МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ
ФАКУЛЬТЕТ НАЗЕМНИХ СПОРУД І АЕРОДРОМІВ
Кафедра аерокосмічної геодезії та землеустрою

ДОПУСТИТИ ДО ЗАХИСТУ
Завідувач кафедри
_____ Юрій ВЕЛИКОДСЬКИЙ
«___» _____2024 р.

КВАЛІФІКАЦІЙНА РОБОТА

(ПОЯСНЮВАЛЬНА ЗАПИСКА)

ЗДОБУВАЧА ОСВІТНЬОГО СТУПЕНЯ «БАКАЛАВР»

Тема: «Використання глибокого навчання для виявлення пошкодженої забудови на космічних знімках»

Виконавець: Козлова Тетяна Дмитрівна, студентка групи ГС 412 Б _____

Керівник: к.ф.-м.н., ст. досл. Великодський Юрій Іванович _____

Нормоконтролер: Іщенко Наталія Федорівна, PhD, доцент _____

Київ 2024

НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ

Факультет наземних споруд і аеродромів

Кафедра аерокосмічної геодезії та землеустрою

Спеціальність 193 «Геодезія та землеустрій»

Освітньо-професійна програма «Геоінформаційні системи і технології»

ЗАТВЕРДЖУЮ

Завідувач кафедри

_____ Юрій ВЕЛИКОДСЬКИЙ

«___» _____2024 р.

ЗАВДАННЯ

на виконання кваліфікаційної роботи

Козловій Тетяні Дмитрівні

1. Тема кваліфікаційної роботи: «Використання глибокого навчання для виявлення пошкодженої забудови на космічних знімках», затверджена наказом ректора від 22.04.2024 року № 601/ст.

2. Термін виконання роботи: з 20 травня 2024 р. по 16 червня 2024 р.

3. Вихідні дані роботи: космічні знімки Maxar з роздільною здатністю 0,5 м, космічні знімки Planet з роздільною здатністю 3,0 м, геопросторові дані будівель міста Маріуполь з OpenStreetMap.

4. Зміст пояснювальної записки: У першому розділі розглянуто теоретичні основи глибокого навчання, включаючи його використання в геоінформаційних системах. У другому розділі описано розробку моделі глибокого навчання для ідентифікації пошкодженої забудови, включаючи збір і обробку навчальних даних та оцінку точності моделі. У третьому розділі описано практичне застосування моделі глибокого навчання в ArcGIS для ідентифікації пошкодженої забудови, проведено аналіз точності шляхом порівняння

фактичних і прогнозованих класифікацій, а також представлено візуалізацію результатів.

5. Перелік обов’язкового ілюстративного матеріалу: 1 таблиця, 20 рисунків, 5 лістингів.

6. Календарний план-графік

№ з/п	Завдання	Термін виконання	Підпис керівника
1.	Визначити тему роботи	20.05.24	
2.	Сформувати зміст роботи	21.05.24 – 24.05.24	
3.	Опрацювати літературні джерела за тематикою роботи	25.05.24 – 28.05.24	
4.	Робота над першим розділом роботи	28.05.24 – 01.06.24	
5.	Робота з тематикою другого розділу роботи	02.06.24– 04.06.24	
6.	Робота над практичною частиною роботи	05.06.24 – 07.06.24	
7.	Формування висновків. Оформлення пояснювальної записки	08.06.24 – 10.06.24	
8.	Підготовка роботи до захисту. Захист роботи	11.06.24 – 16.06.24	

Дата видачі завдання: «20» травня 2024 р.

Керівник кваліфікаційної роботи: _____ Великодський Ю.І.

Завдання прийняв до виконання: _____ Козлова Т.Д.

РЕФЕРАТ

Кваліфікаційна робота на тему: «Використання глибокого навчання для виявлення пошкодженої забудови на космічних знімках»: 59 сторінок, 20 рисунків, 1 таблиця, 5 лістингів, 35 літературних джерел.

Об'єктом дослідження кваліфікаційної роботи є пошкоджена забудова, що знаходиться в зоні бойових дій.

Метою роботи є аналіз можливостей використання глибокого навчання для точного виявлення пошкодженої забудови на космічних знімках. Це включає в себе розробку спеціалізованої моделі глибокого навчання та оцінку її ефективності на основі реальних даних.

Методи дослідження: моделювання; аналіз; опис; класифікація.

Результати бакалаврської роботи рекомендується використовувати для моніторингу інфраструктури, оцінки збитків, а також для швидкого реагування на наслідки воєнних дій.

ГЛИБОКЕ НАВЧАННЯ, МАШИННЕ НАВЧАННЯ, АНАЛІЗ ЗОБРАЖЕНЬ, АНАЛІЗ КОСМІЧНИХ ЗНІМКІВ, КЛАСИФІКАЦІЯ ЗОБРАЖЕНЬ, CNN, TENSORFLOW, KERAS, ARCGIS.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

AI — Штучний інтелект (artificial intelligence);

CNN — Згорткові нейронні мережі (convolutional neural network);

ГН — Глибоке навчання;

OSM — OpenStreetMap;

ГІС — Геоінформаційні системи;

ШНМ — Штучна нейронна мережа.

ВСТУП

Зростаючий інтерес до застосування обчислювальних потужностей у різних сферах життя набуває все більшої популярності. Машинне навчання, яке є важливою складовою сучасних технологій, знаходить дедалі ширше застосування у різних галузях. Особливе місце серед цих алгоритмів займає глибоке навчання, що полягає у тренуванні великих нейронних мереж на значних обсягах даних для виконання складних завдань. Завдяки своїй здатності до автоматичного виділення особливостей та побудови складних моделей, глибоке навчання знаходить широке застосування, від комп'ютерного зору до економічних прогнозів.

Актуальність теми: глибоке навчання стає все більш поширеним інструментом для аналізу зображень, включаючи оцінку супутникових та аерофотознімків. Це важливо для вирішення таких прикладних задач, як моніторинг довкілля, планування міської інфраструктури, управління сільським господарством та ін. Використання глибокого навчання для виявлення пошкодженої забудови на космічних знімках демонструє значну актуальність у галузі геоінформаційних систем (ГІС). У сучасних реаліях нашої країни здатність швидко та точно оцінювати стан забудови є надзвичайно важливою для планування відновлювальних робіт та надання гуманітарної допомоги. Це підкреслює актуальність застосування глибокого навчання для швидкого аналізу даних, створення карт руйнувань та прийняття рішень.

Метою дослідження є аналіз можливостей використання глибокого навчання для виявлення пошкодженої забудови на космічних знімках. Це включає в себе розробку спеціалізованої моделі глибокого навчання та оцінку її ефективності на основі реальних даних.

Для досягнення поставленої мети в кваліфікаційній роботі необхідні розв’язати такі **завдання**:

- розглянути теоретичні основи машинного та глибокого навчання;

- проаналізувати наявні методи глибокого навчання, що застосовуються для аналізу зображень;
- розробити методологію підготовки та попередньої обробки даних для тренування нейронної мережі;
- вибрати оптимальну структуру нейронної мережі для класифікації пошкодженої забудови на основі супутникових зображень;
- провести навчання нейронної мережі на підготовлених даних та оцінити її точність;
- застосувати розроблену модель для аналізу реальних супутникових знімків з метою виявлення пошкодженої забудови частини міста Маріуполя;
- провести порівняння результатів моделі з реальними даними для оцінки її точності на основі побудованої матриці плутанини;
- візуалізувати отримані результати у вигляді тематичної карти.

Об'єкт дослідження: пошкоджена забудова, що знаходиться в зоні бойових дій.

Предмет дослідження: використання глибокого навчання для виявлення пошкодженої забудови на космічних знімках.

Методи дослідження: моделювання – створення моделі глибокого навчання для класифікації пошкодженої забудови на космічних знімках; аналіз – проведення аналізу оцінки точності створеної моделі на різних наборах даних; опис – викладення результатів проведеного дослідження; класифікація – створення класів для чіткої ідентифікації будівель.

Практичне значення отриманих результатів: використання для відновлення інфраструктури, оцінки збитків, а також для швидкого реагування на наслідки воєнних дій.

РОЗДІЛ 1. ТЕОРЕТИЧНІ ОСНОВИ ГЛИБОКОГО НАВЧАННЯ

1.1. Машинне та глибоке навчання

1.1.1. Машинне навчання

Машинне навчання — це галузь штучного інтелекту (AI), що спрямована на створення систем, які навчаються або покращують свою продуктивність на основі введених даних. Штучний інтелект - це загальний термін, який описує системи або машини, що мають здатність схожу на людський інтелект. Хоча терміни "машинне навчання" і "штучний інтелект" часто вживаються поруч, вони не є повністю взаємозамінними. Важливо розуміти, що, хоча всі системи машинного навчання відносяться до штучного інтелекту, не всі аспекти штучного інтелекту є машинним навчанням [1].

Машинне навчання, як область дослідження, що дозволяє комп'ютерам вчитися без явного програмування, становить одну з найцікавіших сучасних технологій. Воно надає комп'ютерам здатність навчання, зробивши їх більш схожими на людину. У сучасний час машинне навчання стає незамінним інструментом для вирішення проблем у різних галузях, таких як:

- комп'ютерний зір;
- обробка природної мови;
- фінансові послуги;
- медицина;
- соціальні мережі.

Машинне навчання бере свій початок із різноманітних джерел даних: числових значень, зображень, або навіть текстової інформації. Ці дані проходять передобробку для використання у ролі навчальних даних на основі яких модель машинного навчання буде навчатися. Після цього виконавці вибирають модель машинного навчання для використання, надають дані та дозволяють

комп'ютерній моделі вивчати закономірності або робити прогнози. Пізніше виконавець може налаштувати модель, включаючи зміну параметрів, для досягнення більш точних результатів. Деякі дані витягуються з навчальних даних, які використовуються як дані для оцінки, перевіряючи, наскільки точно модель машинного навчання працює, коли їй подаються нові дані. У результаті отримуємо модель, яка може бути використана у майбутньому з різними наборами даних.

Машинне навчання включає три основні підкатегорії, кожна з яких використовується для навчання моделей з різними підходами [2]:

1. *Контрольоване машинне навчання* передбачає використання моделі, в якій машини навчаються на позначених наборах даних і здатні робити прогнози на основі цього навчання. Позначені дані передбачають, що вхідні та вихідні параметри вже пов'язані між собою. Машина навчається на основі вхідних даних та їх відповідних виходів. Під час наступних етапів, прогнозування результату за допомогою тестових наборів даних є головною метою. Контрольоване машинне навчання поділяється на дві основні категорії:
    - Класифікація використовує алгоритми, що розв'язують завдання класифікації, де вихідна змінна є категоричною, наприклад, так/ні, правда/брехня, чоловік/жінка тощо. Приклади включають виявлення спаму та фільтрацію електронної пошти.
    - Регресія застосовується для розв'язання завдань, де вхідні та вихідні змінні пов'язані лінійно, і прогнозується безперервний вихід. Наприклад, це може бути прогноз погоди або аналіз ринкових тенденцій. Серед поширених алгоритмів регресії можна виділити просту лінійну регресію, багатовимірну регресію і дерево рішень.
2. *Навчання без нагляду* не потребує прямого втручання спостерігача. Машина навчається на основі непозначених даних і може робити прогнози самостійно. Алгоритм неконтрольованого навчання

призначений для кластеризації несортованих даних на основі їх схожостей, відмінностей та шаблонів. Неконтрольоване машинне навчання поділяється на два основних типи [2]:

- Кластеризація виконує групування об'єктів у кластери на основі схожостей або відмінностей між ними.
- Асоціація досліджує типові зв'язки між змінними у великому наборі даних і визначає взаємозв'язки між ними.

3. *Навчання з підкріпленням* - це процес, що базується на принципі зворотного зв'язку. У цьому підході компонент штучного інтелекту автоматично аналізує оточуюче середовище за допомогою методу проб та помилок, навчається на досвіді та покращує свою продуктивність. Компонент отримує винагороду за кожну успішну дію і покарання за неправильні дії, що сприяє максимізації винагороди за коректні дії. На відміну від навчання під наглядом, у навчанні з підкріпленням відсутні позначені дані, і алгоритми навчаються виключно на основі досвіду. Цей метод поділяється на два типи алгоритмів [2]:
   - Навчання з позитивним підкріпленням полягає у використанні позитивного стимулу після певної дії агента, що збільшує ймовірність повторення такої дії в майбутньому, наприклад, нагорода після успішної дії.
   - Навчання з негативним підкріпленням: спрямоване на зміцнення певної дії, яка допомагає уникнути негативного наслідку.

За роки розвитку штучного інтелекту виникає не лише зростання складності завдань, але й постійне вдосконалення методів їх вирішення. Машинне навчання, у свою чергу, швидко розвивається, генеруючи різноманітні течії, серед яких особливе місце займає глибоке навчання (рис.1.1). Цей підхід відрізняється відповідністю комп'ютерних моделей до самовдосконалення та адаптації. Використовуючи аналіз складних даних та розпізнавання важких

шаблонів, воно дозволяє вирішувати завдання, що потребують великої кількості інформації і глибокого розуміння контексту.

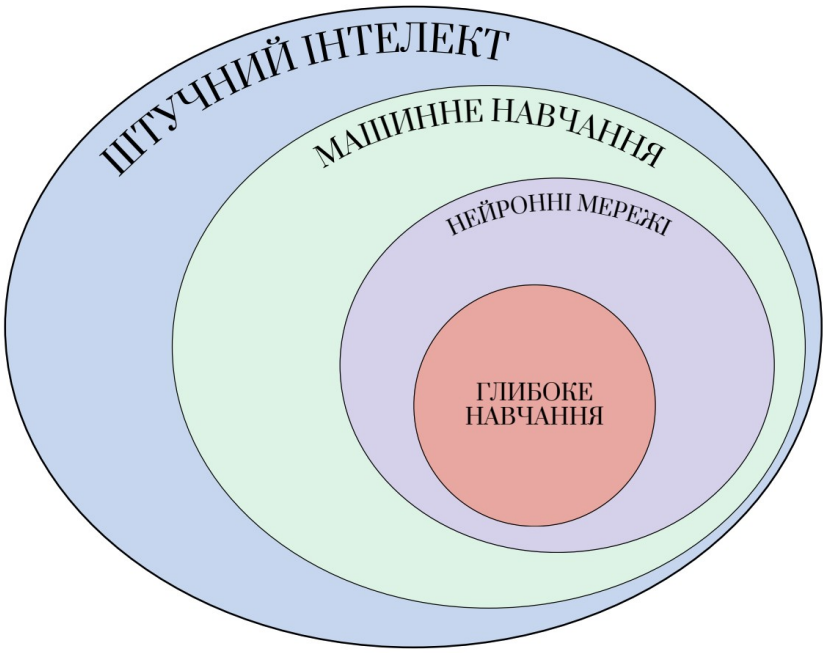

Рис. 1.1. Взаємозв'язок машинного та глибокого навчання

1.1.2. Глибоке навчання

Глибоке навчання (ГН) — це витончена і математично складна еволюція алгоритмів машинного навчання. Останнім часом ця сфера привертає до себе багато уваги, і це цілком аргументовано: останні розробки призвели до результатів, які раніше не вважалися можливими. Підвищення обговорення відбулося в 2012 році, коли нейронна мережа досягла надлюдської продуктивності в задачах розпізнавання зображень [3]. Так само як і у машинному навчанні, це може відбуватися як через контрольоване, так і через неконтрольоване навчання. ГН використовує багаторівневу структуру алгоритмів, яка називається штучною нейронною мережею (ШНМ). Структура такої ШНМ натхненна біологічною нейронною мережею людського мозку, що

призводить до процесу навчання, який набагато ефективніший, ніж у стандартних моделей машинного навчання.

Крайній лівий шар це шар з вхідними даними та позначений блакитним кольором, а крайній правий, відомий як вихідний шар, позначений червоним. Посередині ми бачимо приховані шари, виділені зеленим кольором, і вони не відображаються в навчальній вибірці. Ці шари є обчисленими значеннями, які мережа використовує для виконання своєї "магії". Чим більше прихованих шарів між вхідним та вихідним шарами, тим глибшою стає нейронна мережа. Загалом, будь-яка ШНМ з двома або більше прихованими шарами вважається глибокою нейронною мережею [4].

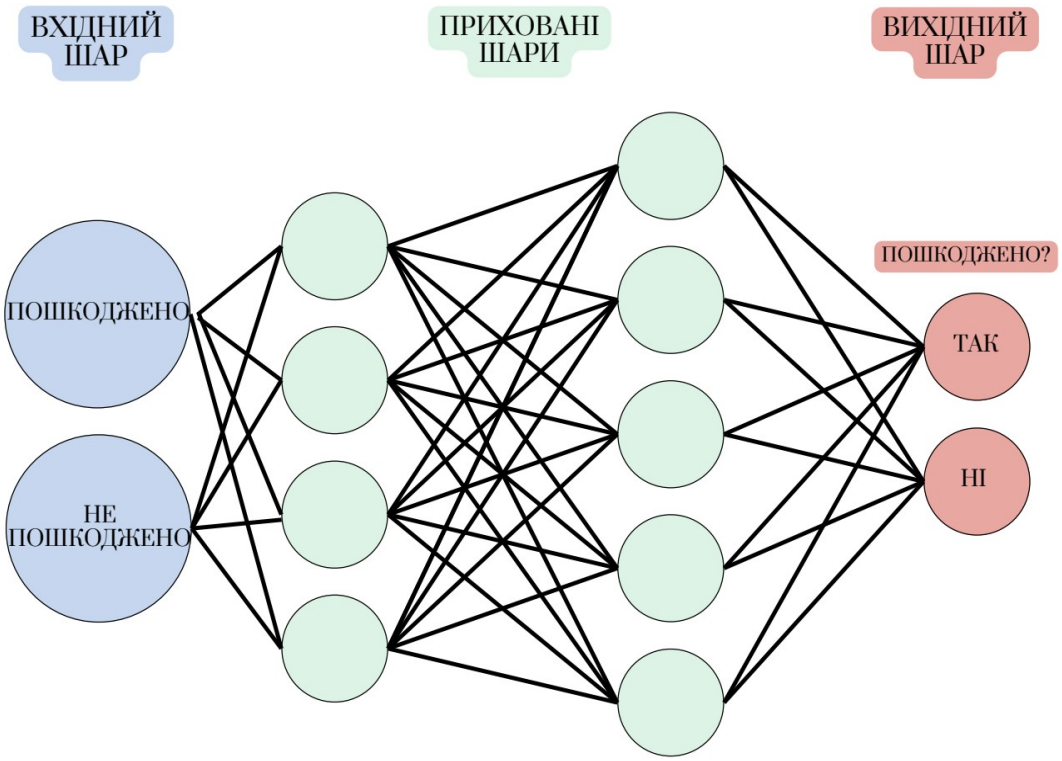

Рис. 1.2. Структура глибокої нейронної мережі на прикладі бінарної класифікації пошкоджених будівель

Нейронні мережі вчаться, отримуючи величезні обсяги даних на вхід та застосовуючи алгоритм зворотного поширення помилки. Процес полягає у поданні даних у мережу, генерації відповіді, порівнянні її з бажаним результатом (за допомогою функції втрат) та коригуванні ваг мережі відповідно до цього порівняння. Цей процес повторюється і після декількох ітерацій ми

можемо побачити досить хороший показник точності, що свідчить про вдале навчання. Але якщо ми подамо мережі вхідні дані без відомого результату, вона зможе видати результат на основі наближеної функції [3]. Якщо розглядати це на прикладі, процес виглядатиме так: ми хочемо, щоб наша мережа навчилася розпізнавати пальмові дерева на зображеннях. Для цього ми надаємо їй навчальний набір, що складається з різноманітних зображень, та порівнюємо отримані результати з реальними даними. Після цього ми коригуємо ваги мережі відповідно до отриманих результатів. З кожною новою ітерацією мережа зазвичай вдосконалюється. У кращому випадку, вона здійснює менше помилок і наближається до бажаного результату. Однак важливо враховувати, що існує ймовірність, коли мережа може навчитися неправильно або перенавчитися, що може призвести до зниження точності. Тепер ми можемо подавати мережі нові зображення, і вона буде повідомляти нас про наявність пальм на них, це представляє собою значний прогрес у розвитку моделі, що відображає високу ефективність та точність її роботи.

Зі зростанням інтересу до глибокого навчання почали розроблятися вдосконалені методи та архітектури, спрямовані на оптимізацію різних процесів. Кожна з архітектур демонструє кращу точність та ефективність у вирішенні конкретних завдань. Моделі можуть бути систематизовані та класифіковані відповідно до їх архітектурних особливостей та призначення, яке вони вирішують:

- *Нейронні мережі прямого зв'язку (FNN)* - відомі також як багатошарові перцептрони, зазвичай складаються з повністю зв'язаних шарів, де кожен нейрон одного шару пов'язаний з кожним нейроном наступного шару. Ця архітектура дозволяє нейронним мережам вивчати нелінійні зв'язки між даними, що робить їх досить ефективними у завданнях класифікації та регресії. Нейронні мережі прямого зв'язку зазвичай мають високу точність, особливо при наявності великої кількості навчальних даних. Вони є менш схильними до перенавчання порівняно з іншими

алгоритмами машинного навчання. Зазвичай використовуються у сфері комп'ютерного зору для розпізнавання облич та класифікації зображень [5].

- *Згорткові нейронні мережі (CNN)* - використовують операцію, відому як згортка, для обробки даних. Основна ідея полягає в тому, що нейрони зв'язуються не з кожним нейроном у наступному шарі, а лише з обмеженою кількістю нейронів. Тому їх основними застосуваннями є комп'ютерне зорове сприйняття та такі додатки, як класифікація зображень, аналіз відео, керування безпілотними транспортними засобами, де вони забезпечують надзвичайну продуктивність. Згорткові нейронні мережі також ідеально поєднуються з іншими типами моделей, такими як рекурентні нейронні мережі та автокодери. Один з таких прикладів - розпізнавання жестів у мові. Зазвичай використовуються у сфері комп'ютерного зору для аналізу відео та класифікації чи пошуку об'єктів на зображеннях [6].
- *Рекурентні нейронні мережі (RNN)* - відрізняються від нейронних мереж прямого зв'язку тим, що спеціалізуються на обробці даних з послідовною або часовою структурою. Ключова особливість RNN полягає у здатності мережі використовувати внутрішню пам'ять для зберігання інформації про попередні обчислення. Це дозволяє RNN ефективно працювати з даними, що мають часову залежність або послідовність, такими як текстові рядки, часові ряди або мовні дані. Кожен вихідний результат мережі RNN обчислюється з урахуванням попередніх результатів і внутрішнього стану мережі. Це дає змогу їм адаптуватися до змін у вхідних даних та вирішувати завдання, що вимагають аналізу послідовностей або динамічного контексту. RNN часто застосовуються в обробці природної мови для машинного перекладу та генерації тексту [5].
- *Генеративні суперницькі мережі (GAN)* - це модель нейронних мереж, яка складається з двох головних компонентів: генератора і дискримінатора. Ці

дві мережі працюють у взаємодії, щоб покращувати якість генерованих зображень чи даних. Генератор відповідає за створення нових екземплярів даних, які намагаються імітувати реальні дані, тоді як дискримінатор намагається розрізняти між справжніми та створеними даними. Ця конкурентна динаміка між ними сприяє навчанню і вдосконаленню обох моделей, що в кінцевому підсумку призводить до покращення якості згенерованих даних. Зазвичай використовуються у сфері комп'ютерного зору для генерації фотореалістичних зображень та відео [5].

1.1.3. Відмінність глибокого навчання від машинного

На просторах інтернету можна зустріти багато статей, присвячених порівнянню глибокого навчання та машинного. Однак, важливо відзначити, що ці особливості глибокого навчання скоріше визначають його унікальність у межах сфери машинного навчання. Обидва ці напрямки наукових досліджень належать до обширної сфери машинного навчання, що стала першочерговою областю інтересу в ряді високотехнологічних галузей. Проте, вони відрізняються своїми підходами, структурою та способами вирішення завдань [7]. Спробуємо розглянути ці відмінності більш детально через таблицю порівняння.

Таблиця 1. 1

Порівняння алгоритмів машинного та глибокого навчання

Параметр	**Машинне навчання**	**Глибоке навчання**
Обсяг даних	Робота з тисячами точок даних	Зазвичай, використання мільйонів точок даних, що дозволяє краще розуміти та працювати зі складними завданнями

Продовження таблиці 1. 1

Втручання людини	Потребує певного втручання для корекції неточностей прогнозу	Мінімальне або відсутнє втручання, оскільки модель може визначити точність прогнозу на основі нейронної мережі
Дані	Зазвичай, використання структурованих даних	Використання неструктурованих даних які обробляються нейронними мережами
Застосування	Можна використовувати для вирішення простих або дещо складних завдань.	Моделі глибокого навчання підходять для вирішення складних завдань.
Ефективність і швидкість запуску	Швидкий запуск, але обмежена ефективність	Більше часу на налаштування, проте може дати швидкі та значущі результати

1.2. Нейронні мережі CNN та їх застосування у комп'ютерному зорі

Як зазначалося раніше, згорткові нейронні мережі в основному застосовуються для аналізу зображень та відео завдяки використанню операції згортки. Основна перевага порівняно зі своїми попередниками полягає в його здатності автоматично виявляти ключові функції без втручання людини. Ця автоматизація робить CNN одним з найбільш найпопулярніших та впливових інструментів у галузі глибокого навчання. Згорткові нейронні мережі забезпечують часткову стійкість до змін масштабу, зсувів, поворотів, зміни ракурсу та інших спотворень. Наприклад, у звичайних повністю з’єднаних мережах кожен нейрон у шарі повністю зв’язаний з кожним нейроном у попередньому та наступному шарах, а у згорткових нейронних мережах застосовуються спільні ваги та локальні зв’язки. Це дозволяє повністю

використовувати двовимірну структуру вхідних даних, таких як сигнали зображення. Ця операція використовує надзвичайно малу кількість параметрів, що спрощує процес навчання та прискорює мережу. Подібно до клітин зорової кори ока, які сприймають лише обмежені області сцени, ці комірки в CNN використовуються для локальної обробки даних, а не для аналізу всієї сцени. Такий підхід дозволяє виявляти локальні кореляції та важливі особливості у вхідних даних, що є ключовим елементом успішного аналізу зображень у нейронних мережах [8].

Архітектура згорткових нейронних мереж складається з кількох рівнів, кожен з яких виконує певну функцію у обробці вхідних даних і вилученні важливих ознак [9]:

1. Вхідний шар — приймає вхідні дані. Кількість вузлів у цьому шарі зазвичай відповідає кількості вхідних функцій чи характеристик, що аналізуються. Наприклад, для зображень кожен піксель може бути окремим вузлом, а для текстових даних - кожне слово чи символ. Вхідний шар може мати також кілька каналів, наприклад, для кольорових зображень це можуть бути окремі канали для червоного, зеленого та синього кольорів. Також важливо підготувати дані перед подачею на вхідний шар, забезпечуючи їхню нормалізацію та приведення до формату від 0 до 1, щоб мережа могла ефективно їх опрацьовувати.
2. Згортковий шар — ключовий шар, складається з набору згорткових фільтрів, які згортають вхідні дані для створення карт ознак. Ядра цих фільтрів генеруються з випадковими значеннями ваг та коригуються під час навчання для виявлення ключових функцій. Згортковий процес включає «ковзання» ядра по всьому зображенню та обчислення скалярного добутку між ядром і вхідним зображенням для створення карти ознак. Ваги у CNN розподілені між всіма пікселями вхідної матриці, що прискорює процес навчання та зменшує обчислювальні витрати.

3. Шар об'єднання (шар пулінгу) — відповідає за зменшення розміру розміру карти ознак, зберігаючи при цьому важливу інформацію.
4. Вихідний шар — взаємодіє з усіма нейронами попереднього шару, і кількість нейронів у цьому шарі відповідає кількості класів, які мережа намагається розпізнати. Наприклад, у випадку класифікації пошкоджених та непошкоджених будівель, кількість нейронів у вихідному шарі буде 2: один нейрон для класу "пошкоджено" і один нейрон для класу "не пошкоджено". Цей шар також можна описати як "шар класифікації".

Важливу роль відіграє вибір функції активації. Ця функція визначає, коли нейрон мережі має активуватися на певному вхідному сигналі, що дозволяє мережі виявляти складні залежності в даних. Нелінійні функції активації використовуються в нейронних мережах, щоб дати можливість моделі навчитися складним речам. Крім того, ці функції повинні бути диференційовані, щоб мережа могла навчатися за допомогою зворотного розповсюдження помилок [10]. Деякі з найпоширеніших типів функцій активації, що використовуються в CNN та інших глибоких мережах, включають ReLU, Sigmoid і Tanh.

Переваги використання згорткових нейронних мереж порівняно з іншими традиційними архітектурами нейронних мереж у контексті комп'ютерного зору [8]:

- здатність до автоматичного вивчення та розподілу ваг, що сприяє зменшенню кількості параметрів, які потрібно навчати в мережі. Це полегшує узагальнення моделі та уникнення перенавчання;
- одночасне вивчення шарів виявлення ознак і класифікації призводить до того, що вихід моделі є високоорганізованим і сильно залежить від вилучених функцій;
- впровадження великомасштабних мереж у реальних завданнях набагато простіше за допомогою CNN, оскільки ця архітектура виявляється більш

ефективною в управлінні великою кількістю параметрів порівняно з альтернативними підходами.

1.3. Застосування глибокого навчання у геоінформаційних системах

Глибоке навчання за останні роки стало потужним інструментом в багатьох сферах, і геоінформаційні системи не є винятком. Хоча інструменти машинного навчання вже давно використовуються, наприклад, для класифікації зображень, інтеграція алгоритмів ГН у програмне забезпечення ГІС є порівняно новим явищем. Особливо важливим і доцільним стає у контексті аналізу зображень, зокрема при роботі з космічними та аерофотознімками. Його застосування дозволяє автоматизувати процеси, які традиційно потребували значних ресурсів та часу [11]. Розглянемо можливість його застосування на конкретних прикладах.

1.3.1. Класифікація об’єктів

Класифікація об’єктів (зображень) є однією з найпопулярніших форм глибокого навчання, в якій комп'ютер присвоює зображенню мітку, наприклад, «кіт» або «собака». Цей підхід може бути застосований у ГІС для категоризації фотографій з геотегами [12].

Наприклад, у сфері моніторингу будівель, класифікація зображень може бути використана для розпізнавання пошкоджених та непошкоджених будівель на основі супутникових знімків або аерофотознімків (рис. 1.3). Така класифікація дозволяє оцінювати рівень пошкоджень, особливо під час військового конфлікту. Ці дані можуть бути використані для розробки відновлення інфраструктури після завершення конфлікту. Завдяки глибокого навчання, можливо досягти результати високої точності та підвищити ефективність процесу аналізу зображень для прийняття рішень.

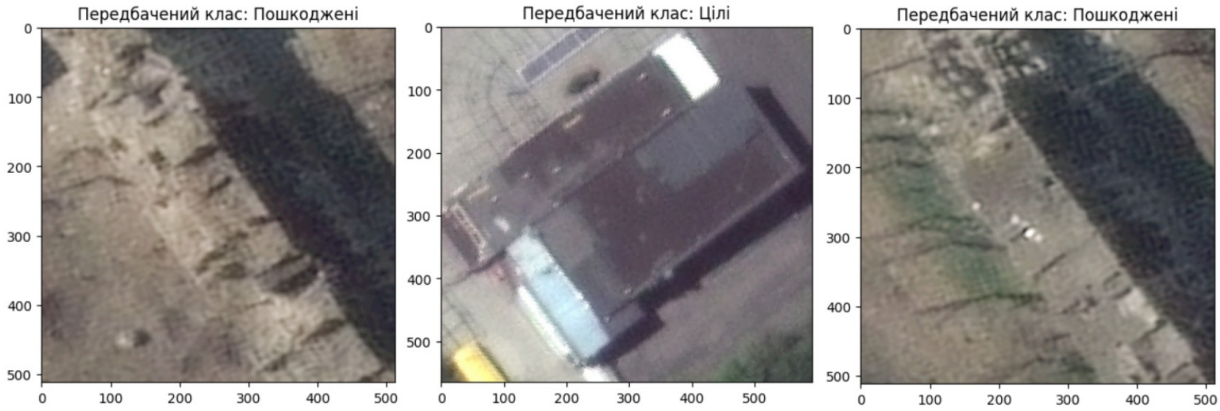

Рис. 1.3. Застосування глибокого навчання для бінарної класифікації пошкодженої забудови

1.3.2. Виявлення об'єктів

Виявлення об'єктів — це процес, при якому комп'ютер виконує пошук об'єктів на зображенні та встановлює їх місцезнаходження [13]. Цей метод є надзвичайно важливим для ГІС, оскільки дозволяє знаходити об'єкти на супутникових, аерофотознімках або знімках, зроблених безпілотниками, і позначати їх на карті.

Наприклад, виявлення об'єктів може бути використане для пошуку пальмових дерев на зображеннях з високою роздільною здатністю для подальшого аналізу (рис. 1.4). Це дозволяє визначити їхнє розташування та розподіл, що корисно для планування сільськогосподарських операцій або оцінки їхнього впливу на місцеву екологію.

1.3.3. Семантична сегментація

Семантична сегментація — це метод обробки зображень, при якому кожному пікселю зображення присвоюється значення певного класу або категорії, тобто класифікація зображення. Цей метод дозволяє створювати детальні класифікаційні карти землекористування.

Рис. 1.4. Застосування глибокого навчання для пошуку пальмових дерев на знімках високої якості

Наприклад, при семантичній сегментації супутникових або аерофотознімків, кожен піксель може бути віднесений до таких категорій, як урбанізація, ліси, водойми, відкриті ґрунти тощо. Це дає змогу створити карту землекористування з високою роздільною здатністю, яка наочно відображає розподіл різних типів земельного покриву та використання земель (рис. 1.5). Семантична сегментація є складним завданням, оскільки вона вимагає обробки та аналізу великої кількості даних на рівні окремих пікселів зображення [13].

Алгоритми глибокого навчання, зокрема згорткові нейронні мережі (CNN), є найбільш поширеними інструментами для виконання цієї задачі, оскільки вони можуть ефективно вивчати патерни та взаємозв'язки в зображенні.

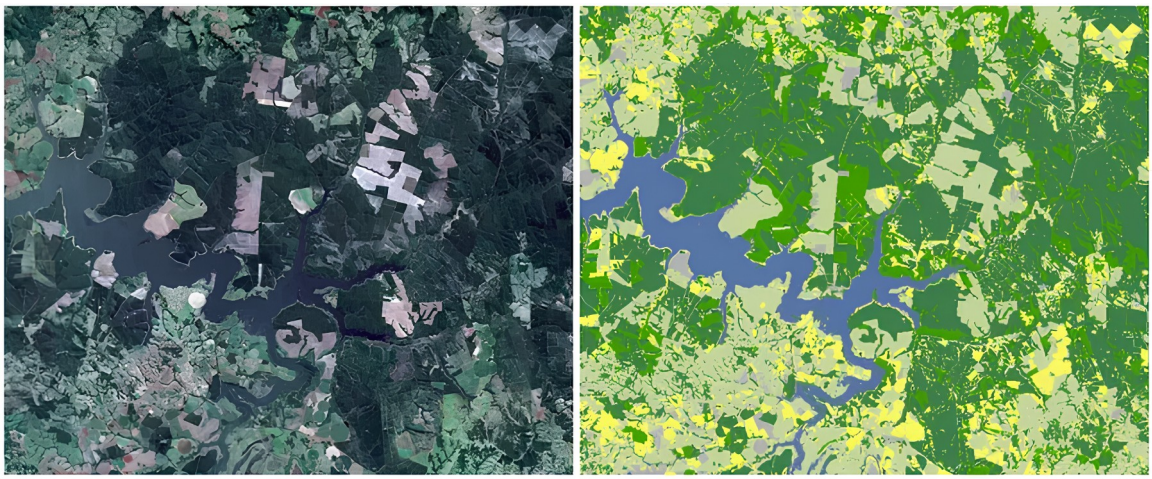

Рис. 1.5. Застосування глибокого навчання для класифікації знімків [14]

1.3.4. Екземплярна сегментація

Екземплярна сегментація — це удосконалений метод виявлення об'єктів, який полягає в тому, що для кожного об'єкта на зображенні малюється індивідуальна маска, тобто об'єкт виділяється як єдиний екземпляр [12]. Це забезпечує більш точне та детальне відокремлення об'єктів порівняно зі звичайним виявленням об'єктів.

Наприклад, під час виявлення слідів будівель на супутникових або аерофотознімках, сегментація екземплярів може визначити кожну будівлю як окремий об'єкт, позначаючи її точні межі. Це дозволяє отримати більш точні дані про кількість, розмір і форму будівель, що корисно для міського планування. Завдяки використанню складних алгоритмів глибокого навчання, сегментація екземплярів може підвищити точність та ефективність аналізу зображень у різних галузях.

Рис. 1.6. Застосування глибокого навчання для створення слідів будівель. *Заливка різного кольору: виявлені маски об’єктів; рамка червоного кольору: відображення рамки виявленої повноцінної маски.*

1.3.5. Виявлення змін

Виявлення змін — метод ідентифікації відмінностей у певних об'єктах або місцевостях між двома різними часовими точками. Ці алгоритми можуть

створювати логічну карту змін, яка показує, де відбулися зміни, та якими вони є [12].

Наприклад, розглянемо моніторинг міста. Завдання виявлення змін можуть бути застосовані для порівняння супутникових знімків міста, зроблених у різний час. Алгоритм глибокого навчання може ідентифікувати, які нові споруди з'явилися, а також відзначити зміни в інфраструктурі, таких як нові дороги чи парки. Ця інформація може бути представлена на карті змін, яка допоможе міському плануванню або оцінці наслідків природних катастроф.

Рис. 1.7. Застосування глибокого навчання для виявлення змін на знімках зроблених у різний час [12]

1.4. Діагностика процесу навчання в глибокому навчанні: проблеми та рішення

Алгоритми глибокого навчання мають свої особливості, які часто виникають під час навчання моделі. Користувачі часто стикаються з такими питаннями, як: чому показники точності надто високі на початковому етапі навчання, але результати моделі під час застосування виявляються поганими; або ж, чому показники з часом знижуються або залишаються стабільними. Ці запитання можуть викликати плутанину та нерозуміння того, які кроки необхідно вжити для покращення навчання моделі. Розглянемо найпоширеніші проблеми, з якими стикаються при навчанні моделі:

- недостатнє навчання;
- надмірне навчання;

1.4.1. Недостатнє навчання

Недостатнє навчання (недонавчання) відбувається, коли модель машинного навчання погано розпізнає навчальний набір. Внаслідок цього модель недостатньо ефективно відображає зв'язок між вхідними та вихідними даними. Це призводить до низької точності прогнозів, навіть для навчальних даних. Як наслідок, така модель видає неточні результати, що призводить до прийняття рішень з високим рівнем похибки, які є близькі до випадкових [15].

Його можна легко виявити за низькою ефективністю на навчальних даних, навіть без використання тестового набору можна спостерігати, що модель має високе зміщення, якщо вона погано працює на навчальному наборі. Ця проблема часто виникає через недостатній обсяг навчальних даних або занадто просту структуру моделі. Для виявлення недонавчання можна проаналізувати графіки втрат та точності моделі: характерним для недонавчання, що зображено на графіках (рис. 1.8), є стабільно високі втрати та низька точність на навчальному наборі, які майже не змінюються з часом.

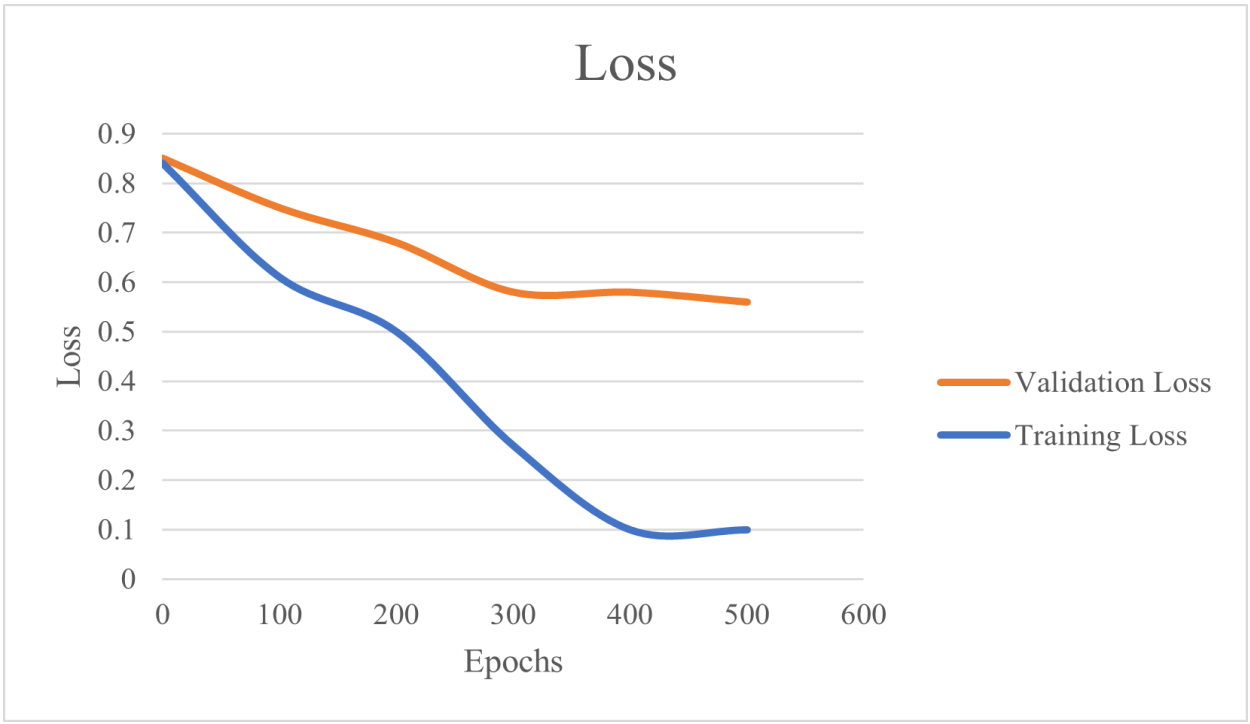

Рис. 1.8. Графік втрат та перевірки при недостатньому навчанні [16]

Для розв'язання проблем недостатнього навчання моделей глибокого навчання можна застосувати різні стратегії, які допоможуть підвищити ефективність навчання та покращити продуктивність моделі. Рішення основних проблем можуть виглядати так [18]:

- Недостатня кількість навчальних даних — якщо ми не можемо традиційно доповнити нашу навчальну вибірку, розв'язання даної проблеми можна досягти за допомогою трансформації даних, наприклад, шляхом повороту, розтягування, стиснення або обрізання зображень. Це створює нові варіації даних, що дозволяє моделі виявляти нові ознаки і підвищувати загальну ефективність.
- Недостатнє навчання — можна подолати шляхом збільшення часу навчання. Важливо не припиняти навчання моделі після кількох ітерацій, оскільки для досягнення якісних результатів необхідний триваліший процес навчання.
- Низька складність моделі — рішенням є збільшення розміру моделі та кількості параметрів, що дозволить моделі краще узагальнювати дані та підвищити її продуктивність.

1.4.2. Надмірне навчання

Надмірне навчання (перенавчання) моделі глибокого навчання виникає, коли модель надто добре підлаштовується під навчальні дані, включаючи випадковий шум і специфічні особливості цих даних. Як результат, модель може мати високу точність на навчальному наборі, але погано працювати з новими даними, наприклад, з валідаційного або тестового набору. У цьому випадку втрати на валідації спочатку знижуються, але потім починають зростати [17].

Перенавчання моделі легко виявити на графіках втрат під час навчання та перевірки (рис. 1.9). Спочатку модель може показувати хороші результати під час тренування, але потім її точність різко погіршується. У такому разі не варто

розраховувати на покращення без корекції моделі, оскільки це свідчить про необхідність виправлення помилок. Причини перенавчання можуть включати надмірну складність моделі або тривале навчання на обмеженому наборі даних.

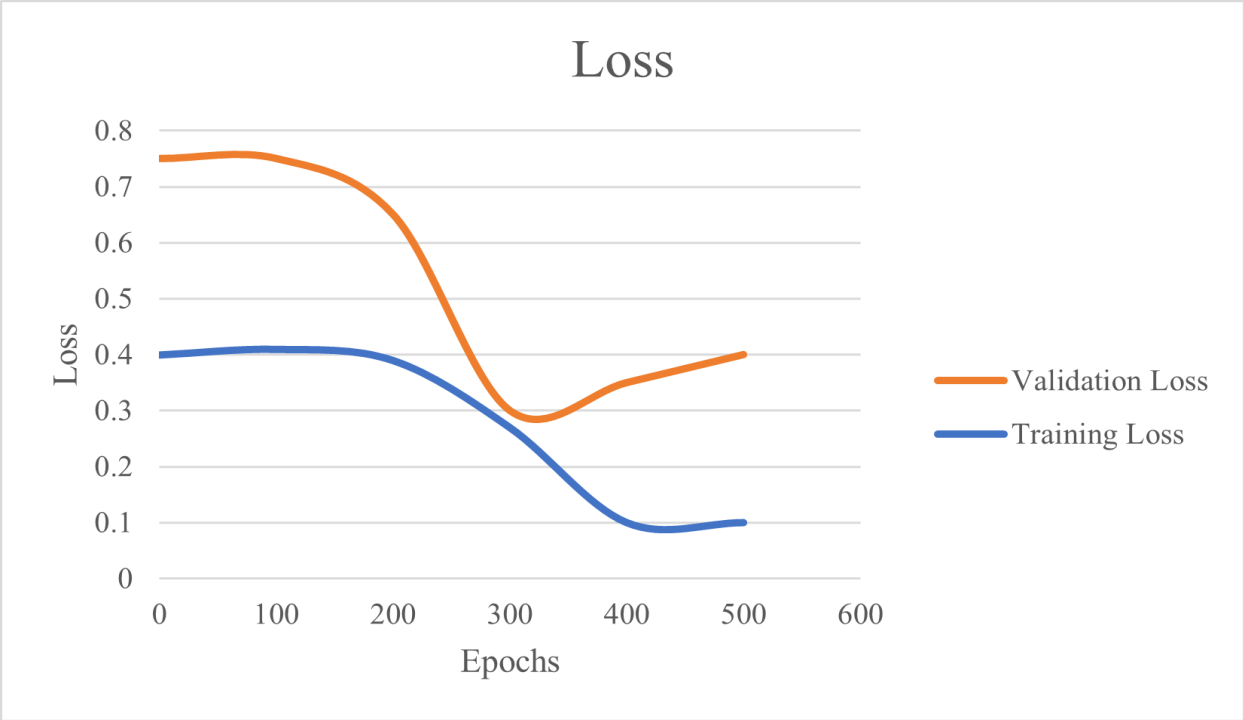

Рис. 1.9. Графік втрат та перевірки при надмірному навчанні [16]

Для вирішення проблем перенавчання моделей глибокого навчання існують різні методи, які допоможуть зменшити складність моделі та покращити її здатність узагальнювати нові дані. Рішення основних проблем можуть виглядати так [19]:

- Зменшення складності моделі —зменшення кількості вхідних функцій або кількості параметрів у моделі допомагає знизити її складність і підвищити здатність узагальнювати.
- Вибір функцій — вибір значущих функцій вручну або за допомогою алгоритмів вибору функцій зменшує кількість вхідних даних і підвищує ефективність моделі.
- Рання зупинка — цей підхід полягає в зупинці навчання моделі, коли досягнуто певних критеріїв, наприклад, коли точність валідації починає знижуватися.

1.4.3. Правильне навчання: оптимальні графіки втрат

Оптимальний випадок навчання знаходиться між недонавчанням та перенавчанням моделі. Його характеризує поступове зменшення втрат як під час навчання, так і під час валідації до стабільного рівня. Втрати на навчальних даних зазвичай нижчі, ніж на валідаційних, що є нормальним явищем. Тому можливий невеликий розрив між кривими втрат на навчанні та валідації.

Графік навчання вважається оптимальним, якщо:

- Графік втрат на навчанні зменшується до стабільного рівня.
- Графік втрат на валідації також зменшується до стабільного рівня та відрізняється незначно від графіка втрат на навчанні.

Для кращого розуміння, нижче наведено графічну ілюстрацію, яка показує, як виглядають криві втрат під час навчання та валідації в ідеальному випадку навчання (рис. 1.10). Тут ми бачимо стабільне зменшення втрат і невеликий розрив між графіками навчання та валідації, що є характерним для оптимального навчання моделі. Також, варто зазначити, що тривале навчання після досягнення хорошого результату може спричинити перенавчання моделі.

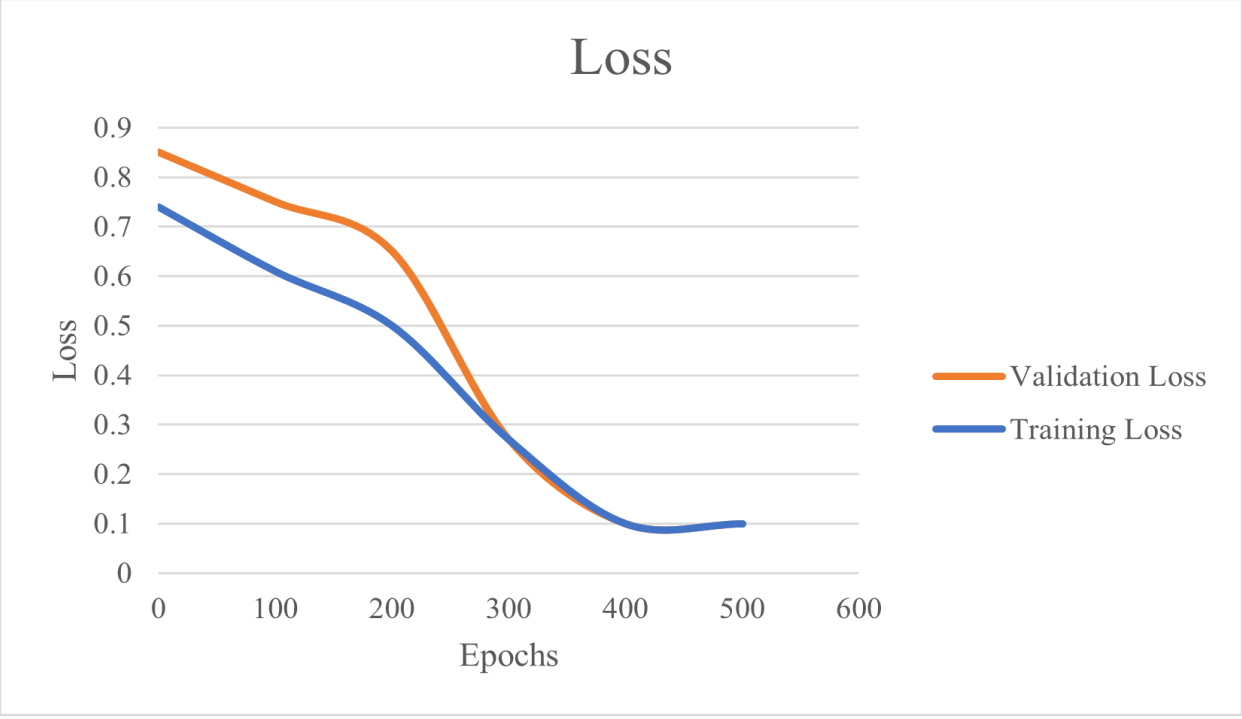

Рис. 1.10. Стабільне зменшення втрат навчання та перевірки [16]

РОЗДІЛ 2. РОЗРОБКА МОДЕЛІ ГЛИБОКОГО НАВЧАННЯ ДЛЯ ІДЕНТИФІКАЦІЇ ПОШКОДЖЕНОЇ ЗАБУДОВИ

2.1. Бібліотеки TensorFlow та Keras: можливості та застосування в глибокому навчанні

Поглиблюючись у сферу глибокого навчання, часто помічаємо, що Keras та TensorFlow є основними інструментами для розробки моделей. Вони застосовуються разом для створення, навчання та оцінки нейронних мереж, які лежать в основі глибокого навчання. Але що таке Keras та TensorFlow і чому вони зазвичай використовуються разом?

TensorFlow — це повноцінна платформа глибокого навчання з відкритим кодом, розроблена компанією Google і випущена в 2015 році. Це бібліотека для математичних обчислень, яка використовується в нейронних мережах і є оптимальною для програмування потоків даних у різних завданнях [20]. Сьогодні TensorFlow є потужним інструментом для розробки та навчання різних моделей, починаючи від простих лінійних регресій і закінчуючи складними нейронними мережами. Для кращого розуміння ефективності та обмежень бібліотеки TensorFlow, розглянемо її основні переваги та недоліки [21].

Переваги TensorFlow:

- TensorFlow забезпечує високий рівень гнучкості, дозволяючи визначати кожен аспект архітектури нейронної мережі та процесу навчання.
- TensorFlow має потужну підтримку спільноти, яка сприяє розробці та вдосконаленню бібліотеки.
- Завдяки можливостям розподіленого обчислення, TensorFlow підходить для великомасштабних завдань машинного навчання. Він використовується у виробничих середовищах та може працювати на GPU, TPU та навіть на кількох машинах.

- TensorFlow демонструє кращу продуктивність порівняно з іншими платформами.
- Вбудований інструмент візуалізації TensorFlow, TensorBoard, є потужним інструментом для моніторингу та налагодження моделей під час навчання.

Недоліки TensorFlow:

- Швидкість TensorFlow нижча порівняно з іншими схожими платформами.
- Новачкам може бути важко освоїти платформу через складність коду.
- TensorFlow не підтримує OpenCL.

Keras — це високорівневий інтерфейс прикладного програмування для роботи з нейронними мережами, розроблений на Python для швидкої роботи з глибокими нейронними мережами. Цей відкритий вихідний код підтримує роботу поверх різних фреймворків, таких як CNTK, TensorFlow і Theano. Головною метою Keras є забезпечення зручність з нейронними мережами. Він не виконує низькорівневі обчислення самостійно, натомість передає їх на обробку відповідній бібліотеці. У середині 2017 року Keras було інтегровано до TensorFlow, і користувачі можуть працювати з ним через модуль `tf.keras`. Водночас Keras все ще функціонує як окрема бібліотека, доступна для використання незалежно від TensorFlow [22].

Переваги Keras:

- Keras розроблений для зручності та інтуїтивного розуміння. Він спрощує більшість складнощів TensorFlow, що робить його чудовим вибором для тих, хто починає знайомитися з глибоким навчанням.
- Keras дозволяє швидко створювати прототипи нейронних мереж, що сприяє оперативному експериментуванню з різними архітектурами.
- Код у Keras зазвичай коротший і зрозуміліший порівняно з еквівалентним кодом на TensorFlow.

- Keras інтегрований у TensorFlow як офіційний високорівневий інтерфейс прикладного програмування з TensorFlow 2.0, забезпечуючи сумісність і взаємодію між ними.

Недоліки Keras:

- Досвідченим користувачам може бракувати повного контролю над кожним аспектом моделей.
- Налаштовувати шари та операції може бути складніше в Keras порівняно з чистим TensorFlow.

Отже, поєднання двох бібліотек — Keras та TensorFlow — є кращим вибором для новачків у сфері глибокого навчання, особливо якщо завдання не вимагає повного контролю над кожним аспектом моделі. Крім того, використання Keras поверх TensorFlow дозволяє швидко прототипувати моделі, експериментувати з різними архітектурами та гіперпараметрами, а також легко інтегрувати моделі в наявні системи. Завдяки широкому набору інструментів та бібліотек, доступних у TensorFlow, можна значно розширити можливості моделей Keras, додавши підтримку для спеціалізованих шарів, функцій втрати та оптимізаторів. Цей дует поєднує простоту Keras з потужністю та гнучкістю TensorFlow, забезпечуючи збалансований підхід до навчання та розробки моделей глибокого навчання.

2.2. Опис та підготовка навчальної вибірки

2.2.1. Збір та обробка навчальних даних

Для створення майбутньої моделі глибокого навчання важливо розуміти теоретичні основи задачі та створити план дій для її реалізації. Першим кроком є підготовка даних для навчання моделі, що зазвичай вимагає значного часу через необхідність створення, очищення, перетворення та розширення вихідних даних для забезпечення їх відповідності вимогам моделі. У нашому випадку це

передбачає використання космічних знімків Maxar (0,5 см на піксель) та геопросторових даних будівель. Для забезпечення доступу до даних про будівлі ми скористалися даними *OpenStreetMap (OSM)*, безкоштовною мапою всього світу, яка була створена у 2004 році у Великій Британії і яку може редагувати кожен користувач. OSM містить дані про дороги, будівлі, підприємства, громадський транспорт, землекористування та багато інших об'єктів. Карту створюють і обслуговують майже 5 мільйонів зареєстрованих користувачів [23].

Overpass Turbo — це онлайн-інструмент для доступу до даних OSM за допомогою мови запитів Overpass. Він дозволяє швидко та зручно отримати вибірку даних про об’єкти з OSM, зокрема інформацію про будівлі, дороги, парки, річки та багато іншого. Overpass Turbo надає можливість створювати запити на основі географічних областей та категорій об'єктів, що дозволяє користувачам витягати потрібні дані з глобальної бази OSM [24].

Використання даного ресурсу дало нам змогу отримати необхідні геопросторові дані про будівлі масті Маріуполя для подальшої обробки та аналізу в ArcGIS. На рис. 2.1, поданому нижче, представлений скриншот з веб-інтерфейсу Overpass Turbo, що демонструє візуальне відображення полігонів будинків Маріуполя. Після імпортування даних до ArcGIS ми змогли здійснити перетворення системи координат, що забезпечує точне узгодження з системою координат космічних знімків. Також, важливо відзначити, що було вирішено виключити полігони малого розміру, які відповідають невеликим будівлям, з набору даних. Замість цього, акцент було зроблено на великих полігонах, що відображають великі будівлі, здатні чітко виокремлюватися на знімках з даною роздільною здатністю.

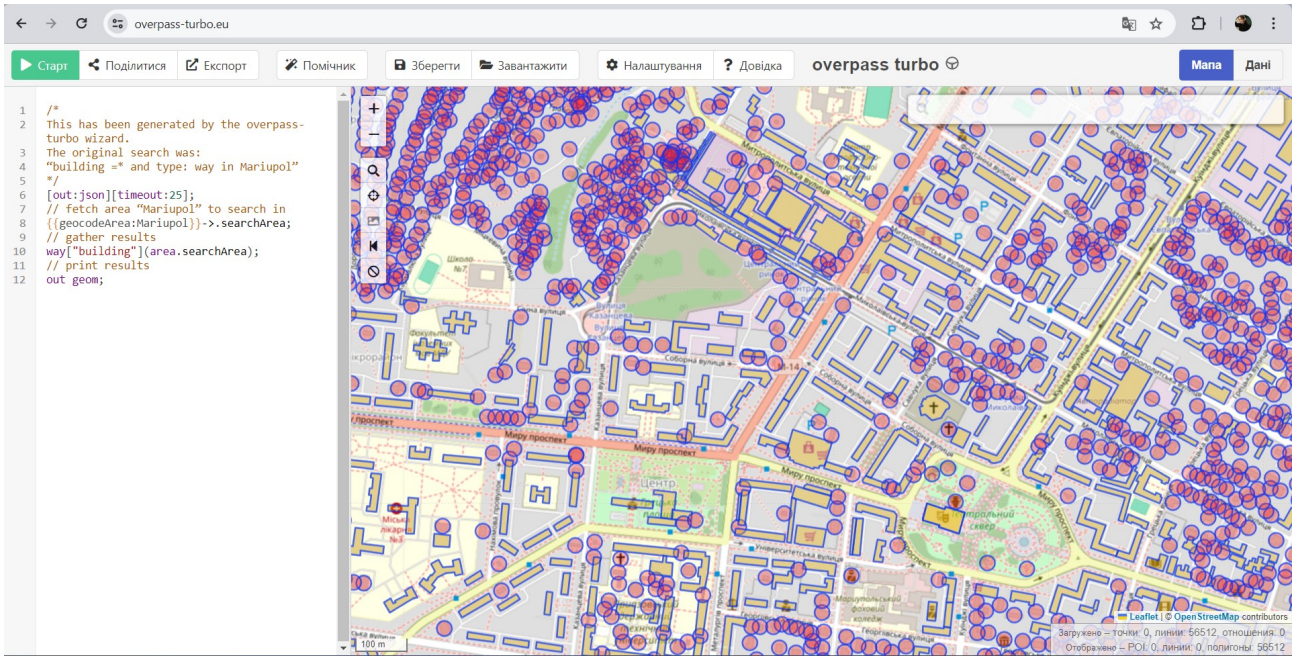

Рис. 2.1. Запит до бази даних OpenStreetMap: отримання інформації про будівлі в місті Маріуполь

Після попередньої підготовки даних можна розпочинати створення міток, що необхідно для подальшого навчання моделі. У таблиці атрибутів шару будівель створюється новий стовпець «Class_Value», де значеннями є цілі числа, що відповідають різним класам об'єктів. У процесі створення міток необхідно ретельно аналізувати кожну будівлю та призначити їй відповідний клас. У цьому дослідженні ми визначили три класи:

- Цілі будівлі (Class_Value = 0), позначені зеленим кольором. Цей клас представляє будівлі, які залишилися неушкодженими.
- Знесені будівлі (Class_Value = 1), позначені фіолетовим кольором. Цей клас включає будівлі, які були знесені з різних причини, ймовірно, через значне пошкодження внаслідок воєнних дій.
- Пошкоджені будівлі (Class_Value = 2), позначені червоним кольором. Цей клас відображає будівлі, які зазнали певного рівня пошкодження.

Наведені класи дозволяють більш точно класифікувати об'єкти для подальшого навчання моделі. Навчальна вибірка включала понад 200 екземплярів кожного класу. Ручна мітка даних гарантує високу якість навчальної вибірки, яка є критично важливою для точного навчання моделі глибокого навчання.

Будівлі, які були визначені як пошкоджені, мали різноманітність ушкоджень, таких як повністю обвалені частини будівлі, видимі ушкодження дахів чи навіть відсутність даху взагалі. Для кожного класу наведено зображення нижче (рис. 2.2), які демонструють відображення будівель відповідного класу, що забезпечує наочність та зручність у роботі з даними.

Рис. 2.2. Класифікація будівель: а — цілі будівлі; б — знесені будівлі; в — пошкоджені будівлі.

Останнім кроком у створенні навчальної вибірки є експорт плиток зображень позначених об'єктів. У середовищі ArcGIS існує широкий спектр інструментів для виконання цього завдання, і важливо підібрати найоптимальніший. Ми використовували спеціальні інструменти бібліотеки глибокого навчання для ArcGIS, які встановлюються окремо, та застосували інструмент "Експорт навчальних даних для глибокого навчання" з наступними параметрами:

- Вхідний растр (Input Raster) — використані космічні знімки.
- Вхідний клас об’єктів або класифікований растр (Input Feature Class Or Classified Raster) — підготовлений шар із позначеними будівлями.
- Розмір плитки X та У (Tile Size X/Tile Size Y) — оптимальним вибором став розмір 256x256 пікселів.
- Формат метаданих (Metadata Format) — найважливіший параметр для нашої задачі — формат Imagenet, де кожна вихідна плитка буде позначена певним класом та розподілена по відповідних папах. Це значно спрощує організацію даних у майбутньому.

- Поле значення класу (Class Value Field) — використовуємо попередньо створене поле «Class_Value».
- Режим кадрування (Crop Mode) — використовується фіксований розмір зображень для спрощення роботи моделі.

Зображення нижче (рис. 2.3) ілюструє результат роботи інструменту — дані, які організовані за класами та розділені за відповідними теками для подальшого навчання моделі.

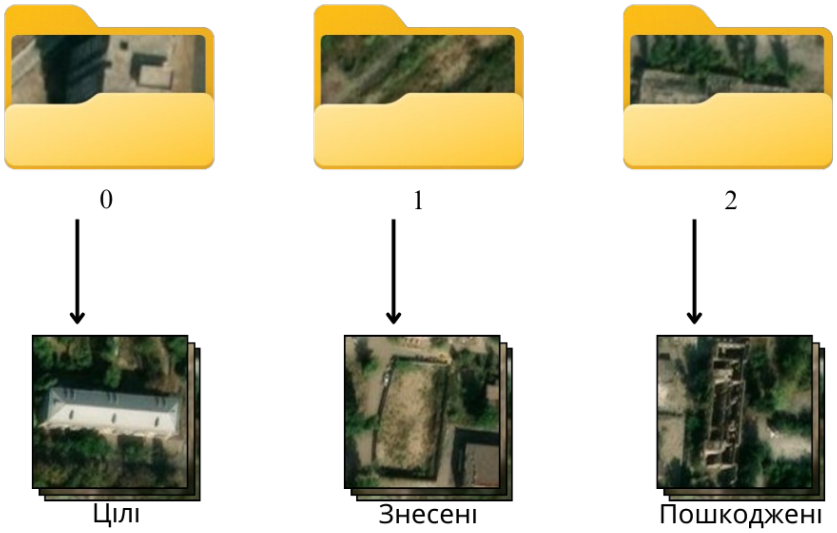

Рис. 2.3. Організація даних навчальної вибірки

2.2.2. Розширення навчальної вибірки

На даний момент навчальна вибірка складає близько 690 екземплярів різних класів. Для ефективного навчання моделі глибокого навчання цього обсягу даних недостатньо. Звісно, якщо є можливість, можна доповнити вибірку, повторюючи раніше зазначені кроки, але це може бути трудомістким та витратним за часом процесом. Натомість, можна скористатися вбудованими інструментами Keras, зокрема генератором даних, який допоможе значно збільшити навчальну вибірку шляхом застосування різних перетворень до зображень. Це дозволяє створити нові варіації на основі наявних даних, що для моделі глибокого навчання є еквівалентом отримання «нових» даних [25].

Для досягнення цієї мети використовується генератор даних Keras (ImageDataGenerator), налаштований на різні види перетворень (лістинг 2.1). Кожне зображення має близько 10 згенерованих версій, застосовуючи випадкові перетворення, як наведено у лістингу нижче.

Лістинг 2.1

```
datagen = ImageDataGenerator(
    rotation_range = 10,  # Обертання зображень у межах 10 градусів
    width_shift_range = 0.05,  # Горизонтальний зсув на 5%
    height_shift_range = 0.05,  # Вертикальний зсув на 5%
    zoom_range = (1.0, 0.7),  # Масштабування зображень від 1.0 до 0.7
    horizontal_flip = True,  # Горизонтальне віддзеркалення зображень
    fill_mode = 'nearest'  # Заповнення порожніх пікселів найближчими значеннями
)
```

Дані дії дозволили значно збільшити обсяг навчальної вибірки, яка тепер містить понад 6,5 тисяч зображень. Це забезпечує надійну основу для розробки моделі високої точності, зменшуючи витрати на збір та підготовку даних. Такий підхід є особливо корисним у випадках, коли обсяг початкової вибірки є обмеженим, але необхідно досягти високої точності та надійності моделі. Таким чином, завдяки ефективному використанню аугментації даних, ми можемо уникнути перенавчання та забезпечити кращу продуктивність моделі без додаткових витрат часу та ресурсів.

2.3. Розробка та оцінка моделі глибокого навчання з використанням TensorFlow та Keras

Після завершення етапу підготовки даних, можна перейти до розробки моделі глибокого навчання. Використовуючи фреймворки TensorFlow та Keras, можна ефективно створювати та налаштовувати нейронні мережі для розв'язання поставленого завдання.

Етап програмування буде виконуватися в середовищі Jupyter Lab за допомогою мови програмування Python. Крім підвантаження бібліотек Keras і TensorFlow необхідно імпортувати інші модулі та сторонні бібліотеки [26]:

- splitfolders — бібліотека призначена для автоматичного розподілення даних на набори для навчання, валідації і тестування;
- tensorflow.keras.preprocessing.image — надає інструменти для роботи з зображеннями перед їхнім використанням у моделях глибокого навчання;
- tensorflow.keras.layers — містить набір класів для побудови різноманітних типів шарів у нейронних мережах;
- tensorflow.keras.models — містить інструменти для побудови нейронних мереж;
- tensorflow.keras.applications.ResNet50 — містить попередньо навчену модель ResNet50, яка буде використана як основа для подальшого навчання.

Далі ми переходимо до обробки навчальної вибірки. Для цього використовується функція «splitfolders.ratio», яка дозволяє розділити дані на навчальний, валідаційний і тестовий набори в зазначеному співвідношенні. Параметр «ratio» визначає, яка частина даних буде призначена для кожного набору. Для нашого завдання ми встановили «ratio = (0.7, 0.2, 0.1)», це означає, що 70% даних буде використано для навчання, 20% для валідації і 10% для тестування [27]. Крім того, за допомогою параметра «output» ми вказуємо шлях до нової теки, де буде збережено, також по текам, розділені дані.

У наступному етапі підготовки до моделювання ми створюємо список класів, які використовуються для класифікації зображень. Після цього ми створюємо об'єкт генератора даних, який дозволяє нам автоматично обробляти зображення для подальшого використання у процесі навчання моделі (лістинг 2.2).

Лістинг 2.2

```
class_names = ['0', '1', '2']
datagen = ImageDataGenerator()
# тренувальний набір даних
train_generator = datagen.flow_from_directory(
    directory=r"D:\Диплом\train",
    classes=class_names,
    target_size=(256, 256),
    batch_size=16,
    class_mode="sparse",
)
# набір даних для перевірки
val_generator = datagen.flow_from_directory(
    directory=r"D:\Диплом\val",
    classes=class_names,
    target_size=(256, 256),
    batch_size=16,
    class_mode="sparse",
)
# тестовий набір даних
test_generator = datagen.flow_from_directory(
    directory=r"D:\Диплом\test",
    classes=class_names,
    target_size=(256, 256),
    batch_size=16,
    class_mode="sparse",
)
```

Генератор даних налаштовується ідентично для всіх наборів даних, включаючи навчальний, валідаційний та тестовий, встановлюючи шлях до теки з зображеннями, розмір зображень (256 на 256 пікселів), розмір пакету (16 зображень), та режим класифікації (sparse). Режим «sparse» означає, що мітки класів представлені у вигляді цілих чисел, а не у формі розріджених векторів. Це особливо корисно в багатокласовій класифікації, де кожне зображення може належати лише одному класу. Відображення випадкових даних тренувального набору зображено на рисунку нижче (рис. 2.4).

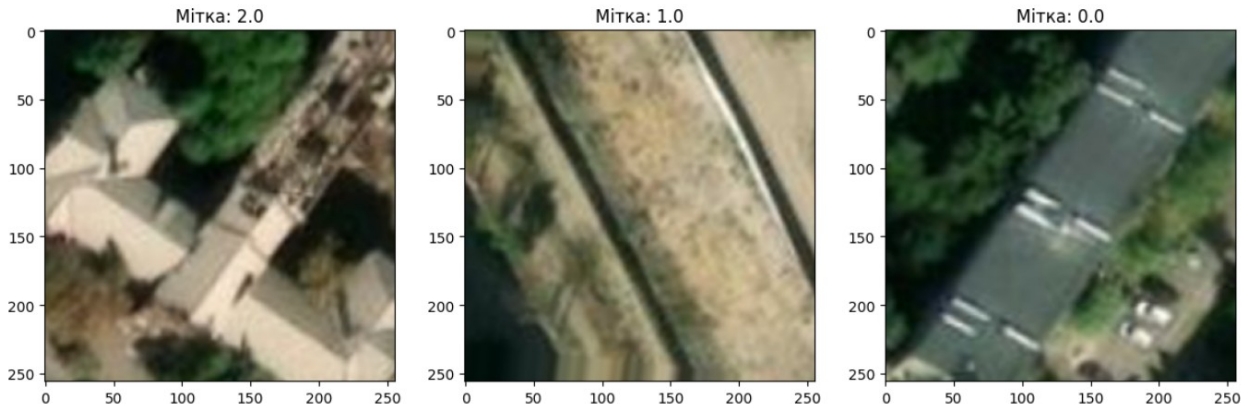

Рис. 2.4. Відображення випадкових даних тренувального набору

У даній задачі ми вирішили, що доцільним буде використання архітектури глибокого навчання ResNet50, шари якої були попередньо натреновані на великому наборі даних (ImageNet) (лістинг 2.3). *ResNet50* є різновидом згорткової нейронної мережі, яка відома своєю глибиною (50 шарів) та ефективністю у вирішенні задач комп'ютерного зору. Основну проблему, яку вирішує ResNet, є проблема деградації глибоких нейронних мереж. У міру збільшення глибини мереж, їхня точність стабілізується, а потім починає швидко погіршуватися. Це погіршення викликане не надмірним пристосуванням, а скоріше складністю оптимізації процесу навчання. ResNet долає цю проблему за допомогою «залишкових блоків» (residual blocks), які забезпечують прямий потік інформації через пропускні зв'язки, що дозволяє уникнути проблеми зникаючого градієнта [28].

Після завантаження ResNet50 ми «заморожуємо» всі її шари, крім верхніх, щоб уникнути їх навчання під час тренування на наших даних. Змінюємо верхні шари структури, додаючи до них нові повнозв'язані шари для відповідної класифікації. Після глобальної підсумовувальної операції, що витягує особливості з останнього згорткового шару, ми додаємо кілька повнозв'язаних шарів з функціями активації «ReLU», що допомагає узагальнити та вирішувати складні завдання класифікації [10].

Щоб запобігти перенавчанню моделі, використовуємо шари «Dropout», які випадковим чином «вимикають» частину нейронів під час тренування.

Завершуємо модель шаром «Dense» з функцією активації «softmax», яка генерує ймовірний розподіл на виході для кожного класу [29, 30]. Кількість виходів відповідає кількості класів, що класифікуються (у нашому випадку 3).

Створення функції «trainModel» надає можливість для тренування моделі з використанням зазначеного оптимізатора, зазначеної кількості епох та вхідних даних. Вона компілює модель з обраним оптимізатором та функцією втрат, а потім тренує модель на тренувальних даних, використовуючи дані валідації для оцінки продуктивності моделі під час тренування (див. лістинг 2.3).

Лістинг 2.3

```
resnet_50 = ResNet50(include_top=False, weights='imagenet',
input_shape=(256,256,3))
for layer in resnet_50.layers:
    layer.trainable = False
# Побудова повної моделі
x = resnet_50.output
x = layers.GlobalAveragePooling2D()(x)
x = layers.Dense(512, activation='relu')(x)  # Повнозв'язаний шар
з 512 нейронами та активацією relu
x = layers.Dropout(0.5)(x)#Шар Dropout з ймовірністю викидання 50%
x = layers.Dense(256, activation='relu')(x)  # Повнозв'язаний шар
з 256 нейронами та активацією relu
x = layers.Dropout(0.5)(x)  # Шар Dropout з ймовірністю викидання
50%
x = layers.Dense(128, activation='relu')(x)  # Повнозв'язаний шар
з 128 нейронами та активацією relu
x = layers.Dropout(0.5)(x)  # Шар Dropout з ймовірністю викидання
50%
x = layers.Dense(64, activation='relu')(x)  # Повнозв'язаний шар з
64 нейронами та активацією relu
x = layers.Dropout(0.5)(x)  # Шар Dropout з ймовірністю викидання
50%
predictions = layers.Dense(3, activation='softmax')(x)  # Вихідний
повнозв'язаний шар з 3 нейронами та активацією softmax
model = Model(inputs = resnet_50.input, outputs = predictions)
def trainModel(model, epochs, optimizer):
    batch_size = 16
    model.compile(optimizer=optimizer,
loss="sparse_categorical_crossentropy", metrics=["accuracy"])
    return model.fit(train_generator,
validation_data=val_generator, epochs=epochs,
batch_size=batch_size)
```

При виклику функції з параметрами «epochs = 10» та «optimizer = «Adam»», ми використовуємо алгоритм оптимізації Adam для навчання моделі

протягом 10 епох (лістинг 2.4). *Adam*, що розшифровується як Adaptive Moment Estimation, є одним із найефективніших методів оптимізації для нейронних мереж. Він налаштовує швидкість навчання кожного параметра на основі історії градієнта, і це налаштування допомагає нейронній мережі ефективно навчатися в цілому [31]. Результатом є набір історичних даних про навчання моделі, який ми призначаємо змінній «model_history». Цей набір даних містить інформацію про кожну епоху навчання, включаючи втрату (loss) та точність (accuracy) на навчальному та валідаційному наборах даних (див. лістинг 2.4).

Лістинг 2.4

```
model_history = trainModel(model = model, epochs = 10, optimizer =
"Adam")
Epoch 1/10
293/293 ------------------ 328s 1s/step - accuracy: 0.5208 - loss: 1.2359
- val_accuracy: 0.9192 - val_loss: 0.2222
Epoch 2/10
293/293 ------------------ 322s 1s/step - accuracy: 0.8771 - loss: 0.3688
- val_accuracy: 0.9297 - val_loss: 0.1909
Epoch 3/10
293/293 ------------------ 323s 1s/step - accuracy: 0.9039 - loss: 0.2704
- val_accuracy: 0.9506 - val_loss: 0.1376
Epoch 4/10
293/293 ------------------ 329s 1s/step - accuracy: 0.9173 - loss: 0.2454
- val_accuracy: 0.9604 - val_loss: 0.0965
Epoch 5/10
293/293 ------------------ 315s 1s/step - accuracy: 0.9430 - loss: 0.1849
- val_accuracy: 0.9648 - val_loss: 0.0935
Epoch 6/10
293/293 ------------------ 299s 1s/step - accuracy: 0.9478 - loss: 0.1563
- val_accuracy: 0.9701 - val_loss: 0.0823
Epoch 7/10
293/293 ------------------ 298s 1s/step - accuracy: 0.9501 - loss: 0.1490
- val_accuracy: 0.9701 - val_loss: 0.0747
Epoch 8/10
293/293 ------------------ 298s 1s/step - accuracy: 0.9534 - loss: 0.1399
- val_accuracy: 0.9746 - val_loss: 0.0850
Epoch 9/10
293/293 ------------------ 301s 1s/step - accuracy: 0.9605 - loss: 0.1210
- val_accuracy: 0.9820 - val_loss: 0.0520
Epoch 10/10
293/293 ------------------ 300s 1s/step - accuracy: 0.9659 - loss: 0.1131
- val_accuracy: 0.9858 - val_loss: 0.0499
```

При обробці кожної епохи, що включає в себе 293 партії даних по 16 зображень у кожній, наша модель витрачає приблизно 5 хвилин. До прикладу,

під час першої епохи точність навчального набору склала 52.08%, втрата — 1.2359, а точність валідаційного набору — 91.92%, втрата — 0.1909. На десятій епосі точність навчального набору підвищилася до 96.59%, втрата зменшилася до 0.1131, точність валідаційного набору також підвищилась до 98.58%, а втрата зменшилась — 0.0499.

Крім того, отримані дані можна відобразити графічно, що дозволить легко порівняти динаміку зміни втрат та точності моделі на навчальному та валідаційному наборах даних протягом всіх епох. Графіки втрат та точності допомагають зрозуміти, як модель вчиться на основі наданих даних, та виявити можливі проблеми, які розглядалися у попередньому розділі. Їх раціонально використовувати при великій кількості епох, але у нашому випадку це також є корисним, ми бачимо стабільне падіння втрат та підвищення точності з невеликим відривом між тренувальними та валідаційними даними (рис. 2.5).

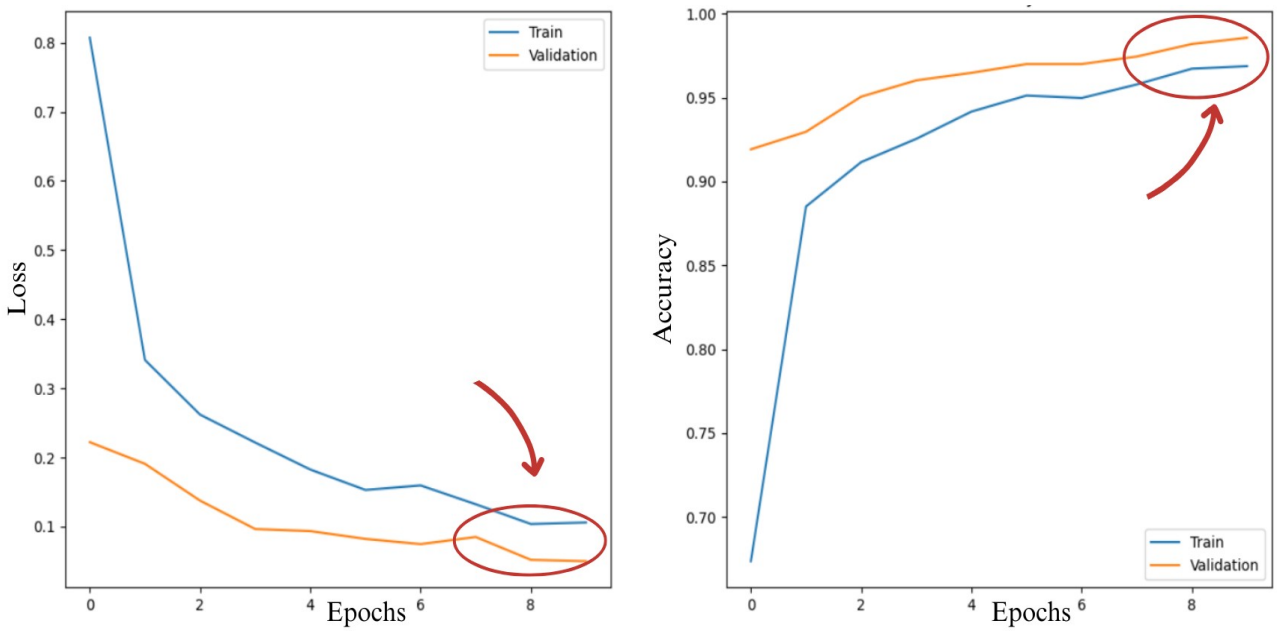

Рис. 2.5. Попередня оцінка навчання: графіки втрат та точності

Після успішного завершення навчання моделі та отримання досить непоганих попередніх результатів, оцінка її остаточної точності стає завершальним кроком. Використовуючи інструменти, доступні в бібліотеці Keras, ми можемо швидко провести цей процес.

Після проведення тестування на 42 партіях даних результати оцінки доступні менше, ніж за хвилину (лістинг 2.5). Середня втрата склала 0.06997781246900558, що вказує на низький рівень помилок у процесі класифікації. Крім того, середня точність моделі на тестових даних, яких вона ще не бачила зовсім, склала 98.66%, що свідчить про високу ефективність у вирішенні поставленої задачі. Ці результати підтверджують готовність моделі для повноцінного застосування у практичному завданні.

Лістинг 2.5

```
test_loss, test_acc = model.evaluate(test_generator)
print("Втрата:", test_loss)
print("Точність:", test_acc * 100)
42/42 ------------------ 35s 820ms/step - accuracy: 0.9915 - loss: 0.0562
Втрата: 0.06997781991958618
Точність: 98.66071343421936
```

РОЗДІЛ 3. ЗАСТОСУВАННЯ МОДЕЛІ ГЛИБОКОГО НАВЧАННЯ ДЛЯ ІДЕНТИФІКАЦІЇ ПОШКОДЖЕНОЇ ЗАБУДОВИ

Глибоке навчання вже застосовувалось для задач ідентифікації пошкодженої забудови [32, 33]. Однак, ці дослідження не включали аналіз міста Маріуполь. Винятком є проєкт UA Damage [34], який займався аналізом міста Маріуполь у 2022-2023 роках, проте цей проект є закритим. На основі порівняння з даними UA Damage можна зробити висновок про суттєві зміни, що відбулись за рік: деякі квартали було знесено, а деякі відбудовано. Тому в даній роботі було вирішено застосувати метод глибокого навчання на основі архітектури ResNet50 для аналізу пошкодженої забудови міста Маріуполь з метою доповнення та розширення наявних результатів.

3.1. Використання моделі глибокого навчання для аналізу пошкодженої забудови у середовищі ArcGIS

Середовище ArcGIS має безліч інструментів глибокого навчання, одним з яких є інструмент «Класифікація об’єктів за допомогою глибокого навчання». Проте, наші спроби використання цього інструменту, який може приймати файл моделі, створеної за допомогою Keras, були обмежені через наявність помилок у програмному забезпеченні.

Внаслідок цього, ми вирішили використовувати альтернативний підхід, щоб досягти бажаного результату. Спочатку ми експортували непозначені окремі плитки об'єктів, які потребували класифікації, застосовуючи відомий спосіб, що використовувався при створенні навчальної вибірки. Після цього ми передали ці плитки безпосередньо до моделі глибокого навчання за допомогою циклу, який здійснював передбачення для кожного зображення та записував його у CSV-файл. В результаті ми отримали таблицю з передбачуваними

класами, яку можна легко приєднати до таблиці атрибутів шару з некласифікованими будівлями як додатковий стовпець у середовищі ArcGIS.

Був розроблений простий інтерфейс на мові Python для полегшення застосування моделі глибокого навчання. За допомогою цього інтерфейсу користувач може завантажити теку з даними та отримати файл з передбаченнями для кожного зображення, перевіривши таблицю безпосередньо у ньому. Інтерфейс дозволяє підвантажити теку з зображеннями, відображаючи кількість елементів у ній, і після натискання кнопки запуску кожне зображення аналізується моделлю глибокого навчання. Процес аналізу супроводжується рядком відображення стану, який показує поточний статус обробки зображень.

Створена таблиця результатів містить три стовпці:

- OBJECTID — ідентифікатор кожного зображення у наборі даних.
- Передбачений клас — містить клас, який був передбачений моделлю для кожного зображення.
- Клас перевірки — залишається порожнім і призначений для подальшого введення користувачем.

Після візуального аналізу результатів користувач має можливість зберегти цю таблицю для подальшого використання (рис. 3.1).

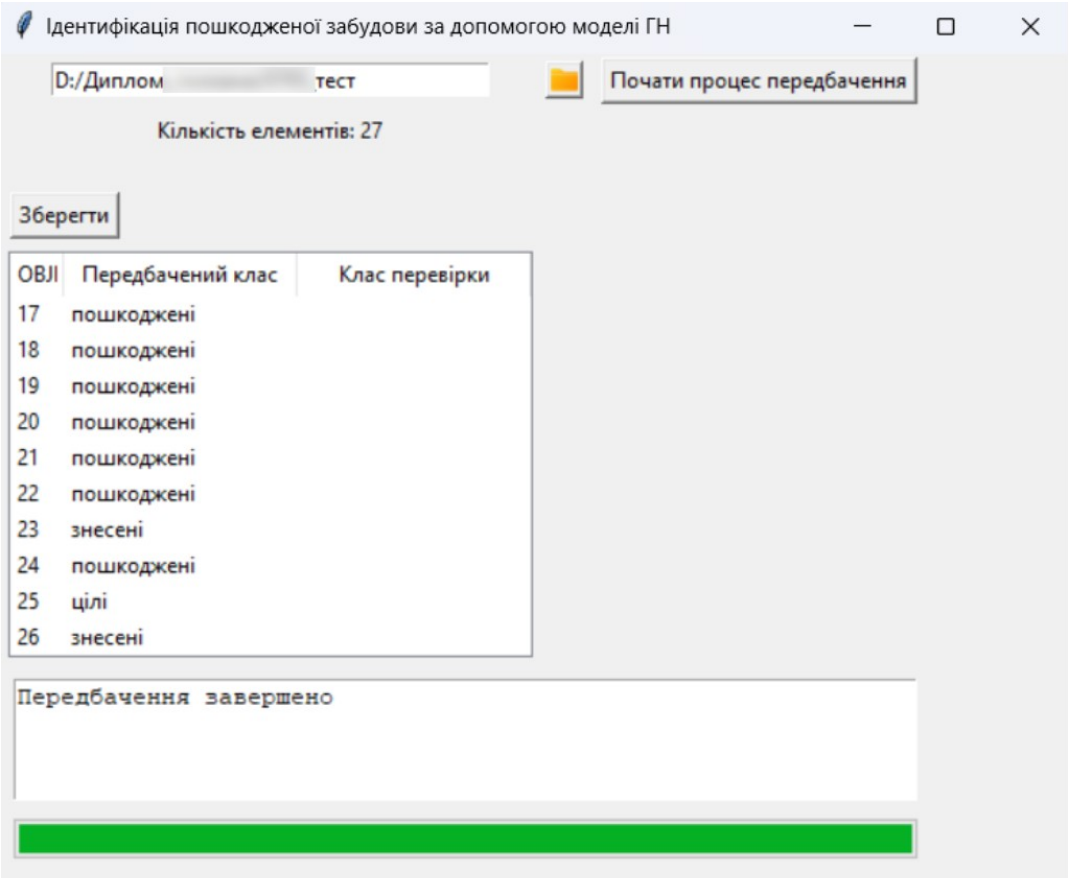

Рисунок 3.1. Інтерфейс користувача для застосування моделі глибокого навчання на підготовлених даних

Після отримання файлу з передбаченнями моделі та використання інструментів ArcGIS ми можемо легко завантажити ці дані у таблицю атрибутів нашого шару з некласифікованими будівлями, щоб провести подальший аналіз (рис. 3.2). Це дозволяє візуалізувати дані, порівняти їх з реальним станом і зробити висновки щодо точності моделі.

Для перевірки точності та ефективності моделі було вибрано 500 будівель Лівобережного району Маріуполя станом на 2024 рік. Цей район, зокрема його західна частина, що знаходиться праворуч від заводу "Азовсталь", зазнав значних пошкоджень через бойові дії. У цій зоні можна спостерігати цілі квартали знесених будинків, також деякі будинки були відновлені, порівнюючи з даними 2023 року.

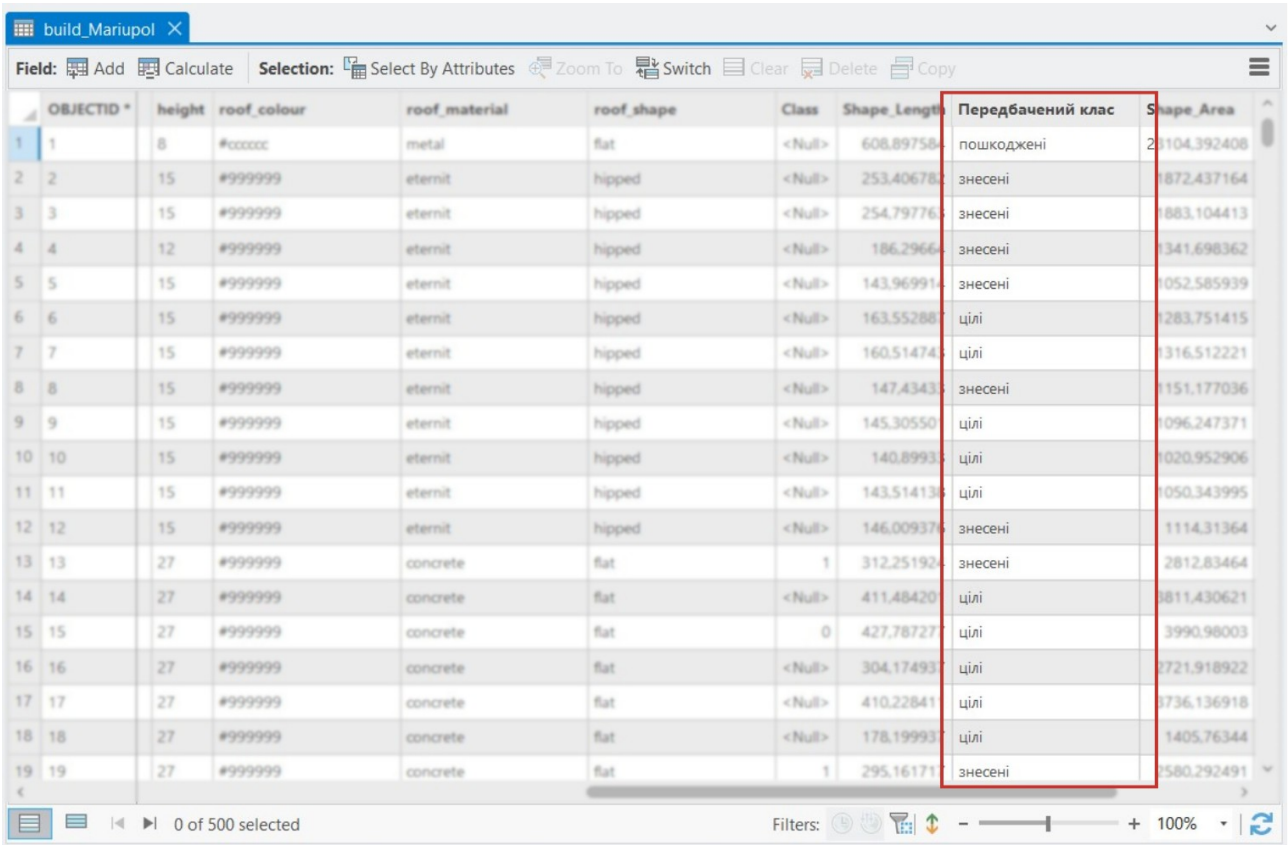

Рисунок 3.2. Підвантаження результатів прогнозування моделі глибокого навчання у середовище ArcGIS

3.2. Аналіз точності. Порівняння реальних та прогнозованих класифікацій

Для оцінки точності прогнозованих результатів ми використовували атрибут "Клас перевірки" та вручну присвоювали фактичний клас кожній будівлі. З 500 об’єктів було проаналізовано та класифіковано 488 будівель, оскільки інші 12 будівель було важко розпізнати на знімках через недостатню роздільну здатність. Це дозволило забезпечити більш точну оцінку результатів класифікації.

Під час аналізу було виявлено 21 відхилення від реальних результатів. Ці відхилення були представлені в матриці плутанини (рис. 3.3), яка допомогла візуалізувати та зрозуміти, які саме типи будівель модель класифікувала неправильно.

Рис. 3.3. Матриця плутанини результатів прогнозування моделі глибокого навчання

Аналізуючи кожен клас окремо, помилки та їх можливі причини розглянуті:

- Помилки у визначенні цілих будівель — в цілому, ці помилки виникли через їх неправильне класифікування як пошкоджені. Це може бути пов'язано зі схожістю текстури даху до пошкодженої або з виникненням незвичних тіней, які сприймаються моделлю як вибухи чи дірки в даху.
- Помилки у визначенні пошкоджених будівель — в основному, помилки полягали у їх невірному класифікуванні як цілих. Це може бути викликано слідами вибухів на даху, які сприймаються як тіні, або

відсутністю частини будівлі, що сприймається моделлю як межа цілої будівлі.

- Помилки у визначенні знесених будівель — лише одна помилка була виявлена у визначенні знесених будівель, ймовірно, через наявність різноманітної текстури на даній території, що може призвести до плутанини для моделі.

Точність моделі оцінювалась за допомогою загальної точності (Overall Accuracy). Загальна точність визначається як відношення правильно класифікованих зразків до загальної кількості зразків. Це дає змогу оцінити загальну ефективність моделі з використанням усіх доступних даних. Формула загальної точності виглядає наступним чином:

$$OA = \frac{\sum_{i=1}^{k} n_{ii}}{n} = \frac{467}{488} = 0.957 \text{ або } 95.7\%, \quad (3.1)$$

де $\sum_{i=1}^{k} n_{ii}$ — слід матриці;

n — загальна кількість зразків.

З підсумків аналізу точності роботи створеної моделі глибокого навчання можна зробити висновок, що найімовірніше, помилки моделі виникають через ускладнення в розрізненні між тінями та слідами вибухів на даху. Це може викликати неправильну класифікацію будівель, особливо в тих випадках, коли вони мають схожі текстури або утворюються складні відблиски та тіні на зображеннях. Незважаючи на це, модель ефективно впоралася з багатьма випадками ідентифікації пошкодженої забудови, що є важливим кроком у діагностиці стану будівельних об'єктів в контексті геоінформаційних систем і використання глибокого навчання.

3.3. Візуалізація отриманих результатів

Згідно з отриманими результатами роботи моделі глибокого навчання, було вирішено створити тематичну карту для наочності та кращого візуального сприйняття (рис. 3.4). У програмному забезпеченні ArcGIS результати класифікації відображено на карті за допомогою режиму відображення даних "Unique Values", що дозволяє окремо показувати та налаштовувати всі значення вибраних полів.

Як зазначалося раніше, найбільших руйнувань зазнала західна частина району. Центральна частина району також має значні пошкодження та невелику кількість знесених будівель, що може бути спричинено інтенсивними бойовими діями. У північній частині району знесені будівлі зустрічаються частково, тоді як у південній частині зафіксовано повністю знесений квартал.

Всього було оцінено 488 будівель. З них:

- 194 будівель залишилися цілими, що складає найбільшу частку серед створених категорій.
- 179 пошкоджених будівель, що складає значну частку об'єктів із різним ступенем руйнувань.
- 115 знесених будівель.

Кількість пошкоджених та знесених будівель разом значно перевищує кількість цілих будівель, що свідчить про масштабні руйнування, які зазнала дана територія. Враховуючи похибки, які виникли у результаті застосування глибокого навчання, висновок про масштаби руйнувань залишається незмінним.

Рис. 3.4. Візуалізація результатів класифікації створеної моделі глибокого навчання на території Лівобережного району Маріуполя

3.4. Аналіз змін забудови Маріуполя з 2023 по 2024 рік

Порівнюючи наші результати з картою руйнувань, представленою проектом UA Damage, ми виявили значні відмінності. Ретельна перевірка показала, що наша модель, яка працювала на знімках 2024 року, не помилилась. Після цього ми проаналізували знімки 2023 року і виявили, що проєкт UA Damage, скоріше за все, використовував знімки 2023. Таким чином, відмінності виникають саме через різний час отримання знімків.

Дані про пошкодження Маріуполя від UA Damage були висвітлені ще у квітні 2022 року, детально описуючи масштаби руйнувань [35]. Хоча нам не відомо точно, чи оновлювалися ці дані після квітня 2022 року, але порівнюючи рівень руйнувань у 2023 році, можна припустити, що дані були оновлені. Оскільки UA Damage не оновлював дані у 2024 році, ми вирішили порівняти ситуацію у 2023 та 2024 роках за наявними знімками. На рис. 3.5 показана карта, аналогічна карті з рис. 3.4, але на ній відмічені лише ті будинки, стан яких змінився порівняно з 2023 роком. Можна спостерігати наступні зміни:

- на півдні міста спостерігається невелика частина відновлених будівель, водночас цілий квартал було знесено, ймовірно, через значні пошкодження, які не підлягали відновленню;
- на північному сході видно значне відновлення забудови, проте деякі поодинокі будинки все ж були знесені;
- у західній частині міста зміни майже не відбулись, відновлено лише декілька будівель.

На основі порівняння з даними UA Damage можна зробити висновок про значні зміни, що відбулися протягом року: деякі квартали було знесено через значні пошкодження, а інші частково або повністю відбудовано.

Рис. 3.5. Візуалізація аналізу змін забудови Маріуполя з 2023 по 2024 рік

ВИСНОВКИ

1. Розроблено модель глибокого навчання на основі ResNet50 для виявлення пошкодженої забудови на космічних знімках Маріуполя 2024 року. Оцінена точність моделі склала на тестових даних – 98,7%, а на реальних даних – 95,7%.
2. Класифіковано пошкоджену забудову Лівобережного району Маріуполя станом на 2024 рік. Значні пошкодження має західна частина, де багато будівель зруйновано або пошкоджено.
3. Побудовано тематичну карту, яка відображає розташування та стан класифікованих будівель. Карту використано для візуалізації даних і полегшення аналізу результатів.
4. На основі порівняння з картою руйнувань Маріуполя від проєкту UA Damage виявлено значні відмінності через різний час отримання знімків: за останній рік деякі квартали було знесено, а інші – частково або повністю відбудовано.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Machine Learning Tutorial. *GeeksforGeeks*: веб-сайт. URL: https://www.geeksforgeeks.org/machine-learning/ (дата звернення: 20.05.2024).
2. What Is Machine Learning? Definition, Types, Applications, and Trends. *Spiceworks*: веб-сайт. URL: https://www.spiceworks.com/tech/artificial-intelligence/articles/what-is-ml/ (дата звернення: 20.05.2024).
3. Deep Learning Algorithms. *The AI Summer*: веб-сайт. URL: https://theaisummer.com/Deep-Learning-Algorithms/ (дата звернення: 20.05.2024).
4. Deep Learning vs. Machine Learning – What's The Difference? *Levity Blog*: веб-сайт. URL: https://levity.ai/blog/difference-machine-learning-deep-learning (дата звернення: 24.05.2024).
5. Deep Learning Algorithms. *Javatpoint*: веб-сайт. URL: https://www.javatpoint.com/deep-learning-algorithms (дата звернення: 23.05.2024).
6. Deep Learning vs. Machine Learning. *Google Cloud*: веб-сайт. URL: https://cloud.google.com/discover/deep-learning-vs-machine-learning (дата звернення: 24.05.2024).
7. Difference Between Machine Learning and Deep Learning. *GeeksforGeeks*: веб-сайт. URL: https://www.geeksforgeeks.org/difference-between-machine-learning-and-deep-learning/ (дата звернення: 25.05.2024).
8. Alzubaidi, L., Zhang, J., Humaidi, A.J. et al. Review of deep learning: concepts, CNN architectures, challenges, applications, future directions. J Big Data 8, 53 (2021). URL: https://doi.org/10.1186/s40537-021-00444-8 (дата звернення: 29.05.2024).
9. Understanding Convolutional Neural Networks: A Beginner's Journey into the Architecture. M*edium*: веб-сайт. URL:

https://medium.com/codex/understanding-convolutional-neural-networks-a-beginners-journey-into-the-architecture-aab30dface10 (дата звернення: 28.05.2024).

10. Activation functions in Neural Networks. *GeeksforGeeks*: веб-сайт. URL: https://www.geeksforgeeks.org/activation-functions-neural-networks/ (дата звернення: 28.05.2024).
11. Т. Д. Козлова, Ю. І. Великодський. Застосування глибокого навчання в геоінформаційних системах. *Аерокосмічна геодезія та землеустрій*: зб. матеріалів доп. учасн. XVI Міжнар. наук.-практ. конф. «ABIA-2023». –К.: НАУ, 2023.
12. Introduction to deep learning. E*sri*: веб-сайт. URL: https://pro.arcgis.com/en/pro-app/latest/help/analysis/deep-learning/what-is-deep-learning-.htm (дата звернення: 25.05.2024).
13. Deep Learning in Geospatial Analysis. M*edium*: веб-сайт. URL: https://medium.com/codex/understanding-convolutional-neural-networks-a-beginners-journey-into-the-architecture-aab30dface10 (дата звернення: 26.05.2024).
14. Machine Learning and Deep Learning for Land Use Classification using ArcGIS Pro. M*edium*: веб-сайт. URL: https://medium.com/@limeira.felipe94/machine-learning-and-deep-learning-for-land-use-classification-using-arcgis-pro-9d304130a51e (дата звернення: 26.05.2024).
15. How to use Learning Curves to Diagnose Machine Learning Model Performance. M*achine Learning Mastery*: веб-сайт. URL: https://machinelearningmastery.com/learning-curves-for-diagnosing-machine-learning-model-performance/ (дата звернення: 28.05.2024).
16. Training and Validation Loss in Deep Learning. *Baeldung*: веб-сайт. URL: https://www.baeldung.com/cs/training-validation-loss-deep-learning (дата звернення: 28.05.2024).

17. Underfitting and Overfitting in Machine Learning. *Baeldung*: веб-сайт. URL: https://www.baeldung.com/cs/ml-underfitting-overfitting#1-cures-for-overfitting (дата звернення: 28.05.2024).
18. Common Problems When Training Deep Learning Models and How to overcome Them. M*edium*: веб-сайт. URL: https://medium.com/@limeira.felipe94/machine-learning-and-deep-learning-for-land-use-classification-using-arcgis-pro-9d304130a51e (дата звернення: 29.05.2024).
19. The Complete Guide on Overfitting and Underfitting in Machine Learning. S*implilearn*: веб-сайт. URL: https://www.simplilearn.com/tutorials/machine-learning-tutorial/overfitting-and-underfitting#what_is_underfitting (дата звернення: 29.05.2024).
20. TensorFlow і машинне навчання. F*oxminded*: веб-сайт. URL: https://foxminded.ua/tensorflow-shcho-tse/ (дата звернення: 30.05.2024).
21. Keras vs. TensorFlow: Understanding the Powerhouse Duo of Deep Learning. S*etronica*: веб-сайт. URL: https://setronica.com/keras-vs-tensorflow-understanding-the-powerhouse-duo-of-deep-learning/?trk=article-ssr-frontend-pulse_little-text-block (дата звернення: 30.05.2024).
22. TensorFlow - Keras. *Tutorialspoint*: веб-сайт. URL: https://www.tutorialspoint.com/index.htm (дата звернення: 31.05.2024).
23. *What is OpenStreetMap? Welcome OpenStreetMap*: веб-сайт. URL: https://welcome.openstreetmap.org/who-is-openstreetmap/ (дата звернення: 31.05.2024).
24. *Overpass-turbo. Overpass turbo*: веб-сайт. URL: https://overpass-turbo.eu/ (дата звернення: 31.05.2024).
25. Use of ImageDataGenerator in CNN and Tensorflow. M*edium*: веб-сайт. URL: https://medium.com/@avinashmachinelearninginfo/use-of-imagedatagenerator-in-cnn-and-tensorflow-a19178c6f457 (дата звернення: 01.06.2024).

26. Keras: The high-level API for TensorFlow. T*ensorflow*: веб-сайт. URL: https://www.tensorflow.org/guide/keras (дата звернення: 01.06.2024).
27. Split your directory automatically to train, validation & test folders. K*aggle*: веб-сайт. URL: https://www.kaggle.com/discussions/general/321868 (дата звернення: 27.05.2024).
28. What is ResNet-50? *Roboflow*: веб-сайт. URL: https://blog.roboflow.com/what-is-resnet-50/ (дата звернення: 24.05.2024).
29. Dropout layer. K*eras*: веб-сайт. URL: https://keras.io/api/layers/regularization_layers/dropout/ (дата звернення: 25.05.2024).
30. Dense layer. K*eras*: веб-сайт. URL: https://keras.io/api/layers/core_layers/dense/ (дата звернення: 20.05.2024).
31. Complete Guide to the Adam Optimization Algorithm. B*uiltin*: веб-сайт. URL: https://builtin.com/machine-learning/adam-optimization (дата звернення: 25.05.2024).
32. КАРТА РУЙНУВАНЬ. Тексти: веб-сайт. URL: https://texty.org.ua/projects/109019/karta-rujnuvan/ (дата звернення: 23.05.2024).
33. Deep Learning for Damage Detection Using Satellite Images. *ELEKS*: веб-сайт. URL: https://eleks.com/research/deep-learning-for-damage-detection-using-satellite-images/ (дата звернення: 24.05.2024).
34. UA Damage. *UA Damage*: веб-сайт. URL: https://www.uadamage.com/ (дата звернення: 23.05.2024).
35. Будуємо Україну разом – Заклик до обміну та створення мереж. *Archined*: веб-сайт. URL: https://www.archined.nl/2022/09/build-ukraine-back-together-a-call-for-exchange-and-creating-networks/ (дата звернення: 03.06.2024).