НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО”

Факультет інформатики та обчислювальної техніки

Кафедра обчислювальної техніки

До захисту допущено:
Завідувач кафедри
_____ Сергій СТІРЕНКО
(підпис)
“__” _____2021 р.

Дипломний проєкт

на здобуття ступеня бакалавра
за освітньо-професійною програмою “Комп’ютерні системи та мережі”
спеціальності 123 “Комп’ютерна інженерія”

на тему: *Модуль розпізнавання об'єктів для системи виявлення вирубок лісу по знімках місцевості*
Виконав: студент *IV* курсу, групи *ІО-71*
(шифр групи)

Шрам Влада Сергіївна _____
(прізвище, ім’я, по батькові) (підпис)

Керівник *ст. викладач Алєнін Олег Ігорович* _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Консультант(нормоконтроль)_____ _____
(назва розділу) (посада, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Рецензент _____ _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Засвідчую, що у цьому дипломному проєкті немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Київ – 2021 р.

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО”

Факультет інформатики та обчислювальної техніки

Кафедра обчислювальної техніки

Рівень вищої освіти – перший (бакалавр)

Освітньо-професійна програма

“Комп’ютерні системи та мережі”

спеціальність 123 “Комп’ютерна інженерія”

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Сергій СТІРЕНКО
(підпис)
“__” _____2021 р.

ЗАВДАННЯ
на бакалаврський дипломний проєкт студента

Шрам Влади Сергіївни

1. Тема проєкту *Модуль розпізнавання об'єктів для системи виявлення вирубок лісу по знімках місцевості*
керівник проєкту *ст. викладач Алєнін Олег Ігорович*,
(прізвище, ім’я, по батькові, науковий ступінь, вчене звання)
затверджені наказом по університету від *11 травня* 2021 року № *1139-с*
2. Термін здачі студентом закінченого проєкту _____ *2021 р.*
3. Вихідні дані до проєкту *технічна документація. теоретичні та статистичні дані.*
4. Зміст розрахунково-пояснювальної записки (перелік питань, які розробляються) *огляд існуючих систем, опис задачі та огляд методів рішення, вибір алгоритмів та засобів реалізації, розробка модуля розпізнавання.*
5. Перелік графічного матеріалу (з точним позначенням обов’язкових креслень): *функціональна схема, принципова схема, структурна схема.*
6. Консультанта проєкту, з вказівкою розділів проєкту, які до них вносяться

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв
Нормоконтроль	Сімоненко В.П.		

7. Дата видачі завдання_____

Календарний план

№ п/п	Найменування етапів дипломного проєкту	Терміни виконання етапів проєкту	Примітки
1.	*Затвердження теми проєкту*	*10.12.2020-15.12.2020*	
2.	*Вивчення та аналіз завдання*	*15.12.2020-10.03.2021*	
3.	*Огляд існуючих методів розпізнавання та детектування*	*10.03.2021-25.03.2021*	
4.	*Аналіз засобів для реалізація*	*25.03.2021-1.04.2021*	
5.	*Програмна реалізація модуля*	*1.04.2021-10.04.2021*	
6.	*Оформлення пояснювальної записки*	*10.04.2021-20.05.2021*	
7.	*Захист програмного продукту*	*25.05.2021*	
8.	*Передзахист*	*26.05.2021*	
9.	*Захист*	*14.06.2021*	

Студент-дипломник Шрам В.С. _____
(підпис)

Керівник проєкту Алєнін О.І. _____
(підпис)

Анотація

В даній бакалаврській дипломній роботі розроблено модуль розпізнавання об'єктів для системи виявлення вирубок лісу по знімках місцевості.

Даний модуль дозволяє отримати чорно-білу маску сегментації супутникового зображення з виявленими на ній вирубками лісу.

Був написаний з використанням бібліотек Tensorflow і Keras.

Annotation

In this bachelor's thesis, an object recognition module has been developed for detection of deforestation using terrain images.

This module allows you to get a black and white segmentation mask of a satellite image with deforestations detected on it.

It was written using the Tensorflow and Keras libraries.

ОПИС АЛЬБОМУ

до дипломного проєкту

на тему: «Модуль розпізнавання об'єктів для системи виявлення вирубок лісу по знімках місцевості»

Київ – 2021 року

№ рядка	Формат	*Значення*	*Найменування*	Кіл. листів	№ екземпляра	*Додаток*
			Документація загальна			
			Знову розроблена			
1	*А4*	***ІАЛЦ.467100.001 ОА***	Модуль розпізнавання об'єктів	1		
			для системи виявлення вирубок			
			лісу по знімках місцевості			
			Опис альбому			
2	*А4*	***ІАЛЦ.467100.002 ТЗ***	Модуль розпізнавання об'єктів	3		
			для системи виявлення вирубок			
			лісу по знімках місцевості			
			Технічне завдання			
3	*А4*	***ІАЛЦ.467100.003 ПЗ***	Модуль розпізнавання об'єктів	65		
			для системи виявлення вирубок			
			лісу по знімках місцевості			
			Пояснювальна записка			
4	*А4*	***ІАЛЦ.467100.004 Д1***	Модуль розпізнавання об'єктів	1		
			для системи виявлення вирубок			
			лісу по знімках місцевості			
			Функціональная схема			
5	*А4*	***ІАЛЦ.467100.004 Д2***	Модуль розпізнавання об'єктів	1		
			для системи виявлення вирубок			
			лісу по знімках місцевості			
			Принципова схема			
6	*А4*	***ІАЛЦ.467200.004 ДЗ***	Модуль розпізнавання об'єктів	1		
			для системи виявлення вирубок			
			лісу по знімках місцевості			
			Структурна схема			

Зм.	***Лист***	***№ докум.***	***Підп.***	***Дата***	

ІАЛЦ.467100.001 ОА

Розроб.	*Шрам В. С.*	*Модуль розпізнавання об'єктів для системи виявлення вирубок лісу по знімках місцевості Опис альбому*	*Літ.* / *Аркуш* / *Аркушів*
Перев.	*Сімоненко В.П.*		Т / 1 / 1
Реценз.			*НТУУ «КПІ імені Ігоря Сікорського», ФІОТ Група ІО - 71*
Н.Контр.	*Сімоненко В.П.*		
Затв.	*Стіренко С.Г.*		

ТЕХНІЧНЕ ЗАВДАННЯ

до дипломного проєкту

на тему: «Модуль розпізнавання об'єктів для системи виявлення вирубок лісу по знімках місцевості»

Київ – 2021 року

ЗМІСТ

1. НАЙМЕНУВАННЯ ТА ОБЛАСТЬ ЗАСТОСУВАННЯ

Дане технічне завдання розповсюджується на розробку модуля розпізнавання об’єктів для системи виявлення вирубок лісу по знімках місцевості.

Область застосування: практичне використання людьми для відслідковування вирубок на різних територіях.

2. ПІДСТАВИ ДЛЯ РОЗРОБКИ

Підставою для розробки є завдання на виконання роботи кваліфікаційно-освітнього рівня «бакалавр комп’ютерної інженерії», затверджене кафедрою обчислювальної техніки Національного технічного університету України «Київський політехнічний інститут».

3. МЕТА ТА ПРИЗНАЧЕННЯ РОЗРОБКИ

Метою даного проєкта є розробка модуля розпізнавання об’єктів для системи вивлення вирубок лісу по знімках місцевості.

4. ДЖЕРЕЛА РОЗРОБКИ

Джерелом розробки є наукові статті та публікації на рахунок розпізнаваннях об’єктів на знімках, а також вже створені аналогічні додатки.

5. ТЕХНІЧНІ ВИМОГИ

5.1.Вимоги до розроблюваного продукту

- Достатня точність розпізнавання.
- Зручний формат для вбудовування у систему.
- Висока швидкість обробки даних.

5.2.Вимоги до програмного забезпечення

- Windows 7 і вище, Ubuntu 16.04 і вище, macOS 10.12.6. і вище;
- Python 3.7 і вище;
- Tensorflow 2.0 і вище;

5.3.Вимоги до апаратної частини

Мінімально:

- Процесор: Intel Core i3 і новіші.
- Оперативна пам’ять: не менше 6 Гб.

6. ЕТАПИ РОЗРОБКИ

Вивчення літератури	05.02.2021
Складання та узгодження технічного завдання	22.03.2021
Розробка структури модуля	29.03.2021
Створення програмної частини	20.04.2021
Тестування	18.05.2021
Налагодження та виправлення помилок	21.05.2021
Оформлення документації дипломної роботи	25.05.2021

ПОЯСНЮВАЛЬНА ЗАПИСКА

до дипломного проєкту

на тему: «Модуль розпізнавання об'єктів для системи виявлення вирубок лісу по знімках місцевості»

Київ – 2021 року

ЗМІСТ

ВСТУП

Ліс - це складна система з великою кількістю різноманітних ланок. Дерева, трави, гриби і різні мікроорганізми – це все складові лісів. Кожна жива істота або рослина є невід'ємною частиною лісової зони.

Значення лісів у житті сучасної людини важко переоцінити. Ліс грає важливу екологічну роль для людини, так як саме дерева сприяють тому, що ми дихаємо чистим повітрям. Також, ліс є не тільки місцем туризму або відпочинку, а й джерелом матеріалів, які забезпечують нас необхідними речами у повсякденному житті: в лісах росте деревина, з якої виготовляються будівельні матеріали, меблі, папір, лікарські продукти та інше. На сьогоднішній день важливість лісів посилюється через те, що їх кількість помітно зменшується і, на жаль, нелегальна вирубка лісів у наш час – це дуже гостра проблема. Нелегальні вирубки спричиняють велику шкоду якості та стану лісового фонду по причині того, що вирубаються високоякісні дерева. Внаслідок нелегальної вирубки деревини в країнах не лише втрачаються кошти, а й погіршується екологічна ситуація навколишнього середовища.

У зв’язку с цим, потрібно створити систему, яка б могла регулювати цю проблему. Найочевиднішим рішенням є аналіз фотографій певної місцевості для виявлення вирубок та відслідковування їх змін. Однією з частин такої системи є модуль розпізнання об’єктів.

Метою дипломної роботи є розробка модуля розпізнавання об’єктів для системи виявлення вирубок лісу по знімках місцевості.

РОЗДІЛ 1. ОГЛЯД ІСНУЮЧИХ РІШЕНЬ

1.1. Супутникові знімки

Космознімки [1] являють собою найбільш реалістичне зображення місцевості. Основа космознімків - дистанційне зондування Землі. Це спостереження за поверхнею Землі наземними, авіаційними та космічними засобами, оснащеними різними видами знімальної апаратури. Космічні знімки є надійною основою для моніторингу процесів і явищ, які знаходять відображення на картах: по знімках оперативно оновлюються карти, складаються прогнози погоди, прогнозується врожайність.

Автоматичне розпізнавання супутникових або аеро-знімків - це найбільш перспективний спосіб отримання інформації про розташування різних об'єктів на місцевості. Супутникові знімки у наш час легкодоступні в майже необмеженій кількості. До переваг космознімків відносяться актуальність даних, оперативність отримання інформації та велике одноразове просторове охоплення. На них можна здійснювати завдання класифікації, сегментації або пошуку аномалій.

Використання супутникових знімків застосовується для широко кола задач, пов’язаних із дослідженнями земельних покровів, будівель, транспорту та інше. Приклади систем, які використовують супутникові знімки наведено нижче.

1.2. Огляд систем, що детектують об’єкти на супутникових знімках

1.2.1. Orbital Insight

Одна із технологій створених у Orbital Insight – це сучасний автомобільний детектор. Цей детектор використовується для обробки 50-сантиметрових супутникових знімків [2] та отримання такої інформації, як поведінка споживачів при покупці. Автомобільний детектор реалізовано за допомогою згорткової нейронної мережі, яка була навчена на тисячах розмічених знімках.

Рисунок 1.1. Приклад автомобільного детектора в Orbital Insight

1.2.2. Сканекс

Сканекс [3] – це система моніторингу природних пожеж, яка була створена російською компанією «Сканекс» і основана на технологіях аналізу супутникової інформації. Головною метою даного сервісу є надання результатів моніторингу пожеж всім зацікавленим користувачам. Проект орієнтований як на звичайних користувачів, так і на фахівців.

Відомості про пожежі надані сервісом є продуктом автоматизованої тематичної класифікації даних дистанційного зондування Землі - космічних знімків, отриманих із супутників Terra, Aqua, NPP, NOAA-20.

Алгоритми детектування пожеж засновані на випромінюванні в інфрачервоному діапазоні - різниця в температурних яскравостях пікселів показує різницю між температурою вогнищ пожежі і земної поверхні, а дані, що надходить з інших спектральних каналів, дають змогу маскувати хмари. Крім того, за допомогою порогових значень здійснюється фільтрація термальних аномалій, які з малою ймовірністю відносяться до пожеж. Таким чином, вихідним продуктом алгоритму є маска пікселів, класифікованих як "термоточкі" пожеж і їх характеристики яскравості.

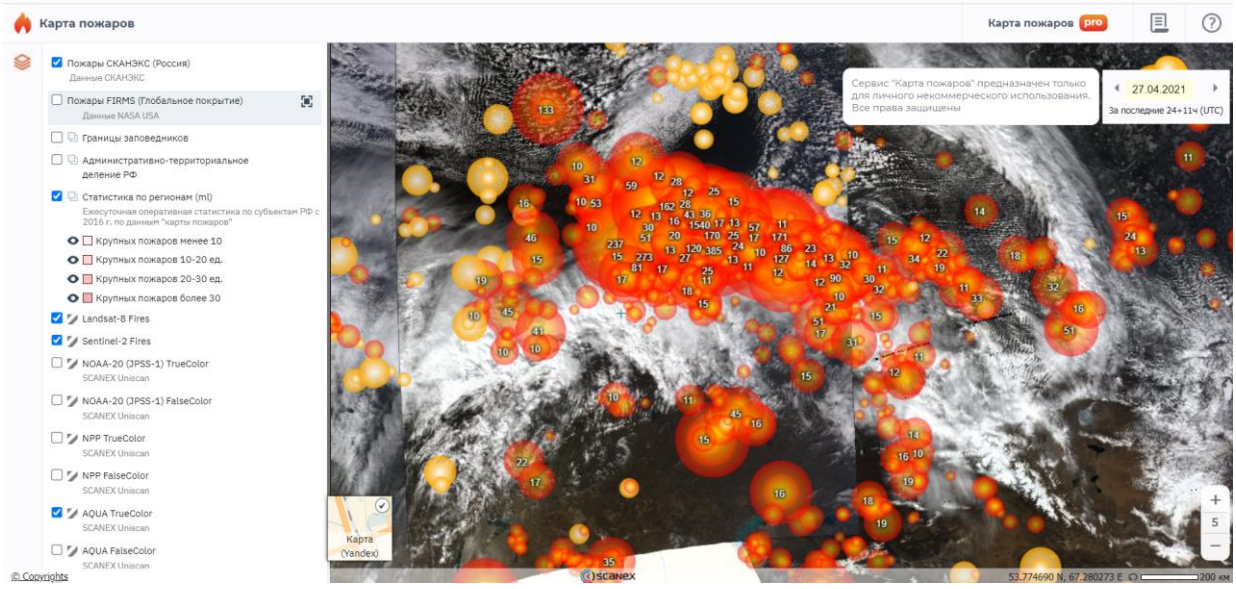


Рисунок 1.2. Система моніторингу Сканекс

1.2.3. Global Forest Watch

Global Forest Watch (Всесвітній лісовий дозор) [4] - це веб-додаток для моніторингу лісового покриву планети в майже реальному часі. Додаток було створено Інститутом світових ресурсів і іншими організаціями, в тому числі: Google, ESRI, Vizzuality і багатьма іншими. Використовуючи супутникові технології, відкриті дані та хмарні обчислення, GFW дозволяє кожному отримати доступ майже в режимі реального часу до інформації про те, де і як змінюються ліси по всьому світу.

Додаток використовується, наприклад, для оцінки збезлісення в різних регіонах планети, для виявлення і відслідковування незаконних вирубок лісів, для оцінки лісових ресурсів та планування їх використання у лісовому господарстві. Використовування даного додатку дозволило виявити, що в 2011-2013 роках РФ і Канада зайняли перші місця в списку країн з найбільшим збезлісенням - перегнавши Бразилію та інші країни з тропічним лісами. У 2015 році дані НАСА про лісові пожежі, які використовувалися в додатку, допомогли виявити незаконне спалювання лісу для отримання землі для сільськогосподарського використання, що призвело до сильного забруднення атмосфери в Південно-Східній Азії.

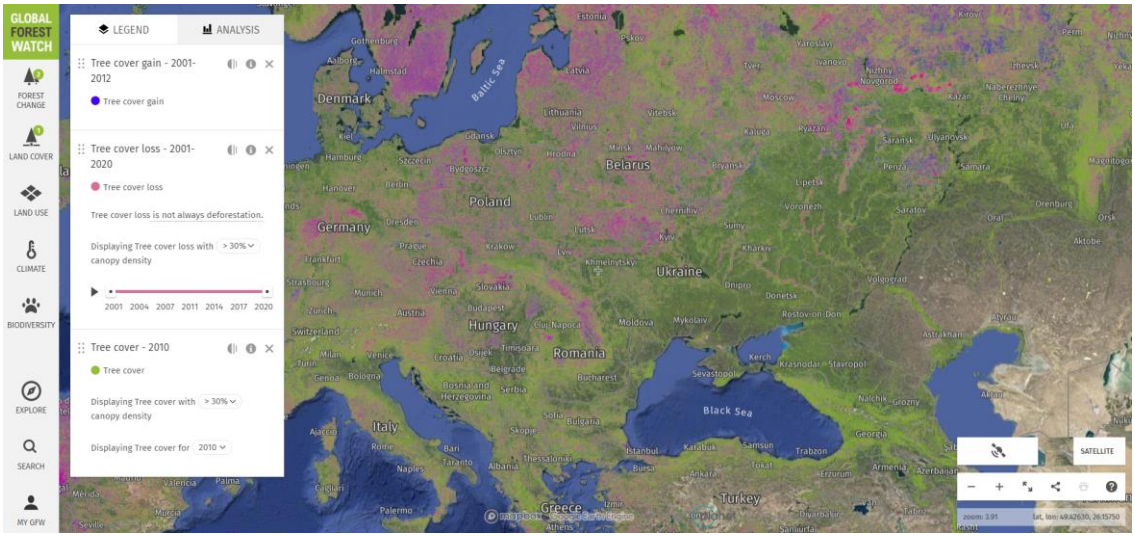


Рисунок 1.3. Інтерфейс карти Global Forest Watch

1.2.4. Deep Green Ukraine

Deep Green Ukraine [5] – це український стартап, який був створений для відслідковування незаконних вирубок лісу. Цей проект виник в Інституті космічних досліджень НАН України, де раніше створювали карти, в яких за основу були взяті супутникові дані, із застосуванням нейронних мереж. Зараз проект реалізують за підтримкою громадських організацій «Лісні ініціативи та суспільство» та «Центр моніторингу влади».

Цей сервіс буде здатний на основі супутникових даних виявляти вирубки, і за допомогою аналізу відповідних документів класифікувати ці вирубки як законні або незаконні. Проект використовує дані Європейського космічного агентства зі супутників Sentinel-1 и Sentinel-2. Зображення з цих супутників об’єднуються і виходить шестиканальний знімок-композит – об’єднання знімків різних території за різний час.

Deep Green Ukraine використовує три типи моделей для навчання: U-Net і дві на основі Fishnet – згорткової нейронної мережі для класифікації зображень. В результаті створюються три карти, які генерують псевдолейбли або псевдомітки – припущені дані, які модель створила на основі маркованих. Найкраща модель продовжує навчання на новому датасеті з вже розмічених даних та псевдоміток.

Показники по виявленню незаконних вирубок досягли 98%. Але розробники вважають, що в перші місяці роботи будуть реєструватися багато

хибних випадків через помилки в державних даних. Проект ще не запущений, але планується бути введеним в експлуатацію у червні 2021 році у чотирьох областях України.

1.3. Постановка задачі

Розроблювана система – це сервіс для виявлення вирубок лісу, який використовує супутникові знімки місцевості. Система складається з веб-додатку для користування сервісом, за допомогою якого користувач зможе отримувати зображення потрібної йому території, ввівши її координати, або завантажувати супутникові зображення і модуля розпізнавання самих вирубок на супутникових зображеннях. Модуль повинен знайти на отриманому зображенні вирубки лісу та надати знайдені вирубки для відображення користувачеві.

1.4. Способи детектування об’єктів на зображеннях

Пошук об’єктів на зображенні можна здійснювати за двома методами: моделювання об’єкта та моделювання фону [9].

Моделювання об’єкта – цей метод зазвичай застосовується, коли фон об’єкта на зображенні не постійний і змінюється. При застосуванні цього методу потрібно знати, що конкретно потрібно знайти на зображенні.

Моделювання фону – цей метод застосовується, коли фон об’єкта не змінюється або мало змінюється. Таким чином можна побудувати модель самого фону, і все що не підходить до цієї моделі є шуканим об’єктом.

Так як у нашому випадку фон змінний, то ми будемо розглядати методи моделювання об’єктів.

Методи пошуку об’єктів на зображенні:

1. Пошук по кольору.
2. Виділення та аналіз контурів.
3. Особливі точки.
4. Методи машинного навчання:
    1. Метод Віоли Джонса
    2. Нейронні мережі

1.4.1. Пошук по кольору

Одним з найпростіших способів знайти об'єкт на зображенні є його пошук по кольору. Пошук по кольору застосовують у тих випадках, коли шуканий об’єкт на зображенні відрізняється кольором від фону, а освітлення рівномірне та не змінюється. Такий алгоритм застосовують для того, щоб прибрати із зображення усе лишнє по ознаці кольору. Отриманим результатом може бути наприклад чорно-біле зображення [7], тобто накладаючи на зображення кольоровий фільтр, кольорове зображення перетворюється у чорно-біле, де білий колір – це об’єкт шуканого кольору, а чорний – фон.

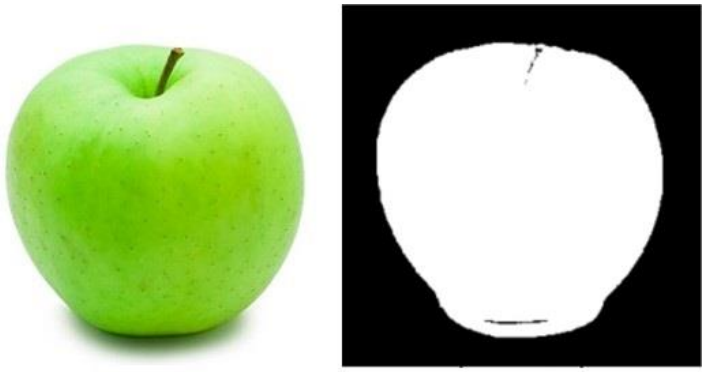


Рисунок 1.4. а) Початкове зображення, б) Зображення після пошуку по кольору

Цей метод передбачає роботу з зображенням як з масивом пікселей RGB. Але RGB представлення зображення не завжди є оптимальним, так як в RGB колір пікселя визначається насиченістю червоного, зеленого і блакитного кольору, а тому визначення діапазону відтінків одного кольору є складною задачею. Тому зображення переводять у формат HSV [6] – hue saturation value. У форматі HSV колір обирається за допомогою параметру Hue – кольоровий тон. Параметри saturation та value – насиченість та яскравість відповідно - дають змогу регулювати насиченість і яскравість обраного кольору. Це дозволяє досить легко задавати діапазони потрібного кольору та його відтінків.

Основним недоліком такого методу є прив’язка до певного кольору та його відтінків, пов’язаних з освітленням. Тобто якщо колір об’єкта зміниться, то даний метод буде працювати некоректно.

1.4.2. Виділення та аналіз контурів

Якщо об’єкт на зображенні явно не відрізняється по кольору від фону [9], або має декілька кольорів, то пошук по кольору не дасть гарних результатів. Тому можна спробувати застосувати метод виділення та аналізу контурів.

Контур об'єкта - це зовнішній обрис об’єкта, що відділяє його від фону. Більшість методів аналізу зображень працюють не з пікселями, а саме з контурами об’єктів. Методи, які працюють із контурами об’єктів в сукупності мають назву контурний аналіз. Також, для більшої ефективності, методи аналізу контурів можна поєднувати з пошуком по кольору.

Метод виділення та аналізу контурів базується на виділенні таких точок зображення, в яких різко змінюється яскравість [8]. Виділення контурів полягає в тому, щоб підсилити перепади яскравості на зображенні. Найпростішим методом виділення контурів є просторове диференціювання функції яскравості. Існує багато методів виділення контурів, основні з яких: Робертса, Лапласа і Собеля.

Оператор Робертса - один з алгоритмів виділення контурів, який обчислює на зображенні суму квадратів різниць між діагонально суміжними пікселями. Оператор Собеля – оператор, який використовує наближення до похідної, що дозволяє виділяти контури в тих місцях, де градієнт найвищий. Результатом застосування оператора Собеля є вектор градієнта яскравості.

а) б)

Рисунок 1.5. а) Початкове зображення, б) Зображення з виділеними контурами

При вираженому об’єкті на фоні або при відсутності перешкод контурний аналіз показує гарні результати, але цей метод є не досить стійким до перешкод, так як наприклад часткова видимість об’єкта або перетин призводять до помилкового знаходження об’єктів або до повної відсутності їх знаходження.

1.4.3. Особливі точки

Особливі точки (key points) [9] – це деякі ділянки зображення, які певним чином виділяються на зображенні. В якості особливих точок можуть виступати екстремуми яскравості, кути, а також інші особливості зображення. Особливі точки, як і контури, можна описати у векторному вигляді. Наприклад, для досягнення не чутливості методу до повороту можна описати взаємне розташування точок як відстані між ними. При повороті об'єктів відстань між цими точками не змінюється – отже метод буде не чутливим до повороту.

Дескриптор – це так звана характеристика особливої точки. Його розраховують по заданій околиці особливої точки, як напрямки градієнтів яскравості різних частин даної околиці. Існують такі методи розрахунку дескрипторів: SIFT, SURF, ORB тощо.

Для пошуку об’єктів на зображенні за допомогою особливих точок ми повинні на зображенні об’єкта знайти його особливі точки та обчислити їх дескриптори. Далі на зображенні, на якому шукається об’єкт, здійснити ті ж дії. Після цього треба порівняти дескриптори особливих точок об’єкта та дескриптори особливих точок, які були знайдені на зображенні. Якщо знайдено певну кількість співпадінь, то шуканий об’єкт є на зображенні.

Такий метод є стійким до поворотів та змін масштабу (в залежності від обраного дескриптора), але він працює ефективно тільки якщо знайдена достатня кількість особливих точок на зображенні об’єкта і вони досить добре збігаються із особливими точками зображення.

1.4.4. Методи машинного навчання

Машинне навчання [12] – це область науки про штучний інтелект, що розробляє алгоритмі, які мають здібність навчатися на основі даних, тобто такі алгоритми шукають певну закономірність у даних.

Складові машинного навчання:

1. Дані – для того щоб навчити машину, наприклад, розпізнавати спам, їй потрібний набір даних, який складається із прикладів спам-листів. Даних потрібно якомога більше для ефективного навчання машини на них.
2. Ознаки – машина повинна знати, на що їй звертати увагу при навчанні і для цього використовують ознаки об’єктів, з яких складається набір даних. Але коли ознак забагато, модель працює повільно та неефективно.
3. Алгоритми – одну й ту ж задачу можна вирішити багатьма способами. Від вибору залежить швидкість роботи моделі, її точність та інше. Але порівнюючи важливість даних та алгоритмів у машинному навчанні, великий та якісний набір даних має більшу значимість, так як на неякісних даних навіть найкращий алгоритм буде працювати погано.

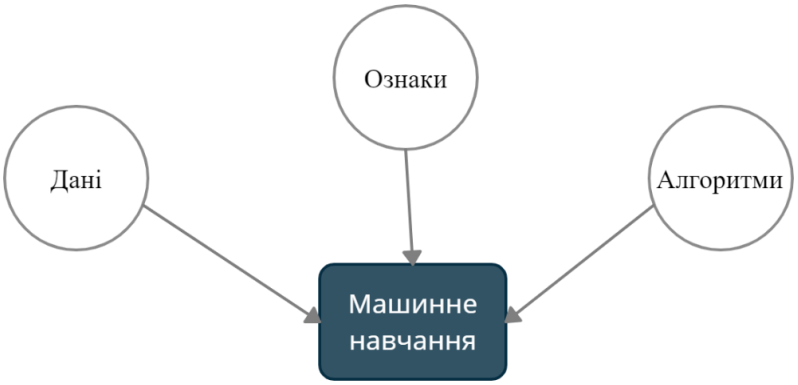


Рисунок 1.6. Складові машинного навчання

Види машинного навчання за ознакою наявності вчителя [10]:

1) Навчання з вчителем – найпоширеніший вид навчання. Таке навчання передбачає навчання системи на розміченому навчальному наборі, тобто для таких даних відповідь вже визначена. Модель навчається на наданих даних і потім робить передбачення для даних, яких вона ще не бачила. Більша частина нейронних мереж використовує навчання з вчителем.

2) Навчання без вчителя - таке навчання передбачає навчання системи на нерозміченому навчальному наборі, намагаючись знайти певні залежності та зробити висновки про подані дані. Однією із задач з цим видом навчання є задача кластерізації. Система намагається знайти схожі ознаки в даних та об’єднати їх у кластери.
3) Навчання з підкріпленням – вид навчання, в якому є агент, який може отримувати винагороди або штрафи за свої дії. Тобто агент з часом навчається певній стратегії, при якій він буде отримувати максимальну винагороду.

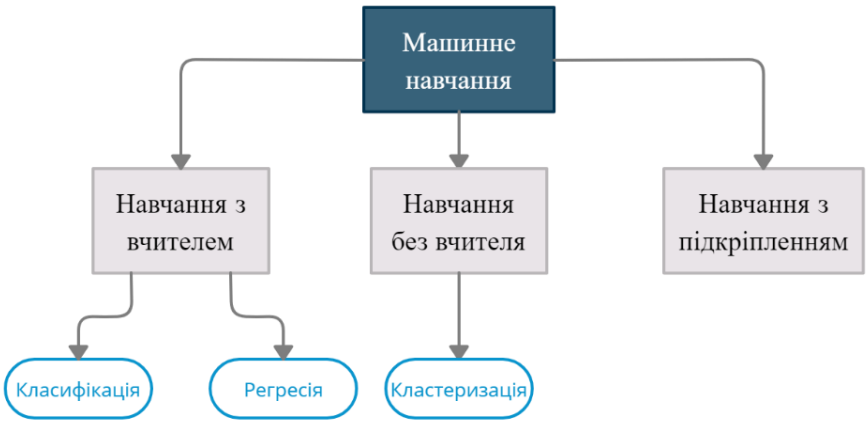


Рисунок 1.7. Види машинного навчання за ознакою наявності вчителя

Види машинного навчання за алгоритмами, що використовуються [12]:

- класичне навчання - відомі алгоритми навчання, які будуються на класичних статистичних алгоритмах і пов'язані з прийняттям рішень на основі даних. Такі алгоритми гарно працюють з невеликими об’ємами даних і зазвичай вихідними значеннями таких алгоритмів є числові значення. Насамперед застосовуються у класифікації, регресії, кластеризації тощо.
- нейронні мережі і глибоке навчання - сучасний підхід до машинного навчання. Вихідні значення таких алгоритмів можуть мати різні формати, наприклад текст, зображення чи звук. Глибокі мережі досягають точності, яка перевищує точність при класичному навчанні, наприклад, у задачах, де потрібно розпізнати зображення чи відео або у задачах машинного перекладу. У задачах пов’язаних з комп'ютерним зором, попередньо

підготовлені мережі для класифікації зображень застосовуються в ході розпізнавання об'єктів і сегментації. Такий підхід полегшує навчання усієї моделі і дає змогу досягти високої продуктивності за менший період часу.

1.4.4.1. Метод Віоли Джонса

Метод Віоли-Джонса – алгоритм, який дозволяє виявляти об’єкти на зображеннях. Був запропонований Полом Віолою і Майлом Джонсоном в 2001 році. Основною задачею цього методу було розпізнавання облич, але його можна використовувати і для розпізнавання інших об’єктів. Перевагами даного методу є високі швидкість роботи і відсоток вірного детектування об'єкта на зображенні [13].

Цей метод детектування об’єктів на зображенні включає в себе чотири основних концепції [11]:

- використання ознак Хаара;
- представлення зображення в інтегральному вигляді;
- навчання системи розпізнавання на основі методу машинного навчання AdaBoost;
- організація «каскадного класифікатора».

В якості ознак для алгоритму авторами були запропоновані ознаки Хаара, які основані на вейвлетах Хаара. Ознак Хаара представляють собою прямокутні області, які складаються з декількох прямокутних областей, світлих та темних [14]. Із двох ознак Хаара будується, наприклад, перший каскад системи по розпізнаванню об’єктів. Присутність функції Хаара визначається за допомогою різниці середнього значення темної області пікселів і середнього значення світлої області пікселів, якщо ця різниця є більшою ніж поріг, який задається в ході навчання, то функція є існуючою. Ознаки Хаара дають точкове значення перепаду яскравості по осі X і Y відповідно.

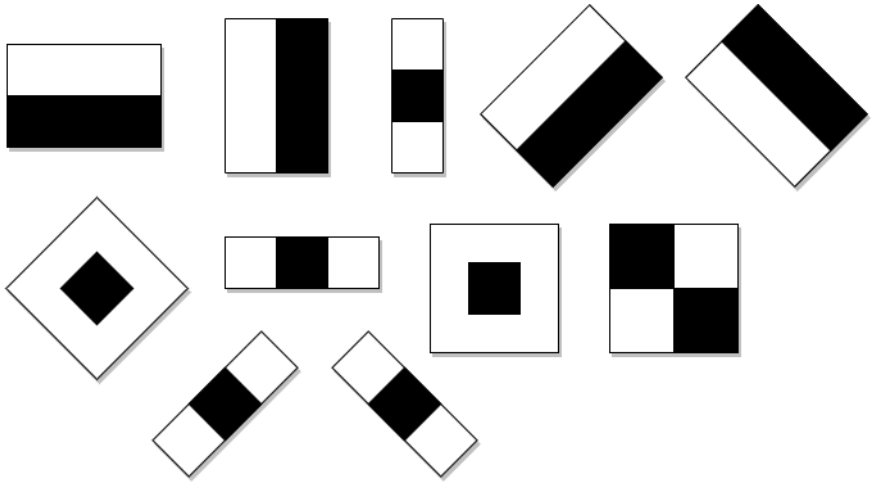


Рисунок 1.8. Ознаки Хаара

Для того, щоб проводити будь-які дії з даними в методі Віоли-Джонса, використовується інтегральне представлення зображень. Однією з корисних особливостей інтегрального представлення є можливість дуже швидко вирахувати суму пікселів довільного прямокутника. Інтегральне уявлення зображення можна представити у вигляді матриці, розміри якої збігаються з розмірами даного зображення, де кожен елемент розраховується як сума інтенсивностей усіх пікселів лівіше та вище за даний [11]. Розрахунок такої матриці займає лінійний час. Це дає змогу досить швидко розраховувати ознаки Хаара для зображення в навчанні і під час розпізнавання.

Для встановлення порогових значень та вибору конкретних функцій Хаара, Віола і Джонс використовують метод машинного навчання під назвою AdaBoost. AdaBoost комбінує багато «слабких» класифікаторів для створення одного «сильного» класифікатора [14]. «Слабкий» класифікатор – це такий класифікатор, який видає правильну відповідь не частіше, ніж випадковий. Такий класифікатор не є ефективним. AdaBoost створює набір слабких класифікаторів і присвоює кожному з них свою вагу. Ця зважена комбінація і є сильним класифікатором.

Каскадна структура класифікатора [11] дає змогу підвищити швидкість виявлення об’єктів, зосереджуючи свою роботу на найбільш інформативних областях зображення.

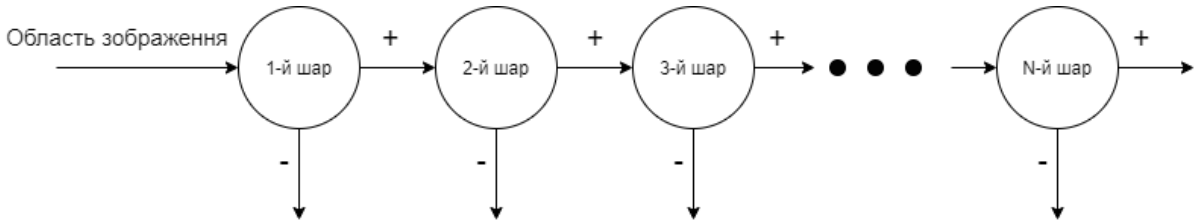


Рисунок 1.9. Структура каскадного класифікатора

Перед початком розпізнавання алгоритм проходить навчання, навчання алгоритму Віола-Джонса - це навчання алгоритму з учителем [14]. Далі алгоритм розпізнавання шукає об’єкти на різних масштабах зображення.

Для навчання з вчителем можлива така проблема як перенавчання. Перенавчання - це надмірна підгонка параметрів моделі до залежностей, що містяться в навчальній множині. Якщо присутнє перенавчання, то модель не набуває здатності до узагальнення – тобто модель не може переносити закономірності з навчальних даних на нові дані.

1.4.4.2. Нейронні мережі

Нейронна мережа представляє собою структуру, яка складається з штучних нейронів, пов'язаних між собою і зовнішнім середовищем за допомогою зав’язків [24]. Кожен із зав’язків має коефіцієнт (вагу), на який множиться значення, яке надходить через цей зв’язок. В процесі своєї роботи, нейронна мережа певним чином перетворює дані. Вид цих перетворень залежить від ваг зв'язків між нейронами, функції активації нейронів, архітектури та конфігурації мережі.

Нейронні мережі представляють собою моделі, які базуються на машинному навчанні, тобто набувають необхідних властивостей при навчанні. Навчання нейронної мережі полягає в послідовній зміні ваг мережі згідно з деяким алгоритмом навчання. Навчання може проходити як з вчителем, так і без вчителя.

Звичайна (повнозв’язна) нейронна мережа складається з трьох компонентів [15]:

- Вхідний шар;
- Приховані (обчислювальні) шари;
- Вихідний шар.

Нейронна мережа для розпізнавання зображень - це, мабуть, найбільш популярний спосіб застосування нейронних мереж.

Стандартні завдання, які вирішуються нейронними мережами із зображеннями [21]:

- ідентифікація об'єктів;
- розпізнавання частин об'єктів або об’єктів;
- семантичне визначення меж об'єктів (дає змогу залишати тільки межі об'єктів на зображенні);
- семантична сегментація;
- виділення нормалей до поверхні (дає змогу перетворювати двовимірні зображення в тривимірні) тощо;

Нейронні мережі ефективні при роботі із зображеннями в зв'язку з тим, що вони мають слабку чутливість до спотворень порівняно з іншими методами.

Багатошаровий перцептрон – нейронна мережа, яка складається із шарів, кожен з яких складається з нейронів. Багатошаровий перцептрон це мережа прямого поширення, тобто вхідний сигнал поширюється в прямому напрямку, від шару до шару. Така мережа складається мінімум з трьох слоїв: вхідний, прихований та вихідний. Навчання відбувається за алгоритмом зворотного поширення помилки [15].

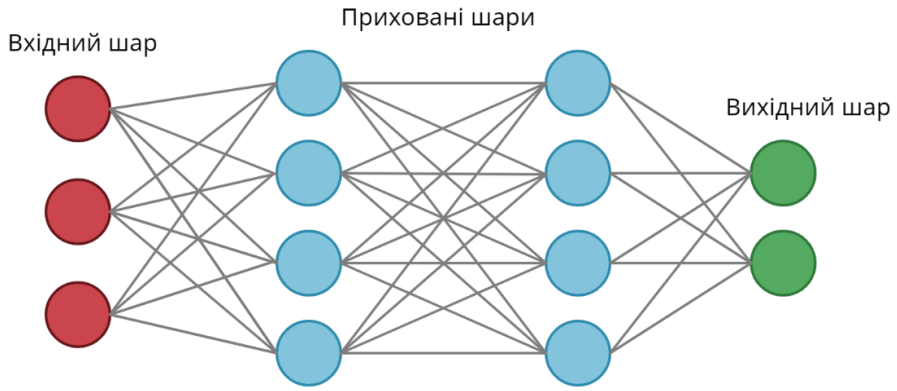


Рисунок 1.10. Багатошаровий перцептрон

Багатошаровий перцептрон є узагальненням одношарового перцептрона Розенблатта [17]. Приклад багатошарового перцептрона зображено на

рисунку 1.10. Кількість вхідних та вихідних нейронів визначається умовами поставленої задачі.

Багатошарові перцептрони застосовуються для вирішення різних задач і мають такі ознаки:

1) Усі нейрони, за виключенням вхідних, використовують нелінійну функцію активації.

Функція активації перевіряє вхідні дані нейрона на предмет того, чи повинні наступні нейрони розглядати цей нейрон як активний або ігнорувати його: вона спочатку нормалізує вхідне значення нейрону і якщо воно більше порогового значення, то цей нейрон вважається активним, якщо менше – то його потрібно ігнорувати. Однією з найбільш популярних форм нелінійної функції активації є сигмоїдальна функція.

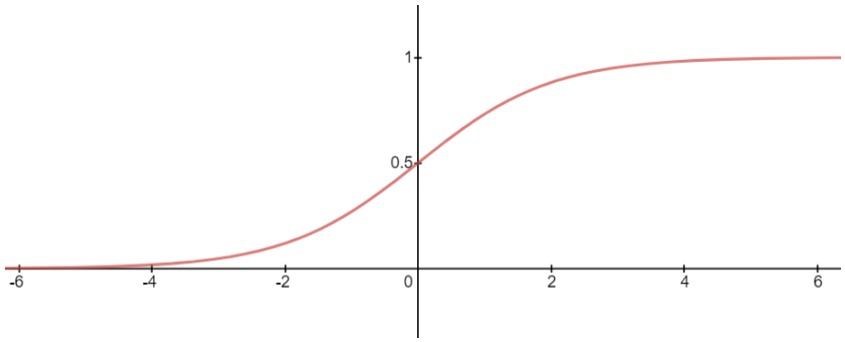


Рисунок 1.11. Графік сигмоїди

Сигмоїда на вході приймає будь-яке дійсне число, а на виході видає дійсне число від 0 до 1, при чому великі позитивні числа перетворюються в одиницю, а великі і по модулю негативні - в нуль. Оскільки вихід сигмоїди добре інтерпретується як рівень активації нейрона – 0 – відсутність активації, 1 – повністю активований - вона широко застосовується в нейронних мережах. Але така функція має недолік: при наближенні до кінців сигмоїди, вихідне значення майже не реагує на зміну вхідних значень. Це впливає на значення градієнту, який використовується при мінімізації помилки в нейронних мережах і створює проблеми при її навчанні.

Наявність нелінійності має значну роль при побудові багатошарового перцептрона, оскільки якщо її не буде, мережа зведеться до звичайного одношарового перцептрона.

2) Декілька прихованих шарів

Багатошаровий перцептрон складається з одного чи декількох приховних шарів, які не є вхідним чи вихідним. Це дозволяє мережі навчатися для розв'язання складних задач, витягуючи найбільш важливі ознаки із вхідної інформації.

3) Висока зв'язність

Багатошаровий перцептрон має високу степінь зв'язності, яка реалізується за допомогою зав'язків між нейронами – синапсами.

Хоча багатошаровий перцептрон використовується для різних класів задач, в тому числі для розпізнавання образів, така архітектура має ряд недоліків, які знижують ефективність роботи такої мережі. Насамперед, на вхід мережі можуть подаватися досить великі за розміром зображення. Для коректного навчання такої мережі потребується збільшувати кількість прихованих нейронів, це у свою чергу призводить до збільшення кількості параметрів [16], а це призводить до зниження швидкості навчання та потребує велику тренувальну вибірку для навчання.

Ще одним недоліком такої архітектури нейронної мережі є те, що вона ігнорує топологію вхідного зображення [16]. Локальні зв'язки пікселів – це важлива складова при роботі із зображенням, так як вони несуть цінну інформацію про зображення і вони формують певні категорії – кути, краї тощо. Згорткова нейронна мережа – це один із видів нейронних мереж, який в значній мірі усуває вище перераховані недоліки багаторшарового перцептрона і гарантує швидке навчання.

Згорткова нейронна мережа [18] – спеціальна архітектура нейронної мережі, яка була запропанована у 1988 році Яном Лекуном і входить у склад технологій глибоко навчання. Ідея згорткових нейронних мереж полягає в чергуванні згорткових та агрегувальних шарів. Навчання зазвичай проходить за допомогою методу зворотнього поширення помилки.

На сьогоднішній момент згорткові нейронні мережі та їх модифікації вважаються найкращими по швидкості та точності алгоритмами для

знаходження об’єктів на зображеннях. Ці нейронні мережі, починаючи з 2012 року, займають перші місця на міжнародному конкурсі по розпізнаванню образів ImageNet. Згорткові нейронні мережі дозволяють використовувати менший обсяг навчальних даних, та розширювати їх можливості для сегментації та розпізнавання об’єктів на зображеннях. Також вони забезпечують часткову стійкість до змін масштабу, поворотів та інших видів спотворень зображення.

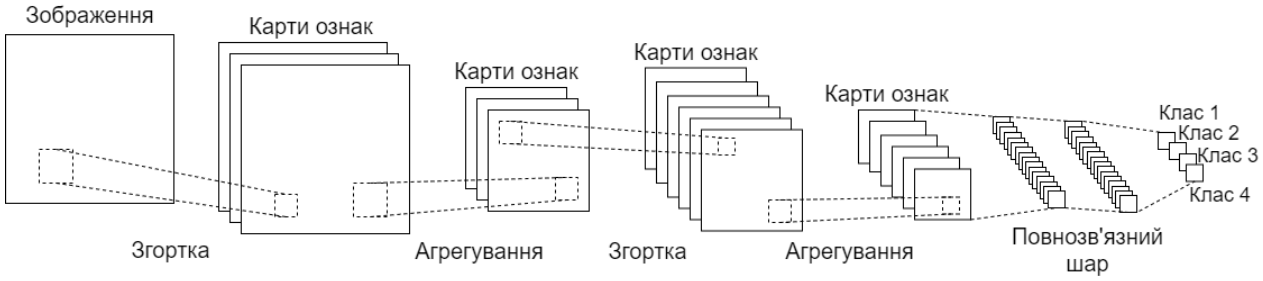


Рисунок 1.12. Загальний вид згорткової нейронної мережі

У згортковій нейронній мережі , на відміну від звичайного перцептрона, у якого кожен нейрон зв’язаний із усіма нейронами попереднього шару [18], у операції згортки використовуються лише невелика матриця ваг.

Операція згортки, зображена на рисунку 1.13 – це операція, при якій фрагмент зображення помножується на ядро згортки (матрицю ваг) по елементах, результати цих множень сумуються і записуються у відповідну позицію вихідного зображення, яка має назву карта ознак. В наступних шарах операція згортки застосовується до карт ознак, які були сформовані на попередніх шарах. Згортковий шар нейронної мережі дозволяє об’єднати значення пікселів, які розташовані поряд для виділення більш загальної ознаки зображення. Матрицю ваг рухають по шару, який обробляється і таким чином формується сигнал активації для нейрона наступного шару у відповідній позиції. Набір ваг у згортковій нейронній мережі не один, так як кожен кодує певні елементи зображення. Ядра згортки формуються під час навчання. Шар згортки зазвичай логістично об’єднують з шаром активації, тобто функція активації вбудована в шар згортки.

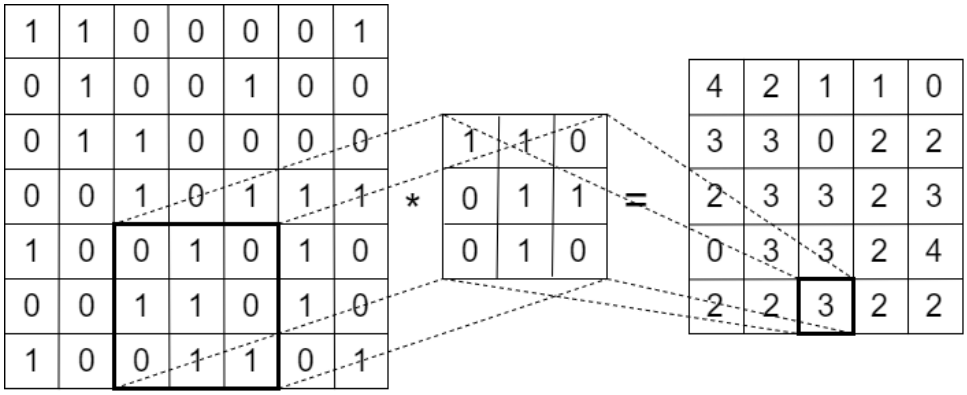


Рисунок 1.13. Операція згортки з фільтром 3x3

Операція агрегування (рисунок 1.14) – операція об'єднання (пулінг) – потрібна для зменшення розмірності сформованих раніше карт ознак, які формуються в результаті операції згортки. Карта ознак проходить нелінійне ущільнення, при якому група пікселів (зазвичай 2 на 2) за допомогою нелінійного перетворення ущільняється до одного пікселя. Зазвичай нелінійне перетворення відбувається за допомогою функції максимуму. Ця операція, окрім прискорення наступних обчислень, дає мережі можливість не перенавчатися. На відміну від згортки, агрегування зазвичай не застосовується до груп пікселів, що перетинаються.

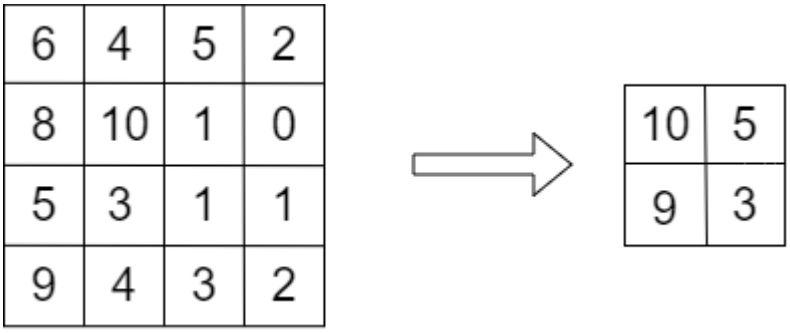


Рисунок 1.14. Операція агрегування з фільтром 2x2, кроком 2 і функцією максимуму

Після проходження декількох шарів згортки та агрегування карта ознак вироджується в вектор або скаляр. В такому вигляді вони потрапляють на шари повнозв'язної нейронної мережі – один або два, які можуть мати різні функції активації. Такі шари використовуються для моделювання складної нелінійної функції, яка найчастіше використовуються для класифікації.

ВИСНОВОК ДО РОЗДІЛУ 1

Отже, у ході дослідження, здійсненого в першому розділі дипломної роботи, було розглянуто такі системи, які аналізують супутникові знімки для різних цілей: детектування автомобілей, пожеж, змін лісу та вирубок. Дані системи мають за мету детектування певних об’єктів на зображеннях, що є і метою даної дипломної роботи.

Також, у розділі було сформульовано постановку задачі та розглянуто базові методи та методи машинного навчання, за допомогою яких можна детектувати об’єкти на зображеннях, і для кожного з них описано їх основні особливості, переваги та недоліки.

РОЗДІЛ 2. ВИБІР ТА ОБҐРУНТУВАННЯ СТРУКТУРИ МОДУЛЯ

2.1. Вибір методу детектування об’єктів

Одним з найважливіших чинників всіх алгоритмів є точність розпізнавання і класифікації даних. Багато алгоритмів розпізнавання [19] є нестійкими до різних змін зображення, на якому буде відбуватися пошук об'єкта. Багато об'єктів, які треба розпізнавати, можуть бути повернуті, спотворені або деформовані. Застосування нейронних мереж, для розпізнавання об'єктів, дозволяє вирішити цю проблему.

Нейронні мережі широко застосовуються для розпізнавання мови, обробки показників з різних сенсорів та для обробки різних типів знімків, наприклад, супутникових чи медичних. У даній роботі дані надано у вигляді супутникових знімків. У сфері розпізнавання образів на зображеннях нейронні мережі засовуються дуже часто і видають гарні і виправдані результати [10].

В ході навчання нейронної мережі для розв’язання певної задачі відбувається налаштування її параметрів, таким чином, ми не повинні вручну визначати параметри моделі – наприклад, обирати ключові ознаки, враховувати їх взаємовідношення тощо, тому що нейронна мережа автоматично витягує їх найкращим чином під час навчання. Виходячи з цього, ми повинні створити гарну тренувальну вибірку. Архітектури нейронних мереж та універсальні та потужні механізми навчання дають змогу створити гнучке налаштування на конкретну задачу, яка розв’язується.

Вирубки мають складний вигляд і структуру – вони відрізняються по варіації форм, кольорів, текстур та розмірів. На супутникових зображеннях можуть бути присутні деякі спотворення, які впливають на розпізнавання. Також присутність на зображеннях складного фону та сторонніх об’єктів (водоймищ, селищ, доріг) знижують надійність розпізнавання. Використання нейронних мереж є ефективним в такому класі задач, так як нейронні мережі є слабо чутливими до спотворень.

Нейронні мережі навчаються на прикладах. Для тренування підбираються необхідні дані, після чого запускається алгоритм навчання, який сприймає

структуру даних автоматично. Однак, від розробника потребується знати, які саме дані потрібно відбирати для навчання та як їх підготовлювати, яку саме архітектуру нейронної мережі обрати. Нейронні мережі не накладають ніяких обмежень на навчальні дані, вони сприймають навчальні дані як є, і намагаються виробляти правдоподібні рішення. Нейронні мережі мають здатність до гарного узагальнення, що означає, що після навчання на кінцевому наборі образів, нейронна мережа буде успішно працювати на усю множину образів, що і потрібно у даній роботі.

Однак не всі мережі підходять для розпізнавання зображень. Одна з найбільш поширених моделей нейронної мережі є багатошаровий перцептрон, але така модель може призвести до сильного збільшення часу і обчислювальної складності процесу навчання. Для усунення цих недоліків застосовується згорткова нейронна мережа.

Згорткова нейронна мережа – це один з найкращих алгоритмів по розпізнаванню та класифікації зображень. При порівнянні з багатошаровим перцептроном має переваги в тому, що кількість ваг, які налаштовуються, значно менша, так як ядро ваг використовується повністю для всього зображення замість того, щоб налаштовувати ваги для кожного пікселя окремо. Це дає можливість при навчанні до узагальнення усієї інформації на зображенні, а не розглядати кожен піксель окремо. Також, згорткова нейронна мережа має стійкість до поворотів та зсувів зображення. Тому доречно використовувати саме цей вид нейронної мережі.

2.2. Вибір виду результату мережі

Комп’ютерний зір [20] – напрямок в області штучного інтелекту, що дозволяє обробляти та аналізувати зображення з метою отримання певної інформації. Нейронні мережі та їх різні архітектури використовуються в передових дослідженнях комп’ютерного зору. Перед вибором архітектури згорткової нейронної мережі слід розглянути техніки комп’ютерного зору та обрати у якому саме вигляді мережа буде видавати результат.

Image Classification – класифікація об’єкта, який знаходиться на зображенні, тобто присвоєння йому деякого класу. Така техніка застосовується до зображень, де один об’єкт є домінуючим на зображенні.

Рисунок 2.1. Класифікація зображення

Object detection - знаходження усіх об’єктів із відомих класів на зображенні та визначення обмежувальної рамки навколо кожного з них. Це техніка на відмінку від класифікації застосовується для пошуку багатьох об’єктів, а не лише для визначення одного домінуючого об’єкта.

Рисунок 2.2. Виявлення об’єктів

Semantic Segmentation – техніка, яка розділяє ціле зображення на піксельні групи певних класів та фону. Якщо декілька об’єктів одного класу перетинаються або перекривають один одного, їх пікселі не відділяються один від одного.

Instanse Segmetation – визначення пікселів, які належать певному класу і об’єкту, подібно до семантичній сегментації, але для кожного об’єкта окремо.

а) б)

в)

Рисунок 2.3. а) Початкове зображення, б) Семантична сегментація, в) Сегментація екземпляру

В залежності від того яку саме задачу треба вирішити, обирається відповідна архітектура згорткової нейронної мережі. У даній дипломній роботі доречніше обирати серед технік Object Detection та Semantic Segmentation, так як звичайна Image Classification не в повному обсязі охоплює ту задачу, яка повинна виконуватися за допомогою розроблюваного модуля – така техніка дозволяє привласнити клас до зображення, наприклад, клас вирубка, але не дозволяє знайти місцезнаходження об’єкта на зображенні. Instance segmentation дозволяє знайти усі об’єкти на зображенні та місця, де саме вони знаходяться, і відділити кожен об’єкт один від одного, але в даній роботі відділення одного об’єкта від іншого не принципове.

При виборі серед Object Detection та Semantic Segmentation слід порівняти деякі особливості цих технік. В таблиці 2.1 перераховано особливості для порівняння та наведено їх характеристику для кожної з технік.

Таблиця 2.1. Порівняння Object Detection та Semantic Segmentation

Особливості	Object Detection	Semantic Segmentation
Знаходження об’єктів на зображенні	Знаходить усі об’єкти на зображенні	Знаходить усі об’єкти на зображенні
Вид виділення об’єктів	Виділяє об’єкти за допомогою прямокутних рамок	Виділяє об’єкти за допомогою маски, відносячи кожний піксель до певного класу
Форма знайдених об’єктів	Не дозволяє визначити форму об’єктів, так як виділяє їх прямокутною рамкою, яка може захоплювати значну частину того, що не відноситься до об’єкту	Дозволяє визначити форму об’єктів і виділяє тільки сам об’єкт
Підготовка навчальних даних	Шуканий об’єкт охоплюється у прямокутну рамку, тобто підготовка датасету досить проста	В найпростішому випадку маска малюється від руки, але за допомого спеціальних програм можна розмітити на об’єкті полігон, з якого потім створюється маска

У даній дипломній роботі доречніше використовувати Semantic Segmentation, так як маска сегментації дозволяє бачити форму об’єктів і в подальшому, якщо використовувати розроблюваний модуль для відслідковування змінити вирубок на певній території, маски досить легко порівнювати.

2.3. Вибір архітектури мережі для семантичної сегментації

Базова архітектура нейронної мережі для сегментації складається з енкодера та декодера. Енкодер витягує ознаки з зображення через фільтри, а

декодер відповідає за створення вихідного зображення – маски. Більшість нейронних мереж для сегментації мають дану архітектуру або певний її варіант.

U-Net [22] – це згорткова нейронна мережа для швидкого та точного сегментування зображення, одна з найбільш ефективних архітектур. Вона була створена у 2015 році для сегментації біомедичних зображень, для яких характерний постійний ракурс. Така архітектура дозволяє працювати з невеликою кількістю зображень для навчання, так як має відносно невелику кількість параметрів, які налаштовуються.

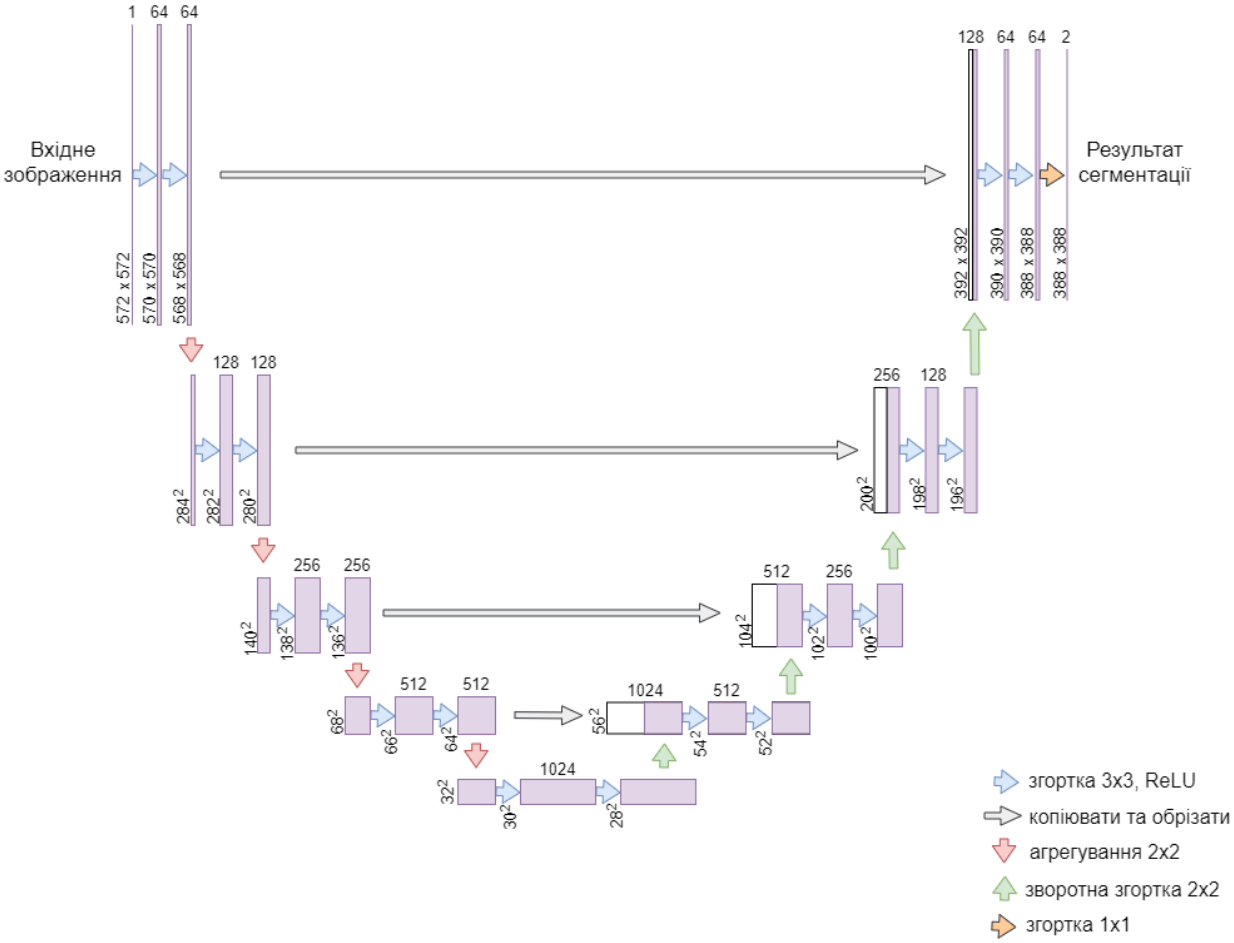


Рисунок 2.4. Архітектура U-Net

U-Net має таку назву через те, що її архітектура виглядає як літера U. Вона складається з двох частин [23]: згорткова (енкодер) та розгорткова (декодер). Перша частина використовується для вилучення із зображення семантичної інформації у вигляді вектора ознак, а друга – для перетворення цього вектору ознак у матрицю нового зображення – маски. Згорткова частина схожа на типову архітектуру згорткової нейронної мережі. Вона складається з двох

підряд згорткових шарів 3x3, після кожного з яких йде функція активції ReLu, і агрегування з функцією максимума 2x2 із кроком 2. Розгорткова частина складається з підвищення розмірності, який розширює карту ознак, після якого слідує згортка 2x2, яка зменшує кількість каналів ознак. Наступним кроком йде конкатенація з відповідним образом обрізаної карти ознак із згорткової частини мережі і дві згортки 3x3, після кожної з яких йде ReLu. Останнім кроком йде згортка 1x1, яка використовується для приведення кожного вектора ознак до кількості класів, яка потребується у задачі. Всього у мережі 23 згорткових шарів. Архітектура даної нейронної мережі зображена на рисунку 2.4.

PSPNet [25] (Pyramid scene parsing network) - це ще одна модель для сегментації поряд з Unet, яка є одною з найбільш відомих, оскільки вона перемогла у ImageNet Scene Parsing Challenge 2016. PSPNet архітектура охоплює контекст всього зображення для прогнозів локального рівня, а тому часто використовується для сегментації міських пейзажів.

Отримавши вхідне зображення, PSPNet використовує попередньо навчену згорткову нейронну мережу , наприклад ResNet, в якій в останніх шарах традиційні згорткові шари замінюються на розширені згорткові шари , наприклад, з нульовим відступом (рисунок 2.5), тобто додаються лишні «підробені» пікселі по краях масиву, що дозволяє не обрізати крайні пікселі при згортці, для вилучення карти ознак. Остаточний розмір карти ознак – 1/8 вхідного зображення.

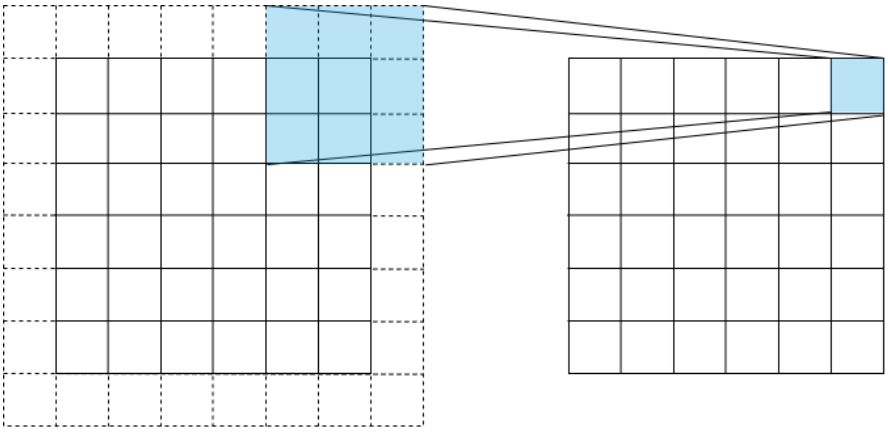


Рисунок 2.5. Операція згортки з нульовим відступом

Pyramid Pooling Module – PPM – це основа частина PSPNet, оскільки дає можливість фіксувати глобальний контекст на зображенні, що допомагає класифікувати пікселі на основі загальної інформації, яка присутня на зображенні. Карта ознак за допомогою пулінгу перетворюється у карти ознак різних розмірів, які на наступному кроці проходять через шар згортки, після чого відбувається підвищення їх розмірності, щоб зробити їх такого ж розміру, як оригінальна карта ознак. Далі отримані карти об'єднуються із початковою картою ознак, яка передається декодеру. Така методика дозволяє створювати загальний контекст, поєднуючи карти ознак різних розмірів.

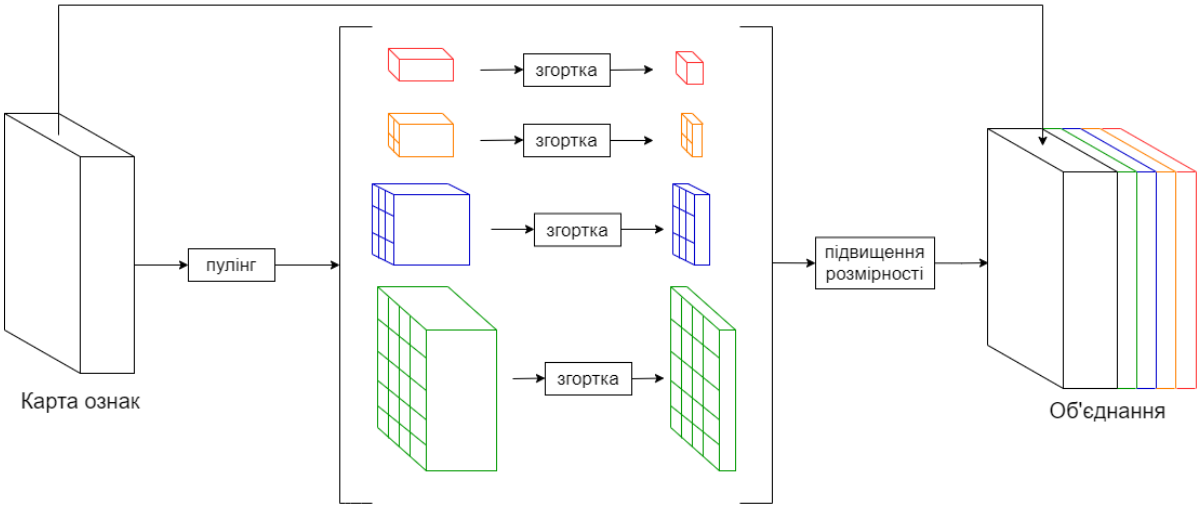


Рисунок 2.6. Pyramid Pooling Module

В якості декодера використовується, наприклад, згортковий шар, за яким слідує восьмикратне підвищення розмірності.

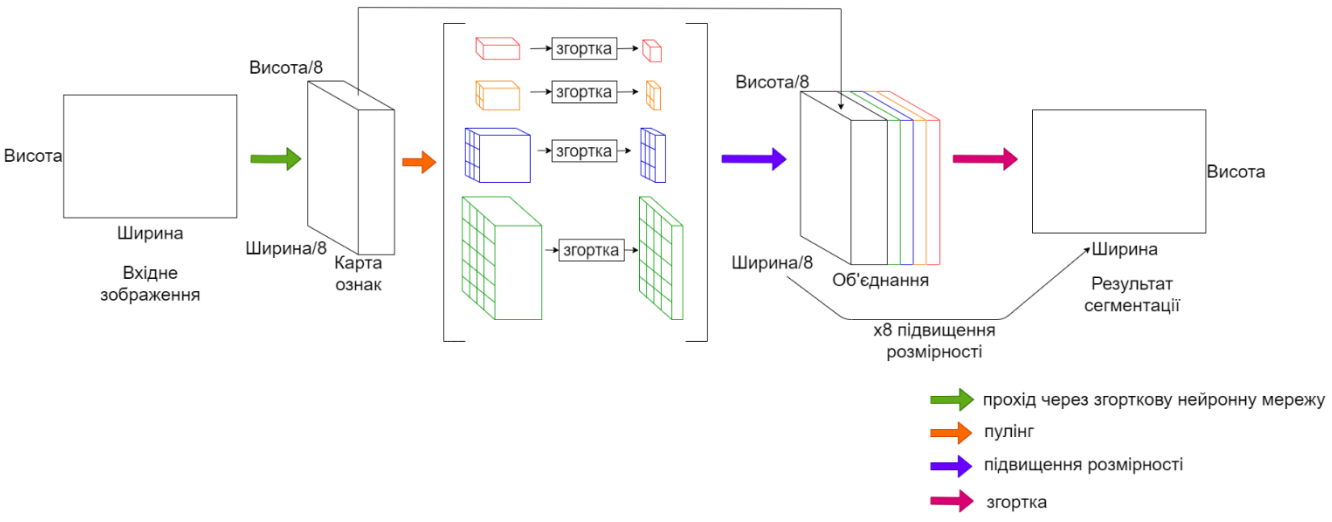


Рисунок 2.7. PSPNet

SegNet є типовим автокодувальником (тобто складається з енкодера та декодера [26]), заснованим на згортковій нейронній мережі. Архітектура данної мережі складається з послідовності нелінійних шарів обробки (енкодерів) та відповідного набору декодерів. Зазвичай, кожен енкодер складається з одного або декількох згорткових шарів із пакетною нормалізацією та функцією активації ReLU з подальшим пулінгом. Головною особливістю Segnet, яка відрізняє її від звичайного згорткового автокодувальника, яка була займана у архітектур, які застосовуються для навчання без вчителя: шари підвищення розмірності декодера пов'язані з відповідними шарами пулінга енкодера [27]. Повторне використання шарів пулінга в процесі декодування має декілька практичних переваг: це зменшує кількість параметрів, які задіюються при навчанні, так як шари підвищення розмірності не навчаються, а отримують інформацію від пулінг шарів, які зберігають індекси активованих пікселів, і покращує окреслення меж об'єктів. Архітектура майже повністю симетрична, окрім шару Softmax в кінці декодера. Цей шар перетворює кожен піксель вихідної матриці в ціле число, яке показує клас кожного пікселя. Всю архітектуру можна навчити, використовуючи стохастичний градієнтний спуск. Недоліком такої архітектури є те, що в ній навчається дуже багато параметрів, а тому потрібна велика навчальна вибірка та довге навчання.

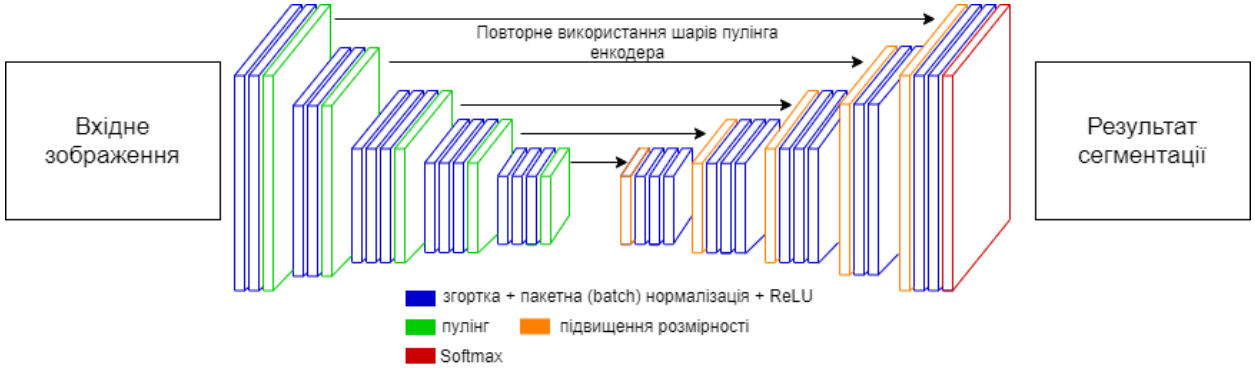


Рисунок 2.8. SegNet

Серед розглянутих архітектур було вирішено використовувати архітектуру U-Net по ряду причин:

- архітектура U-Net була створена для сегментації біомедичних зображень, для яких характерний постійний ракурс, що також відповідає нашій задачі

– для обробки використовуються супутникові знімки, які мають постійний ракурс;

- дана архітектура не потребує великого навчального набору для ефективного навчання, так як має відносно невелику кількість параметрів, які налаштовуються.

2.4. Вибір та обґрунтування програмних засобів для розробки нейронної мережі

2.4.1. Вибір мови програмування

Python [29] – це високорівнева мова програмування загального призначення, яка була створена Гвидо ван Россумом у 1991 році. Python є мультипарадигмовою мовою програмування – вона підтримує імперативне, процедурне, об'єктно-орієнтоване, функціональне програмування тощо. Дана мова є одною з найбільш мінімалістичних і поширених мов програмування.

Python є лідером серед мов програмування штучного інтелекту [28]. Ця мова є гнучкою та простою для розуміння, має відмінну продуктивність при обробці даних. Однією з основних причин, чому Python застосовують при роботі із штучним інтелектом, є те, що у даної мови є багато бібліотек, які спрощують написання коду і зменшують час на розробку, так як дозоляють застосовувати готові рішення замість того, щоб створювати їх з нуля. Прикладами таких бібліотек є Tensorflow, PyTorch, Theano тощо. Тому Python – це найкращий вибір при роботі із штучним інтелектом.

2.4.2. Вибір програмних засобів

Для реалізації нейронної мережі було обрано мову Python, а тому будуть розглянуті бібліотеки, які підтримуються цією мовою.

Tensorflow [30] – це популярна відкрита бібліотека для машинного навчання, яка була розроблена компанією Google у 2015 році для побудов нейронних мереж. Tensorflow може працювати на різних паралельних процесорах – CPU та GPU, спираючись на архітектуру CUDA для обчислення на графічних процесорах. Назва Tensorflow походить від операцій з багатовимірними масивами даних (вектори, матриці тощо), які мають назву

тензор. В основі Tensorflow лежать статичні [34] обчислювальні графи – абстракція, яка представляє обчислення у вигляді орієнтованого ациклічного графа, що дозволяє застосовувати паралелізм для прискорення роботи.

Перевагами Tensorflow [31] є детальна документація, яка включає в себе не тільки версію на офіційному сайті, але і сторонні джерела. Також бібліотека має повний набір інструментів для візуалізації, які спрощують розуміння та оптимізацію додатків, та моделі, створені за допомогою даної бібліотеки, досить легко вбудувати у власну систему. Так як Tensoflow є низькорівневим інструментом, навколо нього стали з'являтися високорівневі обгортки. Кожна така обгортка дозволяє спростити взаємодію з бібліотекою: TFLearn, Keras тощо. Tensorflow постійно розвивається за рахунок відкритого вихідного кода і великого співтовариства розробників. Але Tensorflow має головний недолік: він досить складний у використанні та освоєнні [34].

Keras [35] – бібліотека для роботи з нейронними мережами, написана на Python. Вона направлена на роботу з глибокими нейронними мережами і часто використовується у поєднанні з Tensorflow для полегшення роботи з ним, так як є високорівневим API. Дана бібліотека містить реалізації блоків нейронних мереж, таких як шари, цільові та передавальні функції, оптимізатори та інше.

PyTorch [33] – фреймворк машинного навчання з відкритим вихідним кодом, який був створений у 2017 році на основі Torch – MATLAB-подібної бібліотеки. PyTorch, як і Tensorflow, використовує тензори – багатовимірні масиви, і підтримує їх різні види. Також, цей фреймворк використовує обчислювальні графи, але на відміну від Tensorflow, не статичні, а динамічні. Таким чином, користувач може змінювати їх під час виконання – це є перевагою, наприклад, коли розробник не знає, скільки пам'яті потребується для створення моделі нейронної мережі.

Перевагами даного фреймворка є: просте налагоджування коду, великий вибір готових моделей [32], динамічні обчислювальні графи [34] та можливість створювати власні типи шарів. З недоліків слід виділити відсутність власних засобів візуалізації та недостатню гнучкість в підтримці різних платформ.

Caffe [37] - Convolution Architecture For Feature Extraction – це фреймворк глибокого навчання, розроблений Яньцином Цзя. Реалізований на мові C ++, і підтримує інтерфейс на мові Python. Caffe включає в себе велику кількість попередньо навчених моделей, готових до використання.

Caffe підтримує різні типи машинного навчання, націлених в першу чергу на класифікацію та сегментацію зображень. Caffe забезпечує згорткові нейронні мережі, регіональні згорткові нейронні мережі і повнозв'язні нейронні мережі. Для прискорення навчання застосовується система графічних процесорів (GPU), яка підтримується архітектурою CUDA.

Caffe маніпулює блоб (blob) - багатовимірними масивами даних, які використовуються в паралельних обчисленнях. Навчання в згортковій нейронній мережі реалізується як паралельні багатопроцесорні обчислення блоб від шару до шару (прямим і зворотним ходом).

Перевагами Caffe є велика швидкість навчання та тестування, велика кількість готових до використання моделей, високий рівень взаємодії центрального та графічного процесорів. Недоліками даного фреймворка є недостатня гнучкість [36], оскільки при внесенні змін, необхідно використовувати C++ та CUDA, обмежена підтримка спільноти, недостатня швидкість при роботі з великими масивами чисел.

Для даної роботи було вирішено використовувати Tensorflow у поєднанні з Keras, так як Tensorflow - це потужний інструмент для роботи з нейронними мережами, який є лідером у своїй області, тому що має великі функціональні можливості, зрозумілий синтаксис та потужні засоби візуалізації. У свою чергу Keras надає усе необхідне для моделювання архітектур глибоких нейронних мереж.

OpenCV [38] - бібліотека алгоритмів комп'ютерного зору для обробки зображень з відкритим кодом. Реалізована на C та C ++, також розроблюється для Python, Java та інших мов. Бібліотека OpenCV має понад 500 функцій, що охоплюють багато областей комп'ютерного зору. З моменту у 1999 року OpenCV знайшла місце в багатьох продуктах і наукових дослідженнях, наприклад: зшивання зображень для отримання супутникових і веб-карт,

калібрування камер, поєднання результатів сканування зображень, зменшення шумів на медичних зображеннях, системи контролю виробничих процесів, військові додатки, а також безпілотні повітряні, наземні і підводні транспортні засоби. Ця бібліотека популярна через свою відкритість та можливість безкоштовного використання як в навчальних, так і комерційних цілях. Дана бібліотека буде використовується при розробці модуля для попередньої обробки зображень.

2.5. Набір даних

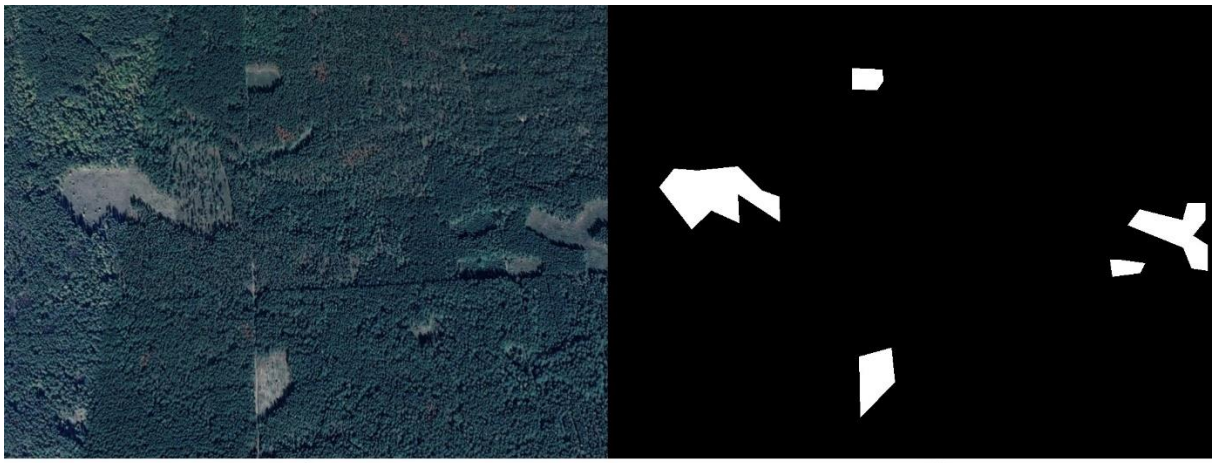
Набір даних, використаний для навчання мережі, був створений за допомогою сервісів Google Earth, Google Maps, ArcGIS Online для створення набору кольорових супутникових знімків та VGG Image Annotator для створення масок. Google Earth [41] – це проект компанії Google, в рамках якого в мережі Інтернет було розміщено супутникові (або в деяких точках аерофото) зображення всієї земної поверхні. Використовуються зображення, отримані як із супутників, так і шляхом аерофотозйомки. В якості основи глобального покриття використовуються знімки з супутника LandSat-7, оброблені компанією EarthSat в рамках проекту NASA зі складання мозаїки всій поверхні суші Землі (GeoCover circa 2000). В даний момент для надання зображень більш високої якості Google використовує супутник Landsat 8.

ArcGIS Online [40] — це комплексне рішення для картографування та аналізу американської компанії ESRI. За допомогою ArcGIS Online можна створювати веб-карти, використовувати готові до роботи ресурси, публікувати картографічні сервіси, здійснювати просторовий аналіз, поширювати дані і отримувати доступ до карт з будь-якого пристрою. Esri регулярно оновлює галерею базових карт за допомогою ресурсів комерційних постачальників даних і перевірених користувачів спільноти по всьому світу. Для даної роботи використовувалася карта World Imagery (Firefly), яка рекомендована Esri, так як постійно оновлюється і є достовірною.

VGG Image Annotator [39] – це інструмент для анотацій зображень, аудіо та відео і працює у веб-браузері. VIA розроблено з використанням HTML, CSS

та Javascript і базується виключно на стандартних функціях, доступних у сучасних веб-браузерах. VIA не залежить від будь-яких зовнішніх бібліотек. Даний сервіс надає можливість розмітити зображення за допомогою полігонів і згенерувати json файл. За допомогою цього файлу та модуля via з бібліотеки dh_segment для роботи з анотаціями, створеними за допомогою VGG Image Annotator, генеруються чорно-білі маски для кожного анотованого зображення.

Набір складається з 325 зображень, зроблених на території України в різних областях. В наборі даних присутні зображення розміром 1024x768 пікселів та 600x600 пікселів. Кожне зображення має чорно-білу маску відповідного розміру, на якій виділено вирубки. Такий набір при навчанні буде поділено на тренувальну вибірку, валідаційну та тестову вибірки – тренувальна вибірка безпосередньо використовується при навчанні, тобто по ній змінюються параметри моделі, валідаційна вибірка використовуються для відслідковування перенавчання мережі, а тестова – для безпосереднього тестування отриманої моделі. При формуванні цих вибірок важливий той факт, щоб ці вибірки не перетиналися, тобто не мали однакових зображень, так як через це модель може видавати некоректні результати для оцінки результату її навчання. Також, якщо у вибірках присутні різні види зображень, наприклад, різні по кольору, то вони мають бути врівноваженими, тобто кількість зображень одного виду не повинна домінувати над кількістю інших. Зазвичай, тестова та валідаційна вибірки становлять 10-20% від всього начального набору зображень.



а) б)

в)

г)

Рисунок 2.9. Приклади зображень з набору даних а) Google Earth, б) Чорно-біла маска, в) ArcGIS Online, г) Google Maps

ВИСНОВОК ДО РОЗДІЛУ 2

Отже, у ході дослідження, проведеного у другому розділі дипломної роботи, було обрано оптимальний метод детектування вирубок на супутникових зображеннях. Для виконання поставленої задачі було обрано застосування нейронних мереж, а саме згорткових, так як це найбільш зручний та ефективний спосіб для роботи із зображеннями. Було розлянуто та обрано вид результату детектування, який буде видавати розроблюваний модуль, а саме семантичну сегментацію та визначено, яка саме архітектура майбутної нейронної мережі найкраще підходить для поставленої задачі.

Для програмної реалізації було обрано мову Python та бібліотеки Tensorflow та Keras. В якості набору даних для навчання виступає власно зібраний набір кольорових супутникових зображень за допомогою сервісів Google Earth, Google Maps та ArcGIS Online та створені для кожного зображення чорно-білі маски за допомогою інструменту VGG Image Annotator та модуля via бібліотеки dh_segment.

РОЗДІЛ 3. РЕАЛІЗАЦІЯ НЕЙРОННОЇ МЕРЕЖІ

3.1. Реалізація класу U_net_model

Однією з найголовніших частин при розробці нейронної мережі є розробка моделі, так як саме її параметри будуть змінюватися у ході навчання і від яких в подальшому залежить якість сегментації. У попередньому розділі для реалізації було обрано модель U-Net, яка дозволяє отримувати гарні результати сегментації при невеликому наборі навчальних даних.

Дана модель реалізована у вигляді класу U_net_model. Даний клас має два методи: create_model та load_mask.

Для безпосереднього створення моделі використовується метод create_model, який примає такі аргументи:

- size – висота та ширина зображення, які визначають розміри вхідного шару мережі;
- img_channels – кількість каналів зображення, тобто кількість числових значень, за допомогою яких кодується колір пікселя. У RGB зображення має три числа для кожного пікселя: червоний, зелений та синій кольори;
- filters – початкова кількість фільтрів (ядер згортки);
- activation – функція активації, яка буде застосовуватись разом із згортковими шарами;
- final_activation – функція активації, яка буде застосовуватись в останньому шарі згортки для приведення кожного вектора ознак до кількості класів, які потрібно визначити у даній задачі. У нашому випадку класа два: білий – вирубка та чорний – фон.

Даний метод створює модель, яка складається з таких шарів:

- Input – вхідний шар, який визначається висотою, шириною та кількістю каналів зображення;
- Conv2D – згортковий шар із кількістю фільтрів filters розміром 3 на 3, функцією активації activation та параметром padding, який приймає значення same, що дозволяє зберігати таку саму висоту і ширину

зображення на виході, як і на вході. Останній шар згортки натомість має єдиний фільтр розміром 1 на 1 та функцію активації final_activation.

```
conv10 = Conv2D(1, (1,1), activation=final_activation)(conv9)
```

Рисунок 3.1. Останній шар згортки

- MaxPool2D – шар агрегування з фільтром 2 на 2 із функцією максимуму, застосовується для зменшення розміру зображення.

```
conv1 = Conv2D(self.filters, (3,3), activation=self.activation, padding='same')(inputs)
conv1 = Conv2D(self.filters, (3,3), activation=self.activation, padding='same')(conv1)
pool1 = MaxPool2D((2,2))(conv1)
```

Рисунок 3.2. Шари згортки та агрегування

- Conv2DTranspose – операція, оберенена до згортки. Даний шар потрібен для здійснення підвищення розмірності зображення, так як шари згортки зменшили його на попередніх кроках. На відміну від простого Upsampling, який також підвищує розмірність, Conv2DTranspose має параметри, що навчаються, що дозволяє навчити модель здійснювати підвищення розмірності найкращим чином.
- Concatenate – операція для об’єднання карт ознак з шару енкодера та декодера, яка використовується для отримання більш точного передбачення.

```
up_conv9 = Conv2DTranspose(filters, (2, 2), strides=(2, 2), padding='same')(conv8)
up_conv9 = concatenate([up_conv9, conv1], axis=3)
conv9 = Conv2D(filters, (3, 3), activation=activation, padding='same')(up_conv9)
conv9 = Conv2D(filters, (3, 3), activation=activation, padding='same')(conv9)
```

Рисунок 3.3. Шари оберненої згортки, операції об’єднання та згортки

Описані шари організовані у відповідності до оригінальної архітектури U-Net, при чому кількість фільтрів у першій частині моделі зростає після кожного проходження шарів згортки та агрегування в два рази до першого шару оберненої згортки. Далі кількість фільтрів зменшується після кожного проходження блоку оберненої згортки, операції об’єднання та згортки також в два рази. Результатом даного методу є модель, створена за допомогою класу

Model з tensorflow.keras.models, який об’єднує усі шари в об’єкт з функціями навчання та виводу.

Також, в класі створено метод load_model, який дає змогу завантажити вже існуючу модель для подальшого її використання для створення маски зображення з виявленими вирубками та донавчання. В якості параметрів даний метод приймає шлях до файлу збереженої моделі та власні функції чи об’єкти, які були задіяні при навчанні, наприклад, метрика, що не є стандартною.

```
def load_model(self, model_path, custom_objects):
    model = tf.keras.models.load_model(model_path,
                                       custom_objects=custom_objects, compile=False)

    return model
```

Рисунок 3.4. Метод load_model класу U_net_model

3.2. Створення масок для зображень

У попередньому розділі було зазначено, що за допомогою онлайн інструменту VGG Image Annotator було створено анотації для кожного супутникового зображення з навчального набору. На рисунку 3.5 показано, що вирубки розмічені полігонами за допомогою даного інструменту. Розмітивши усі зображення полігонами, потрібно експортувати проект у файл з розширенням json.

Рисунок 3.5. Приклад розміченого зображення у VGG Image Annotator

Наступним кроком для створення масок для зображень є встановлення бібліотеки dh_segment і імпортування у файлі binary_mask з неї модуля via для роботи з розмітками, створеними за допомогою VGG Image Annotator. Визначивши в коді шлях до файлу з анотаціями (annotation_file), шлях до зображень (images_dir) та до папки, де будуть збережені створені маски, завантажуємо усі дані з файлу анотацій та самі анотації (рис 3.6).

```
via_data = via.load_annotation_data(annotation_file)
via_annotations = via.load_annotation_data(annotation_file, only_img_annotations=True)
```

Рисунок 3.6. Завантаження даних з файлу анотацій

Далі збираємо анотовані регіони, атрибути та параметри, і створюємо маски для кожного зображення за допомогою функції create_masks (рис.3.7). Після запуску даного коду, отримуємо маски до попередньо розмічених даних для кожного зображення.

```
working_items = via.collect_working_items(via_annotations, collection, images_dir)
if '_via_attributes' in via_data.keys():
    list_attributes = via.parse_via_attributes(via_data['_via_attributes'])
else:
    list_attributes = via.get_via_attributes(via_annotations)

via.create_masks(masks_dir, working_items, list_attributes, collection)
```

Рисунок 3.7. Збір атрибутів і параметрів та створення масок

3.3. Реалізація класу Data

Підготовка даних – друга на рівні з моделлю частина для створення ефективної нейронної мережі. Для того, щоб використовувати зображення із супутника та відповідні їм маски, ці дані треба певним чином підготувати, а саме: розділити зображення на тренувальні, валідаційні та тестові, здійснити їх змінення розміру, провести аугментацію та зібрати дані у формат датасета tensorflow. Спочатку, розглянемо, які саме аргументи потребує конструктор класу Data (рис 3.8) :

- size та img_channels – розмір (висота та ширина), до якого буде приведено зображення та кількість його каналів;
- imgs_path та masks_path – шляхи до папки із зображеннями та масками;

```
class Data:

    def __init__(self, size, img_channels, imgs_path, masks_path,
                 valid_split, test_split, random_state):
        self.size = size
        self.img_channels = img_channels
        self.imgs_path = imgs_path
        self.masks_path = masks_path
        self.valid_split = valid_split
        self.test_split = test_split
        self.random_state = random_state
```

Рисунок 3.8. Конструктор класу Data

- valid_split та test_split – десяткові числа, які вказують відсоток від усієї кількості зображень, який піде на валідаційну та тестову вибірки відповідно;
- random_state – ціле значення, яке керує перемішуванням, яке застосовується до зображень перед розподілом на тренувальну, валідаційну та тестову вибірки.

3.3.1. Поділ на вибірки

Як було зазначено у другому розділі, поділ на три вибірки потрібен для того, щоб навчати модель на тренувальних даних, перевіряти в ході навчання мережу на ознаку перенавчання за допомогою валідаційної вибірки та перевіряти вже остаточно навчену модель на тестовій вибірці. Розділення зображень на вибірки відбувається за допомогою методу create_sets класу Data, який приймає аргумент mode, що визначає які саме вибірки будуть повертатися: усі або тільки тестова. За допомогою визначених раніше шляхів, завантажуємо зображення та маски і використовуючи функцію train_test_split з бібліотеки scikit-learn ділим зображення на вибірки: спочатку відділяємо від усього набору даних валідаційну вибірку, а наступним кроком – тестову. При цьому важливим параметром при поділі даних є перемішування (random_state) , так як дані були зібрані послідовно з різних джерел, а перемішування забезпечує потрапляння зображень з різних джерел в кожну з вибірок, що в подальшому дозволяє моделі при навчанні не налаштовувати свої параметри лише для знімків одного виду. В результаті роботи методу отримаємо три вибірки, що не мають однакових зображень.

```
train_x, valid_x, train_y, valid_y = train_test_split(images, masks,
                    test_size=valid_size, random_state=self.random_state)
train_x, test_x, train_y, test_y = train_test_split(train_x, train_y,
                    test_size=test_size, random_state=self.random_state)
```

Рисунок 3.9. Поділ на вибірки

3.3.2. Створення датасету tensorflow

Наступним кроком при обробці даних після розділення на вибірки є організація зображень у датасет tensorflow для подачі на навчання. Раніше було зазначено, що для подачі зображень на вхід моделі спочатку треба змінити їх розмір та провести аугментацію. Аугментація – це збільшення вибірки навчальних даних за допомогою певних модифікацій вже існуючих даних. Дана методика дозволяє просто та швидко розширити малу навчальну вибірку, за рахунок чого досягається покращення якості моделі та підвищення її стійкості до різних деформацій та шумів. Організація даних у датасет tensorflow проводиться за допомогою методів класу Data train_tf_dataset та valid_tf_dataset для навчальних та валідаційних вибірок відповідно. На рисунку 3.11 приведено код, в якому описується метод tf_parse для завантаження зображень та зміни їх розміру, використовуючи методи prepare_image та prepare_mask. На рисунку 3.10 наведено код методу prepare_image – спочатку зображення зчитується за допомогою функції imread з бібліотеки opencv-python. Наступним кроком, так як зображення, яке поступає на вхід мережі, має бути визначеного розміру, кожне зображення змінюється до цього раніше зазначеного при створенні об’єкта класу Data розміру. Зміна розміру до визначеного дозволяє проводити передбачення для будь-якого розміру вихідного зображення. Кожне значення пікселя на зображенні лежить в діапазоні від 0 до 255, які для прискорення навчання нормалізуються за допомогою ділення кожного значення на 256.

```
def prepare_image(self, path):
    path = path.decode()
    x = cv2.imread(path, cv2.IMREAD_COLOR)
    x = cv2.resize(x, (self.img_height, self.img_width))
    x = x / 256.0
    return x
```

Рисунок 3.10. Метод prepare_image класу Data

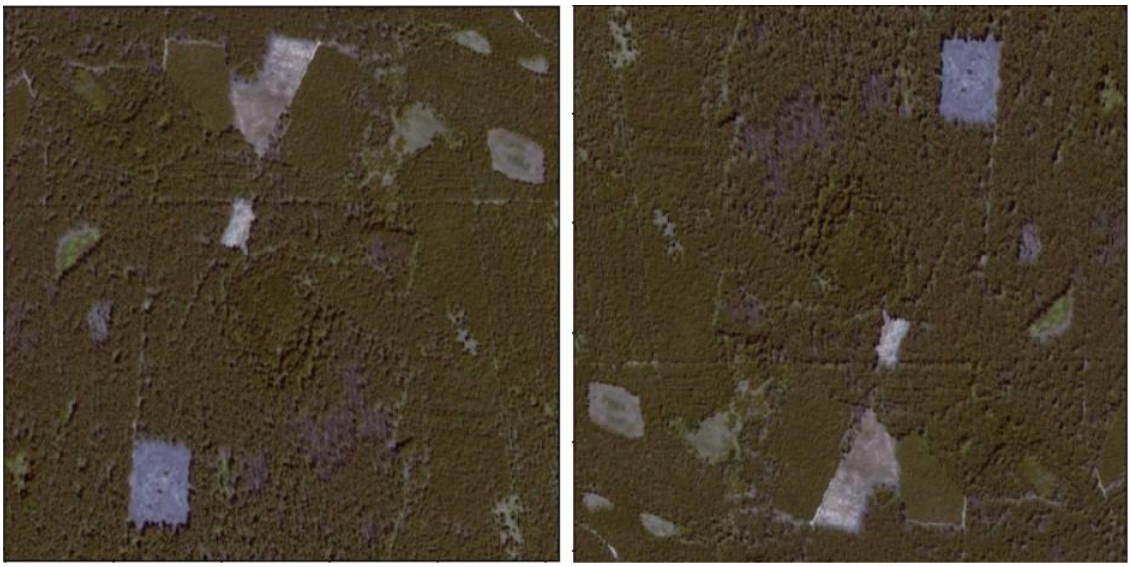
```
def tf_parse(self, x, y):

    def _parse(x, y):
        '''A transformation function to preprocess raw data
                into trainable input. '''
        x = self.prepare_image(x)
        y = self.prepare_mask(y)
        return x, y

    x, y = tf.numpy_function(_parse, [x, y], [tf.float64, tf.float64])
    x.set_shape([self.img_height, self.img_width, self.img_channels])
    y.set_shape([self.img_height, self.img_width, 1])
    return x, y
```

Рисунок 3.11. Метод tf_parse класу Data

Далі проводиться аугментація – попередньо було створено три копії навчальної вибірки: перша залишається без змін, тобто зображення, які були зібрані у чистому вигляді увійдуть у датасет tensorflow для навчання; до другої копії буде застосовано аугментацію у вигляді повороту (рис.3.12), а до третьої – змінення яскравості зображення (рис.3.14). Аугментація проводиться за допомогою функцій з модуля image. Приклад реалізації повороту зображення з навчального набору представлено на рисунку 3.13.

```
def flip(self, x, y):
    x = tf.image.flip_left_right(x)
    x = tf.image.flip_up_down(x)
    y = tf.image.flip_left_right(y)
    y = tf.image.flip_up_down(y)
    return x,y
```

Рисунок 3.12. Метод для реалізації повороту зображення



а) б)

Рисунок 3.13. Приклад здійснення повороту зображення

```
def brightness(self, x, y):
    x = tf.image.stateless_random_brightness(
        x, max_delta=0.2, seed=(0,3))
    return x, y
```

Рисунок 3.14. Метод для зміни яскравості зображення

Останнім кроком дані копії поєднуються в один датасет та поділяються на партії (batch). Валідаційна вибірка у свою чергу теж організовується у датасет tensorflow, але вже без аугментації.

3.4. Навчання

Навчання буде проводитися за допомогою створеного класу Training, що приймає такі агрументи:

- model – створена модель за допомогою методу create_model класу U_net_model;
- data – об’єкт класу Data;
- loss – фукнція витрат;
- batch_size – розмір партії;
- epochs – кількість епох навчання;
- optimizer – оптимізатор;
- save_path – файл, у який буде збережно навчену модель.

Розглянемо усі агрументи, що передаються для створення об’єкту класу Training детальніше.

3.4.1. Параметри

Для того щоб створити об’єкт класу Training, треба визначити параметри для навчання, створити описану вище модель та завантажити дані. Так як у моделі присутні чотири операції зменшення розміру зображення та подальшого його збільшення, то розмір зображення, що подається на перший шар мережі повинен, має бути кратним 16 для коректного навчання. Було вирішено обрати розмір зображень рівним 512 на 512, кількість каналів при цьому дорівнює 3, так як супутникове зображення представлено у RGB. Відсоток валідаційної та тестової вибірок зазвичай лежить в межах від 10 до

20, тому обираємо 20 відсотків і перемішування зі значеннням 40. Дані параметри використовуються при створенні об’єкта класу Data.

Для створення моделі потрібно створити обєкт класу U_net_model і викликати його метод create_model, перед цим визначивши, яку саме кількість фільтрів та функції активації ми будемо застосовувати. В якості функції активації у шарах згортки було обрано ReLu, а в останньому шарі згортки - sigmoid. В оригінальній U-Net кількість фільтрів починається з 32 у першому шарі згортки. При такій початковій кількості фільтрів, загальна кількість параметрів, що тренується буде досягати близько 7,5 мільйонів. Але так як у нас відносно невеликий навчальний набір, доцільніше зменшити цю кількість фільтрів, що дозволить отримати кращий результат на існуючих даних. Тому початкова кількість фільтрів була обрана у розмірі 16, і яка буде збільшуватися в частині енкодера. Відповідно у частині декодера кількість фільтрів буде зменшуватися. З такою кількістю фільтрів кількість параметрів, що навчаються, буде досягати 2 мільйонів.

Наступним кроком необхідно визначити такі параметри навчання як: функцію витрат, оптимізатор, швидкість навчання, розмір партії та кількість епох навчання:

- Функція витрат – це міра того, наскількі добре модель передбачає очікуваний результат. Під час навчання ми намагаємося навчити параметри моделі таким чином, щоб витрати були мінімальними. Було вирішено використовувати binary cross entropy як функію витрат, так як по суті сегментація в даній задачі – це класифікація кожного пікселя до одного з двох класів.
- Оптимізатор – алгоритм, який використовується для зміни параметрів (ваг) моделі в ході навчання для зменшення витрат. На даний момент існує велика кількість оптимізаторів, а тому доцільно провести навчання з декількома з них і виявити найращий результат. Тому будуть проведені навчання з такими оптимізаторами: SGD, RMSprop, Adam.
- Швидкість навчання – значення, яке дозволяє керувати величиною зміни параметрів при навчанні. Данне значення обирається між 0 та 1.

Швидкість навчання підбирається експерементально для кожної зміни параметрів моделі, при чому чим більша швидкість, тим швидше модель буде навчатися, але при великих значеннях швидкості точність налаштування моделі на мінімум функції витрат може знизитися, що призведе до збільшення витрат. І навпаки - якщо значення шкидкості мале, для навчання знадобиться велика кількість кроків, але збільшується точність налаштування алгоритму на мінімум витрат.

- Розмір партії та кількість епох. Розмір партії – це значення, яке визначає кількість зразків, які пройдуть через мережу за одну ітерацію. Кількість ітерацій визначається діленням усього набору даних на розмір партії. Зазвичай, розмір партії обирають меншим ніж уся кількість даних, оскільки це дозволяє пришвидшити навчання та зменшити використання памяті. Кількість епох – кількість прямих та зворотніх проходів усіх навчальних даних. З кожною епохою мережа все краще налаштовується під навчальні дані. При виборі значення кількості епох слід враховувати явища недонавчання (при занадто малій кількості епох) та перенавчання (при великій кількості епох). Розмір партії та кількість епох, як і швидкість навчання, підбираються експерементально.

3.4.2. Метрики

Для оцінки результів навчання моделей використаємо такі стандартні метрики з keras як: loss, accurancy, recall, precision та dice coefficient, написаний власноруч. Коефіцієнт Дайса – це метрика, яка характеризує степінь перекриття сегментів, отриманих в результаті роботи мережі, та сегментів на розмічених навчальних даних. Рахується даний коефіцієнт за такою формулою:

$$dice\_coeff = \frac{2*(A \cap B)}{A+B}, \quad (3.1)$$

де A – множина пікселів сегментів на розмічених масках, B – множина пікселів сегментів, отриманих в результаті роботи мережі. При повному збігу сегментів коефіцієнт Дайса буде дорівнювати 1. На рисунку 3.15 зображено метод класу

Training для розрахунку даного коефіцієнту. Параметр smooth потрібен, щоб уникнути ділення на 0.

```
def dice_coef(y_true, y_pred, smooth = 1.):
    intersection = tf.keras.backend.sum(y_true * y_pred)
    sum = tf.keras.backend.sum(y_true) + tf.keras.backend.sum(y_pred)
    return (2. * intersection + smooth) / (sum + smooth)
```

Рисунок 3.15. Метод для розрахунку коефіцієнта Дайса

3.4.3. Тренування моделей

Отже, перейдемо до тренування моделей з різними параметрами. Навчання проводиться на графічному процесорі GPU відеокарти NVIDIA GeForce RTX 2060 із 6 Гб оперативної пам'яті.

Створивши об'єкт класу Training, викликаємо його метод train, в якому за допомогою методу prepare_data створюються навчальний, валідаційний та тестовий датасети tensorflow. Далі, передана при створенні об'єкта класу Training модель компілюється з обраними параметрами та запускається навчання за допомогою методу fit, в який передаються датасети, кількість епох та кількість кроків всередині епохи. Використовуючи tensorboard, можна слідкувати за процесом навчання. В результаті метода отримуємо показники моделі на тестовій виборці та її час навчання.

```
self.model.fit(train_dataset,
               validation_data=valid_dataset,
               epochs=self.epochs,
               steps_per_epoch=train_steps,
               validation_steps=valid_steps,
               callbacks=tensorboard_callback)
```

Рисунок 3.16. Навчання моделі

В таблиці 3.1 наведено результати тренування моделей та їх показники на тестовій вибірці. В кожному навчанні кількість епох дорівнювала 20, розмір партії – 4 і використовувалась функція витрат binary cross entropy. Швидкість навчання підбиралася експериментально для кожного оптимізатора для отримання найкращого результату.

Таблиця 3.1. Результати навчання моделей із різними параметрами

Параметри	Optimizer = SGD Lr = 0.05	Optimizer = Adam Lr = 0.0003	Optimizer = RMSprop Lr = 0.0003
Loss	0.1597	0.1397	0.2087
Accuracy	0.8843	0.8863	0.8868
Recall	0.5199	0.7748	0.7263
Precision	0.7022	0.7954	0.7888
Dice coefficient	0.4513	0.6624	0.6721
Час навчання	1680.25	1674.69	1710.16

Проаналізувавши таблицю отриманих показників моделей із різними оптимізаторами та швидкостями навчання та порівнявши їх метрики бачимо, що усі моделі показали досить гарні результати на тестовій вибірці. Модель з оптимізатором Adam має найкращі показники, порівняно з RMSprop і SGD оптимізаторами, так як у RMSprop витрати більше на 6%, а у SGD показник dice coefficient менше на 21%. Отже, вирішено обрати модель, яка навчалася з оптимізатором Adam та швидкістю навчання 0.0003, для подальшого її вбудовування у систему виявлення вирубок, так як вона має найменший час навчання і при цьому має найкращі показники метрик. Нижче наведено графіки змін в ході навчання таких метрик як loss та dice coefficient обраної моделі.

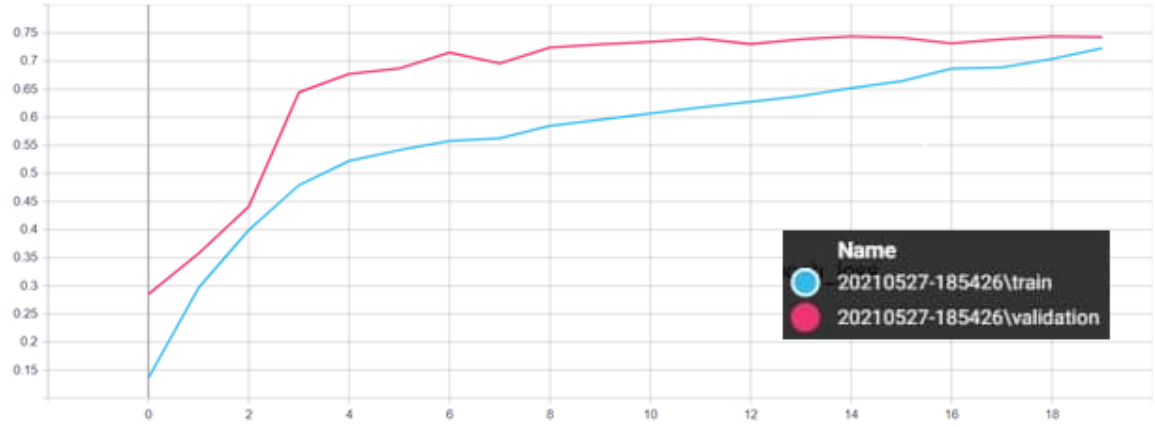


Рисунок 3.17. Графік зміни коефіцієнту Дайса

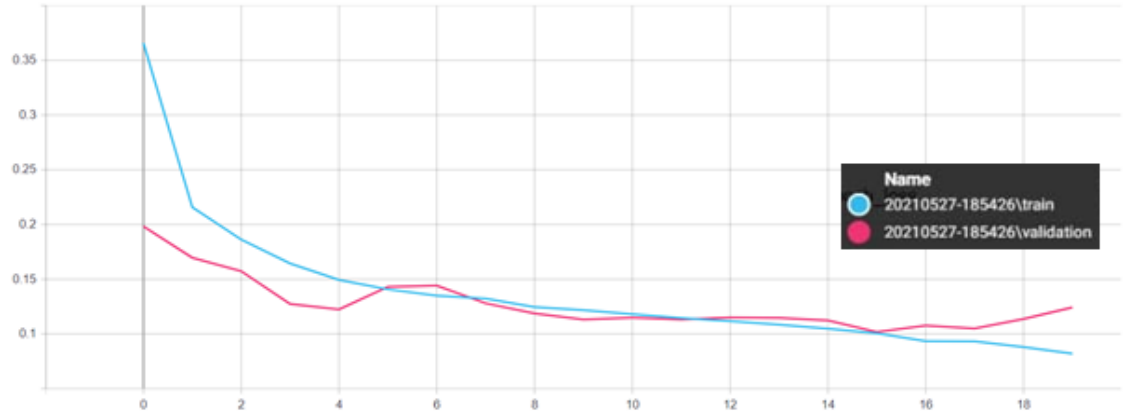


Рисунок 3.18. Графік зміни витрат

3.5. Розробка скрипту для роботи з моделлю

Для вбудовування розробленої моделі у систему відслідковування вирубок і наступного її використання, розроблено клас Prediction, який при створенні об'єкта приймає аргументи model – об'єкт класу U_net_model, model_path – шлях до файлу із збереженою навченою моделлю, image_path – шлях до зображення, для якого потрібно створити маску та save_res_path – шлях, за яким буде збережено результат сегментації. Для початку роботи викликається метод predict, де за допомогою методу load_model завантажується навчена модель по визначеному шляху model_path. Так як було використано власну метрику dice_coeff, вона також вказується при завантаженні моделі. Далі, зображення зчитується по заданому шляху і приводиться до розміру для подачі на вхід мережі за допомогою функції prepare_image. Наступним кроком викликається метод predict завантаженої моделі для проходження зображення через мережу і проведення сегментації. Отриманий результат, тобто маска, зберігається у форматі .png.

3.6. Покращення роботи моделі

Після вбудовування розробленого модуля у систему виявлення вирубок, слід передбачити можливість покращення роботи даного модуля. В ході роботи із системою, база супутникових зображень із вирубками буде поповнюватися і ці зображення можна використати для донавчання моделі. Звичайно, донавчати мережу у такому випадку слід не на кожному новому

зображенні, а після збору певної кількості супутникових зображень, наприклад, такої ж як і при початковому навчанні.

Для донавчання моделі нові супутникові зображення повинні бути розмічені для створення відповідних масок. Це можна зробити двома способами: наприклад, так як VGG Image Annotator дозволяє його використання в наукових та комерційних цілях, даний інструмент можна вбудувати у систему та давати змогу користувачам розмітити зображення самостійно для покращення роботи системи та модуля при, наприклад, отриманні поганої маски сегментації або ж розмітити ці зображення самостійно. Після збереження файлу з розміченими вирубками, файл і зображення використовуються для створення чорно-білих масок за допомогою скрипту з файлу binary_mask.

Отримавши маски та зображення, можна проводити донавчання моделі. Донавчання організовано у вигляді класу Ad_training із наслідування класу Training, так як дані класи мають спільні атрибути та метод dice_coef. Майже усі атрибути даного класу збігаються з атрибутами класу Training, окрім двох доданих:

```
class Ad_training(Training):

    def __init__(self, model, model_path, data, new_data, loss, batch_size, epochs, optimizer, save_path):
        super().__init__(model, data, loss, batch_size, epochs, optimizer, save_path)
        self.new_data = new_data
        self.model_path = model_path
```

Рисунок 3.19. Конструктор класу Ad_training

- new_data – об’єкт класу Data з новими даними для донавчання;
- model_path – шлях до існуючої навченої моделі.

Донавчання проводиться за допомогою методу train: спочатку дані поділяються на навчальну, валідаційну та тестову вибірки, при чому в тестову вибірку входять не лише нові зображення, але і ті, на яких проводилося тестування вже існуючої навченої моделі для того, щоб оцінити роботу донавченої моделі і на попередніх даних. Це дасть змогу об’єктивно порівняти вже існуючу модель та донавчену. Далі існуюча модель завантажується за допомогою функції load_model об’єкту model, який був переданий у

конструктор класу та компілюється з обраними оптимізатором, швидкістю навчання та функцією витрат. Метрики у донавчанні залишаються тими самими, що і при навчанні початкової моделі для подальшого їх порівняння. Застосовуючи метод fit, донавчаємо модель та порівнюємо отримані метрики: якщо показники покращилися, існуюча початкова модель може бути замінена отриманою моделлю в ході донавчання. Варто зауважити, що так як система для виявлення вирубок працює постійно, донавчання слід виконувати не припиняючи роботи існуючої моделі, і тільки отримавши модель із кращими показниками, її слід замінити.

ВИСНОВОК ДО РОЗДІЛУ 3

У даному розділі було розроблено модуль для розпізнавання вирубок на супутникових знімках, а саме такі якого частини:

- реалізовано архітектуру згорткової нейронної мережі U-Net для сегментації та визначено такі її параметри як функції активації, кількість фільтрів та розміри вхідних та результуючих зображень,
- виконано підготовка даних для подачі на вхід мережі – поділ на вибірки, змінення розміру та аугментацію,
- проведено навчання із трьома різними оптимізаторами з різними швидкостями навчання для визначення найкращого з них для даної задачі;
- реалізовано клас для взаємодії з модулем у системи виявлення вирубок для отримання маски сегментації, а також додано можливість донавчання моделі для покращення її якості розпізнавання.

РОЗДІЛ 4. ТЕСТУВАННЯ МОДУЛЯ

4.1. Огляд роботи модуля

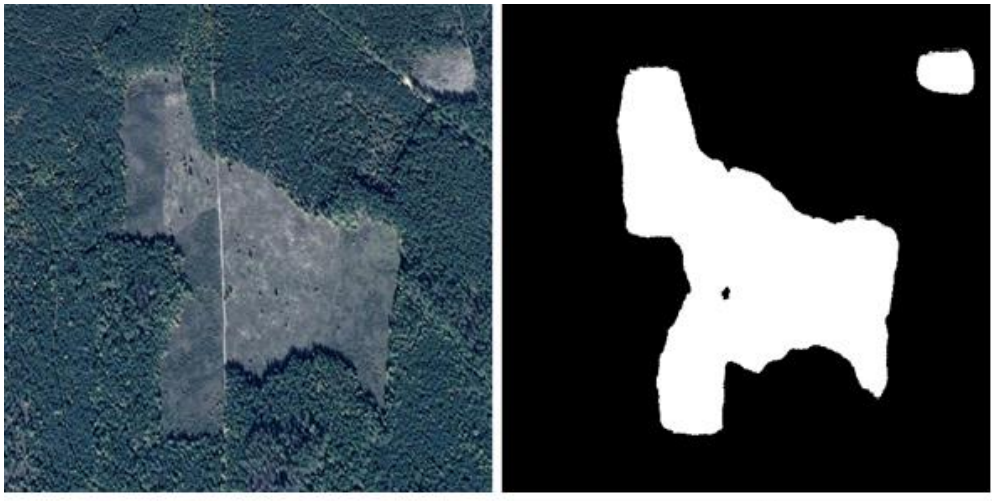
У попередньому розділі було розглянуто моделі з різними параметрами та обрано найкращу з них та написано клас для взаємодії із вже існуючою навченою моделлю. У даному розділі буде проведено тестування модуля на супутникових зображеннях різного виду для оцінки роботи модуля. Усі зображення, на яких перевіряється робота модуля, не входили у навчальний набір.

Для оцінки роботи розробленого модуля були виконані наступні експерименти:

1. Перевірка роботи модуля на зображеннях з різним виглядом вирубок.
2. Перевірка роботи модуля на зображеннях з різним масштабом.
3. Перевірка роботи модуля на зображеннях з наявністю сторонніх об’єктів.

4.1.1. Огляд роботи модуля на зображеннях з різними виглядами вирубок

Для здійснення перевірки роботи модуля на зображеннях з різними виглядами вирубок, наприклад, по формі, кольору або розміру, дані були зібрані з таких сервісів як: Google Earth, Google Maps та ArcGIS Online. На рисунках, які знаходяться нижче, представлені початкові зображення, що подавалося на вхід мережі та результати сегментації для кожного з них.

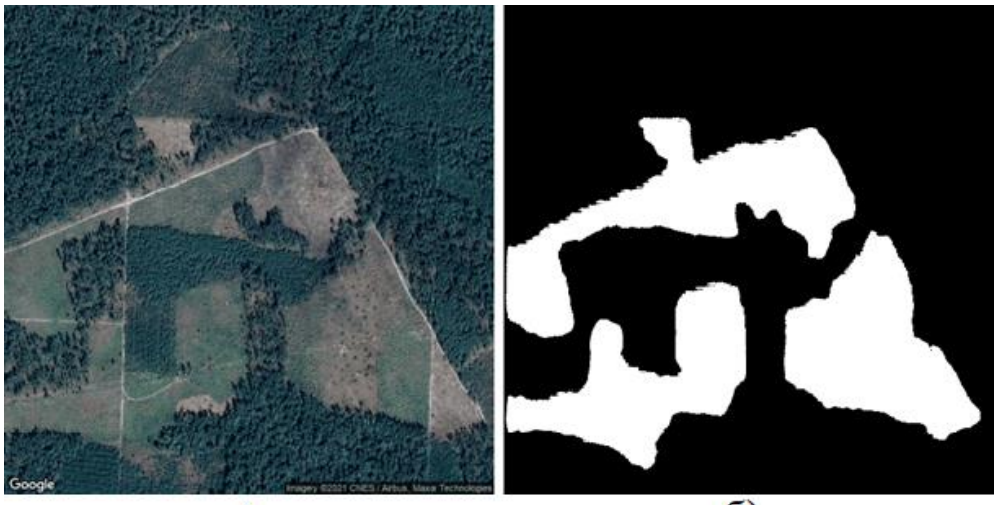


а) б)

Рисунок 4.1. а) Зображення з Google Earth, б) Результат сегментації

а) б)

Рисунок 4.2. а) Зображення з ArcGIS Online, б) Результат сегментації



а) б)

Рисунок 4.3. а) Зображення з Google Maps, б) Результат сегментації

а) б)

Рисунок 4.4. а) Зображення з Google Maps, б) Результат сегментації

Проаналізувавши результати бачимо, що модуль видає прийнятні

результати на знімках з різними видами вирубок, що вказує на те, що модуль можна використовувати для різних видів зображень. Найкращим чином модуль обробляє зображення з вирубками, що мають чіткі межі та що виникли нещодавно, тому на рисунку 4.4 можемо бачити , що модуль погано розпізнав частину вирубки, що частково поросла.

4.1.2. Огляд роботи модуля на зображеннях з різним масштабом

Для даного огляду було зібрано декілька знімків вирубок з різним масштабом, а отже розмір однієї і тієї ж вирубки змінювався в ході зменшення масштабу карти. Після проходу даних знімків через мережу, маємо такі результати:

Рисунок 4.5. а) Вихідне зображення, б) Результат сегментації

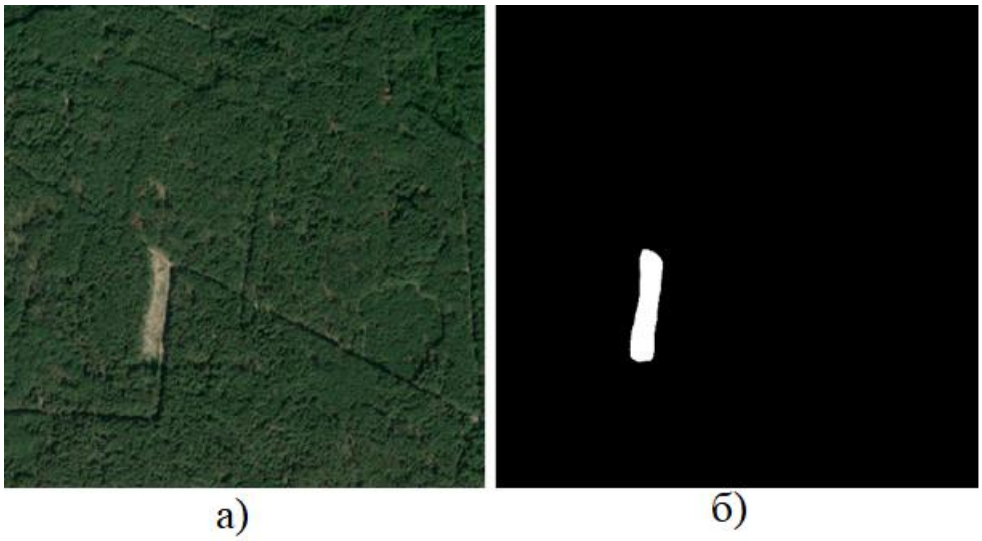


Рисунок 4.6. а) Вихідне зображення, б) Результат сегментації

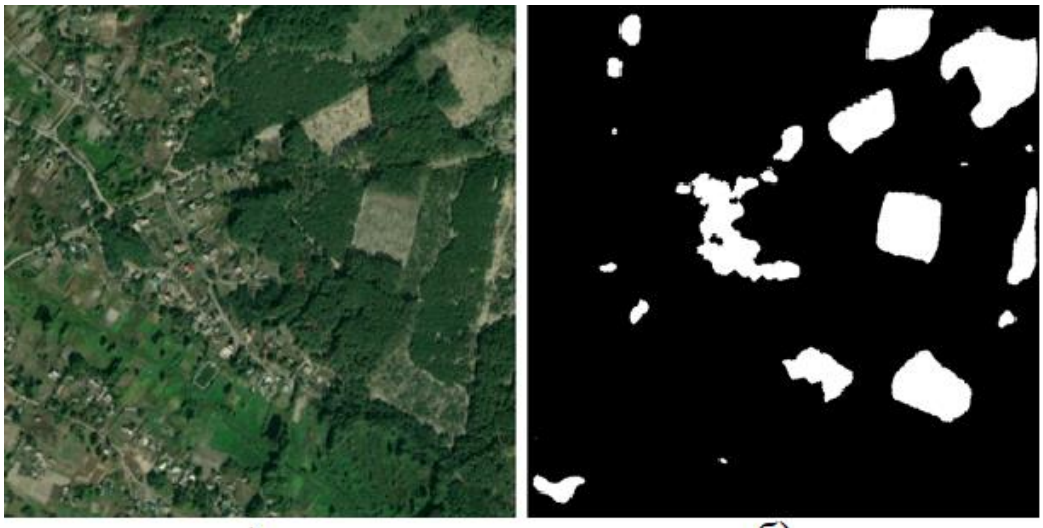
а) б)

Рисунок 4.7. а) Вихідне зображення, б) Результат сегментації

В результаті роботи модуля бачимо, що розпізнавання вирубок на супутникових зображеннях з різним масштабом пройшло досить успішно, але треба зауважити, що так як модель навчалася на знімках, зроблених з висоти, приблизно як на рисунку 4.6, сегментація на таких зображеннях буде проходити краще.

4.1.3. Огляд роботи модуля на зображеннях з наявністю сторонніх об’єктів

Даний огляд проводиться на зображеннях, на яких присутні не тільки вирубки, а й дороги та території селищ. Результат роботи на таких зображеннях представлений нижче.



а) б)

Рисунок 4.8. а) Зображення з частиною селища, б) Результат сегментації

а) б)

Рисунок 4.9. а) Зображення з частиною дороги, б) Результат сегментації

Бачимо, що наявність сторонніх об’єктів на зображеннях зменшує якість розпізнавання. Це викликано тим, що зображень із такими об’єктами у навчальному наборі недостатньо для отримання якісного розпізнавання на таких знімках. При розширення навчального набору, включаючи в нього різноманітні зображення з присутністю сторонніх об’єктів та вирубок, якість розпізнавання збільшиться.

ВИСНОВОК ДО РОЗДІЛУ 4

У даному розділі було представлено тестування розробленого модуля для розпізнавання вирубок. Для оцінки роботи модуля було проведено такі експерименти: робота модуля на зображеннях з різним виглядом вирубок, на зображеннях із різним масштабом та на зображеннях, на яких присутні сторонні об’єкті – селища або дороги. В результаті проведення даних експериментів виявлено, що:

- розроблений модуль здатний розпізнавати вирубки, що мають різні форми, кольори та текстури, при цьому найкращі результати отримуються на зображеннях з вирубками, що мають чіткі межі та що виникли нещодавно. Також модуль має змогу розпізнавати вирубки на зображеннях з різним масштабом;
- при роботі із зображеннями, на яких присутні сторонні об’єкти, такі як частини міст або селищ, дороги та поля, та із зображеннями, на яких вирубки за час певним чином видозмінилися (наприклад, поросли) модуль видає гірші результати. Даний недолік пов'язаний із невеликим розміром навчального набору та вирішується його збільшенням, включаючи в себе велику кількість різноманітних видів та розташувань вирубок.

ВИСНОВКИ

У даній бакалаврській роботі були розглянуті системи для детектування різних об’єктів на супутникових знімках та розглянуто методи для здійснення детектування вирубок лісу. Було обрано та аргументовано оптимальний метод детектування вирубок на супутникових зображеннях, а саме за допомогою згорткових нейронних мереж. Також було визначено вид результату після проходження зображення через мережу, а саме семантичная сегментація, та яка архітектура найкращим чином підходить під поставлену задачу.

Був розроблений модуль розпізнавання вирубок на супутникових зображень, заснований на згортковій нейронній мережі U-Net, який в подальшому буде включений у систему виявлення вирубок лісу. Даний модуль був розроблений за допомогою мови програмування Python і бібліотек Tensorflow і Keras. У ході тестування розробленого модуля було виявлено, що модуль здатний розпізнавати вирубки різних форм, кольорів та розмірів, а при збільшенні навчального набору, якість розпізнавання на зображеннях із сторонніми об’єктами покращиться.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Баранова Н. Космоснимки: что это такое и как их использовать [Електронний ресурс] / Наталья Баранова. – 2018. – Режим доступу до ресурсу: https://te-st.ru/2018/07/24/space-imagery-how-to-use-for-project/.
2. Our Neural Net Dreams of Cars [Електронний ресурс]. – 2016. – Режим доступу до ресурсу: https://orbitalinsight.com/blog/neural-net-dreams-cars.
3. Система оперативного мониторинга СКАНЭКС, сервис «Карта пожаров» [Електронний ресурс] – Режим доступу до ресурсу: https://fires.ru/help.html.
4. Всемирный лесной дозор [Електронний ресурс] – Режим доступу до ресурсу: https://ru.wikipedia.org/wiki/Всемирный_лесной_дозор.
5. Тартачный А. «Зеленые» алгоритмы [Електронний ресурс] / Александр Тартачный. – 2021. – Режим доступу до ресурсу: https://robotdreams.cc/blog/138-zelenye-algoritmy.
6. OpenCV: HSV и поиск объектов по цвету [Електронний ресурс]. – 2017. – Режим доступу до ресурсу: http://itnotesblog.ru/note.php?id=272.
7. Евсегнеев О. OpenCV на python: цветовой фильтр [Електронний ресурс] / Олег Евсегнеев. – 2017. – Режим доступу до ресурсу: https://robotclass.ru/tutorials/opencv-color-range-filter/.
8. Живрин, Я. Э. Методы определения объектов на изображении / Я. Э. Живрин, Нафе Башар Алкзир. // Молодой ученый. — 2018. — № 7 (193). — С. 8-19. — URL: https://moluch.ru/archive/193/48447/.
9. Борисов Е. О задаче поиска объекта на изображении. Часть 1: Базовые методы. [Електронний ресурс] / Евгений Борисов. – 2017. – Режим доступу до ресурсу: http://mechanoid.su/cv-image-detector.html.
10. Нейронные сети: распознавание образов и изображений с помощью ИИ [Електронний ресурс] – Режим доступу до ресурсу: https://center2m.ru/ai-recognition.
11. Наумов Н. Метод Виолы-Джонса (Viola-Jones) как основа для распознавания лиц [Електронний ресурс] / Николай Наумов. – 2011. – Режим доступу до ресурсу: https://habr.com/ru/post/133826/.

12. Нейронные сети, машинное обучение и искусственный интеллект: в чем разница и для чего их используют [Електронний ресурс]. – 2019. – Режим доступу до ресурсу: https://evergreens.com.ua/ru/articles/machine-learning-overview.html.
13. Полищуков Р. М. Распознавание транспортных средств в потоковом видео с машинным обучением // Актуальные исследования. – 2020. – №8 (11). – С. 20-24. – URL: https://apni.ru/article/607-raspoznavanie-transportnikh-sredstv-v-potokov.
14. Азаров Д. Метод распознавания лиц Виолы-Джонса (Viola-Jones) [Електронний ресурс] / Дмитрий Азаров. – 2015. – Режим доступу до ресурсу: https://oxozle.com/2015/04/11/metod-raspoznavaniya-lic-violy-dzhonsa-viola-jones/.
15. Исаков С. Как работает нейронная сеть: алгоритмы, обучение, функции активации и потери [Електронний ресурс] / Станислав Исаков. – 2018. – Режим доступу до ресурсу: https://neurohive.io/ru/osnovy-data-science/osnovy-nejronnyh-setej-algoritmy-obuchenie-funkcii-aktivacii-i-poteri/.
16. Применение нейросетей в распознавании изображений [Електронний ресурс]. – 2009. – Режим доступу до ресурсу: https://habr.com/ru/post/74326/.
17. Многослойный персептрон [Електронний ресурс] – Режим доступу до ресурсу: http://www.aiportal.ru/articles/neural-networks/multi-perceptron.html.
18. Свёрточная нейронная сеть [Електронний ресурс] – Режим доступу до ресурсу: https://ru.wikipedia.org/wiki/Свёрточная_нейронная_сеть.
19. Deep learning вместе с Caffe и Digits часть 1 [Електронний ресурс] – Режим доступу до ресурсу: https://sidstudio.com.ua/sidstudio-blog/deep-learning-vmeste-s-caffe-i-digits-chast-1.
20. Le J. The 5 Computer Vision Techniques That Will Change How You See The World [Електронний ресурс] / James Le. – 2018. – Режим доступу до

ресурсу: https://heartbeat.fritz.ai/the-5-computer-vision-techniques-that-will-change-how-you-see-the-world-1ee19334354b.

21. Маркова С.В., Жигалов К.Ю. ПРИМЕНЕНИЕ НЕЙРОННОЙ СЕТИ ДЛЯ СОЗДАНИЯ СИСТЕМЫ РАСПОЗНАВАНИЯ ИЗОБРАЖЕНИЙ // Фундаментальные исследования. – 2017. – № 8-1. – С. 60-64.
22. U-Net [Електронний ресурс] – Режим доступу до ресурсу: https://ru.wikipedia.org/wiki/U-Net.
23. Ronneberger O. U-Net: Convolutional Networks for Biomedical Image Segmentation / O. Ronneberger, P. Fischer, T. Brox., 2015. – 8 с.
24. Нейронные сети для начинающих. Часть 1 [Електронний ресурс]. – 2016. – Режим доступу до ресурсу: https://habr.com/ru/post/312450/.
25. How PSPNet works? [Електронний ресурс] – Режим доступу до ресурсу: https://developers.arcgis.com/python/guide/how-pspnet-works/.
26. Kendall A. SegNet [Електронний ресурс] / A. Kendall, V. Badrinarayanan, R. Cipolla – Режим доступу до ресурсу: http://mi.eng.cam.ac.uk/projects/segnet/.
27. Badrinarayanan V. SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation / Vijay Badrinarayanan, Alex Kendall, Roberto Cipolla., 2016. – 14 с.
28. ЛУЧШИЙ ЯЗЫК ДЛЯ ПРОГРАММИРОВАНИЯ ИИ: РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ [Електронний ресурс]. – 2020. – Режим доступу до ресурсу: https://www.zfort.com.ua/blog/luchshii-yazyk-dlya-programmirovaniya-ii-rukovodstvo-polzovatelya.
29. Python [Електронний ресурс] – Режим доступу до ресурсу: https://ru.wikipedia.org/wiki/Python.
30. Tensorflow [Електронний ресурс] – Режим доступу до ресурсу: https://ru.wikipedia.org/wiki/TensorFlow.
31. PyTorch и TensorFlow: отличия и сходства фреймворков [Електронний ресурс]. – 2019. – Режим доступу до ресурсу: https://neurohive.io/ru/tutorial/pytorch-vs-tensorflow/.

32. Glayboroda M. Топ-10 фреймворков для искусственного интеллекта: часть первая [Електронний ресурс] / Marina Glayboroda. – 2019. – Режим доступу до ресурсу: https://vc.ru/ml/80391-top-10-freymvorkov-dlya-iskusstvennogo-intellekta-chast-pervaya.
33. PyTorch [Електронний ресурс]. – Режим доступу до ресурсу: https://ru.wikipedia.org/wiki/PyTorch.
34. Komissarenko N. TensorFlow vs PyTorch [Електронний ресурс] / Nick Komissarenko. – 2020. – Режим доступу до ресурсу: https://medium.com/@bigdataschool/tensorflow-vs-pytorch-b41bbd78f061.
35. Keras [Електронний ресурс] – Режим доступу до ресурсу: https://ru.wikipedia.org/wiki/Keras.
36. Сравните десять фреймворков глубокого обучения: TensorFlow не лучший? [Електронний ресурс] – Режим доступу до ресурсу: https://russianblogs.com/article/3268900917/.
37. Caffe [Електронний ресурс] – Режим доступу до ресурсу: https://ru.wikipedia.org/wiki/Caffe.
38. Кэлер А., Брэдски Г. Изучаем OpenCV 3 / пер. с англ. А. А. Слинкина. - М.: ДМК Пресс, 2017. - 826 с.
39. VGG Image Annotator [Електронний ресурс] – Режим доступу до ресурсу: https://www.robots.ox.ac.uk/~vgg/software/via/.
40. Що можна робити за допомогою ArcGIS Online? [Електронний ресурс] – Режим доступу до ресурсу: http://www.esri.ua/sarticle.php?id=4.
41. Google Планета Земля [Електронний ресурс] – Режим доступу до ресурсу: https://ru.wikipedia.org/wiki/Google_Планета_Земля.

ДОДАТОК А

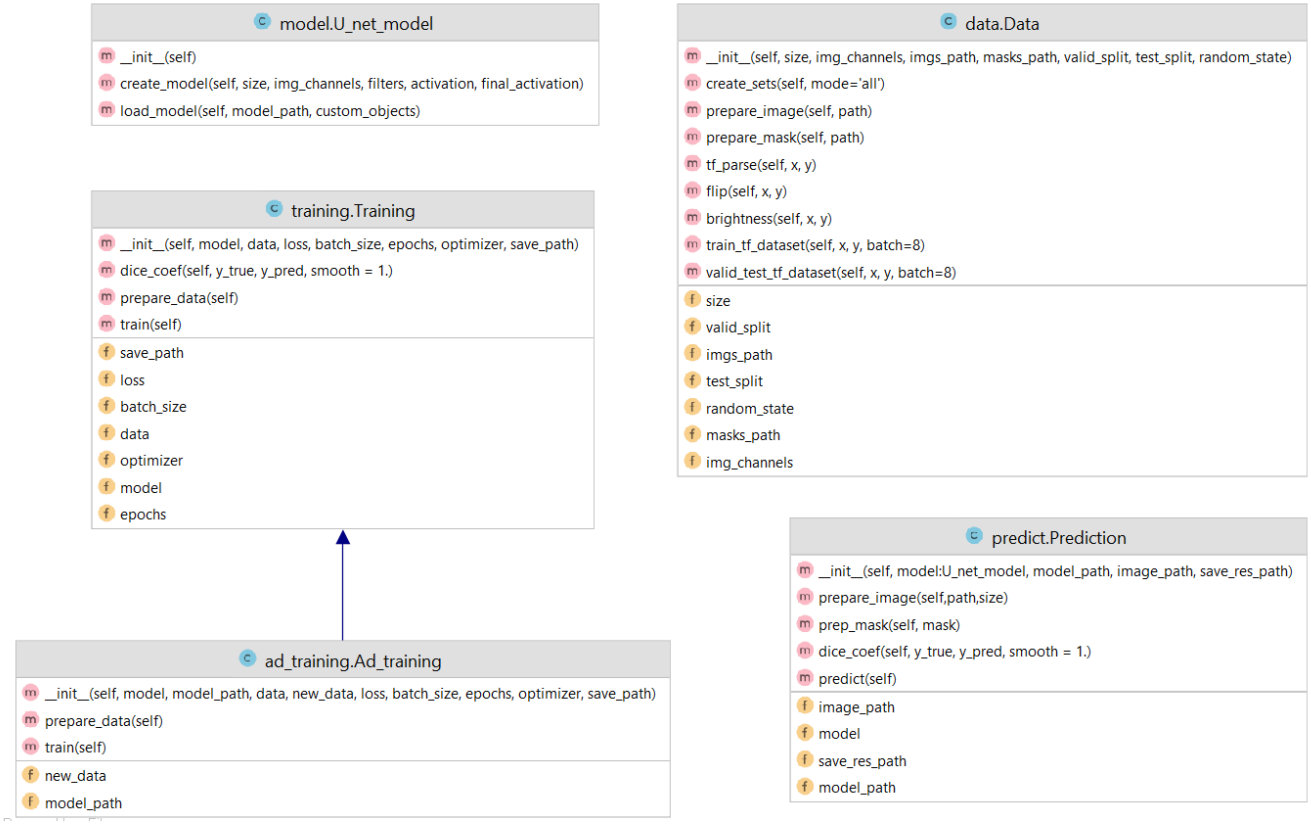
ФУНКЦІОНАЛЬНА СХЕМА

до дипломного проєкту

на тему: «Модуль розпізнавання об'єктів

для системи виявлення вирубок лісу по знімках місцевості»

Київ – 2021 року



					ІАЛЦ.467100.004 Д1		
Зм.		№ документа	Підп.	Дата			
Розробив		Шрам В.С.			Модуль розпізнавання об'єктів для системи виявлення вирубок лісу по знімках місцевості Функціональна схема	Літ.	Аркуш Аркушів
Перевірив		Алєнін О.І.				Т	1 1
						НТУУ «КПІ імені Ігоря Сікорського», ФІОТ Група ІО - 71	
Н.контр.		Сімоненко В.П.					
Затв.		Стіренко С. Г.					

ДОДАТОК Б

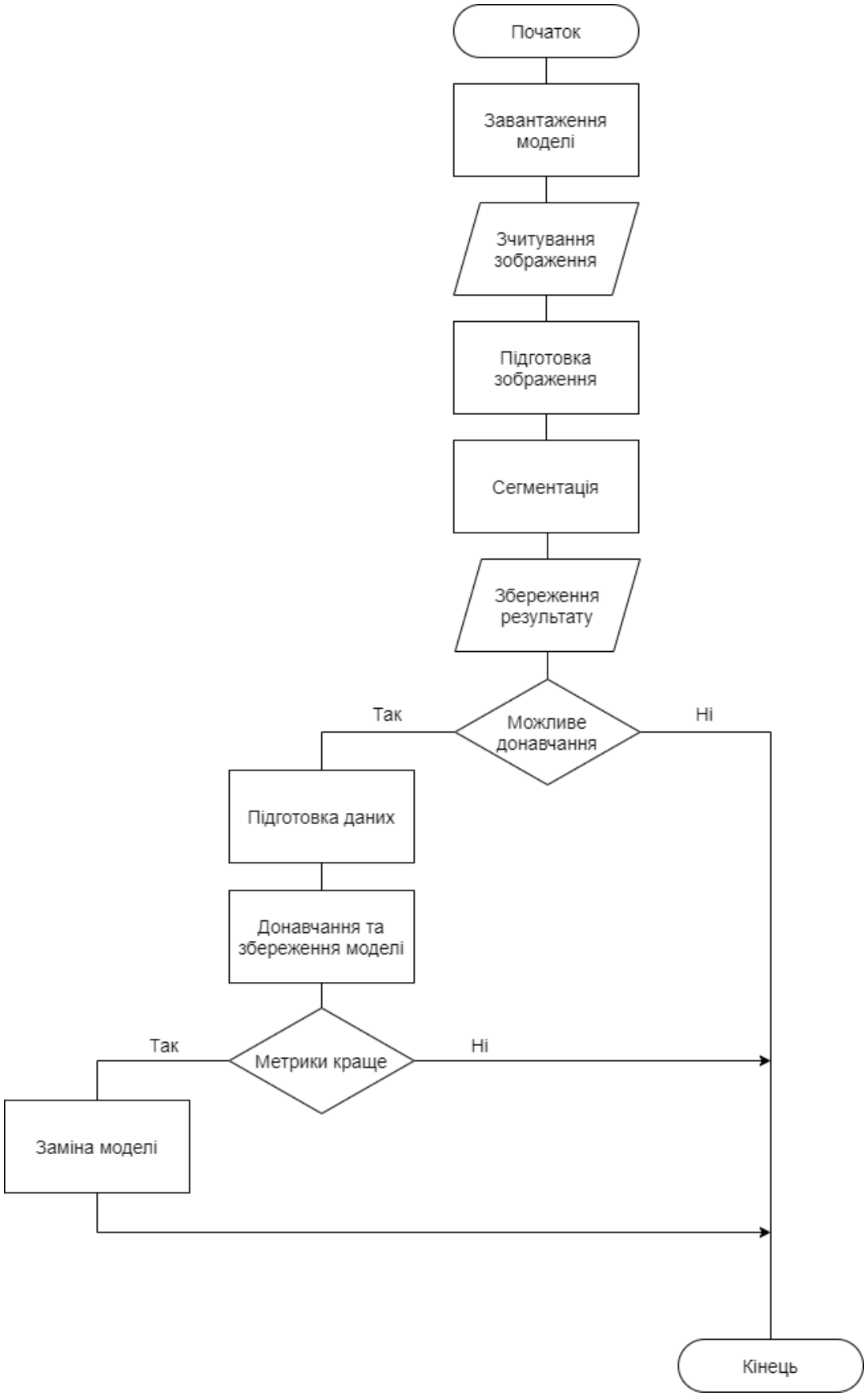
ПРИНЦИПОВА СХЕМА

до дипломного проєкту

на тему: «Модуль розпізнавання об'єктів

для системи виявлення вирубок лісу по знімках місцевості»

Київ – 2021 року



					ІАЛЦ.467100.004 Д2		
Зм.		*№ документа*	*Підп.*	*Дата*			
Розробив		*Шрам В.С.*			*Модуль розпізнавання об'єктів для системи виявлення вирубок лісу по знімках місцевості Принципова схема*	*Літ.*	*Аркуш Аркушів*
Перевірив		*Алєнін О.І.*				Т	1 1
						НТУУ «КПІ імені Ігоря Сікорського», ФІОТ Група ІО - 71	
Н.контр.		*Сімоненко В.П.*					
Затв.		*Стіренко С. Г.*					

ДОДАТОК В

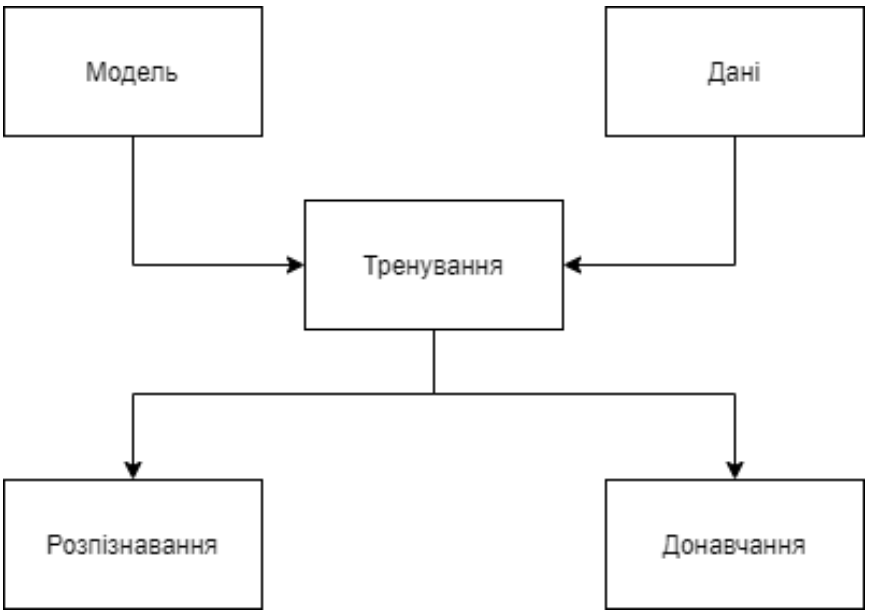
СТРУКТУРНА СХЕМА

до дипломного проєкту

на тему: «Модуль розпізнавання об'єктів

для системи виявлення вирубок лісу по знімках місцевості»

Київ – 2021 року



					ІАЛЦ.467100.004 ДЗ		
Зм.		*№ документа*	*Підп.*	*Дата*			
Розробив		*Шрам В.С.*			*Модуль розпізнавання об'єктів для системи виявлення вирубок лісу по знімках місцевості Структурна схема*	*Літ.*	*Аркуш Аркушів*
Перевірив		*Алєнін О.І.*				Т	1 1
						НТУУ «КПІ імені Ігоря Сікорського», ФІОТ Група ІО - 71	
Н.контр.		*Сімоненко В.П.*					
Затв.		*Стіренко С. Г.*					