МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ЛЬВІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ІВАНА ФРАНКА

Факультет прикладної математики та інформатики

(повне найменування назва факультету)

Кафедра дискретного аналізу та інтелектуальних систем

(повна назва кафедри)

Дипломна робота

«Використання нейронних мереж для оцінки воєнних руйнувань»

Виконала: студентка групи ПМІ-43

спеціальності

122 Комп'ютерні науки

(шифр і назва спеціальності)

Шувар С.О.

(підпис) (прізвище та ініціали)

Керівник Квасниця Г.А.

(підпис) (прізвище та ініціали)

Рецензент

(підпис) (прізвище та ініціали)

Львів – 2023

ЛЬВІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ІВАНА ФРАНКА

Факультет Прикладної математики та інформатики _____
Кафедра Дискретного аналізу та інтелектуальних систем _____
Спеціальність 122 «Комп'ютерні науки»_____
(шифр і назва)

«ЗАТВЕРДЖУЮ»
Завідувач кафедри _____

"31 "серпня 2022 року

З А В Д А Н Н Я

НА ДИПЛОМНУ РОБОТУ СТУДЕНТУ

Шувар Софії Орестівни
(прізвище, ім'я, по батькові)

1 .Тема роботи Використання нейронних мереж для оцінки воєнних руйнувань.____

керівник роботи Квасниця Галина Андріївна, канд. Фіз.-мат. наук_____,
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)
затверджені Вченою радою факультету від **" 13" вересня 2022 року № 15** _____

2. Строк подання студентом роботи **13.06.2023р.** _____

3. Вихідні дані до роботи документація по мові програмування Python, бібліотеки: numpy, matplotlib, pandas, shapely, gdal, google_auth_oauthlib, geopy, opencv, albumentations, torch, torchvision, onn, psycopg, розширювач просторової бази даних для об'єктно-реляційної бази даних PostgreSQL - PostGIS та ГІС-програмоа QGIS. Середовище розробки – PyCharm._____

4. Зміст дипломної роботи (перелік питань, які потрібно розробити) Огляд існуючих підходів до оцінки руйнувань в наслідок воєн або інших катаклізмів, підбір технологій для виконання роботи, аналіз та валідація наданих даних, створення даних для тренування моделей, імплементація архітектури класифікатора, імплементація попередньо навчених моделей за допомогою підходу fine-tuning, імплементація модуля навчання нейронних мереж, імплементація модуля застосування натренованих мереж до нових ортофотопланів, імплементація модуля для тестування та порівняння різних архітектур моделей, наповнення бази даних доступними даними про оцифровані місцевості та імплементація методів використання підходу класифікації для цих даних.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)
Схема обраного підходу до вирішення завдання, представлення прикладів зображень різних рівнів руйнування, графічне представлення гістограми розподілу площ для рівнів руйнування 1, 2, 3, схематичне представлення обраного підходу до створення

навчальних даних моделі, схематичне представлення обраної архітектури моделі, представдлення прикладів застосування метидології розширення даних до тренувального набору, схема матриці невідповідностей для даного завдання, графічне представлення зміни значень функції втрат під час навчання на валідаційному наборі, представлень матриць невідповідностей для тестового набору для архітектури DamageClassifier, гістограма для прівняння метрик отриманих на тестовому наборі в наслідок порівняння різних архітектур моделей, приклади застосування підходу.

6. Консультанти розділів роботи

Розділ	**Прізвище, ініціали та посада консультанта**	**Підпис, дата**	
		завдання видав	**завдання прийняв**
1 - 4	Каратаєв М., керівник відділу машинного навчання компанії Covizmo.		

7. Дата видачі завдання **31 серпня 2022 р.**

КАЛЕНДАРНИЙ ПЛАН

№ з/п	**Назва етапів дипломної роботи**	**Строк виконання етапів роботи**	**Примітка**
1.	Вивчення предметної області, та різних підходів до вирішення завдання.	01.10.2022	
2.	Дослідження наданих даних, створення статистик розподілу площ та рівнів зруйнування, дослідження можливих помилок та невідповідностей у даних.	14.10.2022	
3.	Дослідження середовища Qgis та можливостей PyQgis для нарізання ортофотопланів.	01.11.2022	
4.	Розробка модуля для нарізання ортофотопланів за допомогою можливостей бібліотеки OpenCV.	10.11.2022	
5.	Створення модуля для валідації та фільтрації полігонів для отримання збалансованого датасету.	30.11.2022	
6.	Розробка архітектури моделі DamageClassifier.	15.12.2022	
7.	Розробка модуля для навчання за допомогою бібліотеки PyTorch.	07.01.2023	
8.	Створення модуля для застосування отриманих моделей до нових даних.	02.02.2023	
9.	Створення модуля для тестування результатів на тестовому наборі.	07.03.2023	
10.	Імплементація попередньо навчених моделей за допомогою підходу fine-tuning.	21.03.2023	
11.	Тренування 4-рьох архітектур моделей на отриманому тестовому датасеті.	01.04.2023	
12.	Тестування усіх архітектур та порівняння результатів.	08.04.2023	
13.	Створення бази даних у PostGIS, її наповнення та імплементація модуля для застосування підходу класифікації.	29.04.2023	
14.	Оформлення тексту дипломної роботи.	31.05.2023	

Студент _____ _____
(підпис) (прізвище та ініціали)

Керівник роботи_____
(підпис) (прізвище та ініціали)

Реферат

У даній бакалаврській дипломній роботі досліджено тему “Використання нейронних мереж для оцінки воєнних руйнувань”.

Дипломна робота складається зі вступу, чотирьох розділів, висновку та восьми додатків. Обсяг роботи становить 63 сторінок, 14 рисунків та 5 таблиць. Список використаних джерел містить 35 найменувань.

Метою роботи є створення автоматизованої універсальної системи для оцінки рівня руйнування інфраструктур різних типів на ортофотопланах населених пунктів України. Система включає в себе збір даних для детекції будівель на ортофотопланах та створення класифікатора для визначення рівнів руйнування цих будівель.

Під час аналізу існуючих підходів до класифікації руйнувань, спричинених різноманітними катастрофами, були виявлені недоліки та обмеження, які мають велике значення при аналізі власне воєнних руйнувань. Основним завданням диплому було створення автоматизованої системи для оцінки рівнів руйнування, спричинених війною в Україні у 2022 році. Оригінальність цієї роботи полягає у розподілі задачі на локалізацію та застосуванні алгоритмів класифікації з використанням інформації про координати об'єктів. Ця розробка є критично важливою в українському контексті сучасності. Інформація, отримана в результаті роботи алгоритмів, буде використовуватись для отримання репарацій та компенсацій, а також планування відновлення країни.

У результаті роботи було реалізовано систему для оцінки руйнувань з використанням вже зібраних даних про локалізацію та класифікацію руйнувань за допомогою нейронних мереж. Було проведено порівняння результатів класифікатора, отриманих за допомогою чотирьох архітектур нейронних мереж: мережа, натренована з нуля, а також архітектури InceptionV3, MobilenetV2 та ResNet, які були дотреновані з використанням підходу fine-tuning.

Програмна реалізація була виконана мовою програмування Python з використанням бібліотеки Pytorch для імплементації алгоритмів машинного навчання.

Зміст

ВСТУП

24 лютого 2022 року Росія розпочала повномасштабну неспровоковану війну проти України. Невпинні бомбардування нашої країни російськими військовими завдають неймовірних страждань українському народу. Під час війни тисячі людей загинули або були важко пораненими, а мільйони були змушені залишити свої домівки. Станом на кінець січня 2023 року близько 5,4 мільйона українців були внутрішньо переміщеними особами, а 8 мільйонів стали біженцями у Європі [1]. Крім того, значна шкода була завдана інфраструктурі українських міст та сіл: і за час війни вже знищені або пошкоджені сотні шкіл, дитячих садків, лікарень, закладів культури, сотні мостів, десятки аеропортів, та тисячі квадратних метрів житлових будинків.

Незважаючи на те, що український уряд зосереджується на самій війні, завдяки громадським ініціативам та підтримці уряду України, країн Євросоюзу та інших держав в розробці знаходяться проекти з відбудови країни. Дані ініціативи передбачають широкий спектр заходів від судових позовів до міжнародних судів для уніфікованої оцінки заданих руйнувань та фінансів, необхідних для власне відбудови зруйнованого [2]. Дана робота виконана у рамках одного з таких проектів - RebuildUA.

RebuildUA

Аналітичний центр Київської школи економіки KSE Institute, спільно з Офісом Президента України та Міністерством економіки України, запустили проект "Росія заплатить". Метою цього проекту є збір інформації про пошкоджені об'єкти українських міст та об'єкти, які були зруйновані під час війни. Інформація про збитки збирається на основі аналізу свідчень громадян, інформації від уряду та місцевих органів влади щодо втрат та пошкоджень по всій країні, а також на основі публічних джерел та ініціатив з оцифрування високоточних знімків з дронів. До проекту долучилися численні українські компанії та волонтери.

Зібрані дані та оцінка збитків будуть використовуватися для фіксації воєнних злочинів, порушень прав людини, як докази для судових процесів та позовів проти Росії. Дана інформація необхідна для формування та підкріплення

позовів до міжнародних судів, оскільки необхідно надати зведені докази та реєстр пошкоджених об'єктів з оцінкою втрат, відповідно до обґрунтованої методології. Крім того, зібрана інформація використовуватиметься для відшкодування індивідуальних компенсацій та для отримання репарацій та компенсацій від країни-агресора для відновлення та відбудови України.

Для збереження даних про руйнування населених пунктів України та їх незалежного аналізу використовується онлайн-платформа “Azenzus Vision” [3], яка надає доступ до інформації з дронів та супутникових зображень для аналізу візуальних даних. Ця платформа може використовуватись як цифрова система моніторингу, яка дозволяє зберігати, переглядати та аналізувати зображення. Окрім цього, будь-який користувач може ознайомитись зі статистичними даними через сайт https://damaged.in.ua/, де візуально представлено інформацію зі зруйнованих об’єктів, яку було опрацьовано станом на зараз, адже прозорість та публічність опрацьованих даних є важливою складовою даного проекту.

На даний момент методологія оцифрування зруйнованої інфраструктури відбувається наступним чином: спочатку команда проекту RebuildUA здійснює зйомку дронами та створює ортофотоплани для оцифрування населених пунктів. Використання дронів дозволяє зібрати детальні дані про руйнування, включаючи фото та відео об'єктів під різними кутами, з усіх сторін будівель та на низьких висотах. У випадках, коли зйомка дронами неможлива, наприклад, на окупованих територіях або в місцях бойових дій, використовуються супутникові знімки. Після зйомки команда забезпечує збір фото- та відеоматеріалів, а також додаткової інформації з перевірених джерел для забезпечення повноти вихідних даних. ГІС-фахівці на основі ортофотопланів оцифровують всі будівлі, розпізнають зруйновані об'єкти, класифікують типи будівель та визначають рівень руйнувань.

Ці дані завантажуються до платформи “Azenzus Vision”. Волонтери-аннотатори опрацьовують вивантаженні дані, на основі чого отримується первинна інформація про пошкодження територій, де кожному пошкодженому об’єкту присвоюється унікальний код. В результаті формується звіт, в якому до

кожного об’єкта інфраструктури міститься наступна інформація: унікальний код об’єкта, географічні координати об’єкта та ступінь його пошкодження. Дана інформація надалі обробляється та використовується за потреби, наприклад дизайнери та контент-менеджери презентують зібрану інформацію у вигляді публічних інфографічних звітів.

Станом на 05.26.2023, команда проекту вже успішно оцифрувала більше 100 локацій, включаючи більше ніж 254486 будівель, з яких понад 35000 є зруйнованими. [2] Отримані дані та інформація стануть основою для подальшого планування та відновлення інфраструктури на звільнених територіях, а також допоможуть визначити потреби населення та виправити можливі помилки при наданні допомоги.

Актуальність теми та мета

В даний час процес інспекції територій є досить складним та витратним. Команда анотаторів повинна проглянути велику кількість ортофотопланів, оцифрувати кожну будівлю та визначити рівень пошкодження, що вимагає значної кількості часу та фінансових зусиль.

Більш того, оцінка рівня зруйнувань часто залежить від сприйняття власне анотатора, а також є схильною до невідповідностей, пов’язаних з людськими помилками через втому або з контролем якості, тому нерідко є суб'єктивною. Важливо також зазначити, що час, витрачений на розмічання територій, теж залишається проблемою, адже протягом створення анотацій можливі як ремонтні роботи так і подальші руйнування. Отже, існує потреба у автоматизації цього процесу, оскільки це може забезпечити значну економію часу та коштів, а також підвищити точність отриманих даних.

Одним із способів автоматизації процесу є розробка алгоритмів комп'ютерного зору. Такі алгоритми можуть допомогти автоматично визначати рівень зруйнування будівлі, що дозволить підвищити якість та швидкість процесу

оцінки руйнувань. Крім того, такий підхід дозволить значно зменшити кількість людських помилок та підвищити надійність отриманих даних.

Метою роботи є створення автоматизованої універсальної системи для оцінки рівня руйнування інфраструктур різноманітних типів на ортофотопланах населених пунктів України, що включає збір даних для детекції будівель на ортофотопланах та створення класифікатора рівнів руйнування цих будівель.

РОЗДІЛ 1. АНАЛІЗ ПРОБЛЕМАТИКИ ТА ТЕХНОЛОГІЇ

1.1 Проблематика завдання та різні підходи до її вирішення.

Останні роки характеризуються швидким прогресом у галузі машинного навчання для комп'ютерного зору, особливо у розпізнаванні об'єктів та сегментації зображень. Тому такі алгоритми є цілком придатні для автоматичного вилучення інформації із супутникових знімків. Безліч дослідників застосовують такі алгоритми для оцінки руйнувань спричинених природними катаклізмами такими як цунамі, повені, урагани, землетруси, тощо. Зокрема можна виділити роботу Chuyi Wu, який застосував нейронну мережу Siamese для автоматичного виявлення та класифікації будівель з використанням фотографій до та після катастрофи [4]. Також *ELEKS Data Science Office* опублікували статтю, у якій вирішують проблему оцінки рівня руйнувань шляхом застосування алгоритмів сегментації до зображень до катастрофи та алгоритмів класифікації до післявоєнних знімків використовуючи два набори даних: набір даних xView2 xBD і набір даних, анотований вручну за допомогою публічних зображень Google Earth і Maxar, що складається з руйнувань будівель під час російсько-української війни в 2014 році та російського вторгнення в Україну в 2022 році [5].

Аналізуючи вищезгадані роботи та інші схожі розробки можна виділити такі основні підходи для вирішення поставленого завдання:

1. Детекція лише зруйнованих будівель.
2. Використання зображень до катастрофи для сегментації та зображень після для класифікації.
3. Порівняння ортофотопланів до та після катастрофи.
4. Використання підходів навчання без учителя.

Проблематика цих підходів у тому, що дуже часто будівлі, зруйновані внаслідок війни, є знищеними вщент і навіть людині дуже важко впізнати будівлю в руїнах. Рішення про локалізацію об'єкта в таких випадках приймається спираючись на дані від місцевої адміністрації або свідчення мешканців. Також кількість цілих будівель є значно більшою ніж кількість зруйнованих, тому

неможливо створити збалансований, але при тому достатньо великий набір даних. Так як саме зруйновані будівлі несуть найбільшу важливість, нехтування ними не є допустимим, тому перший підхід не є актуальним.

Проблематика наступних двох підходів полягає у наданих даних, оскільки немає якісних знімків усіх територій до повномасштабного вторгнення, або вони є застарілими та можуть не містити великої кількості будівель.

До останнього методу відноситься підхід, що базується на використанні GAN-моделей, для якого характерним є навчання на великій кількості зображень, що містять лише не зруйновані будівлі. Але на даний момент підходи навчання з учителем дають кращі результати.

Таким чином, враховуючи описану специфіку, було прийняте рішення розділити поставлене завдання на завдання детекції об'єктів на ортофотопланах та завдання класифікації, яке включатиме в себе інформацію про локалізацію об'єктів із попереднього завдання. Схема підходу зображена на рисунку 1.1.

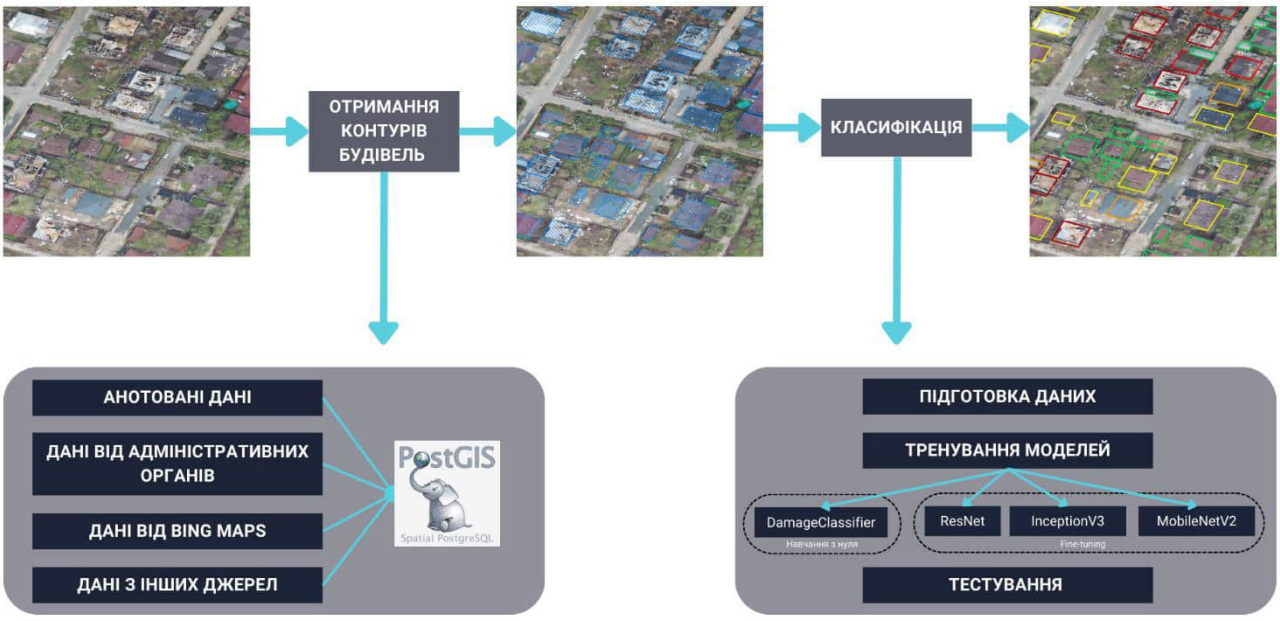


Рисунок 1.1 Схема підходу до вирішення завдання

Оскільки завдання детекції містить проблематику згадану вище, що ускладнює його вирішення за допомогою нейронних мереж, для розв’язання

цього завдання було вирішено об'єднати декілька підходів, а саме дані уже отримані внаслідок роботи анотаторів, дані від доступних відкритих джерел, зокрема Bing Maps, дані від адміністрацій конкретних населених пунктів та дані від інших постачальників. Ці дані зберігаються за допомогою розширювача просторової бази даних для об’єктно-реляційної бази даних PostgreSQL - PostGIS, який додає підтримку географічних об’єктів. З переваг використання PostGIS для поставленого завдання можна виділити наступний функціонал: зберігання просторових даних зокрема багатокутники та мульти-геометрії, як у 2D, так і в 3D; швидкий пошук і отримання просторових даних на основі їх розташування; широкий спектр просторових функцій, які дозволяють фільтрувати та аналізувати просторові дані, вимірювати відстані та площі, тощо; наявність інструментів для обробки геометричних даних і керування ними, наприклад спрощення, перетворення та узагальнення; зберігання та обробка растрових даних, таких як дані про висоту та погоду, геокодування та зворотне геокодування, а також доступ і робота з PostGIS за допомогою сторонніх інструментів, таких як QGIS, GeoServer, ArcGIS, Tableau і MapServer [6].

Щодо завдання класифікації, то для його імплементації доцільно розробити повноцінну систему для класифікації з використанням згорткових нейронних мереж. Вирішення цього завдання варто розділити на етапи: підготовка навчальних даних, тренування моделей та тестування отриманих результатів. Отже, у результаті застосування підходу буде отримано вихідний .geojson файл, що містить інформацію про кожну будівлю на відповідному йому ортофотоплані та інформацію про рівень зруйнування цієї будівлі. Далі буде детально розглянуто кожен етап завдання класифікації.

1.2 Технології

Програмна реалізація була виконана мовою програмування Python, версії 3.10, бібліотеки які використовувались наведені в таблиці 1.1.

Бібліотека	Призначення	Версія
numpy	Бібліотека, що забезпечує роботу з масивами зокрема багатовимірними, а також містить велику колекцію математичних функцій високого рівня для роботи з цими масивами [7]. Використовується для представлення зображень у вигляді багатовимірних масивів і забезпечує швидку векторизовану роботу з цими масивами.	1.24.2
matplotlib	Комплексна бібліотека для створення статичних, анімованих та інтерактивних візуалізацій на Python [8]. Використовується для демонстрації отриманих результатів сегментації та візуалізації графічного представлення порівнянь.	3.7.0
pandas	Бібліотека для маніпуляції та аналізу даних [9]. Використовується для зберігання метричних результатів у вигляді таблиць, а також для збереження та зчитування їх з ".csv" файлів.	1.5.3
shapely	Пакет для теоретико-множинного аналізу та маніпулювання площинними об'єктами за допомогою функцій із широко розповсюдженої бібліотеки GEOS [10]. Використовується для геометричного представлення полігонів, пошуку полігонів що перетинаються, а також для збереження координат полігонів у PostGis.	2.0.1
gdal	Бібліотека для роботи з геопросторовими растровими та векторними даними [11]. Використовується для отримання GPS координат ортофотопланів.	3.6.2
google_auth_oauthlib	Бібліотека, яка забезпечує підтримку протоколу автентифікації OAuth під час роботи з Google API. OAuth 2.0 - це відкритий стандарт авторизації, який дозволяє програмам отримувати доступ до даних користувачів із різних служб без безпосереднього отримання їхніх паролів [12]. Тут Google API необхідне для викачування файлів із сховища.	0.4.1
geopy	Бібліотека, що дозволяє знаходити координати адрес, міст, країн і орієнтирів по всьому світу за допомогою сторонніх геокодувальників та інших джерел даних [13]. Використовується для відстані між географічними об'єктами.	2.3.0
opencv	Відкрита бібліотека комп'ютерного зору, яка призначена для аналізу, класифікації та обробки зображень [14]. Використовується для зчитування зображень, зміни адитивної колірної моделі, зміни розміру зображення, зберігання зображень, а також для застосування алгоритмів обробки зображень.	4.7.0.68
albumentations	Бібліотека комп'ютерного зору, яка підвищує продуктивність глибоких згорткових нейронних мереж [15]. Використовується для застосування різноманітних перетворень до зображень	1.3.0

	перед тренуванням моделі.	
torch	Наукова обчислювальна платформа з широкою підтримкою алгоритмів машинного навчання, яка ставить графічні процесори на перше місце [16]. Використовується для побудови, тренування та тестування нейронних мереж.	1.13.1
torchvision	Бібліотека є частиною Torch. Пакет складається з популярних наборів даних, архітектур моделей і типових перетворень зображень для комп'ютерного зору [17]. Використовується для імплементації попередньо навчених моделей, а також для завантаження даних для тренування.	0.14.1
onnx	Бібліотека для представлення моделей машинного навчання. ONNX визначає загальний набір операторів - будівельних блоків моделей машинного та глибокого навчання і загальний формат файлу [18]. Використовується для експорту моделей.	1.13.0
psycopg	Бібліотека-адаптер бази даних PostgreSQL для мови програмування Python [19].	2.9.6

Таблиця 1.1 Використані технології

РОЗДІЛ 2. ПІДГОТОВКА ДАНИХ

2.1 Зберігання даних

Партнери проекту “RebuildUA” надали дані 69 населених пунктів України, що постраждали від бойових дій. Дані для одного населеного пункту складаються з однієї або кількох пар файлів: ортофотоплану Geotiff формату .tiff з роздільною здатністю в межах від 5 до 7 см на піксель та .geojson файлу з GPS координати та рівнем зруйнування для кожної будівлі на ортофотоплані. Використання GeoTIFF дозволяє зручно працювати з просторовою прив'язкою у формі растрових зображень [20], тоді як GeoJSON надає можливість кодувати різноманітні географічні дані з різними типами геометрії [21]. Обидва ці формати можна легко інтегрувати з ГІС-програмою QGIS та використовувати разом з іншими наборами геопросторових даних. QGIS є потужним і зручним інструментом для роботи з GeoTIFF та GeoJSON, який допомагає в аналізі, візуалізації та керуванні просторовими даними [22].

Всі файли зберігаються у сховищі Google Drive, а окремий Google Sheet файл заповнюється постачальниками у реальному часі та містить всі активні посилання на доступні файли. Оскільки і файли, і основна таблиця заповнюється людьми вручну, усі файли проходять декілька етапів валідації. В цілому допустимими є наступні типи помилок:

1. У таблиці присутнє посилання на файл з невалідним розширенням.
2. Помилка викачування файлу, яка можлива, якщо файл недоступний або був видалений.
3. Помилка читання файлу, що можлива, якщо файл записаний у некоректному форматі.
4. Відсутнє поле “Severity” або “coordinates” в описі певного контура всередині файлу.
5. Рівень зруйнування (“Severity”) не є валідним.
6. GPS-координати виходять за межі .tiff файлу.

Помилки 4, 5, 6 виникають, якщо у файлі присутній хоча б один такий контур. При виявленні будь-якої з цих помилок, невалідні файли відправляються на перезаповнення та перевірку.

В цілому було надано 135 таких пар файлів, які сумарно містять 201976 полігонів, де кожен полігон відповідає контуру будівлі. Окрім вищезгаданих помолок .geojson файли можуть містити дублікати або полігони, які накладаються. Для боротьби з такими контурами застосовувався алгоритм LSH (Locality-sensitive hashing) - це метод дослідження просторової близькості, який використовується для виконання пошуку подібності між багатими на ознаки даними, який зокрема широко використовується під час роботи з ГІС даними [23]. LSH базується на процесі хешування точок даних у групи таким чином, щоб схожі точки, ймовірно, потрапили в одну групу. У використаній інтерпретації процес об’єднання центроїдів багатокутників у сітку можна розглядати як форму кластеризації, коли подібні центроїди групуються в одній комірці сітки. Полігони вважаються дублікатами, якщо значення IOU (Intersection Over Union) перевищує порогове. В такому випадку зберігається полігон із вищим значенням рівня зруйнування та більшою площею. Також на цьому етапі відфільтровуються полігони, які через несуцільність GEOTIFF файлу частково або повністю потрапляють на білий фон.

Рівень зруйнування будівель кодується цілим значенням від 1 до 5. Приклади такої класифікації, а також кількість руйнувань кожного типу, наведені у таблиці 2.1.1.

Класи руйнувань	1	2	3	4	5
	Немає видимих пошкоджень	Ймовірне руйнування	Невеликі пошкодження	Сильне пошкодження	Повністю зруйновано
Фото					
Кількість	180060	10763	5480	1789	3884

Таблиця 2.1.1 Класи руйнувань та їхній розподіл

Оскільки розподіл рівня зруйнованості будинків не є збалансованим, а розмах варіаційного ряду значно перевищує мінімальну варіанту, було вирішено звузити запропонований постачальниками розподіл та збалансувати дані. Таким чином, класи 2 та 3 об'єдналися у клас 2, а класи 4 та 5 у клас 3, з цих полігонів відфільтрувались лише інформативні, тому 184957 полігонів рівнів руйнування 1, 2, 3 були вилучені. В результаті було отримано 3 збалансовані класи руйнувань, а кількість полігонів у кожному класі становить 5673.

2.2 Створення навчальних даних

Оскільки дана робота зосереджена на класифікації пошкоджень кожної будівлі, доцільним є нарізати ортофотоплан на невеликі зображення з будівлею у середині, використовуючи координати контурів кожної будівлі. Оскільки важливо, щоб під час навчання модель навчалась правильно розпізнавати різноманітні ознаки пошкоджень, як от наприклад віконні рами, пошкодження черепиці, різні уламки чи навіть залишки стін, було вирішено при нарізанні ортофотопланів зберегти оригінальну роздільну здатність GEOTIFF файлів. Це було зроблено для того, щоб забезпечити однорідність розмірів і зберегти деталізацію кожної будівлі навіть у випадку великих об'єктів.

Так як обчислювальні машини, які використовувались для навчання є обмеженими, класифікаційний алгоритм доцільно виконувати на зображеннях,

розмір яких лежить в діапазоні від 64 до 1024 пікселів, так як це дозволяє використовувати достатньо великий розмір батчу (кількість зразків, які одночасно передаються в мережу) для швидкого навчання. На рисунку 2.2.1 наведено діаграму розподілу площ усіх будівель окремо для кожного рівня руйнування. Так як кількість будівель, розмір яких перевищує 80000, є малою, вони були вилучені з діаграми, щоб краще проілюструвати тенденцію спаду. Аналізуючи графік бачимо, що зі збільшенням площ кількість будівель стрімко зменшується, а пізніше майже вирівнюється. Так як класи 2 та 3 несуть найбільшу цінність для поставленої задачі, було встановлено розмір зображень для навчання як 256x256 пікселів. Будівлі, розмір яких перевищує вказаний, були розділені на декілька зображень, тому що ці полігони зазвичай відповідають великим промзонам, які і включають в себе декілька будівель.

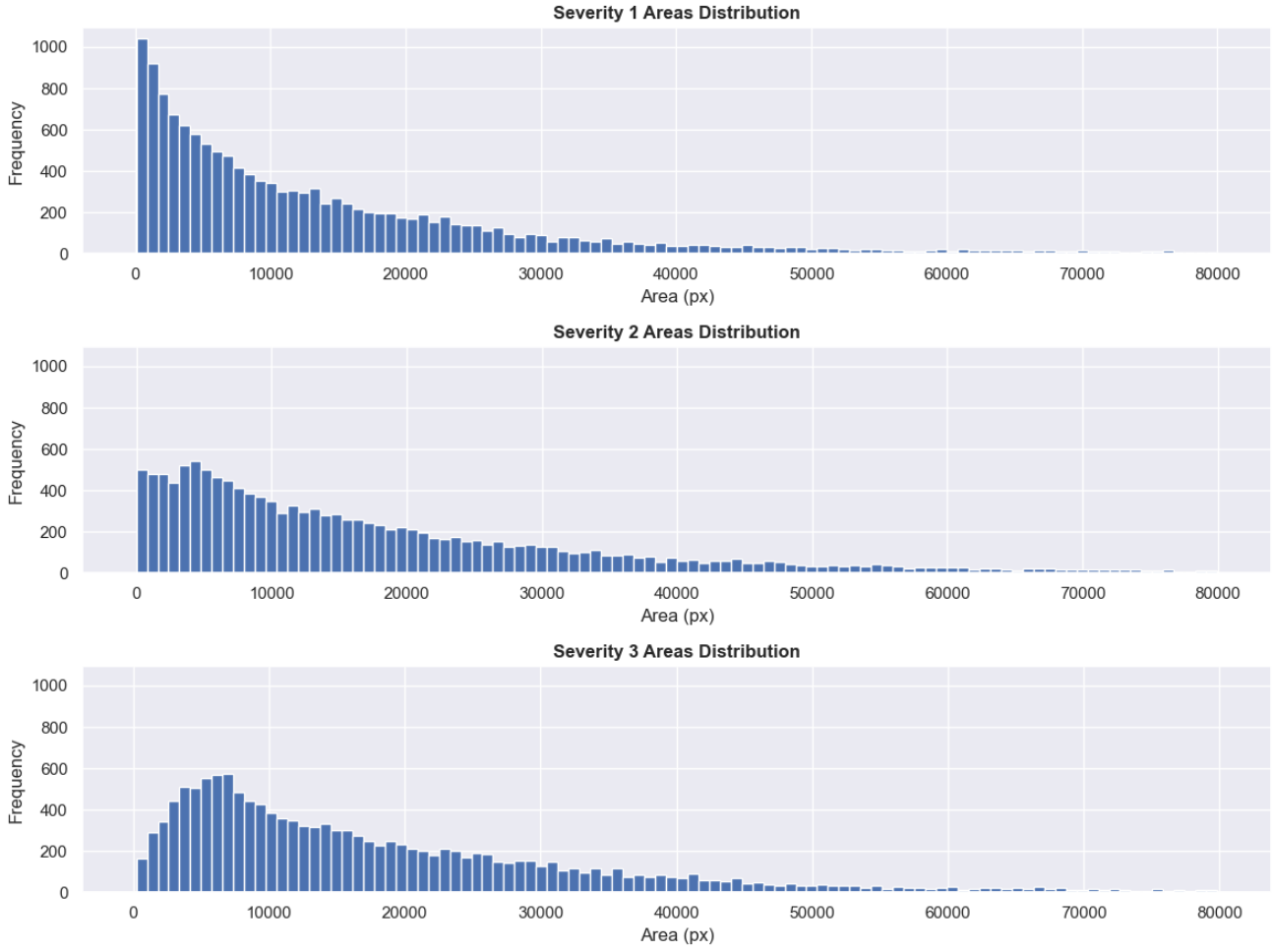


Рисунок 1.2.1 Гістограми розподілу площ будівель у пікселях для різних рівнів руйнування

На додачу, дуже часто оточення будівлі надає критично важливу інформацію та візуальні підказки, які можуть допомогти моделі краще передбачити рівень пошкодження. Однак, якщо включити занадто багато навколишньої території, це збільшує ризик заплутати модель через додавання до неї фрагментів зображення, які містять кілька будівель різного рівня пошкодження, а також інформацію, невідповідну контексту завдання, таку як зелені насадження, пішоходи, автомобілі тощо. Тому для неповного вилучення об’єктів, які не мають відношення до оцінки збитку, але збереження акценту на власне будівлі та різноманітних уламках, що часто потрапляють на фон, було вирішено додати до триканального зображення четвертий - альфа-маску, де центральній будівлі відповідає значення 256, а фону - 40. Значення 40 було вибрано для покращення візуалізацій. Такий підхід повинен допомогти підвищити якість передбачень щодо рівня пошкодження, оскільки модель може краще сконцентруватися на самій будівлі та враховувати тільки важливі деталі. Схема підходу зображена на рисунку 2.2.2.

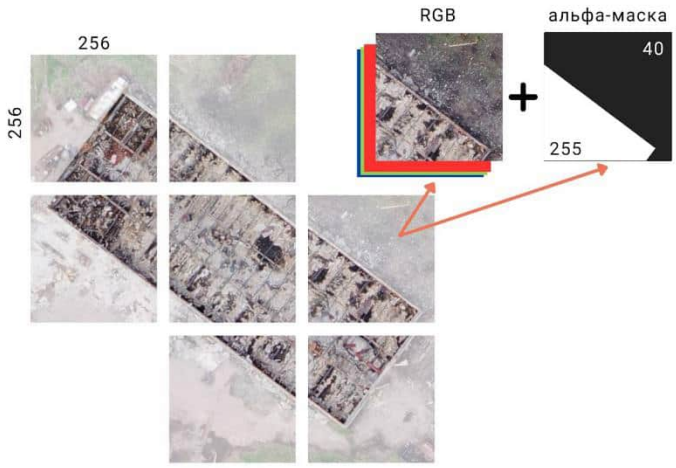


Рисунок 2.2.2 Створення навчальних даних

В результаті для оновлених рівнів зруйнування 1, 2, 3 отримали 6361, 9100, 9996 зображень відповідно. Ці дані використовуються для навчання моделей.

РОЗДІЛ 3. НАВЧАННЯ НЕЙРОННОЇ МЕРЕЖІ

Комп’ютерне бачення (Computer vision) - область дослідження, яка прагне розробити методи, які допомагають комп’ютерам бачити та розуміти вміст візуальних даних, таких як фотографії та відео. Це мультидисциплінарна галузь, яку вважають підгалуззю штучного інтелекту та машинного навчання, яка може передбачати використання спеціалізованих методів і використання загальних алгоритмів навчання [24]. Але після десятиліть досліджень, комп’ютерний зір залишається нерозкритим, принаймні з точки зору відповідності можливостям людського зору. Однією з причин є те, що ми не маємо чіткого уявлення про те, як працює людський зір. Вивчення біологічного зору вимагає розуміння органів сприйняття, таких як очі, а також інтерпретації сприйняття в мозку. Ще одна причина полягає в тому, що конкретний об'єкт можна побачити з будь-якої орієнтації, за будь-яких умов освітлення, з будь-яким типом оклюзії іншими об'єктами тощо [24]. Одним із завдань комп'ютерного зору є завдання класифікації. Його суть полягає у тому, щоб відповісти на запитання до яких з k категорій (класів) належать об'єкти на зображенні. Щоб вирішити це завдання, алгоритму навчання зазвичай пропонують створити функцію $f: \mathbb{R}^n \longrightarrow \{1, \dots k\}$. Коли $y = f(x)$ модель призначає вхідні дані, описані вектором x, до категорії, ідентифікованої числовим кодом y. [25].

Життєвий цикл моделі глибокого навчання створеної завдяки API PyTorch, можна описати наступними кроками:

1. Підготовка даних.
2. Визначення моделі.
3. Навчання моделі.
4. Оцінка моделі.
5. Прогнозування.

У попередньому розділі було розглянуто процес підготовки даних, у цій частині буде висвітлено визначення архітектур моделей та процес тренування.

3.1 Архітектура моделі

Вирішення задач комп'ютерного зору традиційно передбачає використання системи модульних моделей.

Нейронна мережа або модель - метод штучного інтелекту, створений базуючись на структурі та функціонуванні біологічних нейронних мереж. Нейронна мережа машинного навчання складається з нейронів, об'єднаних у шари. З'єднання між нейроном із вхідного шару та нейроном з прихованого шару представлено числом, яке називається вагою: W_i. Спочатку ваги призначаються випадковим чином і пізніше коригуються в процесі навчання. У нейронній мережі прямого зв'язку вихід одного рівня служить входом для наступного рівня. Формули для прямого розповсюдження передбачають застосування обчислення нейронного входу та функції активації до кожного шару в мережі. Функція активації визначає вихід нейрона або шару нейронів на основі вхідних даних, які він отримує. Це вводить нелінійність у мережу, дозволяючи нейронним мережам моделювати складні зв'язки та робити вихід нелінійним відносно вхідного сигналу. Таким чином виходи нейронної мережі, що містить L шарів для кожного шару l від 2 до L та для кожного нейрона j у шарі l, а кожен шар містить N_{l-1} нейронів можна описати так:

$$Output_{Lj} = Activation(w_{l,1,j} \times Output_{l-1,1} + w_{l,2,j} \times Output_{l-1,2} + \cdots + w_{l,N_{l-1},j} \times Output_{l-1,N_{l-1}} + b_{l,j})$$

- $w_{l,1,j}$ представляє вагу між нейроном i у шарі $l-1$ і нейроном j у шарі l.
- $Output_{l-1,i}$ представляє вихід нейрона i на рівні $l-1$.
- $b_{l,j}$ представляє член зміщення нейрона j у шарі l.

Кожна модель була розроблена для певного завдання, наприклад класифікації. Моделі використовуються в конвеєрі з необробленим зображенням на одному кінці та результатом, в даному випадку ймовірністю приналежності до кожного з трьох класів, на іншому кінці [25].

CNN розшифровується як Convolutional Neural Network. Це тип архітектури нейронної мережі, спеціально розроблений для обробки структурованих сіткових даних, тобто таких, що можуть репрезентуватися тензорами, таких як зображення. Він здатен ефективно виявляти та розпізнавати важливі ознаки у вхідних зображеннях подібно до принципу сприйняття зображення людиною, а саме розбиття його на декілька частин. Ця мережа була запропонована Fukushima у 1988 [26].

Нейронна мережа, що використовувалась під час навчання - це архітектура CNN, створена аналогічно до мережі VGG (Visual Geometry Group) [27], що складається з двох основних частин: виділення ознак (feature extraction) і класифікатора. Схематичне представлення цієї моделі зображено на рисунку 3.1.1. Частина виділення ознак відповідає за вивчення представлення вхідного зображення, яке фіксує важливі ознаки для класифікації. Вона складається з кількох згорткових (Convolution) шарів із активацією ReLU, які вивчають просторові особливості зображення, за якими слідують шари максимального об’єднання (Max-pooling), які зменшують дискретизацію ознакових карт (feature maps), щоб зменшити їх просторовий розмір. Batch-нормалізація також застосовується до вихідних даних деяких згорткових шарів для покращення стабільності навчання та продуктивності.

Частина класифікації складається з повністю з’єднаних (fully-connected) шарів, які відображають вивчені ознаки у ймовірності класу. Вихід частини вилучення ознак зрівнюється та проходить через два повністю з’єднані шари з активацією ReLU та batch-нормалізацією. Кінцевий вихідний рівень має 3 вузли і обчислює ймовірності класів.

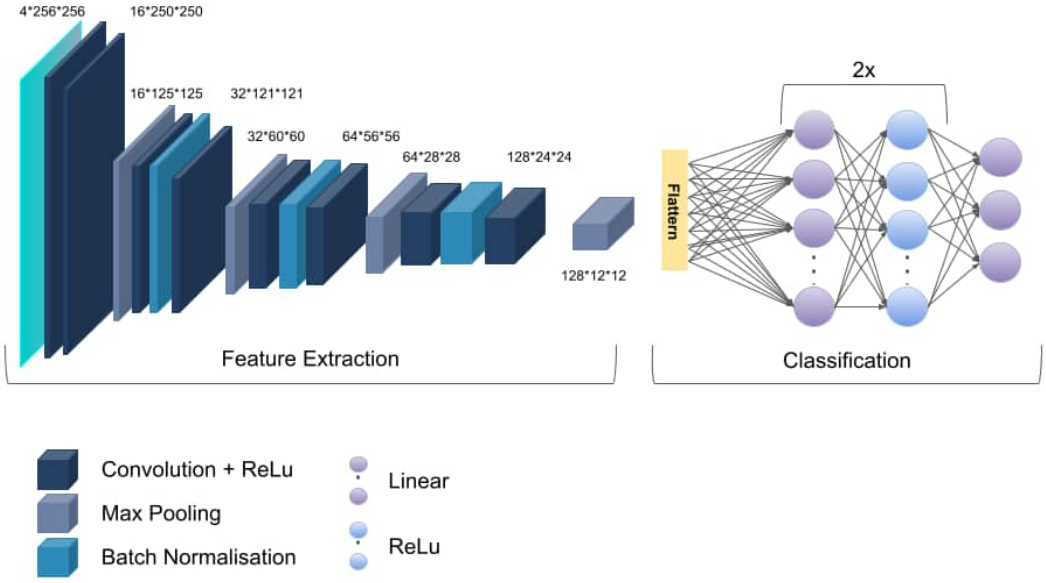


Рисунок 3.1.1 Архітектура моделі

На відміну від оригінальної VGG архітектури, яка характеризується великою кількістю шарів та параметрів, запропонована модель має відносно простішу архітектуру, менше згорткових шарів і меншу кількість параметрів. Простіша архітектура є вигідною в ситуаціях, де набір даних малий. Вона може забезпечити хорошу продуктивність з меншою кількістю параметрів і часу на навчання. Всього модель містить 961299 навчальних параметрів.

Далі буде детально висвітлено кожен шар.

3.1.1 Згортковий шар

Згортковий шар є фундаментальним будівельним блоком згорткових нейронних мереж (CNN). Він призначений для вилучення важливих ознак із вхідних даних. Операція згортки передбачає застосування набору фільтрів до вхідних даних. Кожен фільтр є невеликою матрицю з вагами, які вивчаються в процесі навчання. Фільтр проходить по вхідному тензору з певним кроком, і в кожній позиції виконується поелементне множення між фільтром і вхідними значеннями, результати поелементного множення підсумовуються для отримання єдиного значення. Результат згорткового шару - це тривимірний тензор, відомий як карта ознак, із розмірами (кількість фільтрів, висота виведення, ширина

виходу). Кожен канал на карті ознак відповідає певному фільтру, що фіксує різні шаблони або патерни у вхідних даних.

Операція згортки для одного каналу описується наступною формулою:

$$O(i,j) = \sum_{k=1}^{K} \sum_{l=1}^{K} I(i+k-1, j+l-1) K(k,l)$$

де $K \times K$ - розмір фільтру, $O(i,j)$ значення у i-тому рядку та j-тому стовпці вихідної матриці, $I(i,j)$- значення у i-тому рядку та j-тому стовпці вхідної матриці [25].

Згортковий шар також характеризується такими параметрами як padding, який відповідає за додавання додаткових пікселів або значень навколо вхідних даних перед застосуванням операції згортання. Зазвичай він використовується для збереження просторових розмірів, та stride, який визначає розмір кроку, з яким згортковий фільтр переміщується або ковзає над вхідними даними. Використання більших значень stride може зменшити просторові розміри вихідних ознакових карт, оскільки фільтр пропускає проміжні позиції [24].

3.1.2 Шари максимального об’єднання

Шар максимального об’єднання - це операція зменшення дискретизації. Він працює шляхом поділу вхідних даних на області, що не перекриваються, і вибору максимального значення в кожній області. У наведеній моделі шари maxpooling вставляються після певних згорткових шарів. Ці шари допомагають зменшити розмір ознакових карт і охопити найважливіші ознаки.

Операцію можна описати так:

$$O(i,j) = max_{i \le l \le i+R-1, j \le k \le j+R-1} \{I(l,k)\}$$

де $O(i,j)$ значення у i-тому рядку та j-тому стовпці вихідної матриці, $I(i,j)$- значення у i-тому рядку та j-тому стовпці вхідної матриці.

Важливість цього шару полягає в тому, що: шляхом зменшення дискретизації ознакових карт maxpooling зменшує кількість параметрів і обчислень у наступних шарах, роблячи модель більш ефективною з точки зору обчислень. На додачу, шляхом вибору максимального значення maxpooling зосереджується на найбільш помітних ознаках і зменшує вплив незначних локальних варіацій або шуму [24].

3.1.3 Активація ReLU

Функція активації надає можливість моделювати нелінійні функції, тому вона повинна бути нелінійною. З усіх нелінійних функцій активації, була обрана функція активації ReLU.

ReLU - нелінійна функція активації, що визначається тим, що встановлює від'ємні значення на нуль, зберігаючи додатні значення:

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

де функція лінійна для $x > 0$ і рівна 0, де x - від'ємне.

Було помічено, що CNN з ReLU, як функцією активації, навчається набагато швидше, ніж мережа з іншими функціями активації [28].

3.1.4 Повністю з'єднані шари

Повністю зв'язані шари - це шари, що перетворюють багатовимірну вхідну матрицю у вектор (одновимірний масив). Варто зауважити, що загалом існує більше одного повністю з'єднаного шару, і за кожним із шарів слідує функція активації. Зокрема, останній повністю з'єднаний шар усередині CNN поєднує всі функції з фактично передбаченими об'єктами (рівнем зруйнування 1, 2, 3).

3.1.5 Batch-нормалізація

Batch-нормалізація - це техніка, яка використовується в нейронних мережах для нормалізації вхідних даних кожного рівня шляхом коригування та масштабування активацій. Це допомагає підвищити стабільність і швидкість навчання, дозволяючи мережі навчатися ефективніше. Уточнюючи попереднє

твердження, batch-нормалізація нормалізує вхідні активації шару шляхом віднімання середнього значення та ділення на стандартне відхилення, обчислене для одного батчу. Це допомагає пом’якшити проблему внутрішнього зсуву коваріант, що є явищем, коли розподіл вхідних даних для шару змінюється під час навчання. Завдяки нормалізації вхідних даних batch-нормалізація гарантує, що наступні шари отримають вхідні дані в більш стабільному та узгодженому діапазоні [25].

3.2 Попередня обробка даних

Під час навчання моделі застосовувалась методологія розширення даних - застосування випадкових перетворень до вхідних даних, що є поширеним підходом, завдяки якому модель краще узагальнює нові дані. Методи розширення, які застосовуються до даного набору даних, складаються з горизонтальних і вертикальних обертань, афінних перетворень та змін яскравості та контрасту. Приклад застосування таких перетворень до зображень наведено на рисунку 3.2.1.

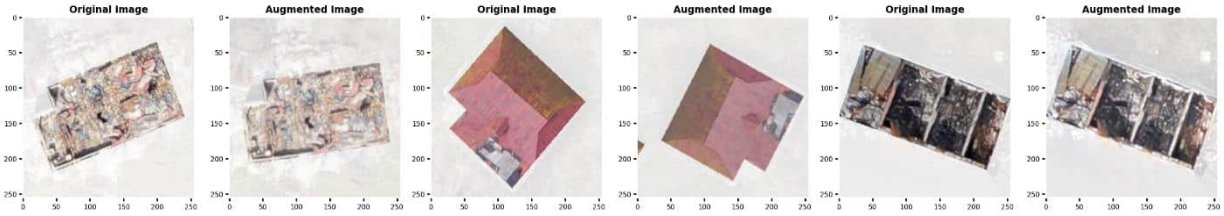


Рисунок 3.2.1. Застосування методології розширення даних до зображень із навчального набору. Зліва оригінальне зображення, справа зображення після застосування випадкових перетворень.

Основна мета доповнення зображення полягає в тому, щоб внести варіативність і різноманітність у навчальні дані. Застосовуючи ці перетворення, доповнені зображення демонструють варіації, які є репрезентативними для сценаріїв реального світу та потенційних варіацій, які можуть виникнути під час тестування.

Під час навчання моделі, важливим є розділення набору даних на навчальний та валідаційний набір - це техніка, яка використовується в машинному навчанні для оцінки та вдосконалення продуктивності моделі під час процесу розробки. Метою такого розділення є надання неупередженої оцінки

ефективності моделі на нових, невідомих даних. Це допомагає запобігти перенавчанню, яке виникає, коли модель стає занадто спеціалізованою для навчальних даних і не може добре узагальнити нові дані. Оцінюючи модель на окремому наборі валідації, можна оцінити її можливості узагальнення та прийняти обґрунтовані рішення щодо покращення її продуктивності. Також значення втрат на валідаційному наборі допомагає у виборі фінальної моделі для тестування [25].

У роботі використовувалось значення розділювача 20%, а це означає, що із 6361, 9100, 9996 зображень для рівнів зруйнування 1, 2, 3, відповідно - 1272, 1820, 1999 зображень становлять валідаційний набір.

3.3 Навчання моделі

Метою згорткових нейронних мереж (CNN) є точна класифікація нових даних у відповідні класи. Щоб досягти цього, необхідною є надійна та точна модель, здатна ідентифікувати навчальні набори даних з мінімальними помилками. Зведення до мінімуму відмінностей між прогнозованими та істинними мітками є поширеним підходом, який часто досягається шляхом мінімізації функції втрат. Однак, оскільки вихідні дані класифікації CNN є категоричними, мінімізація рівняння перехресної ентропії є найкращим варіантом для функції втрат.

Навчання згорткових нейронних мереж передбачає пошук оптимальних параметрів у згорткових шарах. Це досягається за допомогою градієнтного спуску, широко використовуваної техніки для мінімізації функції втрат у нейронних мережах. Зворотне поширення (Backpropagation), частина методу градієнтного спуску, обчислює частинні похідні відносно параметрів, які підлягають навчанню. Оскільки мережі зазвичай складаються з кількох шарів, ланцюгові правила (chain rules) застосовуються ітеративно для обчислення градієнтів.

Значення втрат є важливою мірою ефективності моделі, що представляє відстань між прогнозованими значеннями та справжніми мітками. Бажано

мінімізувати значення втрат до нуля. Навчання проходить через кілька ітерацій, при цьому параметри оновлюються в кожній ітерації, щоб мінімізувати значення втрати. Ініціалізація ваги виконується перед першою ітерацією, забезпечуючи правильне оновлення параметрів шляхом призначення невеликих ненульових випадкових чисел матриці ваг [25].

3.2.1 Функція втрат

Функція втрат - це математичний показник, який кількісно визначає різницю між прогнозованим результатом моделі машинного навчання та справжніми цільовими значеннями. Метою функції втрат є виявлення невідповідності або помилки між прогнозами моделі та істинністю, надаючи єдине скалярне значення, яке можна мінімізувати під час процесу навчання.

Перехресна ентропія - це функція втрат, яка зазвичай використовується в задачах класифікації. Функція визначена різницею між прогнозованим розподілом ймовірностей і справжнім розподілом ймовірностей класів:

$$CrossEntropyLoss = -\sum_i truelabel_i \times \log_e predictedlabel_i$$

де $truelabel_i$ представляє справжню ймовірність або однократно закодований вектор для i-го класу, а $predictedlabel_i$ представляє прогнозовану ймовірність для i-го класу [25].

3.2.2 Зворотне поширення помилки

Зворотне поширення - це алгоритм, який використовується для навчання нейронних мереж шляхом ефективного обчислення градієнтів функції втрат відносно ваг мережі. Це дозволяє мережі вивчати та оновлювати свої параметри на основі помилок, які вона робить під час прямого проходу.

Алгоритм зворотного поширення використовує ланцюгове правило числення для обчислення градієнтів функції втрат. Він працює, поширюючи помилки назад від вихідного рівня до вхідного, регулюючи вагові коефіцієнти на шляху.

У алгоритмі зворотного поширення помилки можна виділити наступні кроки [25]:

- Прямий перехід: вхідні дані передаються через мережу, а активації та виходи кожного рівня обчислюються шар за шаром, доки не буде отримано кінцевий результат.
- Розрахунок втрат: вихід мережі порівнюється зі справжніми цільовими значеннями, а функція втрат використовується для кількісного визначення похибки між прогнозованим і фактичним виходом.
- Зворотний перехід: починаючи з вихідного шару, алгоритм обчислює градієнти функції втрат відносно ваг і зміщень кожного шару. Це робиться шляхом застосування правила ланцюга для обчислення похідної функції втрат відносно активацій кожного рівня.
- Оновлення ваги: після обчислення градієнтів ваги та зміщення кожного шару коригуються за допомогою алгоритму оптимізації, наприклад градієнтного спуску. Градієнти вказують напрямок, у якому слід оновлювати ваги, щоб мінімізувати втрати.
- Зворотне поширення: процес повторюється ітеративно, поширюючи помилки назад по мережі, обчислюючи градієнти та оновлюючи ваги, доки не буде досягнуто конвергенції або критерію зупинки.

3.2.3 Стохастичний градієнтний спуск

SGD - це варіант алгоритму оптимізації градієнтного спуску, який оновлює параметри моделі на основі градієнтів функції втрат, обчислених на випадково вибраній підмножині навчальних даних, відомої як міні-batch. Правило оновлення для SGD можна представити так [25]:

$$\theta(t+1) = \theta(t) - \alpha * \nabla L(\theta(t), x(i), y(i))$$

де:

$\theta(t)$ представляє параметри моделі на ітерації t.

α - швидкість навчання, яка визначає розмір кроку для оновлення параметрів.

$\nabla L(\theta(t), x(i), y(i))$ - це градієнт функції втрат L відносно параметрів моделі θ на ітерації t, обчислений за допомогою одного прикладу навчання $(x(i), y\ (i))$.

У кожній ітерації SGD алгоритм випадковим чином вибирає міні-batch навчальних прикладів і обчислює градієнт функції втрат щодо параметрів, використовуючи ці приклади. Потім він оновлює параметри, віднімаючи добуток швидкості навчання та градієнта від поточних значень параметра. SGD вводить випадковість в оновлення параметрів, що може допомогти уникнути локальних оптимумів і зблизитися до кращого рішення. Крім того, він дозволяє частіше оновлюватись у порівнянні з градієнтним спуском, що робить його обчислювально ефективнішим і придатним для великомасштабних наборів даних.

Швидкість навчання, α, є гіперпараметром, який визначає розмір кроку оновлення параметрів. Він контролює баланс між швидкістю збіжності та стабільністю. Більша швидкість навчання може призвести до швидшої збіжності, але може призвести до перевищення оптимального рішення або нестабільності. З іншого боку, менша швидкість навчання може призвести до повільнішої збіжності, але більш стабільних оновлень.

SGD зазвичай виконується ітеративно протягом кількох епох, де кожна епоха складається з проходження всього навчального набору даних. Алгоритм продовжує оновлювати параметри, доки не буде виконано критерій збіжності, наприклад досягнуто певної кількості ітерацій або покращення функції втрат нижче порогового значення.

3.4 Fine-tuning

Для навчання згорткових нейронних мереж є необхідною велика кількість даних, а також ресурсів для навчання. Мова йде про мільйони зображень, а час навчання може сягати кількох днів.

Одним з поширених підходів, які використовують за відсутності такої кількості тренувальних даних та ресурсів є Fine-tuning. Методологія стосується процесу взяття попередньо-навченої нейронної мережі, яка була навчена на великому наборі даних (такому як ImageNet), і адаптації його до конкретного завдання. Це передбачає повторне використання вивчених ваг попередньо навченої моделі та подальше навчання моделі на новому наборі даних або завданні.[29]

Набір даних ImageNet-1k - це великомасштабний набір даних класифікації зображень, який містить 1281167 навчальних позначених зображень, що належать до 1000 класів. Навчання глибокої нейронної мережі з нуля на такому великому наборі даних вимагає значної кількості обчислювальних ресурсів, часу та позначених даних. Попередньо підготовлені моделі в ImageNet, наприклад ті, що базуються на популярних архітектурах, таких як ResNet, VGG або Inception, тренувалися протягом тривалого часу.[30]

Fine-tuning використовує знання та представлення, отримані попередньо навченою моделлю на ImageNet, і застосовує їх до нового завдання класифікації. Замість випадкової ініціалізації вагових коефіцієнтів нейронної мережі, fine-tuning ініціалізує мережу попередньо навченими ваговими коефіцієнтами. Ці попередньо підготовлені вагові коефіцієнти служать хорошою відправною точкою, оскільки вони вже зафіксували ознаки низького рівня та абстракції вищого рівня з набору даних ImageNet.

Було вирішено порівняти результати отримані внаслідок навчання власної нейронної мережі описаної у попередніх розділах з нуля та частини виділення ознак попередньо навчених мереж: ResNet [31], InceptionV3 [32] і MobileNetV2, - що були дотреновані разом із частиною класифікатора із попередньої моделі. У додатках А, Б, В наведені архітектури цих згорткових нейронних мереж відповідно.

РОЗДІЛ 4. АНАЛІЗ РЕЗУЛЬТАТІВ

Вимірювання ефективності моделі після навчання є вирішальним кроком у оцінці її ефективності та визначенні того, наскільки добре вона узагальнює нові дані.

Такі методи дозволяють оцінити, наскільки добре модель навчилася. Вимірюючи продуктивність, можна отримати уявлення про сильні та слабкі сторони моделі, що дозволяє приймати обґрунтовані рішення щодо її надійності та корисності. Також оцінка продуктивності дає змогу порівнювати різні моделі або варіації однієї моделі. Це допомагає визначити, яка модель краще працює для конкретного завдання чи набору даних, що дозволяє вибрати найбільш прийнятну модель для розгортання. Ще оцінка продуктивності допомагає визначити області, де модель можна покращити. Розуміючи слабкі сторони моделі, можливо точно налаштувати її архітектуру, гіперпараметри або процес навчання для підвищення її продуктивності.

Існують різні способи вимірювання продуктивності навченої моделі. В цій роботі використані лише декілька з них, в наступному розділі буде висвітлено ці методи.

4.1 Методи оцінки отриманих результатів

4.1.1 Матриця невідповідностей

У задачах класифікації матриця невідповідностей - це таблиця, яка підсумовує продуктивність моделі класифікації шляхом представлення прогнозованих і фактичних міток набору даних. Ці матриці є важливою складовою аналізу результатів, коли мова йде про багатокласову класифікацію, зокрема класифікація рівня зруйнування: 1, 2, 3

Матриця невідповідностей для класифікації рівня зруйнування відносно рівня зруйнування 1 зображена на рисунку 4.1.1.1:

Рисунок 4.1.1.1 Матриця невідповідностей

- **True Positive (TP)**: модель правильно передбачила, що приклад належить до певного рівня зруйнування (наприклад, зруйнування 1), коли фактична мітка також відповідає цьому рівню зруйнування.
- **False Negative (FN)**: модель неправильно передбачила, що екземпляр не належить до певного рівня зруйнування (наприклад, зруйнування 1), коли фактична мітка відповідає цьому рівню зруйнування.
- **False Positive (FP)**: модель неправильно передбачила, що екземпляр належить до певного рівня зруйнування (наприклад, зруйнування 1), коли фактична мітка має інший рівень зруйнування.
- **True Negative (TN)**: модель правильно передбачила, що екземпляр не належить до певного рівня зруйнування (наприклад, зруйнування 1), коли фактична мітка також не відповідає цьому рівню зруйнування.

Аналізуючи значення в матриці невідповідностей, можна оцінити продуктивність моделі для кожного рівня зруйнування окремо. А також можна розрахувати такі показники, як точність, запам’ятовування та оцінка F1 для кожного рівня тяжкості на основі значень у матриці. Ці показники дають уявлення про здатність моделі точно класифікувати екземпляри за різними рівнями зруйнування.

4.1.2 Accuracy (точність)

Точність вимірює загальну правильність прогнозів моделі на всіх рівнях зруйнування. Вона розраховується як відношення правильно класифікованих екземплярів до загальної кількості екземплярів.

$$Accuracy = \frac{TP + TN}{TP + FN + FP + TN}$$

4.1.3 Precision

Precision - це показник, який використовується в завданнях класифікації та оцінює частку правильно передбачених екземплярів для певного класу серед усіх екземплярів, передбачених як цей клас. Він пояснює здатність моделі мінімізувати помилкові позитивні прогнози для певного рівня зруйнування.

$$Precision = \frac{TP}{TP + FP}$$

4.1.4 Recall

Recall вимірює частку правильно передбачених випадків для певного рівня серйозності серед усіх фактичних випадків, що належать до цього рівня серйозності. Він спрямований на мінімізацію помилкових негативних прогнозів. Recall показує, наскільки добре модель ідентифікує всі випадки певного рівня зруйнування.

$$Recall = \frac{TP}{TP + FN}$$

4.1.5 F1

Оцінка F1 є середнім гармонійним значенням Precision та Recall. Він фіксує як здатність моделі мінімізувати помилкові позитивні прогнози (precision), так і здатність фіксувати всі випадки певного рівня серйозності (recall). Середнє гармонійне надає більшої ваги нижчим значенням, а це означає, що бал F1 буде низьким, якщо або precision, або recall низькі.

$$F1 = 2 * \frac{Precision \times Recall}{Precision + Recall}$$

4.2 Аналіз та порівняння отриманих результатів

Всі попередньо розглянуті моделі: Damage Classifier (натренований з нуля), ResNet, InceptionV3 і MobileNetV2, - тренувались із початковою швидкістю навчання 0.001. Навчання Damage Classifier з нуля зайняло 12 годин, а моделей, навчених з використанням методу Fine Tuning - приблизно по 6 годин кожна. Графічна карта, що використовувалась для навчання - Nvidia GeForce RTX 3090. На рисунку 4.2.1 наведено графіки, що презентують тенденцію зміни значення втрат на валідаційному наборі в процесі навчання.

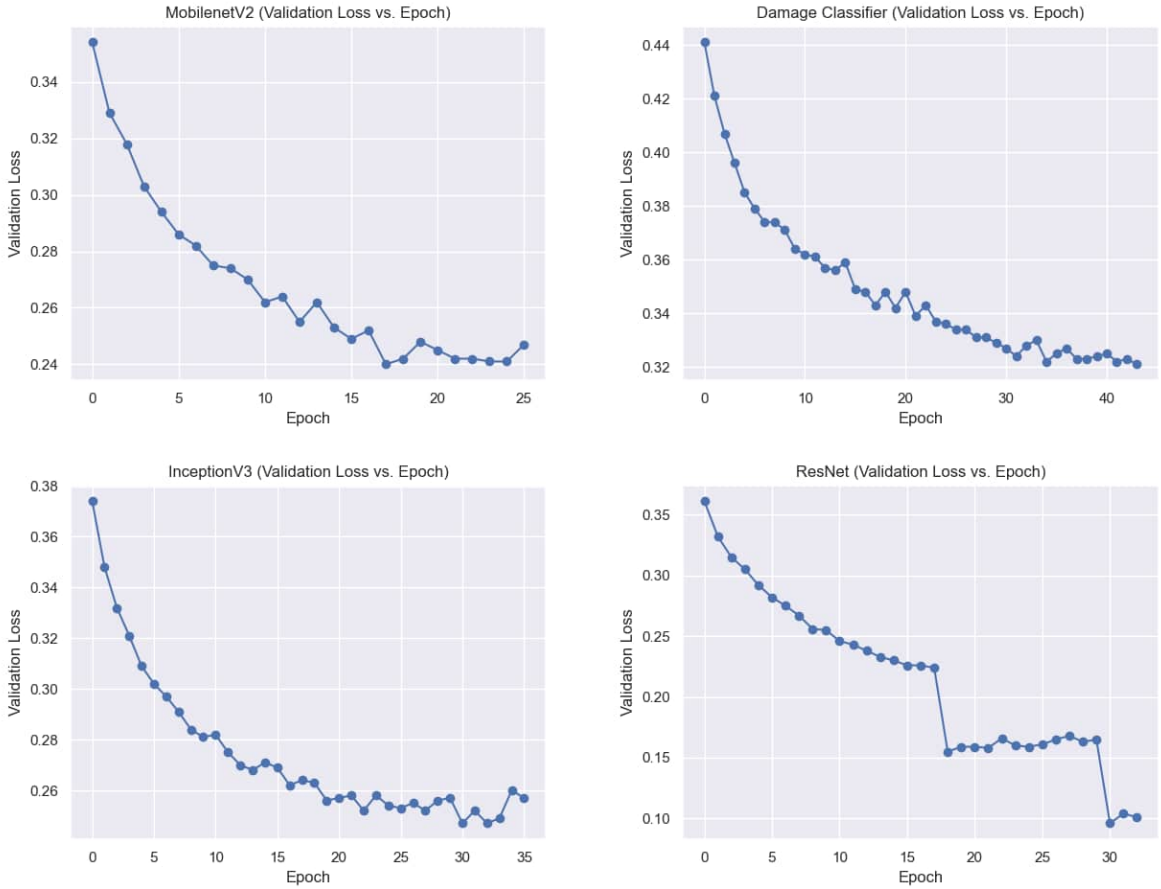


Рисунок 4.2.1 Зміна значень функції втрат під час тренування для валідаційних даних для Damage Classifier, ResNet, InceptionV3 і MobileNetV2 відповідно.

У момент, коли було помічено, що модель починає перенавчатися, було вирішено зупинити тренування. Для кожної моделі цей момент припав на діапазон від 18 до 44 ітерації. Найшвидше достатньо низького значення функції втрат досягла мережа MobileNetV2 на 18 ітерації, а найменше серед усіх значення

втрат на валідаційному наборі досягла модель ResNet - 0.158. Для тестування були обрані моделі, де значення валідаційних втрат є найменшим. В таблиці 4.2.1 наведені обрані моделі.

Архітектура	**Ітерація**	**Значення валідаційних втрат**
Damage Classifier	44	0.321
InceptionV3	33	0.247
MobilenetV2	18	0.240
ResNet	21	0.158

Таблиця 4.2.1 Обрані моделі для тестування

Для тестування підходу використовувались дані із 13 населених пунктів з більш-менш збалансованою класифікацію, що були вилучені з навчальних даних. Оригінальний розподіл рівнів зруйнування на цих територіях наведений в додатку Г.

Матриці невідповідностей виводились для кожного з тестових населених пунктів. Такий підхід дає чітке уявлення про класифікацію на кожному населеному пункті. На рисунку 4.2.2 наведено декілька прикладів матриць невідповідностей для архітектури Damage Classifier. Інші матриці невідповідностей наведені у додатку Д, Е, Є, Ж.

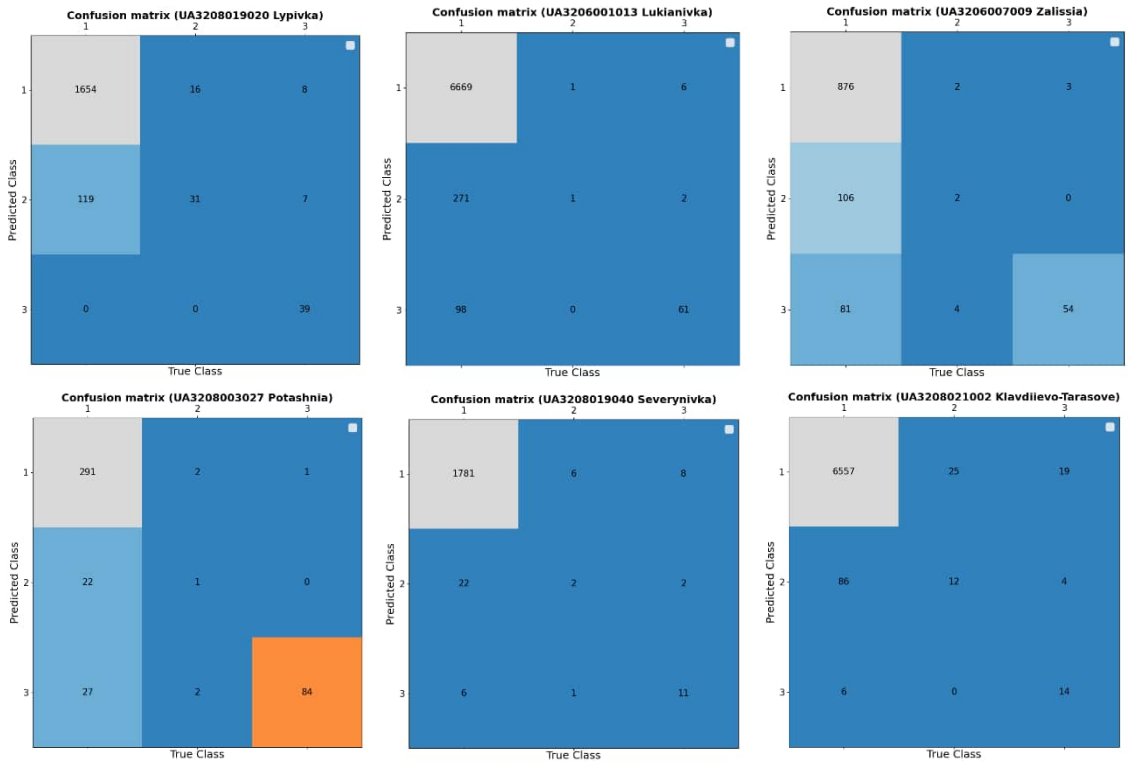


Рисунок 4.2.2 Матриці невідповідностей

Аналізуючи матриці, помітно, що модель добре розпізнає рівні зруйнування 1 та 3, але при тому на деяких прикладах населених пунктів робить невеликі помилки, передбачаючи рівень зруйнування 2.

Для порівняння різних архітектур моделей доцільно використовувати метрики: Accuracy, Precision, Recall та F1. Три останні метрики обчислюються незалежно для кожного рівня руйнування, потім береться їх середнє значення, зважене за кількістю правдивих випадків для кожної мітки. Таким чином забезпечується середньозважена точність для всіх рівнів руйнування. В таблиці 4.2.1 наведено метричні результати для кожної з моделей, а на малюнку 4.2.3 представлена візуалізація цих результатів.

Архітектура	**Accuracy**	**Precision**	**Recall**	**F1**
Damage Classifier	0.954788	0.941404	0.954788	0.947461
InceptionV3	0.945270	0.950096	0.945270	0.947430
MobilenetV2	0.940872	0.939597	0.940872	0.939407
ResNet	0.943870	0.945243	0.943870	0.944521

Таблиця 4.2.1 Метричні результати

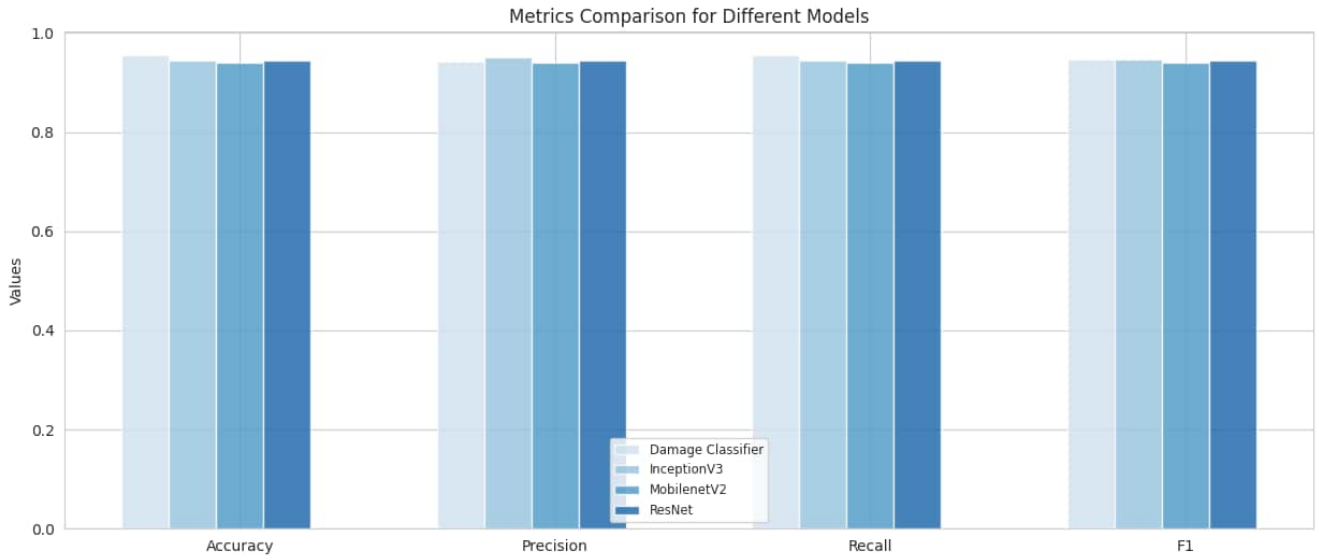


Рисунок 4.2.3 Метрична гістограма

Аналізуючи отримані результати помітно, що найвище значення метрики accuracy показала архітектура DamageClassifier, це вказує на те, що більшу частку зображень було правильно класифіковано за цією моделлю. Найвище значення precision відповідає архітектурі InceptionV3, це означає, що вона краще виконує правильну ідентифікацію позитивних випадків, враховуючи при цьому різні розподіли класів у наборі даних. Тим не менш, усі архітектури показали дуже високі результати передбачень на тестовому наборі, при чому вищеописані метрики обчислені для моделей, навчених з використанням підходу fine-tuning, не мали змістовної переваги відносно моделі, навченої з нуля. Отже, можна зробити висновок, що для того щоб вдосконалити результати, доцільно забезпечити ширший та чистіший навчальний набір.

4.3 Використання системи

Для використання системи, дані усіх .geojson файлів були інтегровані у спільну PostGis базу даних. Також туди були додані дані розмічені компанією Microsoft з відкритого джерела. Ці дані використовуються за відсутності анотованих.

Система розроблена таким чином, щоб класифікувати будівлі за рівнем зруйнування на будь-якому ГІС ортофотоплані .tiff формату.

Оригінальні .tiff-файли мають 4 канали, де 4-й канал містить альфа-маску ортофотоплану. Щоб профільтрувати полігони, які відповідають лише конкретному файлу з бази даних, необхідно за допомогою альфа-маски дістати піксельні координати ортофотоплану. Так як його форма в основному буває зернистою, до 4-ого каналу файлу застосовуються морфологічні операції закриття. Морфологічні операції - це набір прийомів обробки зображень, які змінюють форму та структуру об’єктів у зображенні. Операція закриття, зокрема, складається з двох етапів: розширення з наступною ерозією. Форму та розмір околиці, яка використовується для розширення та ерозії, визначає матриця, яка проходиться по контурах, тим самим згладжуючи їх: у роботі використовується матриця 5x5. Але якщо застосувати таку матрицю до ортофотоплану, то це не призведе до великих покращень, проте, якщо ж збільшити розмір матриці, то внаслідок операції до альфа-маски додається багато зовнішніх додаткових пікселів. Тому впродовж 4-ох кроків альфа-маска ітеративно зменшується вдвічі та до неї застосовується операція морфологічного закриття, потім аналогічно на 4-ох ітераціях маска розширюється з використанням цієї ж операції. На кінець до неї застосовується операція побітового додавання з оригінальним зображенням. Таких підхід забезпечив більш згладжені контори ортофотоплану, до яких можна застосувати операцію бібліотеки OpenCV - findContors(), - яка повертає піксельні координати альфа-маски, які з легкістю переводяться у GPS координати враховуючи інформацію з .tiff файлу. За цими координатами і здійснюється фільтрація бази даних.

Приклади використання наведено на рисунку 4.3.1. Ці приклади слугують підтвердженням метрик з попереднього розділу.

Рисунок 4.3.1 Приклади застосування підходу. Зелений колір - рівень руйнування 1, жовтий колів - рівень руйнування 2, червоний колір - рівень руйнування 3.

ВИСНОВКИ

В результаті виконання цієї роботи була імплементована система для класифікації пошкоджень інфраструктур різноманітних типів на ортофотопланах населених пунктів України, що постраждали від бойових дій. Ця інформація використовуватиметься для оцінки пошкоджень та планування відбудови країни.

Підхід до створення даної системи складається з наступних етапів:

- Підготовка та валідація навчальних даних;
- Розробка архітектури класифікаційної моделі створеної аналогічно до мережі VGG для навчання з нуля на тестовому наборі даних;
- Імплементація архітектур моделей за допомогою підходу fine-tuning для порівняння продуктивності;
- Розробка системи для тестування та порівняння отриманих внаслідок роботи результатів для вибору оптимальної архітектури та власне моделі;
- Інтеграція контурів будівель із різних джерел у спільну PostGis базу даних;
- Розробка системи для модулярного використання системи на довільних ортофотопланах.

Результати, що були отримані в роботі, демонструють, що автоматизована оцінка будівлі за допомогою зображення з дронів є багатообіцяючим підходом, який уже може бути застосовним. Робота була представлена на Всеукраїнській студентській науковій конференції «Наука в Україні: досвід та інновації» [34] та Міжнародній студентській науковій конференції з прикладної математики та комп’ютерних наук “AMICon-2023” [35].

Подальші напрями розвитку цієї роботи можуть включати покращення алгоритмів класифікації, використання більш широкого спектру даних для тренування моделі, включення додаткових параметрів та характеристик для більш деталізованої оцінки пошкоджень, а також інтерфейсу користувача для зручного використання системи. Також можна розглядати можливість розширення системи для автоматизованого виявлення не лише пошкоджень, а й потенційно небезпечних об'єктів або зон, що потребують невідкладного втручання, наприклад розмінування територій.

Загалом, дана робота вносить вагомий внесок у розробку та використання систем для оцінки пошкоджень інфраструктури на основі ортофотопланів. Розробка та результати, отримані внаслідок роботи, є надзвичайно важливими в рамках реалій сьогодення.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. HUMANITARIAN SITUATION IN UKRAINE. (2018, December 5). OCHA. https://www.unocha.org/ukraine/humanitarian-situation-ukraine.
2. RebuildUA | Допомога у відбудові України. (n.d.). RebuildUA - Допомога У Відбудові України. https://rebuildua.net/.
3. Rebuilding Ukraine. (n.d.). Rebuilding Ukraine. https://www.rebuilding-ua.org/.
4. Wu, C., Zhang, F., Xia, J., Xu, Y., Li, G., Xie, J., Du, Z., & Liu, R. (2021, February 28). Building Damage Detection Using U-Net with Attention Mechanism from Pre- and Post-Disaster Remote Sensing Datasets. MDPI. https://doi.org/10.3390/rs13050905.
5. Deep Learning for Damage Detection Using Satellite Images | ELEKS - Software engineering, enterprise software development, consulting. (n.d.). ELEKS - Software Engineering, Enterprise Software Development, Consulting. https://eleks.com/research/deep-learning-for-damage-detection-using-satellite-images/.
6. PostGIS. (n.d.). PostGIS. https://postgis.net/.
7. NumPy. (2023, May 10).
8. Matplotlib — Visualization with Python. (2023, February 14). Matplotlib — Visualization With Python. https://matplotlib.org/ NumPy. https://numpy.org/.
9. pandas - Python Data Analysis Library. (n.d.). Pandas - Python Data Analysis Library. https://pandas.pydata.org/.
10. The Shapely User Manual — Shapely 2.0.1 documentation. (n.d.). The Shapely User Manual — Shapely 2.0.1 Documentation. https://shapely.readthedocs.io/en/stable/manual.html.
11. GDAL — GDAL documentation. (n.d.). GDAL &Mdash; GDAL Documentation. https://gdal.org/.
12. google-auth-oauthlib — google-auth-oauthlib 0.4.1 documentation. (n.d.). Google-auth-oauthlib — Google-auth-oauthlib 0.4.1 Documentation. https://google-auth-oauthlib.readthedocs.io/en/latest/.

13. Welcome to GeoPy’s documentation! — GeoPy 2.3.0 documentation. (n.d.). Welcome to GeoPy’s Documentation! &Mdash; GeoPy 2.3.0 Documentation. https://geopy.readthedocs.io/en/stable/.
14. OpenCV - Open Computer Vision Library. (2022, December 29). OpenCV. https://opencv.org/.
15. Albumentations. (n.d.). Albumentations: Fast and Flexible Image Augmentations. https://albumentations.ai/.
16. PyTorch documentation — PyTorch 2.0 documentation. (n.d.). PyTorch Documentation &Mdash; PyTorch 2.0 Documentation. https://pytorch.org/docs/stable/index.html.
17. torchvision — Torchvision 0.15 documentation. (n.d.). Torchvision &Mdash; Torchvision 0.15 Documentation. https://pytorch.org/vision/stable/index.html.
18. ONNX | Home. (n.d.). ONNX | Home. https://onnx.ai/.
19. PostgreSQL driver for Python — Psycopg. (n.d.). PostgreSQL Driver for Python — Psycopg. https://www.psycopg.org/.
20. GeoTIFF. (n.d.). GeoTIFF. https://trac.osgeo.org/geotiff/.
21. GeoJSON. (n.d.). GeoJSON. https://geojson.org/.
22. Documentation for QGIS 3.28 — QGIS Documentation documentation. (2023, May 25). Documentation for QGIS 3.28 &Mdash; QGIS Documentation Documentation. https://docs.qgis.org/3.28/en/docs/index.html.
23. Pan J., Manocha D. (n.d.). Fast GPU-based Locality Sensitive Hashing for K-Nearest Neighbor Computation.
24. Brownlee J. (2019). Deep Learning for Computer Vision - Image Classification, Object Detection and Face Recognition in Python.
25. Goodfellow I., Bengio Y. Courville A. (2016). Deep Learning.
26. FUKUSHIMA K. (1988). Neocognitron: A Hierarchical Neural Network Capable of Visual Pattern Recognition.

27. Karen Simonyan, Andrew Zisserman (2015). VERY DEEP CONVOLUTIONAL NETWORKS FOR LARGE-SCALE IMAGE RECOGNITION.
28. Krizhevsky A., Sutskever I., Hinton G. E., (2012). ImageNet Classification with Deep Convolutional Neural Networks.
29. Tanveer M. S., Khan U. M. K., Kyung C. (2020, Jun 16). Fine-Tuning DARTS for Image Classification.
30. magenet-1k · Datasets at Hugging Face. (2023, May 22). Imagenet-1k · Datasets at Hugging Face. https://huggingface.co/datasets/imagenet-1k.
31. He K., Zhang X., Ren S., Sun J. (2015, December 10). Deep Residual Learning for Image Recognition.
32. Szegedy C., Vanhoucke V., Ioffe S. Shlens J. (2015, December 11). Rethinking the Inception Architecture for Computer Vision.
33. Sandler M., Howard A. Zhu M., Zhmoginov A., Chen L. (2019, March 21). MobileNetV2: Inverted Residuals and Linear Bottlenecks.
34. (Львів, 2023 р., 25 березня) НАУКА В УКРАЇНІ: ДОСВІД ТА ІННОВАЦІЇ, ЗБІРНИК МАТЕРІАЛІВ І СТУДЕНТСЬКОЇ НАУКОВО-ПРАКТИЧНОЇ КОНФЕРЕНЦІЇ, ВИПУСК 1. https://drive.google.com/file/d/1-Jg3D3NL_h9HfckBtpfhk4sFTXtHzXdQ/view?usp=sharing.
35. (Львів, 2023 р., 4-5 травня) МІЖНАРОДНА СТУДЕНТСЬКА НАУКОВА КОНФЕРЕНЦІЯ ЗПРИКЛАДНОЇ МАТЕМАТИКИ ТА КОМП'ЮТЕРНИХ НАУК МСНКПМКН – 2023. https://ami.lnu.edu.ua/wp-content/uploads/2023/05/ISSCAMCS-2023.pdf.

ДОДАТКИ

Додаток А

Архітектура нейронної мережі ResNet.

```
----------------------------------------------------------------
        Layer (type)               Output Shape         Param #
================================================================
            Conv2d-1         [-1, 64, 128, 128]          12,544
       BatchNorm2d-2         [-1, 64, 128, 128]             128
              ReLU-3         [-1, 64, 128, 128]               0
         MaxPool2d-4           [-1, 64, 64, 64]               0
            Conv2d-5           [-1, 64, 64, 64]           4,096
       BatchNorm2d-6           [-1, 64, 64, 64]             128
              ReLU-7           [-1, 64, 64, 64]               0
            Conv2d-8           [-1, 64, 64, 64]          36,864
       BatchNorm2d-9           [-1, 64, 64, 64]             128
             ReLU-10           [-1, 64, 64, 64]               0
           Conv2d-11          [-1, 256, 64, 64]          16,384
      BatchNorm2d-12          [-1, 256, 64, 64]             512
           Conv2d-13          [-1, 256, 64, 64]          16,384
      BatchNorm2d-14          [-1, 256, 64, 64]             512
             ReLU-15          [-1, 256, 64, 64]               0
       Bottleneck-16          [-1, 256, 64, 64]               0
           Conv2d-17           [-1, 64, 64, 64]          16,384
      BatchNorm2d-18           [-1, 64, 64, 64]             128
             ReLU-19           [-1, 64, 64, 64]               0
           Conv2d-20           [-1, 64, 64, 64]          36,864
      BatchNorm2d-21           [-1, 64, 64, 64]             128
             ReLU-22           [-1, 64, 64, 64]               0
           Conv2d-23          [-1, 256, 64, 64]          16,384
      BatchNorm2d-24          [-1, 256, 64, 64]             512
             ReLU-25          [-1, 256, 64, 64]               0
       Bottleneck-26          [-1, 256, 64, 64]               0
           Conv2d-27           [-1, 64, 64, 64]          16,384
      BatchNorm2d-28           [-1, 64, 64, 64]             128
             ReLU-29           [-1, 64, 64, 64]               0
           Conv2d-30           [-1, 64, 64, 64]          36,864
      BatchNorm2d-31           [-1, 64, 64, 64]             128
             ReLU-32           [-1, 64, 64, 64]               0
           Conv2d-33          [-1, 256, 64, 64]          16,384
      BatchNorm2d-34          [-1, 256, 64, 64]             512
             ReLU-35          [-1, 256, 64, 64]               0
       Bottleneck-36          [-1, 256, 64, 64]               0
           Conv2d-37          [-1, 128, 64, 64]          32,768
      BatchNorm2d-38          [-1, 128, 64, 64]             256
             ReLU-39          [-1, 128, 64, 64]               0
           Conv2d-40          [-1, 128, 32, 32]         147,456
      BatchNorm2d-41          [-1, 128, 32, 32]             256
             ReLU-42          [-1, 128, 32, 32]               0
           Conv2d-43          [-1, 512, 32, 32]          65,536
      BatchNorm2d-44          [-1, 512, 32, 32]           1,024
           Conv2d-45          [-1, 512, 32, 32]         131,072
      BatchNorm2d-46          [-1, 512, 32, 32]           1,024
             ReLU-47          [-1, 512, 32, 32]               0
       Bottleneck-48          [-1, 512, 32, 32]               0
           Conv2d-49          [-1, 128, 32, 32]          65,536
      BatchNorm2d-50          [-1, 128, 32, 32]             256
             ReLU-51          [-1, 128, 32, 32]               0
           Conv2d-52          [-1, 128, 32, 32]         147,456
      BatchNorm2d-53          [-1, 128, 32, 32]             256
             ReLU-54          [-1, 128, 32, 32]               0
           Conv2d-55          [-1, 512, 32, 32]          65,536
      BatchNorm2d-56          [-1, 512, 32, 32]           1,024
             ReLU-57          [-1, 512, 32, 32]               0
       Bottleneck-58          [-1, 512, 32, 32]               0
           Conv2d-59          [-1, 128, 32, 32]          65,536
      BatchNorm2d-60          [-1, 128, 32, 32]             256
             ReLU-61          [-1, 128, 32, 32]               0
           Conv2d-62          [-1, 128, 32, 32]         147,456
      BatchNorm2d-63          [-1, 128, 32, 32]             256
             ReLU-64          [-1, 128, 32, 32]               0
           Conv2d-65          [-1, 512, 32, 32]          65,536
      BatchNorm2d-66          [-1, 512, 32, 32]           1,024
             ReLU-67          [-1, 512, 32, 32]               0
       Bottleneck-68          [-1, 512, 32, 32]               0
```

Conv2d-69	[-1, 128, 32, 32]	65,536
BatchNorm2d-70	[-1, 128, 32, 32]	256
ReLU-71	[-1, 128, 32, 32]	0
Conv2d-72	[-1, 128, 32, 32]	147,456
BatchNorm2d-73	[-1, 128, 32, 32]	256
ReLU-74	[-1, 128, 32, 32]	0
Conv2d-75	[-1, 512, 32, 32]	65,536
BatchNorm2d-76	[-1, 512, 32, 32]	1,024
ReLU-77	[-1, 512, 32, 32]	0
Bottleneck-78	[-1, 512, 32, 32]	0
Conv2d-79	[-1, 256, 32, 32]	131,072
BatchNorm2d-80	[-1, 256, 32, 32]	512
ReLU-81	[-1, 256, 32, 32]	0
Conv2d-82	[-1, 256, 16, 16]	589,824
BatchNorm2d-83	[-1, 256, 16, 16]	512
ReLU-84	[-1, 256, 16, 16]	0
Conv2d-85	[-1, 1024, 16, 16]	262,144
BatchNorm2d-86	[-1, 1024, 16, 16]	2,048
Conv2d-87	[-1, 1024, 16, 16]	524,288
BatchNorm2d-88	[-1, 1024, 16, 16]	2,048
ReLU-89	[-1, 1024, 16, 16]	0
Bottleneck-90	[-1, 1024, 16, 16]	0
Conv2d-91	[-1, 256, 16, 16]	262,144
BatchNorm2d-92	[-1, 256, 16, 16]	512
ReLU-93	[-1, 256, 16, 16]	0
Conv2d-94	[-1, 256, 16, 16]	589,824
BatchNorm2d-95	[-1, 256, 16, 16]	512
ReLU-96	[-1, 256, 16, 16]	0
Conv2d-97	[-1, 1024, 16, 16]	262,144
BatchNorm2d-98	[-1, 1024, 16, 16]	2,048
ReLU-99	[-1, 1024, 16, 16]	0
Bottleneck-100	[-1, 1024, 16, 16]	0
Conv2d-101	[-1, 256, 16, 16]	262,144
BatchNorm2d-102	[-1, 256, 16, 16]	512
ReLU-103	[-1, 256, 16, 16]	0
Conv2d-104	[-1, 256, 16, 16]	589,824
BatchNorm2d-105	[-1, 256, 16, 16]	512
ReLU-106	[-1, 256, 16, 16]	0
Conv2d-107	[-1, 1024, 16, 16]	262,144
BatchNorm2d-108	[-1, 1024, 16, 16]	2,048
ReLU-109	[-1, 1024, 16, 16]	0
Bottleneck-110	[-1, 1024, 16, 16]	0
Conv2d-111	[-1, 256, 16, 16]	262,144
BatchNorm2d-112	[-1, 256, 16, 16]	512
ReLU-113	[-1, 256, 16, 16]	0
Conv2d-114	[-1, 256, 16, 16]	589,824
BatchNorm2d-115	[-1, 256, 16, 16]	512
ReLU-116	[-1, 256, 16, 16]	0
Conv2d-117	[-1, 1024, 16, 16]	262,144
BatchNorm2d-118	[-1, 1024, 16, 16]	2,048
ReLU-119	[-1, 1024, 16, 16]	0
Bottleneck-120	[-1, 1024, 16, 16]	0
Conv2d-121	[-1, 256, 16, 16]	262,144
BatchNorm2d-122	[-1, 256, 16, 16]	512
ReLU-123	[-1, 256, 16, 16]	0
Conv2d-124	[-1, 256, 16, 16]	589,824
BatchNorm2d-125	[-1, 256, 16, 16]	512
ReLU-126	[-1, 256, 16, 16]	0
Conv2d-127	[-1, 1024, 16, 16]	262,144
BatchNorm2d-128	[-1, 1024, 16, 16]	2,048
ReLU-129	[-1, 1024, 16, 16]	0
Bottleneck-130	[-1, 1024, 16, 16]	0
Conv2d-131	[-1, 256, 16, 16]	262,144
BatchNorm2d-132	[-1, 256, 16, 16]	512
ReLU-133	[-1, 256, 16, 16]	0
Conv2d-134	[-1, 256, 16, 16]	589,824
BatchNorm2d-135	[-1, 256, 16, 16]	512
ReLU-136	[-1, 256, 16, 16]	0
Conv2d-137	[-1, 1024, 16, 16]	262,144
BatchNorm2d-138	[-1, 1024, 16, 16]	2,048
ReLU-139	[-1, 1024, 16, 16]	0
Bottleneck-140	[-1, 1024, 16, 16]	0
Conv2d-141	[-1, 512, 16, 16]	524,288
BatchNorm2d-142	[-1, 512, 16, 16]	1,024
ReLU-143	[-1, 512, 16, 16]	0
Conv2d-144	[-1, 512, 8, 8]	2,359,296
BatchNorm2d-145	[-1, 512, 8, 8]	1,024
ReLU-146	[-1, 512, 8, 8]	0

```
          Conv2d-147         [-1, 2048, 8, 8]        1,048,576
     BatchNorm2d-148         [-1, 2048, 8, 8]            4,096
          Conv2d-149          [-1, 2048, 8, 8]       2,097,152
     BatchNorm2d-150          [-1, 2048, 8, 8]            4,096
            ReLU-151          [-1, 2048, 8, 8]                0
      Bottleneck-152          [-1, 2048, 8, 8]                0
          Conv2d-153          [-1, 512, 8, 8]         1,048,576
     BatchNorm2d-154          [-1, 512, 8, 8]             1,024
            ReLU-155          [-1, 512, 8, 8]                 0
          Conv2d-156          [-1, 512, 8, 8]         2,359,296
     BatchNorm2d-157          [-1, 512, 8, 8]             1,024
            ReLU-158          [-1, 512, 8, 8]                 0
          Conv2d-159          [-1, 2048, 8, 8]        1,048,576
     BatchNorm2d-160          [-1, 2048, 8, 8]            4,096
            ReLU-161          [-1, 2048, 8, 8]                0
      Bottleneck-162          [-1, 2048, 8, 8]                0
          Conv2d-163          [-1, 512, 8, 8]         1,048,576
     BatchNorm2d-164          [-1, 512, 8, 8]             1,024
            ReLU-165          [-1, 512, 8, 8]                 0
          Conv2d-166          [-1, 512, 8, 8]         2,359,296
     BatchNorm2d-167          [-1, 512, 8, 8]             1,024
            ReLU-168          [-1, 512, 8, 8]                 0
          Conv2d-169          [-1, 2048, 8, 8]        1,048,576
     BatchNorm2d-170          [-1, 2048, 8, 8]            4,096
            ReLU-171          [-1, 2048, 8, 8]                0
      Bottleneck-172          [-1, 2048, 8, 8]                0
AdaptiveAvgPool2d-173         [-1, 2048, 1, 1]                0
         Flatten-174         [-1, 2048]                       0
          Linear-175          [-1, 256]                 524,544
            ReLU-176          [-1, 256]                       0
     BatchNorm1d-177          [-1, 256]                     512
          Linear-178          [-1, 512]                 131,584
            ReLU-179          [-1, 512]                       0
          Linear-180          [-1, 3]                     1,539
================================================================
Total params: 24,169,347
Trainable params: 24,169,347
Non-trainable params: 0
----------------------------------------------------------------
Input size (MB): 1.00
Forward/backward pass size (MB): 374.29
Params size (MB): 92.20
Estimated Total Size (MB): 467.49
----------------------------------------------------------------
```

Додаток Б

Архітектура нейронної мережі InceptionV3.

```
----------------------------------------------------------------
        Layer (type)               Output Shape         Param #
================================================================
            Conv2d-1         [-1, 32, 127, 127]           1,152
       BatchNorm2d-2         [-1, 32, 127, 127]              64
       BasicConv2d-3         [-1, 32, 127, 127]               0
            Conv2d-4         [-1, 32, 125, 125]           9,216
       BatchNorm2d-5         [-1, 32, 125, 125]              64
       BasicConv2d-6         [-1, 32, 125, 125]               0
            Conv2d-7         [-1, 64, 125, 125]          18,432
       BatchNorm2d-8         [-1, 64, 125, 125]             128
       BasicConv2d-9         [-1, 64, 125, 125]               0
        MaxPool2d-10           [-1, 64, 62, 62]               0
           Conv2d-11           [-1, 80, 62, 62]           5,120
      BatchNorm2d-12           [-1, 80, 62, 62]             160
      BasicConv2d-13           [-1, 80, 62, 62]               0
           Conv2d-14          [-1, 192, 60, 60]         138,240
      BatchNorm2d-15          [-1, 192, 60, 60]             384
      BasicConv2d-16          [-1, 192, 60, 60]               0
        MaxPool2d-17          [-1, 192, 29, 29]               0
           Conv2d-18           [-1, 64, 29, 29]          12,288
      BatchNorm2d-19           [-1, 64, 29, 29]             128
      BasicConv2d-20           [-1, 64, 29, 29]               0
           Conv2d-21           [-1, 48, 29, 29]           9,216
      BatchNorm2d-22           [-1, 48, 29, 29]              96
      BasicConv2d-23           [-1, 48, 29, 29]               0
           Conv2d-24           [-1, 64, 29, 29]          76,800
      BatchNorm2d-25           [-1, 64, 29, 29]             128
      BasicConv2d-26           [-1, 64, 29, 29]               0
           Conv2d-27           [-1, 64, 29, 29]          12,288
      BatchNorm2d-28           [-1, 64, 29, 29]             128
      BasicConv2d-29           [-1, 64, 29, 29]               0
           Conv2d-30           [-1, 96, 29, 29]          55,296
      BatchNorm2d-31           [-1, 96, 29, 29]             192
      BasicConv2d-32           [-1, 96, 29, 29]               0
           Conv2d-33           [-1, 96, 29, 29]          82,944
      BatchNorm2d-34           [-1, 96, 29, 29]             192
      BasicConv2d-35           [-1, 96, 29, 29]               0
           Conv2d-36           [-1, 32, 29, 29]           6,144
      BatchNorm2d-37           [-1, 32, 29, 29]              64
      BasicConv2d-38           [-1, 32, 29, 29]               0
       InceptionA-39          [-1, 256, 29, 29]               0
           Conv2d-40           [-1, 64, 29, 29]          16,384
      BatchNorm2d-41           [-1, 64, 29, 29]             128
      BasicConv2d-42           [-1, 64, 29, 29]               0
           Conv2d-43           [-1, 48, 29, 29]          12,288
      BatchNorm2d-44           [-1, 48, 29, 29]              96
      BasicConv2d-45           [-1, 48, 29, 29]               0
           Conv2d-46           [-1, 64, 29, 29]          76,800
      BatchNorm2d-47           [-1, 64, 29, 29]             128
      BasicConv2d-48           [-1, 64, 29, 29]               0
           Conv2d-49           [-1, 64, 29, 29]          16,384
      BatchNorm2d-50           [-1, 64, 29, 29]             128
      BasicConv2d-51           [-1, 64, 29, 29]               0
           Conv2d-52           [-1, 96, 29, 29]          55,296
      BatchNorm2d-53           [-1, 96, 29, 29]             192
      BasicConv2d-54           [-1, 96, 29, 29]               0
           Conv2d-55           [-1, 96, 29, 29]          82,944
      BatchNorm2d-56           [-1, 96, 29, 29]             192
      BasicConv2d-57           [-1, 96, 29, 29]               0
           Conv2d-58           [-1, 64, 29, 29]          16,384
      BatchNorm2d-59           [-1, 64, 29, 29]             128
      BasicConv2d-60           [-1, 64, 29, 29]               0
       InceptionA-61          [-1, 288, 29, 29]               0
           Conv2d-62           [-1, 64, 29, 29]          18,432
      BatchNorm2d-63           [-1, 64, 29, 29]             128
      BasicConv2d-64           [-1, 64, 29, 29]               0
           Conv2d-65           [-1, 48, 29, 29]          13,824
      BatchNorm2d-66           [-1, 48, 29, 29]              96
      BasicConv2d-67           [-1, 48, 29, 29]               0
           Conv2d-68           [-1, 64, 29, 29]          76,800
      BatchNorm2d-69           [-1, 64, 29, 29]             128
      BasicConv2d-70           [-1, 64, 29, 29]               0
           Conv2d-71           [-1, 64, 29, 29]          18,432
```

BatchNorm2d-72	[-1, 64, 29, 29]	128
BasicConv2d-73	[-1, 64, 29, 29]	0
Conv2d-74	[-1, 96, 29, 29]	55,296
BatchNorm2d-75	[-1, 96, 29, 29]	192
BasicConv2d-76	[-1, 96, 29, 29]	0
Conv2d-77	[-1, 96, 29, 29]	82,944
BatchNorm2d-78	[-1, 96, 29, 29]	192
BasicConv2d-79	[-1, 96, 29, 29]	0
Conv2d-80	[-1, 64, 29, 29]	18,432
BatchNorm2d-81	[-1, 64, 29, 29]	128
BasicConv2d-82	[-1, 64, 29, 29]	0
InceptionA-83	[-1, 288, 29, 29]	0
Conv2d-84	[-1, 384, 14, 14]	995,328
BatchNorm2d-85	[-1, 384, 14, 14]	768
BasicConv2d-86	[-1, 384, 14, 14]	0
Conv2d-87	[-1, 64, 29, 29]	18,432
BatchNorm2d-88	[-1, 64, 29, 29]	128
BasicConv2d-89	[-1, 64, 29, 29]	0
Conv2d-90	[-1, 96, 29, 29]	55,296
BatchNorm2d-91	[-1, 96, 29, 29]	192
BasicConv2d-92	[-1, 96, 29, 29]	0
Conv2d-93	[-1, 96, 14, 14]	82,944
BatchNorm2d-94	[-1, 96, 14, 14]	192
BasicConv2d-95	[-1, 96, 14, 14]	0
InceptionB-96	[-1, 768, 14, 14]	0
Conv2d-97	[-1, 192, 14, 14]	147,456
BatchNorm2d-98	[-1, 192, 14, 14]	384
BasicConv2d-99	[-1, 192, 14, 14]	0
Conv2d-100	[-1, 128, 14, 14]	98,304
BatchNorm2d-101	[-1, 128, 14, 14]	256
BasicConv2d-102	[-1, 128, 14, 14]	0
Conv2d-103	[-1, 128, 14, 14]	114,688
BatchNorm2d-104	[-1, 128, 14, 14]	256
BasicConv2d-105	[-1, 128, 14, 14]	0
Conv2d-106	[-1, 192, 14, 14]	172,032
BatchNorm2d-107	[-1, 192, 14, 14]	384
BasicConv2d-108	[-1, 192, 14, 14]	0
Conv2d-109	[-1, 128, 14, 14]	98,304
BatchNorm2d-110	[-1, 128, 14, 14]	256
BasicConv2d-111	[-1, 128, 14, 14]	0
Conv2d-112	[-1, 128, 14, 14]	114,688
BatchNorm2d-113	[-1, 128, 14, 14]	256
BasicConv2d-114	[-1, 128, 14, 14]	0
Conv2d-115	[-1, 128, 14, 14]	114,688
BatchNorm2d-116	[-1, 128, 14, 14]	256
BasicConv2d-117	[-1, 128, 14, 14]	0
Conv2d-118	[-1, 128, 14, 14]	114,688
BatchNorm2d-119	[-1, 128, 14, 14]	256
BasicConv2d-120	[-1, 128, 14, 14]	0
Conv2d-121	[-1, 192, 14, 14]	172,032
BatchNorm2d-122	[-1, 192, 14, 14]	384
BasicConv2d-123	[-1, 192, 14, 14]	0
Conv2d-124	[-1, 192, 14, 14]	147,456
BatchNorm2d-125	[-1, 192, 14, 14]	384
BasicConv2d-126	[-1, 192, 14, 14]	0
InceptionC-127	[-1, 768, 14, 14]	0
Conv2d-128	[-1, 192, 14, 14]	147,456
BatchNorm2d-129	[-1, 192, 14, 14]	384
BasicConv2d-130	[-1, 192, 14, 14]	0
Conv2d-131	[-1, 160, 14, 14]	122,880
BatchNorm2d-132	[-1, 160, 14, 14]	320
BasicConv2d-133	[-1, 160, 14, 14]	0
Conv2d-134	[-1, 160, 14, 14]	179,200
BatchNorm2d-135	[-1, 160, 14, 14]	320
BasicConv2d-136	[-1, 160, 14, 14]	0
Conv2d-137	[-1, 192, 14, 14]	215,040
BatchNorm2d-138	[-1, 192, 14, 14]	384
BasicConv2d-139	[-1, 192, 14, 14]	0
Conv2d-140	[-1, 160, 14, 14]	122,880
BatchNorm2d-141	[-1, 160, 14, 14]	320
BasicConv2d-142	[-1, 160, 14, 14]	0
Conv2d-143	[-1, 160, 14, 14]	179,200
BatchNorm2d-144	[-1, 160, 14, 14]	320
BasicConv2d-145	[-1, 160, 14, 14]	0
Conv2d-146	[-1, 160, 14, 14]	179,200
BatchNorm2d-147	[-1, 160, 14, 14]	320
BasicConv2d-148	[-1, 160, 14, 14]	0
Conv2d-149	[-1, 160, 14, 14]	179,200

```
BatchNorm2d-150          [-1, 160, 14, 14]             320
BasicConv2d-151          [-1, 160, 14, 14]               0
     Conv2d-152          [-1, 192, 14, 14]         215,040
BatchNorm2d-153          [-1, 192, 14, 14]             384
BasicConv2d-154          [-1, 192, 14, 14]               0
     Conv2d-155          [-1, 192, 14, 14]         147,456
BatchNorm2d-156          [-1, 192, 14, 14]             384
BasicConv2d-157          [-1, 192, 14, 14]               0
 InceptionC-158          [-1, 768, 14, 14]               0
     Conv2d-159          [-1, 192, 14, 14]         147,456
BatchNorm2d-160          [-1, 192, 14, 14]             384
BasicConv2d-161          [-1, 192, 14, 14]               0
     Conv2d-162          [-1, 160, 14, 14]         122,880
BatchNorm2d-163          [-1, 160, 14, 14]             320
BasicConv2d-164          [-1, 160, 14, 14]               0
     Conv2d-165          [-1, 160, 14, 14]         179,200
BatchNorm2d-166          [-1, 160, 14, 14]             320
BasicConv2d-167          [-1, 160, 14, 14]               0
     Conv2d-168          [-1, 192, 14, 14]         215,040
BatchNorm2d-169          [-1, 192, 14, 14]             384
BasicConv2d-170          [-1, 192, 14, 14]               0
     Conv2d-171          [-1, 160, 14, 14]         122,880
BatchNorm2d-172          [-1, 160, 14, 14]             320
BasicConv2d-173          [-1, 160, 14, 14]               0
     Conv2d-174          [-1, 160, 14, 14]         179,200
BatchNorm2d-175          [-1, 160, 14, 14]             320
BasicConv2d-176          [-1, 160, 14, 14]               0
     Conv2d-177          [-1, 160, 14, 14]         179,200
BatchNorm2d-178          [-1, 160, 14, 14]             320
BasicConv2d-179          [-1, 160, 14, 14]               0
     Conv2d-180          [-1, 160, 14, 14]         179,200
BatchNorm2d-181          [-1, 160, 14, 14]             320
BasicConv2d-182          [-1, 160, 14, 14]               0
     Conv2d-183          [-1, 192, 14, 14]         215,040
BatchNorm2d-184          [-1, 192, 14, 14]             384
BasicConv2d-185          [-1, 192, 14, 14]               0
     Conv2d-186          [-1, 192, 14, 14]         147,456
BatchNorm2d-187          [-1, 192, 14, 14]             384
BasicConv2d-188          [-1, 192, 14, 14]               0
 InceptionC-189          [-1, 768, 14, 14]               0
     Conv2d-190          [-1, 192, 14, 14]         147,456
BatchNorm2d-191          [-1, 192, 14, 14]             384
BasicConv2d-192          [-1, 192, 14, 14]               0
     Conv2d-193          [-1, 192, 14, 14]         147,456
BatchNorm2d-194          [-1, 192, 14, 14]             384
BasicConv2d-195          [-1, 192, 14, 14]               0
     Conv2d-196          [-1, 192, 14, 14]         258,048
BatchNorm2d-197          [-1, 192, 14, 14]             384
BasicConv2d-198          [-1, 192, 14, 14]               0
     Conv2d-199          [-1, 192, 14, 14]         258,048
BatchNorm2d-200          [-1, 192, 14, 14]             384
BasicConv2d-201          [-1, 192, 14, 14]               0
     Conv2d-202          [-1, 192, 14, 14]         147,456
BatchNorm2d-203          [-1, 192, 14, 14]             384
BasicConv2d-204          [-1, 192, 14, 14]               0
     Conv2d-205          [-1, 192, 14, 14]         258,048
BatchNorm2d-206          [-1, 192, 14, 14]             384
BasicConv2d-207          [-1, 192, 14, 14]               0
     Conv2d-208          [-1, 192, 14, 14]         258,048
BatchNorm2d-209          [-1, 192, 14, 14]             384
BasicConv2d-210          [-1, 192, 14, 14]               0
     Conv2d-211          [-1, 192, 14, 14]         258,048
BatchNorm2d-212          [-1, 192, 14, 14]             384
BasicConv2d-213          [-1, 192, 14, 14]               0
     Conv2d-214          [-1, 192, 14, 14]         258,048
BatchNorm2d-215          [-1, 192, 14, 14]             384
BasicConv2d-216          [-1, 192, 14, 14]               0
     Conv2d-217          [-1, 192, 14, 14]         147,456
BatchNorm2d-218          [-1, 192, 14, 14]             384
BasicConv2d-219          [-1, 192, 14, 14]               0
 InceptionC-220          [-1, 768, 14, 14]               0
     Conv2d-221          [-1, 192, 14, 14]         147,456
BatchNorm2d-222          [-1, 192, 14, 14]             384
BasicConv2d-223          [-1, 192, 14, 14]               0
     Conv2d-224          [-1, 320, 6, 6]           552,960
BatchNorm2d-225          [-1, 320, 6, 6]               640
BasicConv2d-226          [-1, 320, 6, 6]                 0
     Conv2d-227          [-1, 192, 14, 14]         147,456
```

```
      BatchNorm2d-228          [-1, 192, 14, 14]             384
      BasicConv2d-229          [-1, 192, 14, 14]               0
           Conv2d-230          [-1, 192, 14, 14]         258,048
      BatchNorm2d-231          [-1, 192, 14, 14]             384
      BasicConv2d-232          [-1, 192, 14, 14]               0
           Conv2d-233          [-1, 192, 14, 14]         258,048
      BatchNorm2d-234          [-1, 192, 14, 14]             384
      BasicConv2d-235          [-1, 192, 14, 14]               0
           Conv2d-236            [-1, 192, 6, 6]         331,776
      BatchNorm2d-237            [-1, 192, 6, 6]             384
      BasicConv2d-238            [-1, 192, 6, 6]               0
       InceptionD-239           [-1, 1280, 6, 6]               0
           Conv2d-240            [-1, 320, 6, 6]         409,600
      BatchNorm2d-241            [-1, 320, 6, 6]             640
      BasicConv2d-242            [-1, 320, 6, 6]               0
           Conv2d-243            [-1, 384, 6, 6]         491,520
      BatchNorm2d-244            [-1, 384, 6, 6]             768
      BasicConv2d-245            [-1, 384, 6, 6]               0
           Conv2d-246            [-1, 384, 6, 6]         442,368
      BatchNorm2d-247            [-1, 384, 6, 6]             768
      BasicConv2d-248            [-1, 384, 6, 6]               0
           Conv2d-249            [-1, 384, 6, 6]         442,368
      BatchNorm2d-250            [-1, 384, 6, 6]             768
      BasicConv2d-251            [-1, 384, 6, 6]               0
           Conv2d-252            [-1, 448, 6, 6]         573,440
      BatchNorm2d-253            [-1, 448, 6, 6]             896
      BasicConv2d-254            [-1, 448, 6, 6]               0
           Conv2d-255            [-1, 384, 6, 6]       1,548,288
      BatchNorm2d-256            [-1, 384, 6, 6]             768
      BasicConv2d-257            [-1, 384, 6, 6]               0
           Conv2d-258            [-1, 384, 6, 6]         442,368
      BatchNorm2d-259            [-1, 384, 6, 6]             768
      BasicConv2d-260            [-1, 384, 6, 6]               0
           Conv2d-261            [-1, 384, 6, 6]         442,368
      BatchNorm2d-262            [-1, 384, 6, 6]             768
      BasicConv2d-263            [-1, 384, 6, 6]               0
           Conv2d-264            [-1, 192, 6, 6]         245,760
      BatchNorm2d-265            [-1, 192, 6, 6]             384
      BasicConv2d-266            [-1, 192, 6, 6]               0
       InceptionE-267           [-1, 2048, 6, 6]               0
           Conv2d-268            [-1, 320, 6, 6]         655,360
      BatchNorm2d-269            [-1, 320, 6, 6]             640
      BasicConv2d-270            [-1, 320, 6, 6]               0
           Conv2d-271            [-1, 384, 6, 6]         786,432
      BatchNorm2d-272            [-1, 384, 6, 6]             768
      BasicConv2d-273            [-1, 384, 6, 6]               0
           Conv2d-274            [-1, 384, 6, 6]         442,368
      BatchNorm2d-275            [-1, 384, 6, 6]             768
      BasicConv2d-276            [-1, 384, 6, 6]               0
           Conv2d-277            [-1, 384, 6, 6]         442,368
      BatchNorm2d-278            [-1, 384, 6, 6]             768
      BasicConv2d-279            [-1, 384, 6, 6]               0
           Conv2d-280            [-1, 448, 6, 6]         917,504
      BatchNorm2d-281            [-1, 448, 6, 6]             896
      BasicConv2d-282            [-1, 448, 6, 6]               0
           Conv2d-283            [-1, 384, 6, 6]       1,548,288
      BatchNorm2d-284            [-1, 384, 6, 6]             768
      BasicConv2d-285            [-1, 384, 6, 6]               0
           Conv2d-286            [-1, 384, 6, 6]         442,368
      BatchNorm2d-287            [-1, 384, 6, 6]             768
      BasicConv2d-288            [-1, 384, 6, 6]               0
           Conv2d-289            [-1, 384, 6, 6]         442,368
      BatchNorm2d-290            [-1, 384, 6, 6]             768
      BasicConv2d-291            [-1, 384, 6, 6]               0
           Conv2d-292            [-1, 192, 6, 6]         393,216
      BatchNorm2d-293            [-1, 192, 6, 6]             384
      BasicConv2d-294            [-1, 192, 6, 6]               0
       InceptionE-295           [-1, 2048, 6, 6]               0
AdaptiveAvgPool2d-296           [-1, 2048, 1, 1]               0
          Dropout-297           [-1, 2048, 1, 1]               0
          Flatten-298                [-1, 2048]                0
           Linear-299                 [-1, 256]         524,544
             ReLU-300                 [-1, 256]               0
      BatchNorm1d-301                 [-1, 256]             512
           Linear-302                 [-1, 512]         131,584
             ReLU-303                 [-1, 512]               0
           Linear-304                   [-1, 3]           1,539
================================================================
```

```
Total params: 22,444,035
Trainable params: 22,444,035
Non-trainable params: 0
----------------------------------------------------------------
Input size (MB): 1.00
Forward/backward pass size (MB): 158.24
Params size (MB): 85.62
Estimated Total Size (MB): 244.86
----------------------------------------------------------------
Loaded checkpoint from 32-th epoch with loss   0.2470.
Epoch 0
Terminated
```

Додаток В

Архітектура нейронної мережі MobileNetV2.

```
----------------------------------------------------------------
        Layer (type)               Output Shape         Param #
================================================================
            Conv2d-1         [-1, 32, 128, 128]           1,152
       BatchNorm2d-2         [-1, 32, 128, 128]              64
             ReLU6-3         [-1, 32, 128, 128]               0
            Conv2d-4         [-1, 32, 128, 128]             288
       BatchNorm2d-5         [-1, 32, 128, 128]              64
             ReLU6-6         [-1, 32, 128, 128]               0
            Conv2d-7         [-1, 16, 128, 128]             512
       BatchNorm2d-8         [-1, 16, 128, 128]              32
  InvertedResidual-9         [-1, 16, 128, 128]               0
           Conv2d-10         [-1, 96, 128, 128]           1,536
      BatchNorm2d-11         [-1, 96, 128, 128]             192
             ReLU6-12         [-1, 96, 128, 128]               0
           Conv2d-13           [-1, 96, 64, 64]             864
      BatchNorm2d-14           [-1, 96, 64, 64]             192
             ReLU6-15           [-1, 96, 64, 64]               0
           Conv2d-16           [-1, 24, 64, 64]           2,304
      BatchNorm2d-17           [-1, 24, 64, 64]              48
 InvertedResidual-18           [-1, 24, 64, 64]               0
           Conv2d-19          [-1, 144, 64, 64]           3,456
      BatchNorm2d-20          [-1, 144, 64, 64]             288
             ReLU6-21          [-1, 144, 64, 64]          ,    0
           Conv2d-22          [-1, 144, 64, 64]           1,296
      BatchNorm2d-23          [-1, 144, 64, 64]             288
             ReLU6-24          [-1, 144, 64, 64]               0
           Conv2d-25           [-1, 24, 64, 64]           3,456
      BatchNorm2d-26           [-1, 24, 64, 64]              48
 InvertedResidual-27           [-1, 24, 64, 64]               0
           Conv2d-28          [-1, 144, 64, 64]           3,456
      BatchNorm2d-29          [-1, 144, 64, 64]             288
             ReLU6-30          [-1, 144, 64, 64]               0
           Conv2d-31          [-1, 144, 32, 32]           1,296
      BatchNorm2d-32          [-1, 144, 32, 32]             288
             ReLU6-33          [-1, 144, 32, 32]               0
           Conv2d-34           [-1, 32, 32, 32]           4,608
      BatchNorm2d-35           [-1, 32, 32, 32]              64
 InvertedResidual-36           [-1, 32, 32, 32]               0
           Conv2d-37          [-1, 192, 32, 32]           6,144
      BatchNorm2d-38          [-1, 192, 32, 32]             384
             ReLU6-39          [-1, 192, 32, 32]               0
           Conv2d-40          [-1, 192, 32, 32]           1,728
      BatchNorm2d-41          [-1, 192, 32, 32]             384
             ReLU6-42          [-1, 192, 32, 32]               0
           Conv2d-43           [-1, 32, 32, 32]           6,144
      BatchNorm2d-44           [-1, 32, 32, 32]              64
 InvertedResidual-45           [-1, 32, 32, 32]               0
           Conv2d-46          [-1, 192, 32, 32]           6,144
      BatchNorm2d-47          [-1, 192, 32, 32]             384
             ReLU6-48          [-1, 192, 32, 32]               0
           Conv2d-49          [-1, 192, 32, 32]           1,728
      BatchNorm2d-50          [-1, 192, 32, 32]             384
             ReLU6-51          [-1, 192, 32, 32]               0
           Conv2d-52           [-1, 32, 32, 32]           6,144
      BatchNorm2d-53           [-1, 32, 32, 32]              64
 InvertedResidual-54           [-1, 32, 32, 32]               0
           Conv2d-55          [-1, 192, 32, 32]           6,144
      BatchNorm2d-56          [-1, 192, 32, 32]             384
             ReLU6-57          [-1, 192, 32, 32]               0
           Conv2d-58          [-1, 192, 16, 16]           1,728
      BatchNorm2d-59          [-1, 192, 16, 16]             384
             ReLU6-60          [-1, 192, 16, 16]               0
           Conv2d-61          [-1, 64, 16, 16]           12,288
      BatchNorm2d-62          [-1, 64, 16, 16]              128
 InvertedResidual-63          [-1, 64, 16, 16]               0
           Conv2d-64          [-1, 384, 16, 16]          24,576
      BatchNorm2d-65          [-1, 384, 16, 16]             768
             ReLU6-66          [-1, 384, 16, 16]               0
           Conv2d-67          [-1, 384, 16, 16]           3,456
      BatchNorm2d-68          [-1, 384, 16, 16]             768
             ReLU6-69          [-1, 384, 16, 16]               0
           Conv2d-70          [-1, 64, 16, 16]           24,576
```

```
      BatchNorm2d-71         [-1, 64, 16, 16]             128
 InvertedResidual-72         [-1, 64, 16, 16]               0
           Conv2d-73         [-1, 384, 16, 16]          24,576
      BatchNorm2d-74         [-1, 384, 16, 16]             768
             ReLU6-75         [-1, 384, 16, 16]               0
           Conv2d-76         [-1, 384, 16, 16]           3,456
      BatchNorm2d-77         [-1, 384, 16, 16]             768
             ReLU6-78         [-1, 384, 16, 16]               0
           Conv2d-79         [-1, 64, 16, 16]           24,576
      BatchNorm2d-80         [-1, 64, 16, 16]              128
 InvertedResidual-81         [-1, 64, 16, 16]               0
           Conv2d-82         [-1, 384, 16, 16]          24,576
      BatchNorm2d-83         [-1, 384, 16, 16]             768
             ReLU6-84         [-1, 384, 16, 16]               0
           Conv2d-85         [-1, 384, 16, 16]           3,456
      BatchNorm2d-86         [-1, 384, 16, 16]             768
             ReLU6-87         [-1, 384, 16, 16]               0
           Conv2d-88         [-1, 64, 16, 16]           24,576
      BatchNorm2d-89         [-1, 64, 16, 16]              128
 InvertedResidual-90         [-1, 64, 16, 16]               0
           Conv2d-91         [-1, 384, 16, 16]          24,576
      BatchNorm2d-92         [-1, 384, 16, 16]             768
             ReLU6-93         [-1, 384, 16, 16]               0
           Conv2d-94         [-1, 384, 16, 16]           3,456
      BatchNorm2d-95         [-1, 384, 16, 16]             768
             ReLU6-96         [-1, 384, 16, 16]               0
           Conv2d-97         [-1, 96, 16, 16]           36,864
      BatchNorm2d-98         [-1, 96, 16, 16]              192
 InvertedResidual-99         [-1, 96, 16, 16]               0
          Conv2d-100         [-1, 576, 16, 16]          55,296
     BatchNorm2d-101         [-1, 576, 16, 16]           1,152
            ReLU6-102         [-1, 576, 16, 16]               0
          Conv2d-103         [-1, 576, 16, 16]           5,184
     BatchNorm2d-104         [-1, 576, 16, 16]           1,152
            ReLU6-105         [-1, 576, 16, 16]               0
          Conv2d-106         [-1, 96, 16, 16]           55,296
     BatchNorm2d-107         [-1, 96, 16, 16]              192
InvertedResidual-108         [-1, 96, 16, 16]               0
          Conv2d-109         [-1, 576, 16, 16]          55,296
     BatchNorm2d-110         [-1, 576, 16, 16]           1,152
            ReLU6-111         [-1, 576, 16, 16]               0
          Conv2d-112         [-1, 576, 16, 16]           5,184
     BatchNorm2d-113         [-1, 576, 16, 16]           1,152
            ReLU6-114         [-1, 576, 16, 16]               0
          Conv2d-115         [-1, 96, 16, 16]           55,296
     BatchNorm2d-116         [-1, 96, 16, 16]              192
InvertedResidual-117         [-1, 96, 16, 16]               0
          Conv2d-118         [-1, 576, 16, 16]          55,296
     BatchNorm2d-119         [-1, 576, 16, 16]           1,152
            ReLU6-120         [-1, 576, 16, 16]               0
          Conv2d-121         [-1, 576, 8, 8]             5,184
     BatchNorm2d-122         [-1, 576, 8, 8]             1,152
            ReLU6-123         [-1, 576, 8, 8]                 0
          Conv2d-124         [-1, 160, 8, 8]            92,160
     BatchNorm2d-125         [-1, 160, 8, 8]               320
InvertedResidual-126         [-1, 160, 8, 8]                 0
          Conv2d-127         [-1, 960, 8, 8]           153,600
     BatchNorm2d-128         [-1, 960, 8, 8]             1,920
            ReLU6-129         [-1, 960, 8, 8]                 0
          Conv2d-130         [-1, 960, 8, 8]             8,640
     BatchNorm2d-131         [-1, 960, 8, 8]             1,920
            ReLU6-132         [-1, 960, 8, 8]                 0
          Conv2d-133         [-1, 160, 8, 8]           153,600
     BatchNorm2d-134         [-1, 160, 8, 8]               320
InvertedResidual-135         [-1, 160, 8, 8]                 0
          Conv2d-136         [-1, 960, 8, 8]           153,600
     BatchNorm2d-137         [-1, 960, 8, 8]             1,920
            ReLU6-138         [-1, 960, 8, 8]                 0
          Conv2d-139         [-1, 960, 8, 8]             8,640
     BatchNorm2d-140         [-1, 960, 8, 8]             1,920
            ReLU6-141         [-1, 960, 8, 8]                 0
          Conv2d-142         [-1, 160, 8, 8]           153,600
     BatchNorm2d-143         [-1, 160, 8, 8]               320
InvertedResidual-144         [-1, 160, 8, 8]                 0
          Conv2d-145         [-1, 960, 8, 8]           153,600
     BatchNorm2d-146         [-1, 960, 8, 8]             1,920
            ReLU6-147         [-1, 960, 8, 8]                 0
          Conv2d-148         [-1, 960, 8, 8]             8,640
```

```
      BatchNorm2d-149         [-1, 960, 8, 8]             1,920
            ReLU6-150         [-1, 960, 8, 8]                 0
          Conv2d-151          [-1, 320, 8, 8]           307,200
      BatchNorm2d-152          [-1, 320, 8, 8]               640
InvertedResidual-153          [-1, 320, 8, 8]                 0
          Conv2d-154          [-1, 1280, 8, 8]          409,600
      BatchNorm2d-155          [-1, 1280, 8, 8]            2,560
            ReLU6-156          [-1, 1280, 8, 8]                0
      BatchNorm2d-157          [-1, 1280, 8, 8]            2,560
          Conv2d-158          [-1, 640, 6, 6]         7,373,440
            ReLU-159          [-1, 640, 6, 6]                 0
       MaxPool2d-160          [-1, 640, 3, 3]                 0
         Flatten-161          [-1, 5760]                      0
          Linear-162          [-1, 256]               1,474,816
            ReLU-163          [-1, 256]                       0
      BatchNorm1d-164          [-1, 256]                     512
          Linear-165          [-1, 512]                 131,584
            ReLU-166          [-1, 512]                       0
          Linear-167          [-1, 3]                     1,539
================================================================
Total params: 11,208,611
Trainable params: 11,208,611
Non-trainable params: 0
----------------------------------------------------------------
Input size (MB): 1.00
Forward/backward pass size (MB): 200.72
Params size (MB): 42.76
Estimated Total Size (MB): 244.48
----------------------------------------------------------------
```

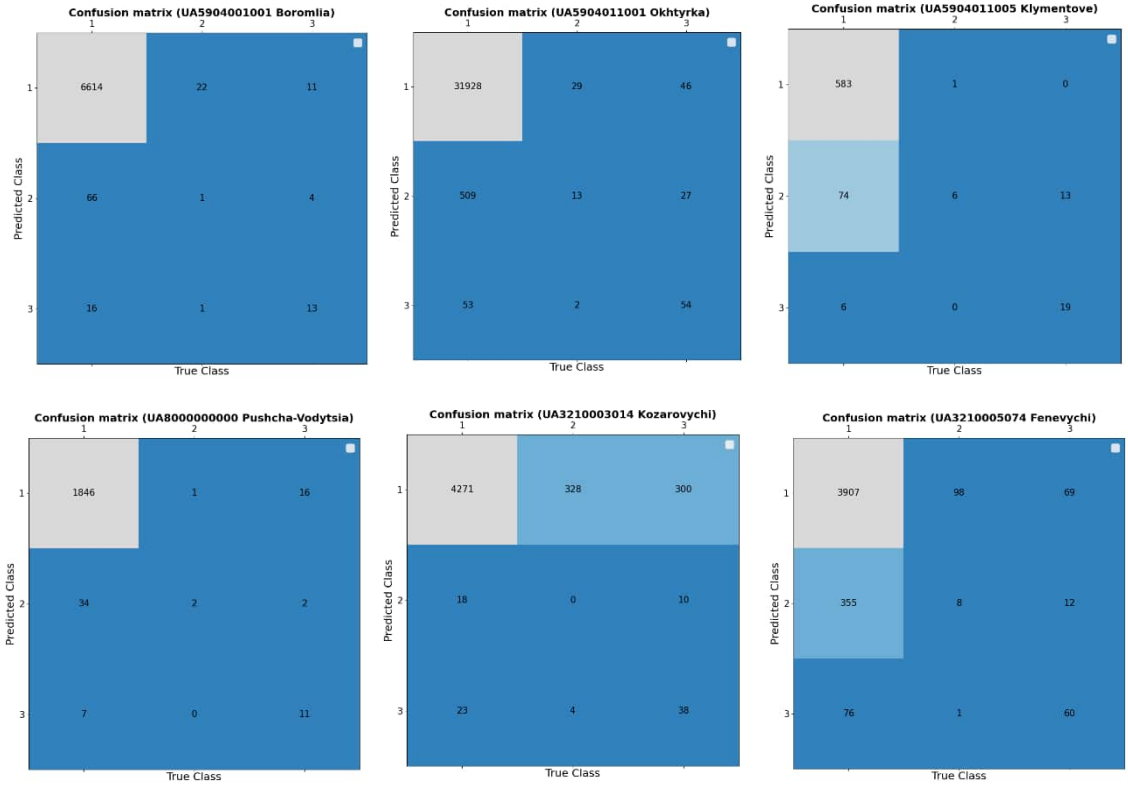
Додаток Г

Розподіл рівнів руйнування тестового набору даних

Населений пункт	**Р.З. 1**	**Р.З. 2**	**Р.З. 3**	**Р.З. 4**	**Р.З. 5**
Lukianivka	6676	265	9	65	94
Zalissia	881	50	58	54	85
Potashnia	294	13	10	7	106
Kozarovychi	4899	26	2	13	52
Fenevychi	4074	314	61	47	90
Boromlia	6659	48	23	20	10
Okhtyrka	32003	421	128	48	61
Klymentove	584	42	51	10	15
Fasivochka	773	27	24	7	24
Klavdiievo-Tarasove	6601	95	7	6	14
Lypivka	1678	124	38	33	1
Pushcha-Vodytsia	1863	18	20	6	12
Severynivka	1795	20	6	8	10

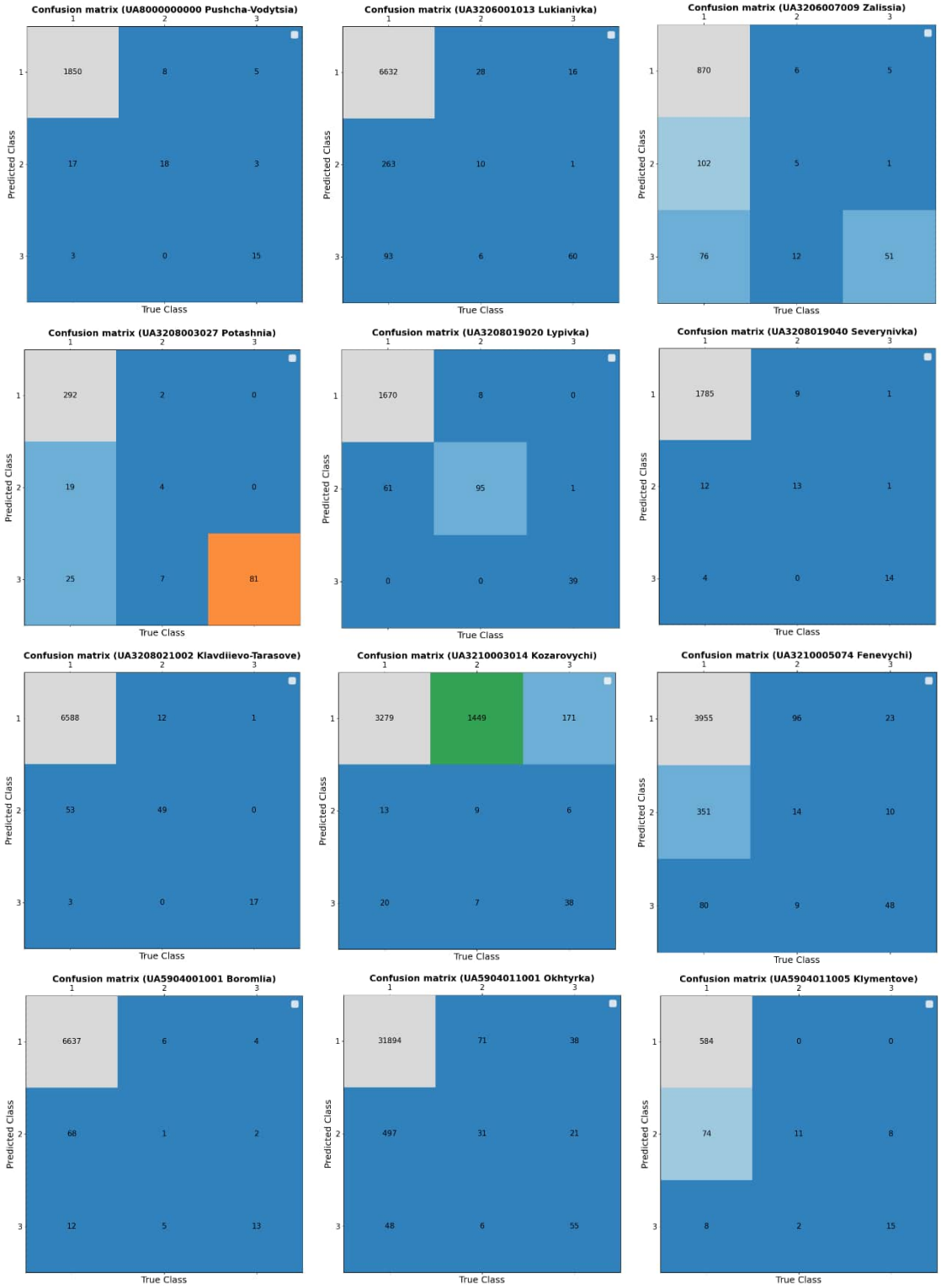
Додаток Д

Матриці невідповідностей для мережі DamageClassifier.



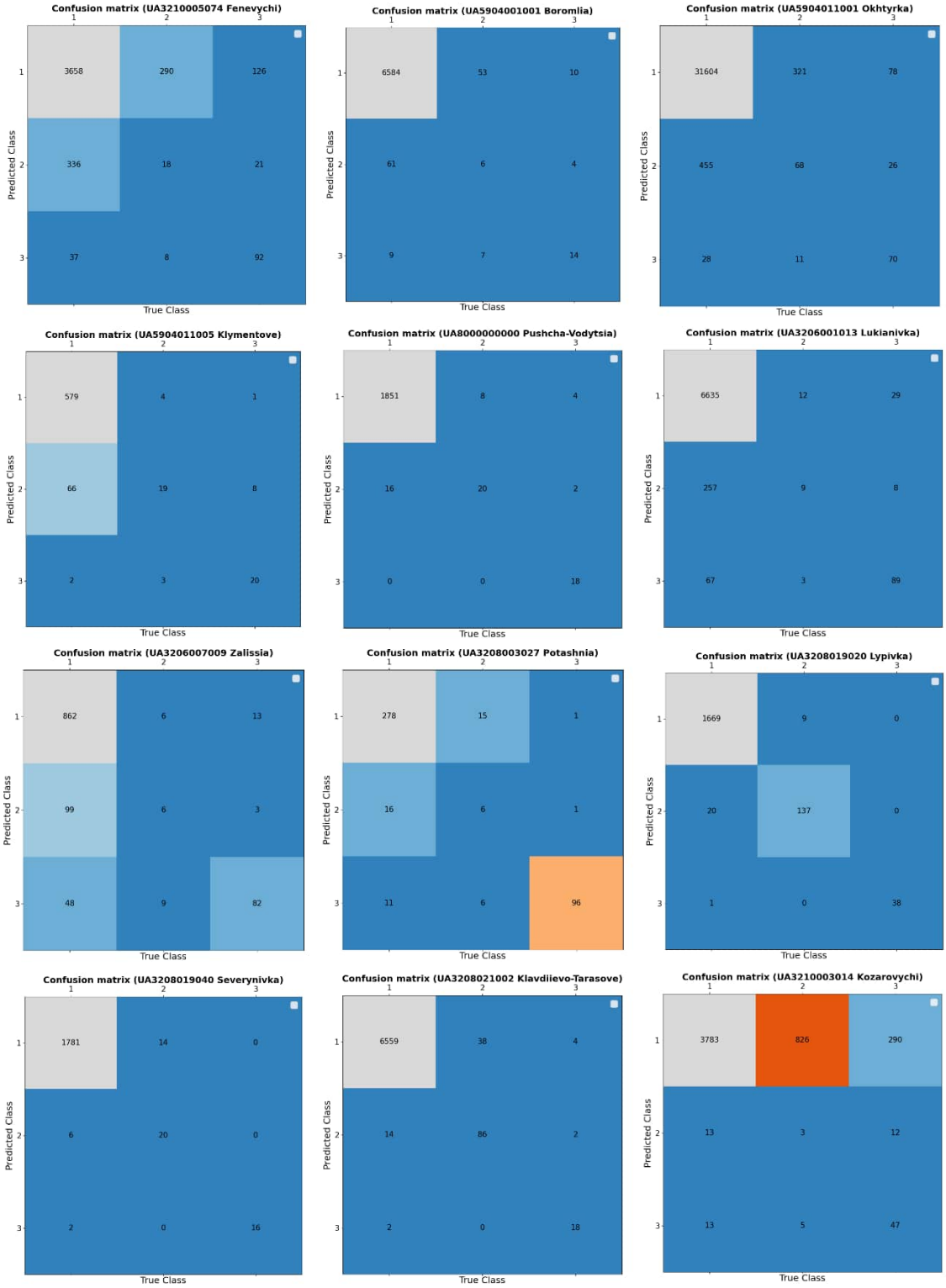
Додаток Е

Матриці невідповідностей для мережі ResNet.



Додаток Є

Матриці невідповідностей для мережі InceptionV3.



Додаток Ж

Матриці невідповідностей для мережі MobileNetV2