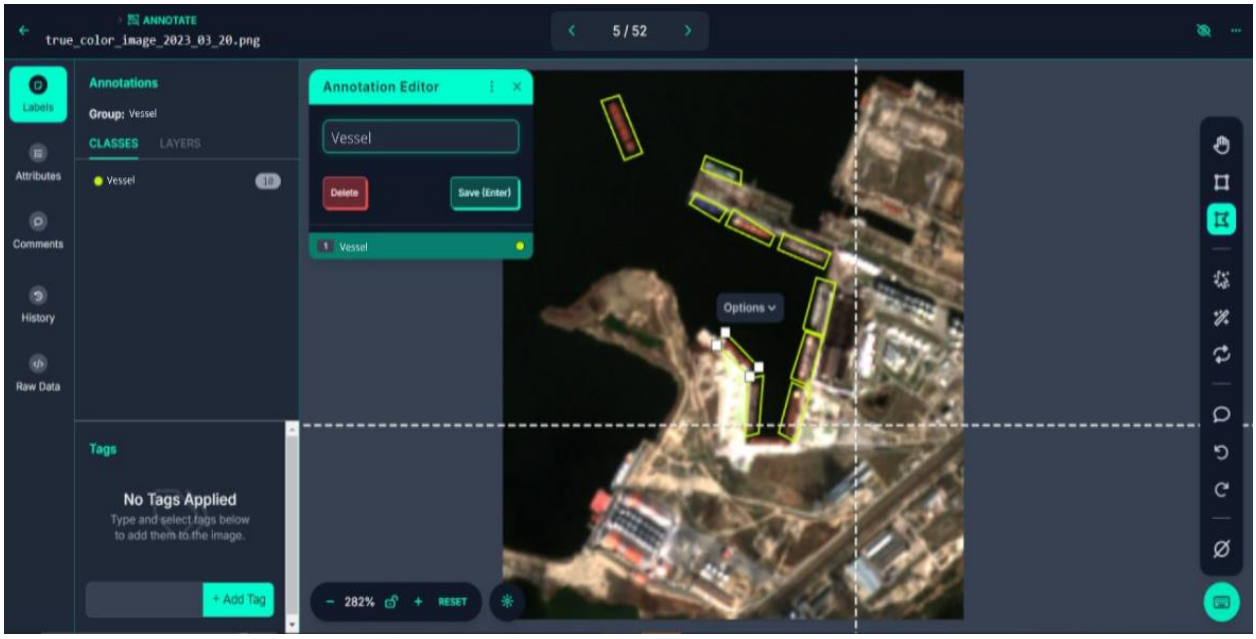


Рисунок 2.12 – Анотація суден на знімку в Roboflow

Після анотації даних їх необхідно розбити на групи. Розбиття даних на три частини - тренувальну, валідаційну та тестову - є фундаментальним принципом у машинному навчанні (рис. 2.13 –2.14). Це дозволяє:

- Навчати модель: тренувальна вибірка використовується для того, щоб модель "навчилася" виявляти закономірності в даних.
- Оцінювати модель під час навчання: валідаційна вибірка використовується для оцінки того, наскільки добре модель узагальнюється і для підбору оптимальних гіперпараметрів.
- Об'єктивно оцінювати кінцеву модель: тестова вибірка використовується для остаточної оцінки моделі після завершення навчання і підбору гіперпараметрів. Це дозволяє отримати неупереджену оцінку того, наскільки добре модель працюватиме на нових, раніше не бачених даних.

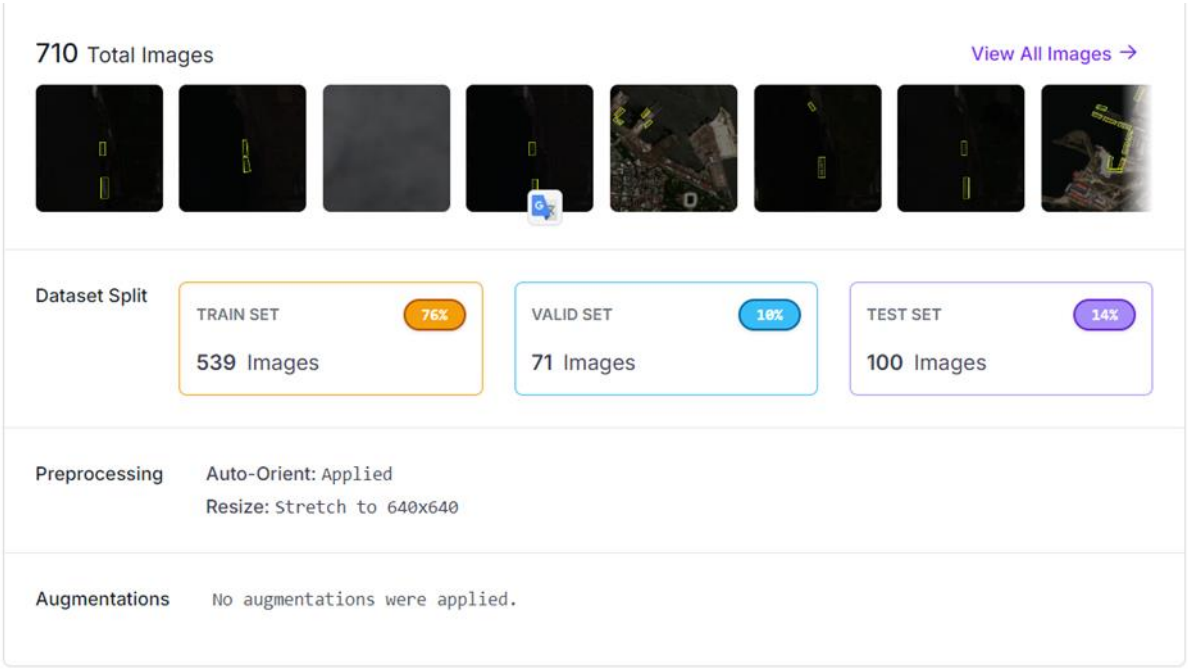


Рисунок 2.13 - Розділення набору оптичних даних

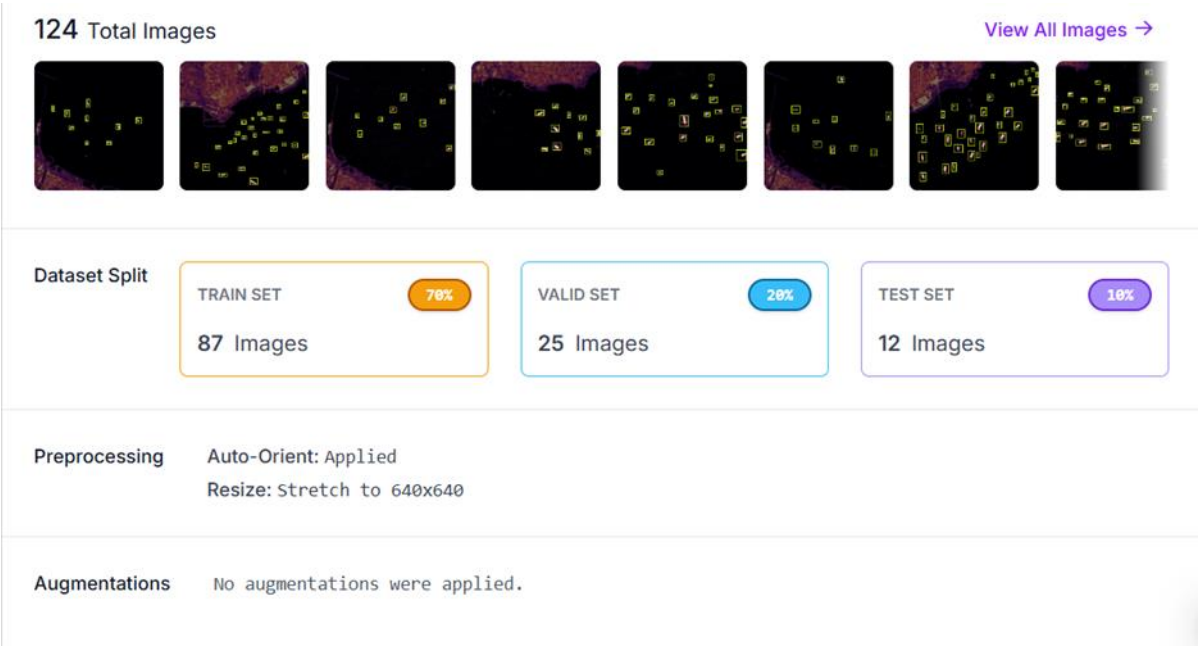


Рисунок 2.14 – Розділення набору даних SAR

Після підготовки датасету наступає етап навчання нейронної мережі. Навчати нейронну мережу можна на персональному комп'ютері. Для цього необхідно інсталювати необхідні бібліотеки і модулі, такі як, PyTorch та програмно-апаратну архітектуру паралельних обчислень (CUDA). Для

навчання нейронної мережі необхідно мати персональний комп’ютер з хорошими технічними характеристики. Рекомендуються процесори з великою кількістю ядер та потоків. Також необхідно мати потужну відеокарту. Чим більше відеопам'яті, тим більшого розміру моделі можна навчати. Також необхідна велика кількість оперативної пам'яті (мінімум 16 ГБ, бажано 32 ГБ або більше), вона необхідна для розміщення даних та проміжних обчислень.

Іншим варіантом є використання для навчання платформи Google Colaboratory, також відомої як Google Colab, вона була розроблена компанією Google Research у 2017 році. Це безкоштовне хмарне онлайн-середовище Jupyter Notebook, яке дозволяє навчати моделі машинного навчання на CPU, GPU та TPU.

Google Colab можна використовувати незалежно від специфікацій та конфігурацій локального комп'ютера. Все, що потрібно для використання платформи це обліковий запис Google і веб-браузер.

Навчання нейронної мережі в Google Colab

Для початку необхідно змінити в налаштуваннях Google Colab середовище виконання з CPU на GPU. Перевірити це і отримати інформацію про доступне відео-ядро можна командою !nvidia-smi (рис. 2.15), так як за свою основу Google Colab бере Jupyter Notebook, то весь код запускається в окремих комірках, що дає змогу одразу бачити результат виконання необхідних команд (Додаток В).

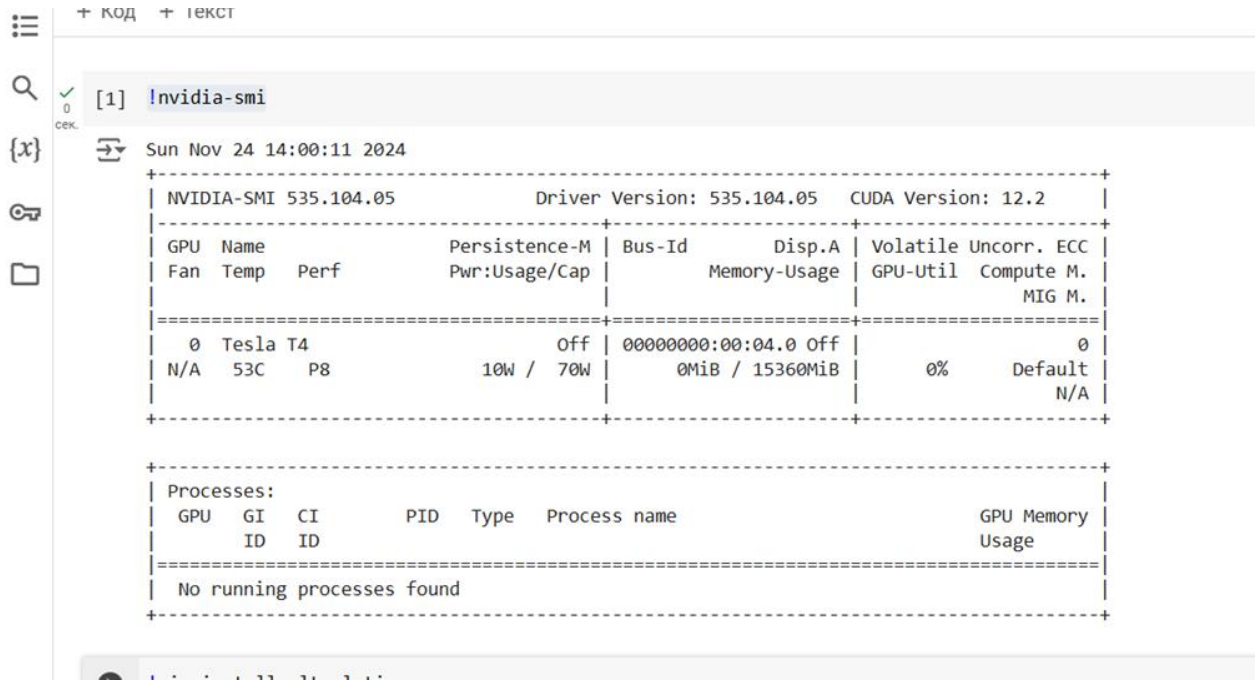


Рисунок 2.15 – Інформація про відео-ядро Google Colab

Так як в Google Colab заздалегідь встановлені всі основні бібліотеки для машинного навчання, такі як TensorFlow, Keras та PyTorch, то для роботи з YOLOv8 необхідно встановити лише бібліотеку ultralytics та імпортувати необхідні модулі, щоб підготувати середовище для роботи (рис. 2.16).

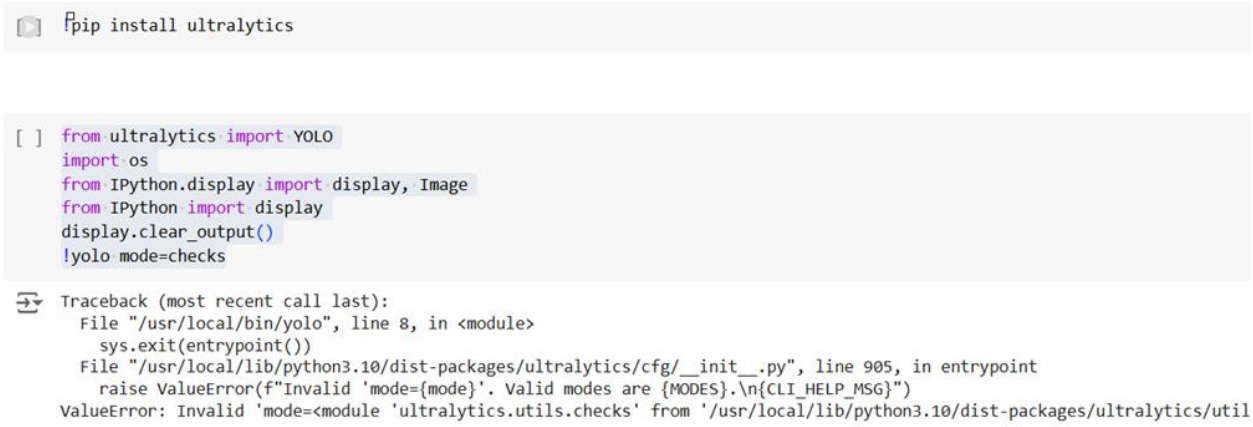


Рисунок 2.16 – Імпорт необхідних компонентів в Google Colab

Наступним кроком є підключення датасету з Roboflow, для цього необхідно вказати ключ вашого проекту, його версію та для якої нейромережі генерувати дата сет (рис. 2.17).

```
[ ]  !pip install roboflow

     from roboflow import Roboflow
     rf = Roboflow(api_key="[redacted]")
     project = rf.workspace("education-pwtb2").project("sar-aurpa")
     version = project.version(1)
     dataset = version.download("yolov8")
```

Рисунок 2.17 – Підключення датасету

Для роботи з датасетом нейромережа YOLOv8 потребує спеціальної структури папок для набору даних. Спеціальна структура допомагає підтримувати порядок у великих наборах даних і полегшує процес навчання моделі. Також необхідно вказати в конфігураційному файлі місце знаходження зображень та відповідних ним анотацій (рис. 2.18).

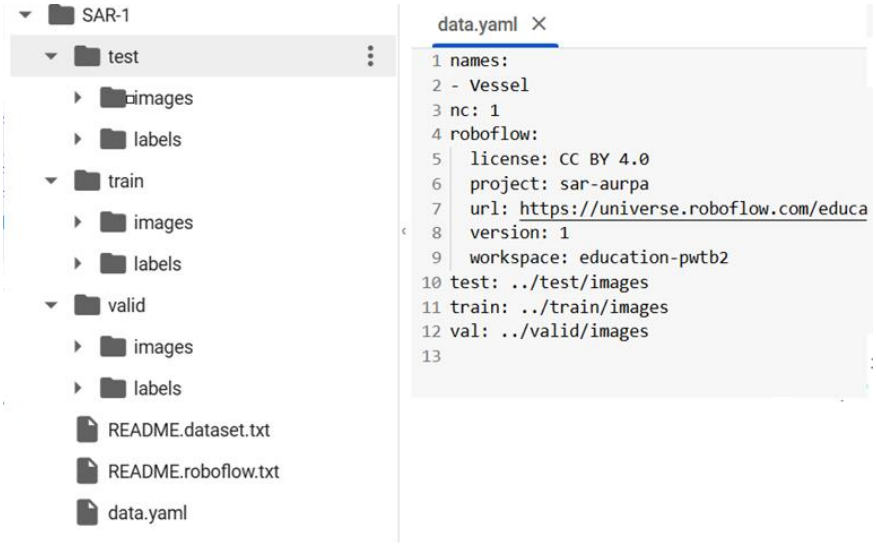


Рисунок 2.18 – Структура папок YOLOv8

Після налаштування конфігураційного файлу можна переходити до навчання нейромережі. Для цього необхідно прописати в окрему комірку

команду !yolo task=detect mode=train model=yolov8m.pt data={dataset.location}/data.yaml epochs=50 imgsz=640, де вказати параметри для навчання, такі як задача - в даному випадку це детектування, кількість епох для навчання та розмір зображень датасету. Після запуску цієї команди почнеться навчання нейромережі (рис. 2.19).

```
Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
1/100      3.67G      3.826      6.272      3.243         14        640: 100% 10/10 [00:13<00:00,  1.39s/it]
           Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100% 1/1 [00:01<00:00,  1.58s/it]
             all          5          6    0.00333      0.833    0.00739    0.00265

Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
2/100      3.6G      2.837      4.299      2.545         20        640: 100% 10/10 [00:03<00:00,  2.59it/s]
           Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100% 1/1 [00:00<00:00,  4.89it/s]
             all          5          6     0.0252      0.167     0.0244    0.00789

Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
3/100      3.62G      2.376      4.115      2.011          9        640: 100% 10/10 [00:02<00:00,  3.61it/s]
           Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100% 1/1 [00:00<00:00, 11.57it/s]
             all          5          6      0.181      0.167     0.0355     0.0106

Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
4/100      3.72G      2.349      3.946       1.96         16        640: 100% 10/10 [00:02<00:00,  3.62it/s]
           Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100% 1/1 [00:00<00:00, 11.75it/s]
             all          5          6     0.0254      0.167     0.0115    0.00447

Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
5/100      3.7G      2.303      3.708      1.926         10        640: 100% 10/10 [00:03<00:00,  2.74it/s]
           Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100% 1/1 [00:00<00:00,  6.21it/s]
             all          5          6    0.00121      0.167   0.000895   0.000351
```

Рисунок 2.19 – Процес навчання нейромережі

Після того як модель завершить навчання, вона створить шлях /runs/detect/train, в якому знаходитиметься сама навчена нейромережа з назвою файлу best.pt і файли з метриками для оцінки.

Розглянемо результати навчання нейромережі для детектування суден на супутникових знімках для оптичного (рис. 2.21) і для SAR- (рис. 2.23) діапазонів. Вибірка даних для оптичного діапазону містила в собі 710 супутникових знімків з яких в навчанні брали участь 539. Слід зауважити, що відсотків 25 або більше знімків були з високою хмарністю і не несли в собі потрібних даних. Нейромережа навчалася на 20 епохах (рис. 2.20).

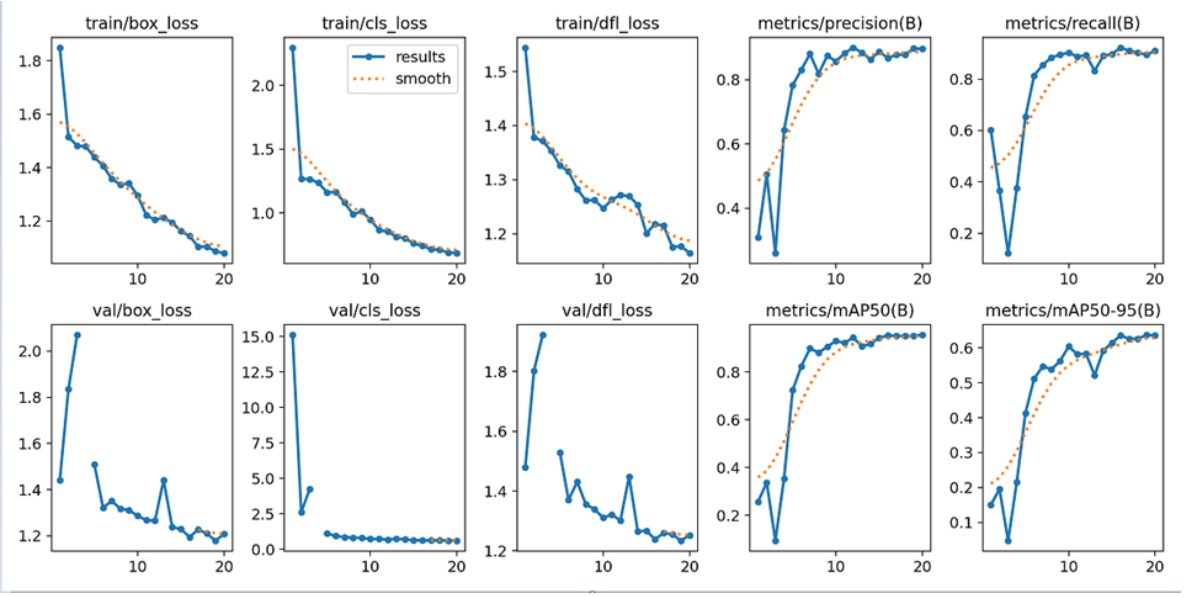


Рисунок 2.20 – Оціночні метрики для нейромережі,
що навчалася на оптичних даних

Рисунок 2.21 – Результат роботи нейромережі, навченої на оптичних знімках

Вибірка даних для SAR діапазону містила в собі 124 знімки, з яких в навчанні приймали участь 87 фото. Нейромережа навчалася на 50 епохах (рис. 2.22).

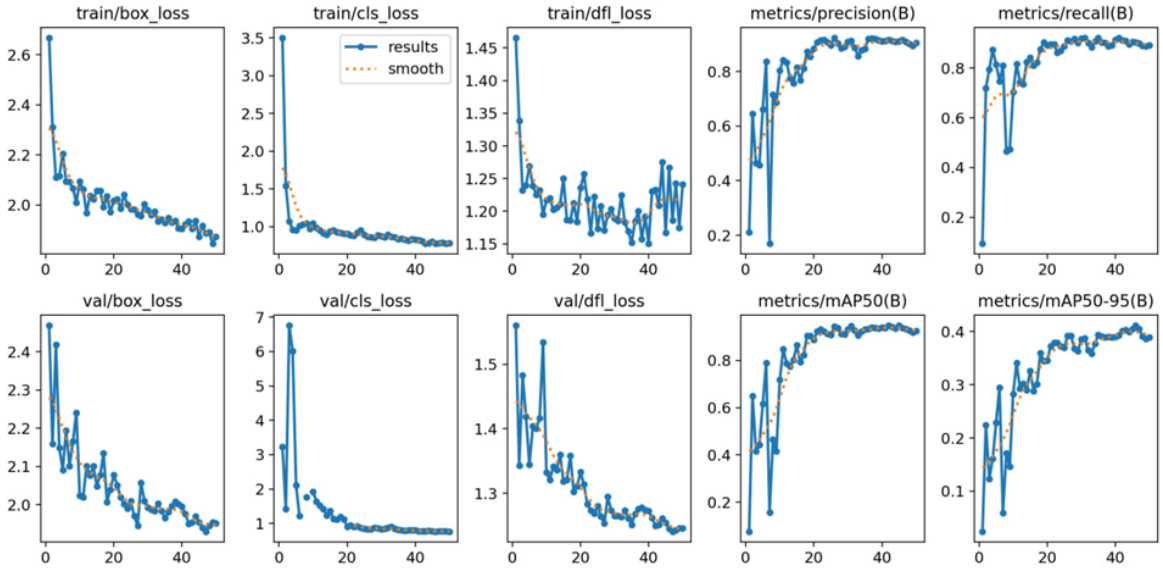


Рисунок 2.22 - Оціночні метрики для нейромережі,
що навчалася на SAR-даних

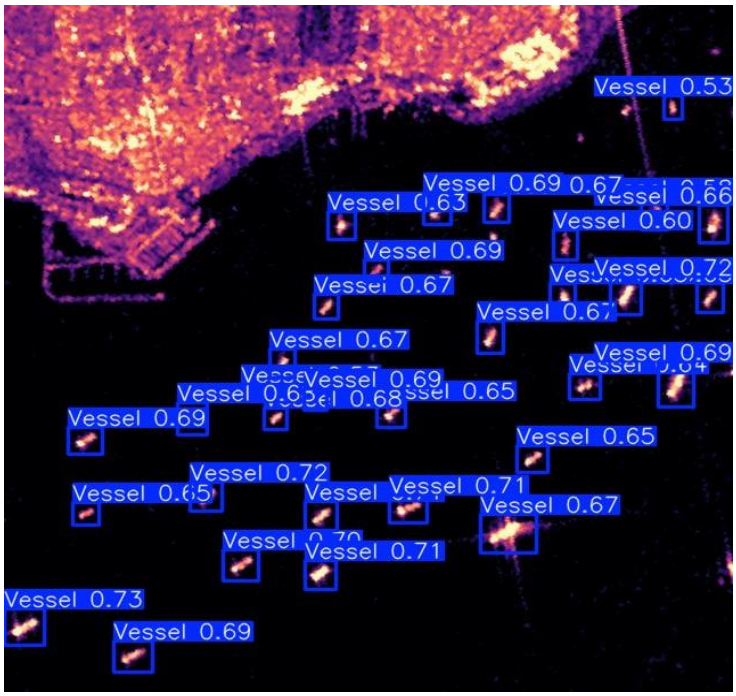


Рисунок 2.21 – Результат роботи нейромережі, навченої на SAR-знімках

Навчання Yolov8 за допомогою датасету створеного з використанням даних AIS

Для ефективного вирішення задачі детектування необхідні великі і якісні датасети, анотовані вручну. Однак, ручне створення таких датасетів є

трудомістким процесом. Можна розглянути інший підхід, який дозволяє автоматизувати цей процес за допомогою інтеграції даних AIS. Це дозволить значно скоротити час, необхідний для створення датасетів. Основним джерелом даних AIS може слугувати сайт MarineTraffic. Цей сайт надає інформацію в реальному часі про рух суден та про їх фактичне місце перебування в гаванях або портах.

Для того, щоб отримати дані про місце знаходження судна з сайту можна скористатися скрапінгом. Скрапінг – це процес автоматичного збору даних з веб-сторінок. Приклад скрапера для парсингу даних про судна в Чорному морі наведено в додатку Г. Скрапер збирає дані і автоматично створює відповідні директорії створюючи папки за датами і часом парсингу Дані про судна збираються в форматі JSON (рис. 2.22).


```
{
    "LAT": "40.89596",
    "LON": "27.4698",
    "SPEED": "0",
    "COURSE": "339",
    "HEADING": "203",
    "ELAPSED": "2",
    "DESTINATION": "TRTEK",
    "FLAG": "LR",
    "LENGTH": "399",
    "ROT": "0",
    "SHIPNAME": "MSC LEANNE",
    "SHIPTYPE": "7",
    "SHIP_ID": "4849568",
    "WIDTH": "58",
    "L_FORE": "159",
    "W_LEFT": "25",
    "DWT": "202461",
    "GT_SHIPTYPE": "11"
},
```


Рисунок 2.22 – Дані про судно отримані з MarineTraffic

Коли ми маємо супутникові знімки, що розбиті за датами і дані про місце знаходження суден за ці дні, то необхідно розробити скрипт, який буде виявляти судна, які знаходяться в межах супутникового знімку (додаток Ґ). Для цього необхідно виділити область інтересу аналогічно тому, як це

зроблено на рисунку 1.12, після чого необхідно створити функцію, яка б перевіряла, чи знаходиться судно в межах заданого квадрату (рис. 2.23).

```
22  bounding_box = {
23      'lon_min': aoi_coords_wgs84[0],
24      'lat_min': aoi_coords_wgs84[1],
25      'lon_max': aoi_coords_wgs84[2],
26      'lat_max': aoi_coords_wgs84[3]
27  }
28  def is_within_bounding_box(lat, lon, bounding_box):
29      return (bounding_box["lat_min"] <= lat <= bounding_box["lat_max"] and
30              bounding_box["lon_min"] <= lon <= bounding_box["lon_max"])
31
```

Рисунок 2. 23 – Функція для перевірки, чи знаходиться судно в межах супутникового знімку

Далі необхідно перебрати папки з даними. Ітерація іде по папках за датами. Для кожної дати перевіряється наявність папки map. Якщо папка існує, перевіряється наявність файлів з даними про судна. Скрипт зчитує дані з JSON-файлів та перевіряє, чи потрапляють координати судна в заданий квадрат. Якщо так то інформація про судно додається до списку ships_in_area (рис. 2.24).

```
            if os.path.getsize(map_file_path) > 0:
                with open(map_file_path, 'r', encoding='utf-8') as f:
                    data = json.load(f)

                for ship in data['data']['rows']:
                    lat = float(ship['LAT'])
                    lon = float(ship['LON'])

                    if is_within_bounding_box(lat, lon, bounding_box):
                        ship['date'] = date_obj.strftime('%Y-%m-%d')
                        ship['time'] = time_str
                        ships_in_area.append(ship)
            else:
                print(f"Skip empty file: {map_file_path}")
        except json.JSONDecodeError:
            print(f"Error reading JSON in file: {map_file_path}")
        except Exception as e:
            print(f"Enother error with the file  {map_file_path}: {e}")
```

Рисунок 2.24 – Створення списку зі знайденими суднами

Після чого цей список зберігається в JSON файл (рис. 2.25).


```
[
    {
        "LAT": "44.15615",
        "LON": "28.6461",
        "SPEED": "0",
        "COURSE": "21",
        "HEADING": "277",
        "ELAPSED": "2",
        "DESTINATION": "RO CND",
        "FLAG": "BS",
        "LENGTH": "189",
        "ROT": "0",
        "SHIPNAME": "CS SONOMA",
        "SHIPTYPE": "7",
        "SHIP_ID": "6942478",
        "WIDTH": "32",
        "L_FORE": "161",
        "W_LEFT": "11",
        "DWT": "58810",
        "GT_SHIPTYPE": "6",
        "date": "2024-11-25",
        "time": "12:00"
    }
]
```


Рисунок 2.25 – Дані про судна, що знайдені в межах супутникового знімку

Маючи супутникові знімки та список знайдених суден, необхідно перетворити географічні координати в пікселі на зображенні (додаток Д). Для

того, щоб пов’язати координати суден з MarineTraffic з супутниковим знімком (рис.2.26).

```
37  def geo_to_pixel(lon, lat, bbox, img_width, img_height):  1 usage
38      x_frac = (lon - bbox['lon_min']) / (bbox['lon_max'] - bbox['lon_min'])
39      y_frac = (lat - bbox['lat_min']) / (bbox['lat_max'] - bbox['lat_min'])
40
41      x_pixel = int(x_frac * img_width)
42      y_pixel = int((1 - y_frac) * img_height)  # Координати по Y перевернуті
43      return x_pixel, y_pixel
```

Рисунок 2.26 – Функція перетворення координат в пікселі

Наступний фрагмент коду перетворює координати суден в відповідні цим координатам пікселі і створює навколо них рамку в 70 на 70 пікселів, яку анотує в файл анотації (рис. 2.27).

```
85      detections = []
86      for ship_info in ships_data:
87          if ship_info["date"] == ship["date"]:
88              lat_ship = ship_info['lat']
89              lon_ship = ship_info['lon']
90              if is_within_bounding_box(lat_ship, lon_ship, bounding_box):
91                  x_center, y_center = geo_to_pixel(lon_ship, lat_ship, bounding_box, img_width, img_height)
92                  width, height = 70, 70
93                  detections.append((x_center, y_center, width, height))
94
95      if detections:
96          create_yolo_annotation(image_name.replace( _old: '.png', _new: ''), detections, img_width, img_height)
```

Рисунок 2.27 – Автоматичне створення анотації

Розглянемо проблематику автоматичної анотації даних. Вона полягає в тому, що неможна точно визначити обмежувальну рамку для судна, тому для великих суден рамка може бути замалою, а для маленьких суден завеликою. Інша проблема полягає в тому, що не всі судна, які є на супутниковому знімку стоять з ввімкненими AIS-транспондерами, що не дає автоматично анотувати ці судна (рис. 2.28). Таким чином, більшість суден залишаються не анотованими, через що моделі складно визначити необхідні залежності в даних. Також сама кількість супутникових знімків, на яких були

ідентифіковані судна є набагато меншою, ніж вся кількість знімків, яку можна отримати з супутника.

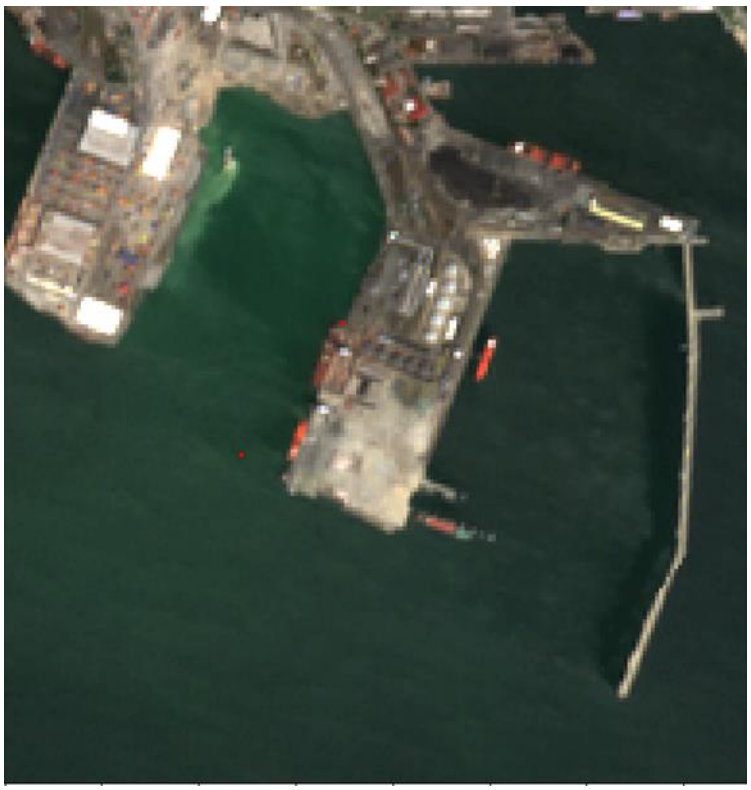


Рисунок 2.28 – Червоними точками візуалізовано дані отримані з AIS

Перевіримо роботу такого підходу для створення датасету на практиці. Вдалося зібрати і анотувати 84 супутникові знімки для оптичного діапазону з 5 різних портів. З цих знімків 79 пішло в тренувальну вибірку а 5 в тестову. Нейромережа навчалася на 100 епохах (рис. 2.29 - 2.30).

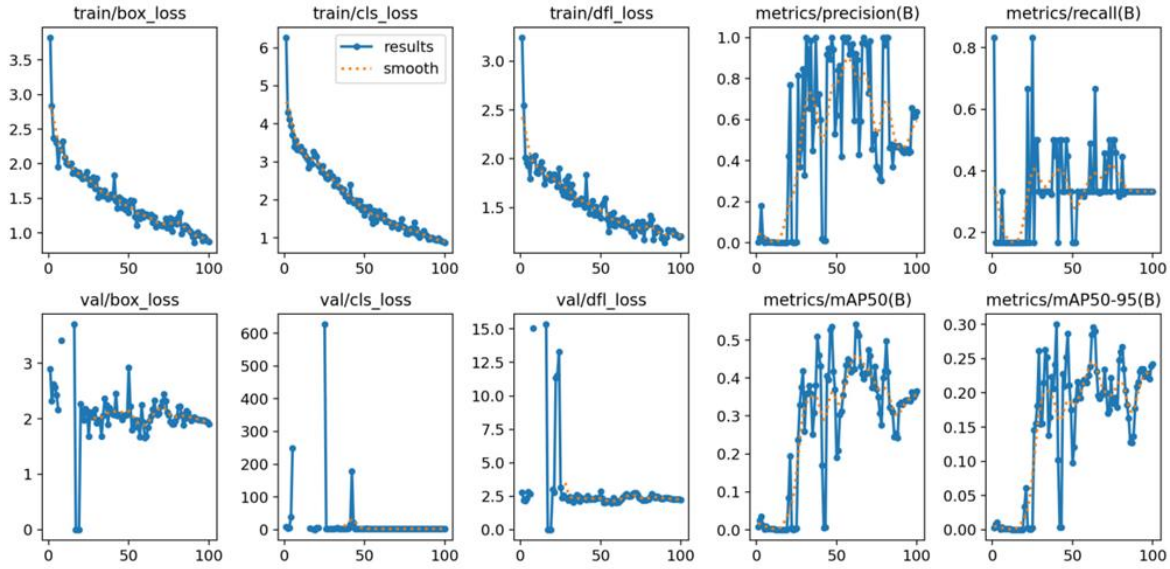


Рисунок 2.29 – Оціночні метрики нейромережі з автоматизованою анотацією

Рисунок 2.30 – Результат роботи нейромережі на автоматизованому датасеті

Висновок

У другому розділі було проведено дослідження застосування нейромереж для детекції суден на супутникових знімках. Були розглянуті різні архітектури нейромереж та проведено їх порівняння. Вибір YOLOv8 був обумовлений високою швидкістю роботи та точністю детектування.

Особливу увагу було приділено процесу підготовки даних для навчання нейромережі, а також створенню датасету на основі даних AIS. Навчання моделі здійснювалося в середовищі Google Colab, що дозволило ефективно використовувати обчислювальні ресурси.

Результати експериментів показали високу ефективність ручного підходу до створення датасету. Отримані моделі дозволяють точно виявляти судна на супутникових знімках, що відкриває широкі можливості для застосування. Автоматизоване створення датасету показало себе гірше через малий об'єм даних, що призводить до недостатньої узагальненості моделі. Також дані AIS можуть містити помилки та неточності, що ускладнює процес автоматичної розмітки. Збір більшого об’єму AIS-даних та розроблення алгоритмів їх нормалізації та стандартизації покращили б результати моделі.

					171.6321М.КР.ПЗ.РЗ		
Змн.	*Арк.*	*№ документу*	*Підпис*	*Дата*			
Розробник		Вишневський В.В.			Оцінка трафіку порту за даними AIS та супутниковими знімками	*Літ.*	*Арк.* 60 / *Аркушів*
Консультант							
Н. контр.		Добрінова Л. П.				НУК імені адмірала Макарова	
Керівник		Дьяконов О.С.					
Зав. каф.		Рябенький В. М.					

РОЗДІЛ 3. ОЦІНКА ТРАФІКУ ПОРТУ ЗА ДАНИМИ AIS ТА СУПУТНИКОВИМИ ЗНІМКАМИ

3.1. Автоматизоване завантаження даних AIS

Супутникові дані та дані AIS відкривають нові можливості для дослідження морської діяльності. Супутникові знімки надають детальну інформацію про морське середовище та прибережні зони. Дані AIS містять інформацію про рух суден в реальному часі, що дозволяє отримати більш повну картину. Джерела супутникових даних та алгоритми роботи з ними наведені в розділі 1.

Джерелами даних AIS можуть слугувати такі веб-сервіси як:

- MarineTraffic: один з найбільш популярних і доступних сервісів, який надає детальну інформацію про рух суден у всьому світі. Має зручний інтерфейс, дозволяє фільтрувати дані за різними параметрами (тип судна, порт призначення тощо) та будувати траєкторії руху [16].
- VesselsValue: крім відстеження суден, цей сервіс надає детальну інформацію про кожне судно, включаючи його технічні характеристики, історію зміни власників та ринкову вартість.
- AISHub: ще один популярний сервіс, який надає доступ до даних AIS в реальному часі. Має API для розробки власних додатків.

В даній роботі було обрано платформу MarineTraffic, так як вона має розширений функціонал та пропонує широкий спектр функцій, які дозволяють отримати детальну інформацію про рух суден, включаючи їх відстеження в реальному часі та історичні дані, що дозволяє аналізувати їхні маршрути та поведінку в минулому.

Також платформа надає детальну інформацію про судна, таку як порт приписки, технічні характеристики та багато іншого. Для того, щоб отримувати інформацію з сайту MarineTraffic, необхідно скористатися скрапінгом.

Скрапінг - це потужний інструмент, який дозволяє автоматично збирати дані з веб-сайтів. Скрапінг ідеально підходить для збору великих масивів даних, які було б нереально збирати вручну. Наприклад, можна зібрати ціни на товари з різних сайтів, відгуки клієнтів, новини, статті, контактну інформацію компаній тощо. Зібрані дані можна структурувати та зберігати у базах даних для подальшого аналізу та використання. В нашому випадку необхідно отримати з сайту інформацію про фактичне місце знаходження суден. На сайті ця інформація зберігається у вигляді необроблених JSON-даних (рис. 3.1).

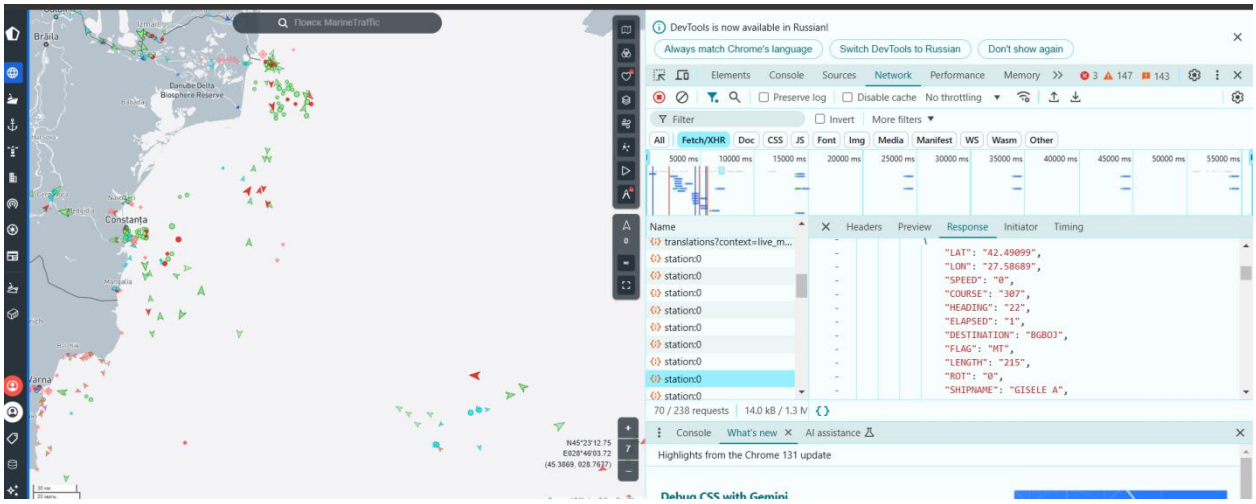


Рисунок 3.1 – Необхідні дані у вигляді відповіді сервера на запит сайту

Скрипт для автоматичного завантаження даних про фактичне місце знаходження суден наведено в додатку Г. В коді використовуються бібліотека Selenium. Це набір інструментів і бібліотек з відкритим кодом, що автоматизує тестування веб-сайтів та веб-додатків. Selenium імітує роботу браузера, що дозволяє отримувати автоматизований доступ до веб-сторінок та робить можливим завантаження даних та збереження їх у файли JSON. Більшість сайтів використовують Cloudflare, так як він є потужним інструментом для захисту веб-сайтів від різних загроз, включаючи автоматизовані атаки. Він активно використовується для запобігання скрапінгу даних та інших видів зловживань. Cloudflare аналізує патерни

трафіку, такі як швидкість запитів, тип запитів, використовувані заголовки HTTP та інші параметри. Для виявлення аномальної активності характерної для роботів сайт MarineTraffic використовує цей інструмент для захисту своїх даних. Тому необхідно замаскувати дії автоматизації. Для цього слід видалити певні властивості з глобального об'єкта Windows, які можуть бути використані для виявлення автоматизації браузера (рис. 3.2).

```
53
54      driver.execute_cdp_cmd( cmd: 'Page.addScriptToEvaluateOnNewDocument', cmd_args: {
55          'source': '''
56              delete window.cdc_adoQpoasnfa76pfcZLmcfl_Array;
57              delete window.cdc_adoQpoasnfa76pfcZLmcfl_Promise;
58              delete window.cdc_adoQpoasnfa76pfcZLmcfl_Symbol;
59              delete window.cdc_adoQpoasnfa76pfcZLmcfl_JSON;
60              delete window.cdc_adoQpoasnfa76pfcZLmcfl_Proxy;
61              delete window.cdc_adoQpoasnfa76pfcZLmcfl_Object;
62              '''
63      })
64
```

Рисунок 3.2 – Властивості, що видають автоматизацію браузера

Скрипт обходить блокування Cloudflare і завантажує дані про всі судна в Чорному морі. Далі він створює робочу директорію, в якій створює піддиректорію з датою куди зберігає файли з даними по ділянках карти (map). Кожен map-файл містить інформацію про судна, таку як його фактичні координати, ім’я, курс і т. д. (рис. 2.22). Також з сайту MarineTraffic можна отримати детальну інформацію про кожне конкретне судно (рис. 3.3).

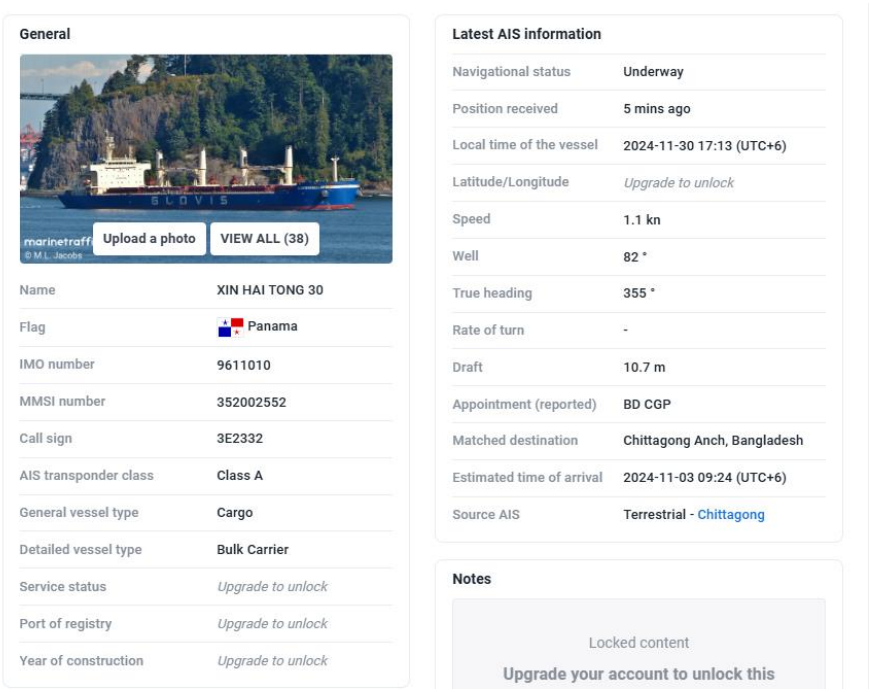


Рисунок 3.3 – Детальна інформація про судно на сайті MarineTraffic

Для цього необхідно скористатися бібліотекою BeautifulSoup – це потужна та універсальна бібліотека Python, яка використовується для веб-скрапінгу з метою вилучення даних із файлів HTML та XML. Вона розбирає веб-сторінку на частини, що дозволяє отримувати певні дані, такі як заголовки, посилання та таблиці (Додаток Е). Скрипт для кожного судна з вхідного JSON-файлу формує URL для сторінки судна на сайті MarineTraffic, відкриває сторінку в браузері та отримує HTML-код сторінки.

За допомогою BeautifulSoup [17] парситься HTML, знаходяться всі рядки таблиці. Для кожного рядка витягуються назва параметра (з тегу <th>) та його значення (з тегу <td>). Дані додаються до словника ship_data. Словник ship_data додається до списку collected_data (рис. 3.3).

```
with open(input_json, 'r', encoding='utf-8') as f:
    ship_list = json.load(f)
collected_data = []
try:
    for ship in ship_list:
        ship_name = ship.get('SHIPNAME')
        ship_id = ship.get('SHIP_ID')

        if not ship_name or not ship_id:
            print(f"Пропускаємо судно з неповними даними: {ship}")
            continue
        url = f'https://www.marinetraffic.com/en/ais/details/ships/shipid:{ship_id}/vessel:{ship_name}'
        print(f"Парсимо дані для судна: {ship_name} ({url})")
        driver.get(url)
        time.sleep(20)
        html_source = driver.page_source
        soup = BeautifulSoup(html_source, features: 'html.parser')
        rows = soup.find_all('tr')
        ship_data = {"SHIPNAME": ship_name, "SHIP_ID": ship_id}
        for row in rows:
            th_tag = row.find('th')
            td_tag = row.find('td')
            if th_tag and td_tag:
                key = th_tag.get_text(strip=True)
                value = td_tag.get_text(strip=True)
                ship_data[key] = value

        collected_data.append(ship_data)
```

Рисунок 3.3 – Парсинг HTML-сторінки за допомогою бібліотеки BeautifulSoup

В результаті роботи цих двох скриптів (Додаток Г, Е) можна отримати дані про фактичне місце заходження судна та детальну інформацію про ці судна.

3.2. Моніторинг трафіку порту за даними AIS

Виділивши область конкретного порту і відфільтрувавши дані AIS, знайшовши інформацію про всі судна, що заходили в порт за конкретний проміжок часу (Додаток Ґ) та зібравши детальну інформацію про кожне судно (Додаток Е), можна проаналізувати діяльність порту (Додаток Є).

Маючи інформацію про тип судна, можна зробити припущення про тип вантажу, що воно перевозить. Для цього треба створити словник з типовими випадками і можливими вантажами (рис. 3.4).

```python
DETAILED_VESSEL_TYPE_TO_CARGO = {
    "Bulk Carrier": "Сипучий вантаж",
    "Container Ship": "Контейнери",
    "Oil Tanker": "Нафтопродукти",
    "Gas Carrier": "Газ",
    "Chemical Tanker": "Хімічні речовини",
    "General Cargo Ship": "Загальний вантаж",
    "LPG Tanker": "Скраплений газ",
    "Other": "Інший",
    "Crude Oil Tanker": "Нафтопродукти"
}
```

Рисунок 3.4 - Словник переводить тип судна у тип вантажу

За ідентифікатором судна код отримує інформацію про нього з обох JSON-файлів. Знаходить тип судна і визначає відповідний тип вантажу. За значенням дедвейту DWT визначається маса вантажу, якщо вона доступна. Визначається зміна осадки між початковим і фінальним значенням. Результат використовується для визначення статусу судна (рис. 3.5):

- Loading: осадка збільшилась (завантаження).
- Unloading: осадка зменшилась (розвантаження).
- No significant change: осадка не змінилась.

```python
    file1_records = [record for record in file1 if record['SHIP_ID'] == ship_id]
    file2_records = [record for record in file2 if record['SHIP_ID'] == ship_id]
    if not file1_records or not file2_records:
        return None

    detailed_type = file1_records[0].get('Detailed vessel type', 'Other')
    cargo_type = DETAILED_VESSEL_TYPE_TO_CARGO.get(detailed_type, "Невідомий")

    cargo_mass = file1_records[0].get('DWT') or file2_records[0].get('DWT')

    initial_draught = float(file1_records[0]['Draught'].replace(' m', ''))
    final_draught = float(file1_records[-1]['Draught'].replace(' m', ''))
    draught_change = final_draught - initial_draught

    if draught_change > 0:
        loading_status = "Loading"
    elif draught_change < 0:
        loading_status = "Unloading"
    else:
        loading_status = "No significant change"
```

Рисунок 3.5 – Код, що аналізує дані про судно

Також за даними дати першої появи судна і дати коли інформація про судно зникає, можна визначити часовий діапазон активності кожного судна. Проаналізувавши всі судна, код виводить статистику про кожне з них, і статистику про загальну кількість суден різних типів, що заходили в порт (рис. 3.6).

```
for res in results:
    print(f"Судно: {res['ship_name']}")
    print(f"Тип вантажу: {res['cargo_type']}")
    print(f"Маса вантажу: {res['cargo_mass']} тонн")
    print(f"Час входу в порт: {res['entered_port']}")
    print(f"Час виходу з порту: {res['left_port'] if res['left_port'] else 'Судно ще в порту'}")
    print(f"Початкова осадка: {res['initial_draught']} м")
    print(f"Кінцева осадка: {res['final_draught']} м")
    print(f"Зміна осадки: {res['draught_change']} м")
    print(f"Статус: {res['loading_status']}")
    print("-" * 40)

print("Статистика за типами суден:")
for ship_type, summary in ship_type_summary.items():
    print(f"Тип судна: {ship_type}")
    print(f"Кількість суден: {summary['count']}")
    print(f"Перша дата: {summary['first_date']}")
    print(f"Остання дата: {summary['last_date']}")
    print("-" * 40)

print(f"Загальна кількість суден: {total_ships}")
```

Рисунок 3.6 – Виведення результатів моніторингу AIS даних

Розглянемо результат роботи коду на прикладі порту в місті Констанца, що розташований в Румунії. Моніторинг проводився в період з 18.11.2024 по 26.11.2024, також була додана інформація за 30.11.2024, що сумарно дає 9 днів спостережень (рис. 3.7 – 3.8).

```
Судно: SIMON STEVIN
Тип вантажу: Невідомий
Маса вантажу: 35815 тонн
Час входу в порт: 2024-11-22 12:00:00
Час виходу з порту: 2024-11-23 13:00:00
Початкова осадка: 6.5 м
Кінцева осадка: 6.5 м
Зміна осадки: 0.0 м
Статус: No significant change
----------------------------------------
Судно: ASAHI PRINCESS
Тип вантажу: Нафтопродукти
Маса вантажу: 105372 тонн
Час входу в порт: 2024-11-20 12:00:00
Час виходу з порту: 2024-11-30 12:00:00
Початкова осадка: 4.5 м
Кінцева осадка: 4.5 м
Зміна осадки: 0.0 м
Статус: No significant change
----------------------------------------
Судно: ETNA
Тип вантажу: Сипучий вантаж
Маса вантажу: 48549 тонн
Час входу в порт: 2024-11-30 12:00:00
Час виходу з порту: 2024-11-30 12:00:00
Початкова осадка: 8.5 м
Кінцева осадка: 8.5 м
Зміна осадки: 0.0 м
Статус: No significant change
----------------------------------------
Судно: ARUNA CIHAN
Тип вантажу: Сипучий вантаж
Маса вантажу: 57411 тонн
Час входу в порт: 2024-11-18 12:00:00
Час виходу з порту: 2024-11-21 12:00:00
Початкова осадка: 9.2 м
Кінцева осадка: 9.2 м
Зміна осадки: 0.0 м
Статус: No significant change
----------------------------------------
```

Рисунок 3.7 – Статистика для кожного судна

```
Статистика за типами суден:
Тип судна: Bulk Carrier
Кількість суден: 3
Перша дата: 2024-11-18 12:00:00
Остання дата: 2024-11-30 12:00:00
----------------------------------------
Тип судна: Pipe Burying Vessel
Кількість суден: 1
Перша дата: 2024-11-22 12:00:00
Остання дата: 2024-11-22 12:00:00
----------------------------------------
Тип судна: Crude Oil Tanker
Кількість суден: 1
Перша дата: 2024-11-20 12:00:00
Остання дата: 2024-11-20 12:00:00
----------------------------------------
Загальна кількість суден: 5
```

Рисунок 3.8 – Статистика діяльності порту

3.3. Моніторинг трафіку порту за супутниковими знімками

Зазвичай для моніторингу трафіку порту використовуються автоматизовані системи ідентифікації суден, такі як система AIS, що дозволяє відстежувати координати суден у реальному часі. Проте, система AIS має обмеження: не всі судна завжди мають ввімкнені транспондери, особливо в умовах високої небезпеки, таких як військові конфлікти. Наприклад, під час російсько-української війни після обстрілів портової інфраструктури судна часто вимикають транспондери, що робить дані AIS недоступними або неповними, що ускладнює точний моніторинг трафіку в порту. В таких випадках супутникові знімки можуть стати альтернативним джерелом для оцінки трафіку.

Супутники надають можливість отримувати зображення територій порту з висоти, що дозволяє фіксувати навіть ті судна, які вимкнули свої транспондери або взагалі не оснащені системою AIS. Скрипт наведений в Додатку Ж використовує алгоритми комп'ютерного зору для автоматичного аналізу зображень порту зроблених супутником.

Алгоритм завантажує супутникові знімки та попередньо натреновану модель YOLOv8, яка була навчена для ідентифікації суден на зображеннях. Процес навчання моделі розписано в підрозділі 2.3. Код перебирає всі знімки у вказаній директорії і на кожному детектує судна, зберігаючи їх координати та клас. Зберігаються тільки координати тих суден в точності детекції яких нейромережа впевнена більше ніж в 50% (рис. 3.9).

```
for image_name in os.listdir(image_folder):
    image_path = os.path.join(image_folder, image_name)
    image = cv2.imread(image_path)

    results = model.predict(source=image_path, conf=0.5)
    detections = results[0].boxes.xyxy
    classes = results[0].boxes.cls
    confidences = results[0].boxes.conf

    current_ships = []
    for i in range(len(detections)):
        x1, y1, x2, y2 = map(int, detections[i])
        confidence = float(confidences[i])
        if confidence > 0.5:
            current_ships.append([x1, y1, x2, y2, confidence])

```

Рисунок3.9 – Фрагмент коду для детектування суден

Далі за допомогою функції наведеної на рисунку 3.10 код порівнює списки координат поточних та попередніх суден, щоб визначити їх стаус:

- entering_ships - нові судна, що з'явились на зображенні (переміщення в порт).
- exiting_ships - судна, яких немає на поточному зображенні (вийшли з порту).
- standing_ships - судна, які не змінили своїх координат (стоять на місці).

Для цього порівнюються координати суден на двох зображеннях, і якщо відстань між координатами суден менша ніж поріг threshold, судна вважаються однаковими.

```python
def compare_ships(current_ships, previous_ships, threshold=50):  1 usage
    entering_ships = []
    exiting_ships = []
    standing_ships = []
    for current in current_ships:
        found_match = False
        for prev in previous_ships:
            if np.linalg.norm(np.array(current[:2]) - np.array(prev[:2])) < threshold:
                standing_ships.append(current)
                found_match = True
                break
        if not found_match:
            entering_ships.append(current)
    for prev in previous_ships:
        found_match = False
        for current in current_ships:
            if np.linalg.norm(np.array(prev[:2]) - np.array(current[:2])) < threshold:
                found_match = True
                break
        if not found_match:
            exiting_ships.append(prev)
    return entering_ships, exiting_ships, standing_ships
```

Рисунок 3.10 – Функція для визначення статусу судна

Після обробки кожного зображення, на ньому малюються прямокутники навколо суден:

- Зелені прямокутники для суден, що увійшли в порт (рис. 3.11, а).
- Червоні прямокутники для суден, що вийшли з порту (рис. 3.11, б).
- Блакитні прямокутники для суден, що залишилися на місці (рис. 3.11, б).

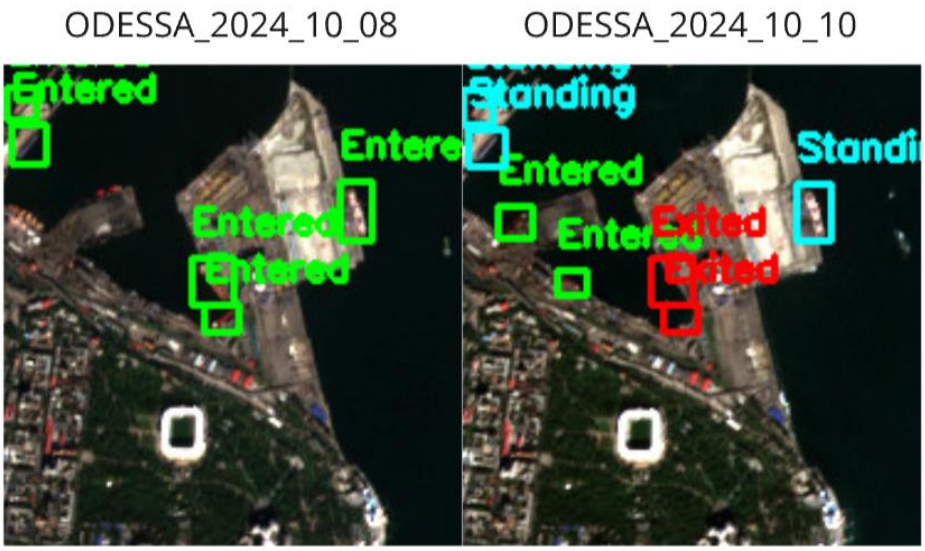
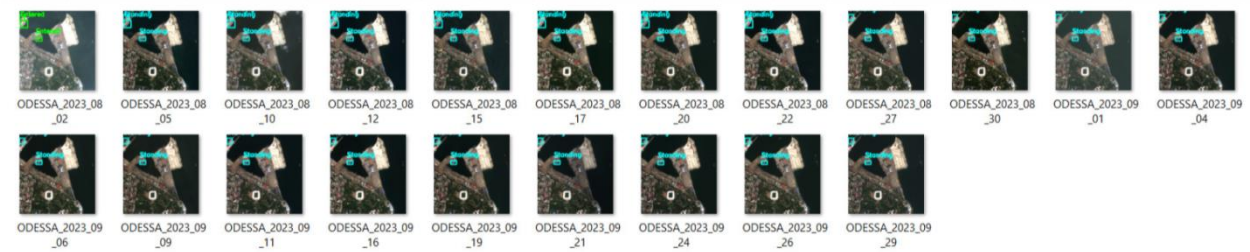


Рисунок 3.11 – Візуалізація статусу судна на зображенні

Код автоматично відслідковує рух суден в порту на основі зображень, отриманих за певні проміжки часу, і також надає корисну статистику для аналізу морського трафіку.

Розглянемо роботи коду на прикладі порту Одеси. Проаналізувавши супутникові знімки за два місяці з 2023-08-02 по 2023-09-29, за допомогою алгоритму отримуємо таку статистику (рис. 3.12).




```
Статистика по всіх знімках:
З 2023-08-02 по 2023-09-29:
Зайшло в порт: 3 суден
Вийшло з порту: 0 суден
Візуалізація завершена!
```

Рисунок 3.12 – Дослідження роботи порту Одеси

Як видно з результату роботи алгоритму, діяльність порту нульова. Це пов’язано з другим етапом російської блокади українських портів, який почався 18 липня 2023 року.

Висновок

У розділі наведено варіанти аналізу трафіку порту, перший заснований на даних AIS, другий полягає в аналізі супутникових знімків. Описано процес автоматизованого завантаження даних AIS з веб-сайту MarineTraffic, що дозволяє ефективно збирати та аналізувати інформацію про рух суден у портових зонах.

Запропонований алгоритм моніторингу трафіку порту на основі даних AIS включає в себе:

- автоматичне завантаження та обробку даних AIS;
- аналіз руху суден та визначення їхніх характеристик, таких як тип, дедвейт та осадка;
- визначення змін осадки для оцінки процесів завантаження та розвантаження суден.

Алгоритм для оцінки трафіку порту за супутниковими знімками базується на використанні нейронних мереж для детектування об'єктів. Цей підхід дозволяє:

- використовувати супутникові знімки для візуалізації та аналізу руху суден;
- визначати кількість суден, що заходять у порт та виходять з нього.

Запропоновані методики можуть бути основою для розробки алгоритму комплексної оцінки трафіку порту, комбінуючи дані з різних джерел для досягнення більшої точності та надійності.

					171.6321М.КР.ПЗ.Р4	
Змн.	*Арк.*	*№ документу*	*Підпис*	*Дата*		
Розробник		Вишневський В.В.			Охорона праці	*Літ.* / *Арк.* 75 / *Аркушів*
Консультант		Тубальцев А. М.				
Н. контр.		Добрінова Л. П.				НУК імені адмірала Макарова
Керівник		Дьяконов О.С.				
Зав. каф.		Рябенький В. М.				

РОЗДІЛ 4. ОХОРОНА ПРАЦІ

4.1. Вимоги до облаштування робочого місця

Раціональне облаштування робочого місця відіграє вирішальну роль у створенні комфортних, безпечних і продуктивних умов для роботи з комп’ютером. Правильна організація дозволяє мінімізувати шкідливий вплив на здоров’я працівників, зменшити ризик професійних захворювань і підвищити ефективність виконання завдань.

Ергономіка робочого місця: Правильна ергономіка робочого місця є важливою для забезпечення комфортних умов роботи, збереження здоров'я та підвищення продуктивності працівника. Основні вимоги до організації робочого простору:

- Робочий стіл: Столешниця робочого столу повинна бути достатньо просторою, щоб вмістити не тільки монітор, клавіатуру та мишу, але й інші необхідні предмети, такі як документи, канцелярія, телефон та інші аксесуари. Це дозволить зберігати порядок на робочому місці і забезпечити доступність усіх інструментів без зайвих рухів. Оптимальна висота столу складає 70–75 см, що дозволяє зручно працювати без перенапруження плечей і спини. Важливо, щоб поверхня стільниці була виконана з матового матеріалу, який не створює відблисків, оскільки яскраві відображення можуть спричиняти навантаження на очі та погіршувати видимість.
- Стілець: Для забезпечення належної підтримки тіла працівника стілець повинен мати регульовану висоту, підлокітники та підтримку для поперекового відділу хребта. Це дає можливість налаштувати стілець під індивідуальні параметри користувача, що сприяє підтримці правильної постави і запобігає розвитку болів у спині та шиї. Висота стільця повинна бути такою, щоб ноги працівника знаходилися в природному положенні — зігнуті під кутом 90 градусів у колінному суглобі, а ступні стояли рівно на підлозі або на спеціальній підставці.

Це знижує навантаження на суглоби та покращує циркуляцію крові, що важливо для здоров'я нижніх кінцівок. Підлокітники повинні підтримувати руки в природному положенні, щоб уникнути напруги в плечах та зап'ястках при роботі за комп'ютером.

Розташування обладнання на робочому місці: Правильне розташування комп'ютерного обладнання на робочому місці є важливим аспектом ергономіки, який допомагає забезпечити зручність і комфорт під час тривалої роботи. Ось основні рекомендації щодо розміщення обладнання:

- Відстань між очима та екраном: Для комфортної роботи екран повинен знаходитися на відстані 50–70 см від очей користувача. Це дозволяє зменшити навантаження на очі, запобігти їх перевтомі та розмитості зору. Занадто близьке розташування екрану може призвести до напруги очей, а занадто велике відстань — до необхідності постійно напружувати зір, що може викликати дискомфорт і головний біль.
- Положення екрану: Верхній край монітора повинен бути трохи нижче рівня очей, з кутом нахилу в 10–15 градусів. Це дозволяє підтримувати природне положення шиї та запобігає її перегину. Перегинання шиї при неправильно налаштованому моніторі може призвести до напруги в шийному відділі хребта, головного болю та дискомфорту. Таким чином, правильне розташування екрана знижує ризик розвитку таких проблем.
- Позиція монітора: Монітор слід розміщувати прямо перед собою, щоб зменшити необхідність повертати голову та шию. Постійне нахиляння або повороти голови можуть призвести до перенапруги м’язів шиї, що з часом викликає біль і дискомфорт. Розташування екрана прямо перед користувачем дозволяє зберегти природне положення шиї, знижуючи навантаження на неї.
- Розташування клавіатури: Клавіатура повинна бути розташована на рівні ліктів, що дозволяє тримати руки в природному положенні.

Зап’ястя повинні залишатися прямими під час друку, без перегинів. Якщо клавіатура занадто висока або низька, це може спричинити напругу в зап’ястях і ліктях, що з часом може призвести до болю та інших проблем. Оптимальна висота клавіатури дозволяє знижувати ризик розвитку таких захворювань, як синдром карпального каналу.

- Розташування миші: Миша повинна бути розміщена поруч із клавіатурою, так, щоб рука могла легко переміщатися між ними без зайвого навантаження. Важливо, щоб миша була на тому ж рівні, що й клавіатура, що дозволяє утримувати руки в природному положенні та запобігає перенапруженню зап’ясть і рук. Якщо миша знаходиться далеко від клавіатури, користувач буде змушений постійно розтягувати руку, що призведе до втоми та болю.

Освітлення робочого місця: Правильне освітлення робочого місця є важливим чинником для забезпечення комфортних умов праці, зменшення навантаження на очі та підтримки загального здоров'я. Невірно організоване освітлення може спричиняти дискомфорт, головний біль, втому очей та погіршення зору. Ось основні рекомендації для належного освітлення:

- Природне освітлення: Природне освітлення, яке надходить через вікна, є найкращим варіантом для робочого місця. Однак важливо, щоб вікно розташовувалося збоку від працівника, а не прямо перед ним чи позаду. Якщо працівник правша, вікно має бути розташоване зліва, а для шульги – справа. Це дозволяє уникнути прямого попадання сонячного світла на екран комп’ютера, що може викликати відблиски, погіршення контрасту та знижувати чіткість зображення. Відблиски на екрані можуть серйозно знизити ефективність роботи, тому правильне розташування вікна є важливим аспектом.
- Штучне освітлення: Штучне освітлення на робочому місці має бути рівномірним, щоб не виникало різких перепадів яскравості, які можуть спричиняти дискомфорт та напругу в очах. В ідеалі для освітлення

робочого місця варто використовувати настільні лампи з матовими плафонами, які сприяють рівномірному розподілу світла на робочій поверхні. Важливо, щоб світло не потрапляло безпосередньо на екран, оскільки це може створювати відблиски та знижувати видимість, а також негативно впливати на зір.

- Запобігання відблискам: Пряме потрапляння світла на монітор може значно ускладнити роботу, тому слід уникати його за будь-яких умов. Для цього можна використовувати різні засоби, такі як штори або жалюзі на вікнах, що дозволяють контролювати рівень природного освітлення. Крім того, спеціальні антивідблискові екрани або плівки можуть бути встановлені на монітор, щоб зменшити негативний вплив від сонячних променів або яскравого штучного освітлення.
- Якість освітлення: Освітлення робочого місця повинно бути достатньо яскравим, щоб працівник міг комфортно працювати, але не занадто інтенсивним, щоб не створювати додаткового навантаження на очі. Для цього необхідно вибирати лампи з правильними параметрами.

Вимоги до мікроклімату робочого місця Забезпечення комфортного мікроклімату в робочому приміщенні є важливим фактором, що безпосередньо впливає на самопочуття працівника. Оптимальні умови мікроклімату сприяють зниженню втоми, підвищенню концентрації та працездатності, що, в свою чергу, позитивно впливає на продуктивність праці. Ось основні вимоги до мікроклімату, які необхідно враховувати при організації робочого місця:

- Температурний режим: Оптимальна температура в робочому приміщенні для комфортної роботи за комп’ютером повинна знаходитися в межах від 18 до 22°C. Занадто висока температура може призвести до перегріву, зниження концентрації та швидкої втоми, тоді як занадто низька температура викликає неприємні відчуття холоду, що також знижує працездатність. Варто звертати увагу на те, що

температура може залежати від пори року та кількості людей у приміщенні, тому важливо підтримувати температурний баланс в межах цього діапазону для забезпечення комфортної роботи.

- Вологість повітря: Вологість повітря має бути в межах 40–60%, що є ідеальним показником для здоров'я та ефективної роботи. Надмірна сухість повітря може спричиняти подразнення дихальних шляхів, сухість шкіри та очей, а також зниження концентрації уваги. В свою чергу, занадто висока вологість може призвести до утворення плісняви та зниження якості повітря. Для підтримання оптимального рівня вологості в приміщенні можна використовувати зволожувачі повітря, особливо в зимовий період, коли опалення може значно знижувати рівень вологості в приміщеннях.
- Провітрювання приміщення: Регулярне провітрювання робочого приміщення є важливим для підтримки здорового мікроклімату. Свіже повітря сприяє покращенню загального самопочуття, підвищенню працездатності і концентрації. Крім того, постійний приток свіжого повітря допомагає знижувати концентрацію вуглекислого газу в кімнаті, що є критичним для нормального функціонування організму. Для цього слід забезпечити регулярне провітрювання, хоча б раз на годину, навіть якщо приміщення оснащене кондиціонерами чи іншими системами вентиляції. Провітрювання допомагає уникнути накопичення шкідливих газів і підтримує оптимальний рівень кисню.
- Системи вентиляції та кондиціювання: Важливим аспектом є наявність ефективної системи вентиляції або кондиціонування, яка дозволяє підтримувати необхідні умови мікроклімату протягом робочого дня. Системи вентиляції мають забезпечувати безперервний приплив свіжого повітря та відведення забрудненого, що сприяє зниженню ризику захворювань та зберігає комфорт в приміщенні. Проте варто уникати надмірної інтенсивності роботи кондиціонерів, оскільки це

може призвести до сухості повітря і створення дискомфорту для працівників.

4.2. Заходи для запобігання шкідливим факторам

Підвищений рівень статичного струму може стати причиною різноманітних проблем при роботі з комп’ютерним обладнанням, зокрема порушенням його нормальної роботи та навіть пошкодженням компонентів. Для мінімізації впливу статичної електрики та збереження безпеки як для користувача, так і для комп’ютерної техніки, слід дотримуватись низки спеціальних заходів, спрямованих на запобігання накопиченню і розряду статичного заряду:

- Заземлення обладнання: Одним із найбільш ефективних способів боротьби з підвищеним рівнем статичного струму є належне заземлення всіх компонентів комп’ютерної техніки, включаючи корпус, монітор, клавіатуру та периферійні пристрої (мишу, принтер тощо). Всі електричні пристрої повинні бути підключені до заземлених розеток. Заземлення допомагає уникнути накопичення статичного заряду, що знижує ризик виникнення коротких замикань або пошкоджень електронних компонентів. Крім того, для додаткового захисту обладнання від статичного струму можна використовувати спеціальні антистатичні мати, що забезпечують безпеку під час роботи з чутливими до електростатичних розрядів компонентами, такими як материнські плати чи пам'ять.
- Використання антистатичних пристроїв: Під час складання, ремонту або обслуговування комп’ютерного обладнання важливо використовувати антистатичні мати або браслети, які розсіюють накопичений статичний заряд і зменшують ризик його передачі на деталі. Антистатичні браслети зазвичай мають спеціальний провід, який підключається до заземленої поверхні, що дозволяє постійно

утримувати користувача на однаковому потенціалі з обладнанням і запобігає виникненню небажаних електростатичних розрядів.

- Зниження накопичення зарядів: Щоб зменшити ймовірність накопичення статичного заряду, слід обмежити рухи, які можуть призвести до тертя об пластикові або синтетичні поверхні, такі як килимки чи ковдри. Пластик та синтетичні матеріали є хорошими ізоляторами і здатні легко накопичувати статичний заряд, що підвищує ризик його розряду в найбільш несподіваний момент, що може пошкодити електроніку. Якщо можливо, слід використовувати поверхні, які не сприяють накопиченню статичних зарядів, або забезпечити додатковий захист за допомогою антистатичних матеріалів.
- Регулярне вологе прибирання робочого місця: Регулярне очищення робочого місця допомагає зменшити кількість пилу та статичних зарядів, що можуть накопичуватись на поверхнях. Для цього варто використовувати вологу тканину або спеціальні антистатичні засоби, які ефективно знижують рівень статичного заряду. Вологе прибирання також покращує загальний рівень чистоти і комфорту, а також допомагає підтримувати в хорошому стані комп’ютерну техніку.

Заходи для зниження фізичних навантажень під час роботи з ПК: Довга робота за комп'ютером може призвести до значних фізичних навантажень, що негативно впливає на здоров'я. Однак, дотримання простих правил та використання правильних методів може значно знизити ризики виникнення болю в спині, шиї, плечах і очах, а також покращити загальне самопочуття під час роботи.

- Ергономічне обладнання: Одним із найважливіших кроків у зниженні фізичних навантажень є використання ергономічних меблів. Це включає стіл і стілець, що мають відповідну підтримку для спини, що допомагає уникнути болю в хребті. Стілець повинен мати регульовану

висоту, можливість налаштування кута нахилу спинки і підлокітники, що забезпечують зручну позицію рук. Крім того, клавіатура та миша повинні бути регульованими, щоб їх можна було адаптувати до вашої постави, знижуючи навантаження на зап'ястя та лікті. Використання таких аксесуарів, як підставка для монітора, також дозволяє тримати екран на рівні очей, що запобігає перенапруженню шиї.

- Правильна постава: Підтримка правильної постави є ключовим фактором у запобіганні фізичних проблем. Ваше тіло повинно бути в природному положенні, що мінімізує навантаження на спину та суглоби. Тримайте спину прямою, не сутультеся. Плечі повинні бути розслабленими, а ноги розташовані таким чином, щоб стегна та коліна перебували на одному рівні, з кутом 90 градусів в колінах. Це дозволяє зменшити навантаження на поперек і зберегти комфорт під час тривалих періодів роботи.
- Регулярні паузи та розтяжки: Важливо регулярно робити перерви для розслаблення м'язів. Кожні 30–40 хвилин роботи варто вставати і рухатися, навіть якщо це лише легка прогулянка по кімнаті або прості розтягуючі вправи для шиї, плечей і кистей. Це допомагає зняти напругу, покращити кровообіг і уникнути затискань в м'язах. Просте підняття рук, повороти голови, нахили шиї допоможуть розслабити м'язи та повернути енергію.
- Вправи для очей: Тривала робота за комп'ютером може викликати значне навантаження на очі, що веде до їх втоми і сухості. Для запобігання цьому необхідно виконувати спеціальні вправи для зняття напруги з очей. Однією з таких вправ є фокусування погляду на віддалених об'єктах протягом кількох секунд, а потім повернення погляду до екрана. Такі перерви дозволяють вашим очам відпочити від постійного фокусування на близьких об'єктах.

- Фізична активність та регулярні вправи: Включення фізичної активності в щоденний графік також є важливим фактором, що сприяє зниженню м'язової напруги. Це може бути не лише спеціальні вправи, але й прогулянки на свіжому повітрі, йога або заняття фітнесом. Залишаючи час для фізичних вправ, ви покращуєте загальний стан здоров’я, стимулюєте кровообіг і запобігаєте розвитку болів у спині та шиї.

Заходи для запобігання перевтомі зорового апарату: Тривала робота за комп'ютером може значно навантажувати зорову систему, що призводить до втоми, сухості очей, порушення концентрації та зниження зорової гостроти. Для запобігання перевтомі зору важливо враховувати кілька аспектів організації робочого місця та режиму роботи.

- Оптимальна позиція та освітлення: Одним із основних факторів для збереження здоров'я очей є правильне розташування монітора. Важливо, щоб верхній край екрану був на рівні очей, а монітор знаходився на відстані 50–70 см від очей. Це дозволяє знижувати навантаження на зір і підтримувати правильну поставу. Крім того, важливо забезпечити достатнє освітлення робочого місця. Рекомендується використовувати поєднання природного та штучного освітлення, щоб уникнути відблисків і зайвого контрасту. Для цього можна використовувати лампи з м'яким, рівномірним світлом або спеціальні настільні лампи з матовими плафонами, що сприяють розсіюванню світла.
- Налаштування екрана: Для комфортної роботи з монітором важливо налаштувати екран таким чином, щоб зображення не викликало додаткового навантаження на очі. Це означає, що яскравість екрана має відповідати рівню освітлення в приміщенні, а контрастність та кольорова температура повинні бути підібрані так, щоб не дратувати

очі. В ідеалі, кольори на екрані повинні бути м'якими, не дуже яскравими, щоб зменшити негативний вплив на зорову систему

- Фільтри синього світла: Довготривале перебування перед екраном комп'ютера може призвести до перевантаження очей через вплив синього світла, що випромінюється монітором. Синє світло є шкідливим, оскільки сприяє порушенню сну, викликаючи зниження вироблення мелатоніну — гормону, що регулює циркадні ритми. Щоб знизити негативний вплив синього світла, можна використовувати спеціальні фільтри або програми, які зменшують випромінювання синього спектра, особливо в темний час доби. Такі фільтри допомагають знизити навантаження на очі та покращити якість сну.
- Режим роботи та відпочинок: Для підтримки здоров'я очей необхідно регулярно давати їм відпочинок. Один із способів — це метод 20-20-20: кожні 20 хвилин роботи з монітором слід дивитися на об'єкти, що знаходяться на відстані 20 футів (приблизно 6 метрів), протягом 20 секунд. Це дозволяє зняти напругу з очей і повернути їх у природний стан. Також важливо забезпечити собі регулярні перерви протягом дня, щоб зменшити навантаження на зорові м'язи. Крім того, для збереження зору важливий якісний сон. Відновлення після дня, проведеного за комп'ютером, дозволяє очам відпочити і відновити свої функції.

4.3. Аналіз шкідливих та небезпечних факторів на робочих місцях

Відповідно до наказу [18], робота з комп’ютером може створювати різноманітні ризики для здоров’я працівників. Шкідливі фактори залежать від тривалості використання комп’ютера, якості організації робочого місця та індивідуальних особливостей користувача. Розгляньмо основні загрози детальніше:

Статичне навантаження та неправильна постава: Тривала робота за комп’ютером часто пов’язана з необхідністю перебувати в одній і тій самій

позі протягом тривалого часу, що створює значне статичне навантаження на м’язи та опорно-руховий апарат. Основними проблемами, які виникають, є:

- Неправильна постава: Тривале сидіння за комп’ютером без належної організації робочого місця сприяє формуванню неправильної постави. Це може проявлятися сутулістю, нахилом голови вперед, округленням плечей і скручуванням хребта. Такі звички поступово можуть стати постійними, викликаючи дискомфорт і біль.
- М’язове напруження: Статичне навантаження призводить до надмірного напруження м’язів шиї, спини, плечового пояса та рук. Зокрема, тривала нерухомість сприяє розвитку м’язового спазму, що може викликати біль і погіршення рухливості.

Ці проблеми є основою для виникнення та розвитку серйозних хронічних захворювань:

- Остеохондроз шийного та грудного відділів хребта: Тривала неправильна постава під час роботи за комп’ютером посилює навантаження на міжхребцеві диски, що сприяє їхньому поступовому руйнуванню. Це може призводити до болю, обмеження рухливості та навіть компресії нервових закінчень.
- Синдром зап’ястного каналу: Тривала робота з клавіатурою та мишею створює надмірне навантаження на кисть, що може спричиняти набряк та здавлювання серединного нерва. Основними симптомами є оніміння пальців, слабкість у кисті та біль, який посилюється вночі.
- Порушення кровообігу в нижніх кінцівках: Тривале перебування у статичній позі, особливо з неправильною посадкою (наприклад, схрещеними ногами), призводить до здавлювання судин. Це може викликати набряки, варикозне розширення вен та навіть утворення тромбів у тяжких випадках.

Психоемоційне навантаження: Робота з комп’ютером вимагає постійної зосередженості, оперативного прийняття рішень і виконання

кількох завдань одночасно. Такий ритм діяльності значно впливає на психоемоційний стан працівника, створюючи низку проблем, серед яких:

- Хронічний стрес: Постійний високий рівень інтенсивності роботи, необхідність відповідати на численні запити, дотримуватися дедлайнів і швидко реагувати на змінювані умови часто спричиняють розвиток хронічного стресу. Це проявляється підвищеною тривожністю, дратівливістю, втомою та навіть фізичними симптомами, такими як головний біль або підвищений тиск.
- Емоційне виснаження: Виконання одноманітних завдань або рутини, яка не вимагає творчого підходу, поступово викликає втрату інтересу до роботи. Це може супроводжуватися відчуттям професійного вигорання, зниженням продуктивності та відсутністю мотивації до виконання навіть простих завдань.
- Порушення сну: Використання комп’ютера у вечірній час, особливо перед сном, негативно впливає на біологічні ритми організму. Блакитне світло екранів пригнічує вироблення мелатоніну — гормону, що відповідає за регуляцію сну. Це може спричиняти труднощі із засинанням, поверхневий сон та відчуття втоми навіть після тривалого відпочинку.
- Зниження когнітивних функцій: Постійна багатозадачність призводить до перевантаження мозку, що з часом може викликати труднощі із зосередженням, погіршення пам’яті та зниження здатності швидко обробляти інформацію.

Напруження органів зору: Тривале перебування за монітором є основною причиною зорового перенапруження, відомого як комп’ютерний зоровий синдром (КЗС). Його основні симптоми:

- Сухість, печіння та почервоніння очей.
- Погіршення гостроти зору.
- Головний біль, викликаний втомою очей.

Причини цих проявів часто пов’язані з недостатнім освітленням робочої зони, невідповідними налаштуваннями монітора (наприклад, низька роздільна здатність, мерехтіння екрана) та неправильною відстанню між очима та екраном.

Невідповідний мікроклімат робочого місця: Умови мікроклімату, такі як температура, вологість повітря та його якість, суттєво впливають на самопочуття й продуктивність працівників. Недотримання оптимальних параметрів створює дискомфорт і може стати причиною погіршення здоров’я. Основні проблеми, пов’язані з невідповідним мікрокліматом, включають:

Підвищена сухість повітря: Використання кондиціонерів або обігрівачів протягом тривалого часу знижує рівень вологості в приміщенні. Це може спричинити:

- Сухість слизових оболонок, що підвищує ризик захворювань верхніх дихальних шляхів.
- Подразнення шкіри, свербіж і почуття стягнутості.
- Зоровий дискомфорт, включаючи відчуття сухості очей, яке посилюється під час роботи за монітором.

Недостатня вентиляція: У приміщеннях із поганою системою вентиляції накопичується вуглекислий газ, що викликає:

- Відчуття задухи та загальну втому.
- Погіршення концентрації уваги та зниження продуктивності.
- Головний біль і запаморочення, особливо у щільно закритих офісах без доступу до свіжого повітря.

Температурні перепади: Різкі зміни температури, наприклад, під час переходу з нагрітого приміщення в холодний коридор або навпаки, можуть:

- Сприяти зниженню імунітету та підвищувати ризик простудних захворювань.

- Викликати дискомфорт і додатковий стрес для організму, оскільки він змушений адаптуватися до зміни умов.
- Погіршувати загальне самопочуття, знижуючи рівень енергії та ефективність роботи.

Шумовий вплив: Шум у робочому середовищі, навіть якщо він має низький рівень, може суттєво впливати на комфорт працівника та ефективність виконання завдань. Зокрема, постійний гул, створений роботою вентиляторів комп'ютера, кондиціонерів чи іншого офісного обладнання, є фактором, який часто недооцінюють. Основні негативні наслідки шуму:

- Дискомфорт і дратівливість: Навіть тихий, але постійний шум, наприклад, від кулерів чи вентиляційної системи, може викликати відчуття дискомфорту, що негативно впливає на загальне самопочуття.
- Зниження концентрації уваги: Монотонні звуки можуть відволікати, заважаючи фокусуватися на виконанні складних завдань. Це особливо актуально для роботи, що потребує високої уваги до деталей або творчого підходу.
- Накопичення втоми: Постійне шумове тло сприяє підвищенню стомлюваності, оскільки організм перебуває в стані подразнення через неможливість повного розслаблення.
- Ризик виникнення головного болю: Тривалий вплив шуму здатен провокувати головний біль, особливо у поєднанні з іншими негативними факторами, такими як недостатня вентиляція чи напруження очей.

4.4. Домедична допомога постраждалим від електричного струму

Робота за комп'ютером, хоча і здається безпечною, може нести серйозні ризики, зокрема для здоров'я, через можливість ураження електричним струмом. Це може статися з різних причин: від несправностей в електропроводці до неправильно підключеного обладнання або недостатньої

ізоляції проводів. Електротравми можуть бути різної тяжкості — від легких ушкоджень до критичних станів, які загрожують життю людини.

Згідно з Порядком надання домедичної допомоги постраждалим від електричного струму та блискавки, який був зареєстрований у Міністерстві юстиції України 7 липня 2014 року під номером 775/25552, електротравма визначається як ушкодження, спричинене впливом електричного струму або розрядом блискавки. У документі прописано необхідні дії, які повинні бути виконані для надання першої допомоги постраждалим від електричного струму.

Електричний струм може мати різний вплив на організм залежно від декількох факторів: сили струму, тривалості контакту, шляху проходження через тіло людини та інших умов. Травми можуть варіюватися від локальних ушкоджень, таких як опіки в місцях контакту з джерелом струму, до серйозних загальних порушень, які можуть включати:

- Місцеві пошкодження: часто це опіки, які виникають на шкірі в місцях контакту з джерелом струму.
- Загальні порушення організму: електричний струм може викликати серйозні порушення функцій внутрішніх органів, таких як серце чи легені. Одним з найнебезпечніших наслідків є зупинка серця або дихання, що може призвести до смерті без негайної допомоги.

Для того щоб уникнути небезпечних ситуацій, потрібно дотримуватися кількох основних правил безпеки під час роботи з електричними приладами:

- Використання обладнання з належним заземленням: це дуже важливо для запобігання коротким замиканням та інших небезпек. Переконайтеся, що ваше обладнання підключено до правильно заземлених розеток.
- Регулярна перевірка електропроводки: проводьте періодичні огляди проводки та електричних систем, щоб уникнути перегріву чи пошкоджень, які можуть призвести до небезпеки.

- Застосування сертифікованої техніки: використовуйте лише сертифіковані прилади та кабелі, які відповідають стандартам безпеки.

Дії під час надання домедичної допомоги постраждалим від електричного струму. Якщо ви стали свідком ураження людини електричним струмом, важливо діяти швидко, але при цьому обережно, аби не наражати себе на небезпеку. Дії, які потрібно здійснити в разі електротравми, можна умовно поділити на кілька етапів:

- Перевірка безпеки: Перше, що потрібно зробити — це переконатися, що джерело струму більше не становить загрози. Якщо електроприлад або проводка ще під напругою, необхідно ізолювати себе від джерела струму. Для цього можна використовувати сухі предмети, такі як дерев'яні палички чи пластикові трубки, щоб відштовхнути постраждалого від джерела електричного струму.
- Припинення дії струму: Найбільш безпечним варіантом є вимкнення електроживлення за допомогою рубильника або вимикача. Якщо вимкнути струм неможливо, використовуйте ізолюючі предмети, щоб відісунути джерело струму від потерпілого.
- Огляд постраждалого: Перевірте, чи є у постраждалого ознаки свідомості. Це можна зробити, звернувшись до нього або злегка потрясши за плечі.
- Оцініть його дихання: чи є рух грудної клітки, чи чути звуки дихання. Також перевірте пульс.

Виклик екстреної допомоги: Негайно зателефонуйте на екстрену лінію та повідомте оператора про місце події. Важливо точно вказати адресу та, по можливості, описати стан постраждалого.

ВИСНОВКИ

У ході виконання роботи розроблено комплексний підхід до моніторингу портового трафіку, що поєднує супутникові дані, автоматизовані методи їх обробки та інформацію з автоматичної ідентифікаційної системи (AIC).

Було проведено огляд безкоштовних джерел супутникових знімків і розроблено скрипт для їх автоматичного завантаження, що забезпечує зручність та швидкість доступу до актуальних даних дистанційного зондування Землі.

Навчено нейронну мережу YOLOv8 для задачі ідентифікації суден на супутникових знімках, що демонструє високу ефективність і точність розпізнавання об’єктів.

Описано процес автоматичного завантаження даних AIC із використанням парсингу веб-ресурсу MarineTraffic, що дозволяє інтегрувати ці дані у моніторинг судноплавства.

Запропоновано два програмні рішення для моніторингу портового трафіку: одне на основі аналізу даних AIC, інше – на основі аналізу супутникових знімків за допомогою нейронних мереж.

Результати роботи демонструють перспективність поєднання різних джерел даних і сучасних алгоритмів машинного навчання для задач спостереження за судноплавством. Запропоновані методи можуть бути адаптовані для інших галузей, пов’язаних із просторовим аналізом, зокрема екологічного моніторингу, безпеки морських перевезень і планування транспортної логістики.

Таким чином, реалізовані підходи створюють основу для побудови ефективних систем моніторингу, здатних забезпечити гнучкість і точність у вирішенні актуальних задач, пов’язаних із морським транспортом

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Types of Satellite Imagery [Електронний ресурс] Режим доступу: https://satpalda.com/blogs/types-of-satellite-imagery/
2. EOS DATA ANALYTICS [Електронний ресурс] Режим доступу: https://eos.com/blog/free-satellite-imagery-sources/
3. Copernicus Data Space Ecosystem [Електронний ресурс] Режим доступу: https://dataspace.copernicus.eu/
4. Sentinel-2 [Електронний ресурс] Режим доступу: https://dataspace.copernicus.eu/explore-data/data-collections/sentinel-data/sentinel-2
5. Sentinel-1 [Електронний ресурс] Режим доступу: https://dataspace.copernicus.eu/explore-data/data-collections/sentinel-data/sentinel-1
6. MatplotLib [Електронний ресурс] Режим доступу: https://matplotlib.org/
7. NumPy [Електронний ресурс] Режим доступу: https://numpy.org/
8. sentinelHUB [Електронний ресурс] Режим доступу: https://www.sentinel-hub.com/
9. Faster R-CNN vs YOLO vs SSD — Object Detection Algorithms [Електронний ресурс] Режим доступу: https://medium.com/ibm-data-ai/faster-r-cnn-vs-yolo-vs-ssd-object-detection-algorithms-18badb0e02dc
10. DigitalOcean [Електронний ресурс] Режим доступу: https://www.digitalocean.com/community/tutorials/faster-r-cnn-explained-object-detection
11. ArcGIS API for Python [Електронний ресурс] Режим доступу: https://developers.arcgis.com/python/latest/guide/how-ssd-works/
12. Ultralytics [Електронний ресурс] Режим доступу: https://www.ultralytics.com/ru/yolo
13. YOLOv8 [Електронний ресурс] Режим доступу: https://blog.roboflow.com/what-is-yolov8/

14. VisoAI [Електронний ресурс] Режим доступу: https://viso.ai/computer-vision/image-annotation/
15. RoboFlow [Електронний ресурс] Режим доступу: https://roboflow.com/
16. MarineTraffic [Електронний ресурс] Режим доступу: https://www.marinetraffic.com/en/ais/home/centerx:70.9/centery:42.3/zoom:10
17. beautifulsoup4 [Електронний ресурс] Режим доступу: https://pypi.org/project/beautifulsoup4/
18. Міністерство соціальної політики України. (2018). Наказ № 207 від 14.02.2018 р. Про затвердження Вимог щодо безпеки та захисту здоров'я працівників під час роботи з екранними пристроями.

ДОДАТОК А

Код для завантаження супутникових знімків оптичного діапазону

```
import matplotlib.pyplot as plt
import numpy as np
from sentinelhub import (
    SHConfig,
    DataCollection,
    SentinelHubCatalog,
    SentinelHubRequest,
    BBox,
    bbox_to_dimensions,
    CRS,
    MimeType,
)
from datetime import datetime

def plot_image(image, factor=1, clip_range=None, title="Image"):

    image = image * factor
    if clip_range:
        image = np.clip(image, clip_range[0], clip_range[1])

    plt.figure(figsize=(10, 10))
    plt.imshow(image)
    plt.title(title)
    plt.axis('off')
    plt.show()


def get_square_bbox(lat, lon, size_km):
    km_per_deg_lat = 111
    km_per_deg_lon = 85

    lat_shift = size_km / (2 * km_per_deg_lat)
    lon_shift = size_km / (2 * km_per_deg_lon)

    return [lon - lon_shift, lat - lat_shift, lon + lon_shift, lat +
lat_shift]

config = SHConfig('cdse')
print(config)

lat, lon = 44.16213907418022,28.654980018196966
square_size_km = 2
aoi_coords_wgs84 = get_square_bbox(lat, lon, square_size_km)
resolution = 10
aoi_bbox = BBox(bbox=aoi_coords_wgs84, crs=CRS.WGS84)
aoi_size = bbox_to_dimensions(aoi_bbox, resolution=resolution)

print(f"Image shape at {resolution} m resolution: {aoi_size} pixels")

catalog = SentinelHubCatalog(config=config)

start_date = datetime(2024, 11, 1)
end_date = datetime(2024, 11, 30)
time_interval = (start_date.strftime('%Y-%m-%d'), end_date.strftime('%Y-%m-
%d'))

search_iterator = catalog.search(
```

```
    DataCollection.SENTINEL2_L2A,
    bbox=aoi_bbox,
    time=time_interval,
    fields={"include": ["id", "properties.datetime"], "exclude": []},
)

results = list(search_iterator)
print("Total number of results:", len(results))

evalscript_true_color = """
    //VERSION=3

    function setup() {
        return {
            input: ["B02", "B03", "B04"],
            output: { bands: 3 }
        };
    }

    function evaluatePixel(sample) {
        return [sample.B04, sample.B03, sample.B02];
    }
"""

for idx, result in enumerate(results):
    capture_date = result['properties']['datetime']
    time_interval_single = (capture_date, capture_date)
    print(f"Завантаження знімка за {capture_date}...")

    request_true_color = SentinelHubRequest(
        evalscript=evalscript_true_color,
        input_data=[
            SentinelHubRequest.input_data(
                data_collection=DataCollection.SENTINEL2_L2A.define_from(
                    name="s2l2a",
service_url="https://sh.dataspace.copernicus.eu"
                ),
                time_interval=time_interval_single,
                other_args={"dataFilter": {"mosaickingOrder": "leastCC"}},
            )
        ],
        responses=[SentinelHubRequest.output_response("default",
MimeType.PNG)],
        bbox=aoi_bbox,
        size=aoi_size,
        config=config,
    )

    true_color_imgs = request_true_color.get_data()
    image = true_color_imgs[0]

    image_brightened = image * 3.5
    image_brightened = np.clip(image_brightened, 0, 255).astype(np.uint8)

    image_filename = f"Constanta_{capture_date[:10].replace('-', '_')}.png"
    plt.imsave(image_filename, image_brightened)
    print(f"Збережено знімок: {image_filename}")

print("Завантаження всіх знімків завершено.")
```

ДОДАТОК Б

Код для завантаження SAR-знімків

```
import matplotlib.pyplot as plt
import numpy as np
from sentinelhub import (
    SHConfig,
    DataCollection,
    SentinelHubCatalog,
    SentinelHubRequest,
    BBox,
    bbox_to_dimensions,
    CRS,
    MimeType,
)
from datetime import datetime

def plot_image(image, factor=1, clip_range=None, title="Image"):
    image = image * factor
    if clip_range:
        image = np.clip(image, clip_range[0], clip_range[1])

    plt.figure(figsize=(10, 10))
    plt.imshow(image, cmap='magma')
    plt.title(title)
    plt.axis('off')
    plt.show()

def save_image(image, filename, clip_range=(-20, 0)):
    image = np.clip((image - clip_range[0]) / (clip_range[1] -
clip_range[0]), 0, 1)
    plt.imsave(filename, image, cmap='magma')

config = SHConfig('cdse')
if config.sh_client_id == '' or config.sh_client_secret == '':
    print("Потрібно налаштувати Sentinel Hub credentials.")

lat, lon = 46.48507,30.75807
square_size_km = 2
aoi_coords_wgs84 = [lon - square_size_km / 170, lat - square_size_km / 222,
                    lon + square_size_km / 170, lat + square_size_km / 222]
resolution = 10
aoi_bbox = BBox(bbox=aoi_coords_wgs84, crs=CRS.WGS84)
aoi_size = bbox_to_dimensions(aoi_bbox, resolution=resolution)

print(f"Image shape at {resolution} m resolution: {aoi_size} pixels")

catalog = SentinelHubCatalog(config=config)

start_date = datetime(2024, 9, 1)
end_date = datetime(2024, 10, 1)
time_interval = (start_date.strftime('%Y-%m-%d'), end_date.strftime('%Y-%m-
%d'))

search_iterator = catalog.search(
    DataCollection.SENTINEL1_IW,
    bbox=aoi_bbox,
    time=time_interval,
    fields={"include": ["id", "properties.datetime"], "exclude": []},
)
```

```
results = list(search_iterator)
print("Total number of SAR results:", len(results))

evalscript_sar = """
    function setup() {
        return {
            input: ["VV", "VH", "dataMask"],
            output: { bands: 3, sampleType: "FLOAT32" }
        };
    }

    function toDb(linear) {
        return 10 * Math.log10(linear);
    }

    function evaluatePixel(sample) {
        var vv_db = toDb(sample.VV);
        var vh_db = toDb(sample.VH);
        var mask = sample.dataMask;
        return [vv_db * mask, vh_db * mask, mask];
    }
"""

for idx, result in enumerate(results):
    capture_date = result['properties']['datetime']
    time_interval_single = (capture_date, capture_date)

    print(f"Завантаження SAR знімка за {capture_date}...")

    request_sar = SentinelHubRequest(
        evalscript=evalscript_sar,
        input_data=[
            SentinelHubRequest.input_data(

data_collection=DataCollection.SENTINEL1_IW.define_from("s1iw",
service_url=config.sh_base_url),
                time_interval=time_interval_single,
                other_args={
                    "dataFilter": {
                        "resolution": "HIGH",
                        "mosaickingOrder": "mostRecent",
                        "orbitDirection": "ASCENDING",
                    },
                    "processing": {
                        "backCoeff": "SIGMA0_ELLIPSOID",
                        "orthorectify": True,
                        "demInstance": "COPERNICUS",
                        "speckleFilter": {
                            "type": "LEE",
                            "windowSizeX": 3,
                            "windowSizeY": 3,
                        },
                    },
                },
            )
        ],
        responses=[SentinelHubRequest.output_response("default",
MimeType.TIFF)],
        bbox=aoi_bbox,
        size=aoi_size,
        config=config,
    )
```

```
        sar_imgs = request_sar.get_data()
        sar_image_vv = sar_imgs[0][:, :, 0]
        sar_image_vh = sar_imgs[0][:, :, 1]
        data_mask = sar_imgs[0][:, :, 2]

        combined_sar_image = (sar_image_vv + sar_image_vh) / 2 * data_mask

        image_filename = f"ODESSA_{capture_date[:10].replace('-', '_')}.png"
        save_image(combined_sar_image, image_filename)
        print(f"Збережено знімок: {image_filename}")

print("Завантаження всіх SAR знімків завершено.")
```

ДОДАТОК В

Код для навчання нейромережі

```
!nvidia-smi

!pip install ultralytics

from ultralytics import YOLO
import os
from IPython.display import display, Image
from IPython import display
display.clear_output()
!yolo mode=checks

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

import os
os.environ["WANDB_MODE"] = "disabled"

!yolo task=detect mode=train model=yolov8m.pt
data=/content/drive/MyDrive/Model2/vessel2.yaml epochs=100 imgsz=640 batch
= 8

Image(filename=f'/content/runs/detect/train/confusion_matrix.png',
width=600)

Image(filename=f'/content/runs/detect/train/results.png', width=600)

!yolo task=detect mode=val
model=/content/runs/detect/train/weights/best.pt data
=/content/drive/MyDrive/Model2/vessel2.yaml

!yolo task=detect mode=predict
model=/content/runs/detect/train/weights/best.pt conf=0.5
source=/content/drive/MyDrive/Model2/images/val

import glob
from IPython.display import Image, display

for image_path in glob.glob(f'/content/runs/detect/predict/*.jpg'):
  display(Image(filename=image_path, height=600))
  print("\n")
```

ДОДАТОК Г

Парсер даних про судна

```
import argparse
import time
import pytz
from datetime import datetime
import os
import re
import codecs
from selenium import webdriver
from selenium.webdriver.chrome.service import Service
import  json

parser = argparse.ArgumentParser()
parser.add_argument('--path',required=True, help='path to work directory')

parser.print_help()
args = parser.parse_args()
print(args)

current_time_str = datetime.now(tz=pytz.UTC).strftime("%d_%m_%Y_%H_00")
time_str = datetime.now(tz=pytz.UTC).strftime("%H_00")
sub_dir_name_str = datetime.now(tz=pytz.UTC).strftime("%d_%m_%Y")

max_attempts = 5
current_attempt = 0
failed_flag = True

options = webdriver.ChromeOptions()
options.add_argument("--disable-blink-features=AutomationControlled")

s = Service()

while (current_attempt<max_attempts and failed_flag == True):
    failed_flag = False
    current_attempt=current_attempt+1
    driver = webdriver.Chrome(service=s, options=options)
    driver.execute_cdp_cmd('Page.addScriptToEvaluateOnNewDocument', {
        'source': '''
            delete window.cdc_adoQpoasnfa76pfcZLmcfl_Array;
            delete window.cdc_adoQpoasnfa76pfcZLmcfl_Promise;
            delete window.cdc_adoQpoasnfa76pfcZLmcfl_Symbol;
            delete window.cdc_adoQpoasnfa76pfcZLmcfl_JSON;
            delete window.cdc_adoQpoasnfa76pfcZLmcfl_Proxy;
            delete window.cdc_adoQpoasnfa76pfcZLmcfl_Object;
            '''
    })

    p= driver.window_handles[0]


main_url="https://www.marinetraffic.com/en/ais/home/centerx:34.3/centery:44.1
/zoom:6"

map_urls=["https://www.marinetraffic.com/getData/get_data_json_4/z:6/X:18/Y:1
1/station:0","https://www.marinetraffic.com/getData/get_data_json_4/z:6/X:18/
Y:12/station:0","https://www.marinetraffic.com/getData/get_data_json_4/z:6/X:
19/Y:11/station:0","https://www.marinetraffic.com/getData/get_data_json_4/z:6
/X:19/Y:12/station:0"]
```

```
    try:
        os.mkdir(args.path)
    except OSError as error:
        print(error)

    try:
        os.mkdir(args.path+"\\"+sub_dir_name_str)
    except OSError as error:
        print(error)

    try:
        os.mkdir(args.path+"\\"+sub_dir_name_str+"\\"+"map")
    except OSError as error:
        print(error)

    driver.maximize_window()
    driver.get(main_url)
    time.sleep(5)

    i=1
    for url in map_urls:
        try:
            os.mkdir(args.path + "\\" + sub_dir_name_str+ "\\" + "map" + "\\"
+ time_str)
        except OSError as error:
            print(error)
        print("Scraping data of MT map")
        driver.execute_script("window.open('');")
        driver.switch_to.window(driver.window_handles[1])
        driver.get(url)
        time.sleep(2)
        n=os.path.join(args.path+"\\"+sub_dir_name_str+"\\"+ "map" + "\\"
+time_str,"map_" +str(i)+ ".json")
        with codecs.open(n, "w", "utf?8") as f:
            h = driver.page_source
            result = re.sub(r'<.*?>', '', h)
            result = (json.dumps(json.loads(result), indent=4))
            f.write(result)
            if (result.find('Access Denied') != -1):
                failed_flag=True
        time.sleep(1)
        driver.close()
        driver.switch_to.window(p)
        i=i+1

    time.sleep(2)
    driver.quit()
```

ДОДАТОК Ґ

Код, що знаходить судна в межах знімку

```
import os
import json
from datetime import datetime

base_path = r'G:\Вірші\Книги\Tropo\data'
#base_path = r'G:\UniverNUK\1327_Python\NewPycharm\ODESSA'
lat, lon = 46.48507,30.75807
square_size_km = 2

def get_square_bbox(lat, lon, size_km):
    km_per_deg_lat = 111
    km_per_deg_lon = 85
    lat_shift = size_km / (2 * km_per_deg_lat)
    lon_shift = size_km / (2 * km_per_deg_lon)
    return [lon - lon_shift, lat - lat_shift, lon + lon_shift, lat +
lat_shift]

aoi_coords_wgs84 = get_square_bbox(lat, lon, square_size_km)
bounding_box = {
    'lon_min': aoi_coords_wgs84[0],
    'lat_min': aoi_coords_wgs84[1],
    'lon_max': aoi_coords_wgs84[2],
    'lat_max': aoi_coords_wgs84[3]
}
def is_within_bounding_box(lat, lon, bounding_box):
    return (bounding_box["lat_min"] <= lat <= bounding_box["lat_max"] and
            bounding_box["lon_min"] <= lon <= bounding_box["lon_max"])

def find_ships_in_files(base_path, bounding_box):
    ships_in_area = []
    for date_folder in os.listdir(base_path):
        date_folder_path = os.path.join(base_path, date_folder)
        if os.path.isdir(date_folder_path):
            try:
                date_obj = datetime.strptime(date_folder, '%d_%m_%Y')
            except ValueError:
                continue
            map_folder_path = os.path.join(date_folder_path, 'map')
            if os.path.exists(map_folder_path):
                time_folder = None
                if "12_00" in os.listdir(map_folder_path):
                    time_folder = "12_00"
                elif "13_00" in os.listdir(map_folder_path):
                    time_folder = "13_00"

                if time_folder:
                    time_folder_path = os.path.join(map_folder_path,
time_folder)
                    time_str = time_folder.replace('_', ':')

                    for map_file in os.listdir(time_folder_path):
                        if map_file.startswith('map_') and
map_file.endswith('.json'):
                            map_file_path = os.path.join(time_folder_path,
map_file)
                            try:
                                if os.path.getsize(map_file_path) > 0:
```

```
                                    with open(map_file_path, 'r',
encoding='utf-8') as f:
                                        data = json.load(f)

                                    for ship in data['data']['rows']:
                                        lat = float(ship['LAT'])
                                        lon = float(ship['LON'])

                                        if is_within_bounding_box(lat, lon,
bounding_box):
                                            ship['date'] =
date_obj.strftime('%Y-%m-%d')
                                            ship['time'] = time_str
                                            ships_in_area.append(ship)
                                else:
                                    print(f"Skip empty file:
{map_file_path}")
                            except json.JSONDecodeError:
                                print(f"Error reading JSON in file:
{map_file_path}")
                            except Exception as e:
                                print(f"Enother error with the file
{map_file_path}: {e}")

    return ships_in_area

ships_in_area = find_ships_in_files(base_path, bounding_box)

output_file = 'ships_in_areaNEWgetFullInfo.json'
with open(output_file, 'w', encoding='utf-8') as f:
    json.dump(ships_in_area, f, indent=4, ensure_ascii=False)

print(f"Found {len(ships_in_area)} vessel. Data saved file {output_file}.")
```

ДОДАТОК Д

Код для автоматичної анотації суден

```
import os
import matplotlib.pyplot as plt
import numpy as np
from PIL import Image
import json

image_folder =
r"G:\UniverNUK\1327_Python\NewPycharm\YUZHNY\YUZHNY_RESIZE_FOTO"
output_folder =
r"G:\UniverNUK\1327_Python\NewPycharm\YUZHNY\YUZHNY_NEW_Annotation"
annotations_folder = os.path.join(output_folder, "annotations")
os.makedirs(output_folder, exist_ok=True)
os.makedirs(annotations_folder, exist_ok=True)

json_file_path = r"G:\UniverNUK\1327_Python\NewPycharm\ships_in_areaNEW.json"

with open(json_file_path, 'r', encoding='utf-8') as f:
    ships_data = json.load(f)
)
def get_square_bbox(lat, lon, size_km):
    km_per_deg_lat = 111
    km_per_deg_lon = 85

    lat_shift = size_km / (2 * km_per_deg_lat)
    lon_shift = size_km / (2 * km_per_deg_lon)

    return [lon - lon_shift, lat - lat_shift, lon + lon_shift, lat +
lat_shift]

def is_within_bounding_box(lat, lon, bounding_box):
    return (bounding_box["lat_min"] <= lat <= bounding_box["lat_max"] and
            bounding_box["lon_min"] <= lon <= bounding_box["lon_max"])

def geo_to_pixel(lon, lat, bbox, img_width, img_height):
    x_frac = (lon - bbox['lon_min']) / (bbox['lon_max'] - bbox['lon_min'])
    y_frac = (lat - bbox['lat_min']) / (bbox['lat_max'] - bbox['lat_min'])

    x_pixel = int(x_frac * img_width)
    y_pixel = int((1 - y_frac) * img_height)
    return x_pixel, y_pixel

def create_yolo_annotation(filename, detections, img_width, img_height):
    annotation_path = os.path.join(annotations_folder, f"{filename}.txt")
    with open(annotation_path, "w") as f:
        for (x_center, y_center, width, height) in detections:
            x_center /= img_width
            y_center /= img_height
            width /= img_width
            height /= img_height
            f.write(f"0 {x_center} {y_center} {width} {height}\n")

lat, lon = 46.6208,31.0252
square_size_km = 2
aoi_coords_wgs84 = get_square_bbox(lat, lon, square_size_km)

bounding_box = {
    'lon_min': aoi_coords_wgs84[0],
```

```
    'lat_min': aoi_coords_wgs84[1],
    'lon_max': aoi_coords_wgs84[2],
    'lat_max': aoi_coords_wgs84[3]
}

for ship in ships_data:
    ship_date = ship["date"].replace("-", "_")
    image_name = f"YUZHNY_{ship_date}.png"
    image_path = os.path.join(image_folder, image_name)

    if os.path.exists(image_path):
        print(f"Знайдено зображення для дати {ship['date']}: {image_name}")

        image = Image.open(image_path)
        img_width, img_height = image.size

        detections = []
        for ship_info in ships_data:
            if ship_info["date"] == ship["date"]:
                lat_ship = ship_info['lat']
                lon_ship = ship_info['lon']
                if is_within_bounding_box(lat_ship, lon_ship, bounding_box):
                    x_center, y_center = geo_to_pixel(lon_ship, lat_ship,
bounding_box, img_width, img_height)
                    width, height = 70, 70
                    detections.append((x_center, y_center, width, height))

        if detections:
            create_yolo_annotation(image_name.replace('.png',''), detections,
img_width, img_height)
            print(f"Збережено анотації для зображення {image_name}")
    else:
        print(f"Зображення для дати {ship['date']} не знайдено.")
```

ДОДАТОК Е

Парсер даних про конкретне судно

```
from selenium import webdriver
from selenium.webdriver.chrome.service import Service
import time
from bs4 import BeautifulSoup
import json

input_json =
r'G:\UniverNUK\1327_Python\NewPycharm\ships_in_areaNEWgetFullInfo.json'
output_json = 'ships_data_output2.json'

options = webdriver.ChromeOptions()
options.add_argument("--disable-blink-features=AutomationControlled")

s = Service()
driver = webdriver.Chrome(service=s, options=options)

driver.execute_cdp_cmd('Page.addScriptToEvaluateOnNewDocument', {
    'source': '''
        delete window.cdc_adoQpoasnfa76pfcZLmcfl_Array;
        delete window.cdc_adoQpoasnfa76pfcZLmcfl_Promise;
        delete window.cdc_adoQpoasnfa76pfcZLmcfl_Symbol;
        delete window.cdc_adoQpoasnfa76pfcZLmcfl_JSON;
        delete window.cdc_adoQpoasnfa76pfcZLmcfl_Proxy;
        delete window.cdc_adoQpoasnfa76pfcZLmcfl_Object;
        '''
})


with open(input_json, 'r', encoding='utf-8') as f:
    ship_list = json.load(f)
collected_data = []
try:
    for ship in ship_list:
        ship_name = ship.get('SHIPNAME')
        ship_id = ship.get('SHIP_ID')

        if not ship_name or not ship_id:
            print(f"Пропускаємо судно з неповними даними: {ship}")
            continue
        url =
f'https://www.marinetraffic.com/en/ais/details/ships/shipid:{ship_id}/vessel:
{ship_name}'
        print(f"Парсимо дані для судна: {ship_name} ({url})")
        driver.get(url)
        time.sleep(20)
        html_source = driver.page_source
        soup = BeautifulSoup(html_source, 'html.parser')
        rows = soup.find_all('tr')
        ship_data = {"SHIPNAME": ship_name, "SHIP_ID": ship_id}
        for row in rows:
            th_tag = row.find('th')
            td_tag = row.find('td')
            if th_tag and td_tag:
                key = th_tag.get_text(strip=True)
                value = td_tag.get_text(strip=True)
                ship_data[key] = value
```

```
            collected_data.append(ship_data)

    except Exception as ex:
        print(f"Помилка: {ex}")
    finally:
        driver.close()
        driver.quit()

    with open(output_json, 'w', encoding='utf-8') as f:
        json.dump(collected_data, f, ensure_ascii=False, indent=4)

    print(f"Збір даних завершено та збережено в {output_json}")
```

ДОДАТОК Є

Аналіз роботи порту за даними AIS

```
import json
from datetime import datetime

with
open(r'G:\UniverNUK\1327_Python\NewPycharm\CONSTANTA\CONSTANTA_Json\FullInfo.
json', 'r') as f1,
open(r'G:\UniverNUK\1327_Python\NewPycharm\CONSTANTA\CONSTANTA_Json\Ship_in_A
rea_Full.json', 'r') as f2:
    file1_data = json.load(f1)
    file2_data = json.load(f2)

DETAILED_VESSEL_TYPE_TO_CARGO = {
    "Bulk Carrier": "Сипучий вантаж",
    "Container Ship": "Контейнери",
    "Oil Tanker": "Нафтопродукти",
    "Gas Carrier": "Газ",
    "Chemical Tanker": "Хімічні речовини",
    "General Cargo Ship": "Загальний вантаж",
    "LPG Tanker": "Скраплений газ",
    "Other": "Інший",
    "Crude Oil Tanker": "Нафтопродукти"
}

def parse_datetime(date_str, time_str):
    return datetime.strptime(f"{date_str} {time_str}", "%Y-%m-%d %H:%M")

def analyze_ship_activity(ship_id, file1, file2):

    file1_records = [record for record in file1 if record['SHIP_ID'] ==
ship_id]
    file2_records = [record for record in file2 if record['SHIP_ID'] ==
ship_id]
    if not file1_records or not file2_records:
        return None

    detailed_type = file1_records[0].get('Detailed vessel type', 'Other')
    cargo_type = DETAILED_VESSEL_TYPE_TO_CARGO.get(detailed_type,
"Невідомий")

    cargo_mass = file1_records[0].get('DWT') or file2_records[0].get('DWT')

    initial_draught = float(file1_records[0]['Draught'].replace(' m', ''))
    final_draught = float(file1_records[-1]['Draught'].replace(' m', ''))
    draught_change = final_draught - initial_draught

    if draught_change > 0:
        loading_status = "Loading"
    elif draught_change < 0:
        loading_status = "Unloading"
    else:
        loading_status = "No significant change"

    records_by_date = {}
    for record in file2_records:
        date = record['date']
        if date not in records_by_date:
            records_by_date[date] = []
```

```
            records_by_date[date].append(record)

        sorted_dates = sorted(records_by_date.keys())
        entered_port = parse_datetime(sorted_dates[0],
records_by_date[sorted_dates[0]][0]['time'])
        left_port = None

        all_records_by_date = {}
        for record in file2_data:
            date = record['date']
            if date not in all_records_by_date:
                all_records_by_date[date] = []
            all_records_by_date[date].append(record)

        for i in range(len(sorted_dates) - 1):
            current_date = sorted_dates[i]
            next_date = sorted_dates[i + 1]

            if ship_id not in [record['SHIP_ID'] for record in
all_records_by_date[next_date]]:
                left_port = parse_datetime(current_date,
records_by_date[current_date][-1]['time'])
                break

        last_date = sorted_dates[-1]
        next_date_after_last = sorted([d for d in all_records_by_date.keys() if d
> last_date])

        if not any(ship_id in [record['SHIP_ID'] for record in
all_records_by_date.get(date, [])] for date in next_date_after_last):
            left_port = parse_datetime(last_date, records_by_date[last_date][-
1]['time'])

        return {
            'ship_name': file1_records[0]['SHIPNAME'],
            'cargo_type': cargo_type,
            'cargo_mass': cargo_mass,
            'entered_port': entered_port,
            'left_port': left_port,
            'detailed_type': detailed_type,
            'initial_draught': initial_draught,
            'final_draught': final_draught,
            'draught_change': draught_change,
            'loading_status': loading_status
        }

    ship_ids = set(record['SHIP_ID'] for record in file1_data)
    results = []
    ship_type_summary = {}

    for ship_id in ship_ids:
        result = analyze_ship_activity(ship_id, file1_data, file2_data)
        if result:
            results.append(result)
            ship_type = result['detailed_type']
            if ship_type not in ship_type_summary:
                ship_type_summary[ship_type] = {
                    'count': 0,
                    'first_date': result['entered_port'],
                    'last_date': result['entered_port']
                }
            ship_type_summary[ship_type]['count'] += 1
            if result['entered_port'] <
```

```
ship_type_summary[ship_type]['first_date']:
            ship_type_summary[ship_type]['first_date'] =
result['entered_port']
        if result['entered_port'] >
ship_type_summary[ship_type]['last_date']:
            ship_type_summary[ship_type]['last_date'] =
result['entered_port']

total_ships = len(ship_ids)

for res in results:
    print(f"Судно: {res['ship_name']}")
    print(f"Тип вантажу: {res['cargo_type']}")
    print(f"Маса вантажу: {res['cargo_mass']} тонн")
    print(f"Час входу в порт: {res['entered_port']}")
    print(f"Час виходу з порту: {res['left_port'] if res['left_port'] else
'Судно ще в порту'}")
    print(f"Початкова осадка: {res['initial_draught']} м")
    print(f"Кінцева осадка: {res['final_draught']} м")
    print(f"Зміна осадки: {res['draught_change']} м")
    print(f"Статус: {res['loading_status']}")
    print("-" * 40)

print("Статистика за типами суден:")
for ship_type, summary in ship_type_summary.items():
    print(f"Тип судна: {ship_type}")
    print(f"Кількість суден: {summary['count']}")
    print(f"Перша дата: {summary['first_date']}")
    print(f"Остання дата: {summary['last_date']}")
    print("-" * 40)

print(f"Загальна кількість суден: {total_ships}")
```

ДОДАТОК Ж

Аналіз роботи порту за супутниковими даними

```
import os
import cv2
from ultralytics import YOLO
import numpy as np
from datetime import datetime

model = YOLO("best.pt")

image_folder = r"G:\UniverNUK\1327_Python\NewPycharm\ODESSA\Odesa_8"
output_folder = r"G:\UniverNUK\1327_Python\NewPycharm\ODESSA\ress"
os.makedirs(output_folder, exist_ok=True)

previous_ships = []

total_entering = 0
total_exiting = 0
first_date = None
last_date = None

def compare_ships(current_ships, previous_ships, threshold=30):
    entering_ships = []
    exiting_ships = []
    standing_ships = []

    current_coordinates = [ship[:2] for ship in current_ships]
    previous_coordinates = [ship[:2] for ship in previous_ships]

    for current in current_ships:
        found_match = False
        for prev in previous_ships:
            if np.linalg.norm(np.array(current[:2]) - np.array(prev[:2])) <
threshold:
                standing_ships.append(current)
                found_match = True
                break
        if not found_match:
            entering_ships.append(current)

    for prev in previous_ships:
        found_match = False
        for current in current_ships:
            if np.linalg.norm(np.array(prev[:2]) - np.array(current[:2])) <
threshold:
                found_match = True
                break
        if not found_match:
            exiting_ships.append(prev)

    return entering_ships, exiting_ships, standing_ships

for image_name in os.listdir(image_folder):
    image_path = os.path.join(image_folder, image_name)

    image = cv2.imread(image_path)

    results = model.predict(source=image_path, conf=0.5)  # Мінімальна
впевненість 50%
```

```
    detections = results[0].boxes.xyxy  # Координати рамок
    classes = results[0].boxes.cls  # Класи об'єктів
    confidences = results[0].boxes.conf  # Впевненість

    current_ships = []
    for i in range(len(detections)):
        x1, y1, x2, y2 = map(int, detections[i])  # Координати рамки
        confidence = float(confidences[i])  # Рівень впевненість

        if confidence > 0.5:
            current_ships.append([x1, y1, x2, y2, confidence])

    entering_ships, exiting_ships, standing_ships =
compare_ships(current_ships, previous_ships)

    # Оновлення статистики
    total_entering += len(entering_ships)
    total_exiting += len(exiting_ships)

    image_name_without_extension = os.path.splitext(image_name)[0]

    date_str = image_name_without_extension.split('_')[1:4]  # Витягнення
року, місяця, дня
    image_date_str = "-".join(date_str)  # Перетворення у формат YYYY-MM-DD
    image_date = datetime.strptime(image_date_str, "%Y-%m-%d")

    if first_date is None or image_date < first_date:
        first_date = image_date
    if last_date is None or image_date > last_date:
        last_date = image_date

    print(f"Enter ships: {len(entering_ships)}, Exit ships:
{len(exiting_ships)}, Standing ships: {len(standing_ships)}")

    previous_ships = current_ships

    for ship in entering_ships:
        x1, y1, x2, y2, _ = ship
        cv2.rectangle(image, (x1, y1), (x2, y2), (0, 255, 0), 2)  # Зелений
для нових суден
        cv2.putText(image, "Entered", (x1, y1 - 10),
cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0, 255, 0), 2)

    for ship in exiting_ships:
        x1, y1, x2, y2, _ = ship
        cv2.rectangle(image, (x1, y1), (x2, y2), (0, 0, 255), 2)  # Червоний
для вийшлих суден
        cv2.putText(image, "Exited", (x1, y1 - 10), cv2.FONT_HERSHEY_SIMPLEX,
0.5, (0, 0, 255), 2)

    for ship in standing_ships:
        x1, y1, x2, y2, _ = ship
        cv2.rectangle(image, (x1, y1), (x2, y2), (255, 255, 0), 2)  # Жовтий
для стоячих суден
        cv2.putText(image, "Standing", (x1, y1 - 10),
cv2.FONT_HERSHEY_SIMPLEX, 0.5, (255, 255, 0), 2)

    output_path = os.path.join(output_folder, image_name)
    cv2.imwrite(output_path, image)

    print(f"Processed and saved: {output_path}")

if first_date and last_date:
```

```
    print(f"\nСтатистика по всіх знімках:")
    print(f"З {first_date.strftime('%Y-%m-%d')} по {last_date.strftime('%Y-
%m-%d')}:")
    print(f"Зайшло в порт: {total_entering} суден")
    print(f"Вийшло з порту: {total_exiting} суден")

print("Візуалізація завершена!")
```