МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Чорноморський національний університет
імені Петра Могили
Факультет комп'ютерних наук
Кафедра інтелектуальних інформаційних систем

ДОПУЩЕНО ДО ЗАХИСТУ
Завідувач кафедри інтелектуальних
інформаційних систем, д-р техн. наук, проф.
_____Ю. П. Кондратенко
«____»_____2022 р.

БАКАЛАВРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

ЗАСТОСУВАННЯ ЗГОРТКОВИХ НЕЙРОННИХ МЕРЕЖ ДЛЯ ФОРМУВАННЯ КОНТУРУ БУДІВЛІ

Спеціальність 122 «Комп'ютерні науки»

122 – БКР – 401.21810102

Виконав студент 4-го курсу, групи 401
_____ В. О. Балутін
«20» червня 2022 р.

Керівник: канд. тех. наук, доц.
_____Є. В. Сіденко
«20» червня 2022 р.

Миколаїв – 2022

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Чорноморський національний університет ім. Петра Могили
Факультет комп'ютерних наук
Кафедра інтелектуальних інформаційних систем

Рівень вищої освіти **бакалавр**
Спеціальність **122 «Комп'ютерні науки»**
(шифр і назва)
Галузь знань **12 «Інформаційні технології»**
(шифр і назва)

ЗАТВЕРДЖУЮ

Завідувач кафедри інтелектуальних інформаційних систем, д-р техн. наук, проф.
_____ Ю. П. Кондратенко
«____» _____2021 р.

З А В Д А Н Н Я
на виконання кваліфікаційної роботи

Видано студенту групи 401 факультету комп'ютерних наук Володимиру Олександровичу Балутіну.

1. Тема кваліфікаційної роботи «Застосування згорткових нейронних мереж для формування контуру будівлі».

Керівник роботи Сіденко Є. В., канд. тех. наук, доц.

Затв. наказом Ректора ЧНУ ім. Петра Могили від «___» _____ 20___ р. № ____

2. Строк представлення кваліфікаційної роботи студентом «___» _____ 20__ р.

3. Вхідні (початкові) дані до роботи: згорткові нейронні мережі для формування контуру будівлі; бібліотеки та технології для розробки згорткових нейронних мереж.

Очікуваний результат роботи: застосунок, який за допомогою нейронної мережі формує контур будівлі.

4. Перелік питань, що підлягають розробці (зміст пояснювальної записки):

– огляд на сучасні технології та фреймворки, які використовуються для розробки згорткових нейронних мереж;

– пошук аналогів розроблюваної системи;

– огляд методів і технологій генерації вихідного коду для подальшого його компілювання;

– тестування розроблюваної системи на коректність функціонування та при високому рівні навантаження.

5. Перелік графічного матеріалу: презентація.

6. Завдання до спеціальної частини:

- проаналізувати наявні умови праці в робочому приміщенні;
- визначити достатній рівень показників для робочого приміщення, де проводяться роботи з розробки системи;
- визначити основні правила техніки безпеки при роботі з комп'ютерною технікою.

7. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис
Спеціальна частина з охорони праці	О. В. Макарова, ст. викладач кафедри екології	

Керівник роботи __канд. тех. наук, доц. Сіденко Є. В._____

(наук. ступінь, вчене звання, прізвище та ініціали)

_____

(підпис)

Завдання прийнято до виконання ___Балутін В. О.

(прізвище та ініціали)

_____

(підпис)

Дата видачі завдання «_23_» _____листопада____ 2021 р.

КАЛЕНДАРНИЙ ПЛАН
виконання кваліфікаційної роботи

Тема: Застосування згорткових нейронних мереж для формування контуру будівлі

№	Найменування роботи	Початок	Закінчення	Примітки
1	Подання заяви на затвердження теми та керівників БКР	26.10.2021	27.10.2021	
2	Отримання завдання на виконання БКР	23.11.2021	23.11.2021	
3	Складання календарного плану на період виконання БКР	09.12.2021	10.12.2021	
4	Отримання завдання на переддипломну практику	18.05.2022	18.05.2022	
5	Проходження переддипломної практики, збір та аналіз матеріалів до БКР	23.05.2022	04.06.2022	
6	Оформлення звіту з переддипломної практики	04.06.2022	06.06.2022	
7	Налаштування середовища для розробки застосунку	28.03.2022	29.03.2022	
8	Формування базових функціональних вимог до застосунку	01.04.2022	04.04.2022	
9	Створення згорткової нейронної мережі	22.05.2022	26.05.2022	
10	Тренування розробленої згорткової нейронної мережі	26.05.2022	28.05.2022	
11	Оформлення звіту БКР	10.05.2022	16.06.2022	
12	Попередній захист БКР на засіданні комісії кафедри	30.05.2022	31.05.2022	
13	Консультація з керівником БКР	04.06.2022	10.06.2022	
14	Виправлення помилок в звіті	26.05.2022	01.06.2022	
15	Подання БКР рецензенту	16.06.2022	18.06.2022	
16	Подання БКР, її електронної копії та інших документів (відгуку, рецензії) до захисту	20.06.2022	22.06.2022	
17	Захист БКР перед екзаменаційною комісією (ЕК)	27.06.2022	29.06.2022	

Розробив студент _____Балутін В. О._____ _____
(прізвище та ініціали) (*підпис*)

Керівник роботи __канд. тех. наук, доц. Сіденко Є. В ___ _____
(наук. ступінь, вчене звання, прізвище та ініціали) *(підпис)*

«_11_» _____11_____ 2021 р.

АНОТАЦІЯ

бакалаврської кваліфікаційної роботи студента групи 401 ЧНУ ім. Петра Могили

Балутіна Володимира Олександровича

Тема: «Застосування згорткових нейронних мереж для формування контуру будівлі»

Об'єкт роботи – процеси формування контуру будівлі.

Предмет роботи – згорткові нейронні мережі для формування контуру будівлі.

Метою бакалаврської кваліфікаційної роботи є формування контуру будівлі на основі розроблених нейронних мереж різних програмних бібліотек.

Робота складається з фахового розділу і спеціальної частини з охорони праці. Пояснювальна записка складається зі вступу, трьох розділів, висновків та додатків.

У першому розділі проведено аналіз предметної області, визначено аналоги розроблюваної системи та розглянуто останні публікації.

У другому розділі проаналізовано сучасні технології для розробки згорткових нейронних мереж та визначено технологічний стек, який використано для розробки.

У третьому розділі описано процес проєктування застосунку, який за допомогою згорткової нейронної мережі формує контур будівлі, виділено нюанси розробки, проведено тестування застосунку та описано покроковий алгоритм роботи з розробленим застосунком.

В результаті розроблено застосунок для формування контуру будівель за допомогою згорткової нейронної мережі.

Бакалаврська кваліфікаційна робота містить 92 сторінок, 45 рисунків, 25 використаних джерел та 4 додатків.

Ключові слова: машинне навчання, нейронні мережі, згорткові нейронні мережі, U-Net, Mask R-CNN.

ANNOTATION

bachelor's degree work of a student of group 401 ChNU. Petra Mogili

Balutin Vladimir Alexandrovich

Topic: "The use of convolutional neural networks to form the contour of the building"

Object of work - the processes of forming the contour of the building.

The subject of work - convolutional neural networks for the formation of the contour of the building.

The purpose of the bachelor's thesis is to form the contour of the building on the basis of developed neural networks of various software libraries.

The work consists of a professional section and a special section on labor protection. The explanatory note consists of an introduction, three chapters, conclusions and appendices.

In the first section the analysis of the subject area is carried out, analogues of the developed system are defined and the last publications are considered.

The second section analyzes modern technologies for the development of convolutional neural networks and identifies the technological stack used for development.

The third section describes the process of application design, which with the help of a convolutional neural network forms the contour of the building, highlights the nuances of development, testing the application and describes a step-by-step algorithm for working with the developed application.

As a result, an application was developed to form the contour of buildings using a convolutional neural network.

The bachelor's thesis contains 92 pages, 45 figures, 25 sources used and 4 appendices.

Keywords: machine learning, neural networks, convolutional neural networks, U-Net, Mask R-CNN.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ ... 3
ВСТУП ... 5
1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ НА ПОСТАНОВКА ЗАДАЧІ ... 7
1.1 Аналіз предметної області ... 7
1.2 Останні дослідження та публікації ... 9
1.3 Постановка задачі ... 14
2 ТЕХНОЛОГІЇ, МЕТОДИ, ПІДХОДИ, АЛГОРИТМИ ДЛЯ ВИРІШЕННЯ ЗАДАЧІ ФОРМУВАННЯ КОНТУРУ БУДІВЕЛЬ ... 15
2.1 Комп'ютерні технології в картографії ... 15
2.2 Машине навчання ... 16
2.3 Штучні нейронні мережі ... 19
2.3 Згорткові нейронні мережі ... 26
3 РЕАЛІЗАЦІЯ ЗГОРТКОВОЇ НЕЙРОННОЇ МЕРЕЖІ ДЛЯ ФОРМУВАННЯ КОНТУРУ БУДІВЛІ ... 37
3.1 Опис початкового набору даних ... 37
3.2 Реалізація Mask R-CNN архітектури на Python ... 48
3.3 Реалізація U-Net архітектури на Python ... 56
4 ОХОРОНА ПРАЦІ ... 61
ВИСНОВКИ ... 73
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ ... 74
ДОДАТОК А Файл класу з налаштуванням Mask R-CNN ... 74
ДОДАТОК Б Файл запуску застосунку на архітектурі Mask R-CNN ... 84
ДОДАТОК В Файл класу з налаштуванням U-Net ... 74
ДОДАТОК Г Файл запуску застосунку на архітектурі U-Net ... 74

ПЕРЕЛІК СКОРОЧЕНЬ

ВДТ – Візуальний дисплейний термінал

ПЕОМ – Персональні електронні обчислювальні машини

КПО – Коефіцієнт природної освітленості

ПК – Персональний комп’ютер

ЕОМ – Електронна обчислювальна машина

ССБП – Система стандартів безпеки праці

МБІ – Морфологічний будівельний індекс

CNN – Згоркова нейронна мережа;

FCN – Повна згоркова мережа;

MFCNN – Багатофункціональної згорткової Нейронної Мережі

GAF – Глобальний модуль уваги

Пояснювальна записка

до кваліфікаційної роботи

на тему:

«ЗАСТОСУВАННЯ ЗГОРКОВИХ НЕЙРОННИХ МЕРЕЖ ДЛЯ ФОРМУВАННЯ КОНТУРУ БУДІВЛІ»

Спеціальність 122 «Комп'ютерні науки»

122 – БКР – 401.21810102

Виконав студент 4-го курсу, групи 401

_____*В. О. Балутін*_____

(підпис, ініціали та прізвище)

«___» _____ 2022 р.

Керівник: _____канд. тех. наук, доц._____

(наук. ступінь, вчене звання)

_____ *Є. В. Сіденко* _____

(підпис, ініціали та прізвище)

«___» _____ 2022 р.

Миколаїв – 2022

ВСТУП

Останнім часом, машинне навчання отримало дуже значне поширення в дослідженнях і було включено до різних програм, включаючи аналіз тексту, виявлення спаму, рекомендації відео, класифікацію зображень та пошук мультимедійних концепцій. З швидко зростаючим попитом на машини, що навчаються, для вирішення багатьох складних завдань глибоке навчання за останні кілька років перетворилася на галузь інтересів дослідників. Глибоке навчання є похідним від звичайної нейронної мережі, але значно перевершує своїх попередників. Крім того, глибоке навчання одночасно використовує перетворення та графові технології для створення багаторівневих моделей навчання.

Згорткова нейронна мережа (CNN) - це підхід до глибокого навчання, який широко використовується для вирішення складних завдань. Він долає обмеження традиційних підходів до машинного навчання.

Актуальність теми**.** Інформація про просторовий розподіл та зміни будівель має широкий спектр застосувань у міських дослідженнях, таких як міське планування, управління стихійними лихами, оцінка чисельності населення та оновлення карт. Місцеві бюро містобудування та управління природними ресурсами витрачали велику кількість людських та матеріальних ресурсів для отримання точних растрових (тобто об'єктів карти, що описуються матрицею пікселів, де кожен піксель містить пов'язані особливості, окреслені дискретними вершинами, де кожна вершина визначає координати просторових об'єктів) будівель. Космічні та бортові технології забезпечують численні зображення дистанційного зондування, які стають все більш важливими для отримання інформації про будинки. У ранніх дослідженнях суміш пікселів була одним із важливих факторів, що впливають на виділення будівель, коли були доступні тільки супутникові зображення з високою або низькою роздільною здатністю. В даний час передова технологія дистанційного зондування пропонує зображення дуже високої роздільної здатності з просторовою роздільною здатністю менше метра, що робить їх привабливими для вилучення точних полігонів контуру

будівель. У зображеннях з дуже високою роздільною здатністю вплив змішаних пікселів незначний, а складність сцени стає новою проблемою для вилучення слідів будівлі. В даний час багато урядових відомств та промислових компаній використовують ручні методи для розмежування векторних даних контурів будівель за зображеннями дистанційного зондування з високою роздільною здатністю, щоб отримати векторні карти, що відповідають вимогам точності зйомки та картографування. Оскільки ручне анотування вимагає багато часу та досвіду, необхідно розробити ефективну та надійну схему автоматичного вилучення полігонів контурів будівель із зображень дистанційного зондування.

Об’єктом дослідження є згорткові нейронні мережі.

Предметом дослідження є методи згорткових нейронних мереж для формування контуру будівлі.

Метою бакалаврської кваліфікаційної роботи є формування контуру будівлі на основі розроблених нейронних мереж різних програмних бібліотек;

Для досягнення поставленої мети необхідно вирішити наступні завдання:

- проаналізувати сучасний стан задачі побудови карт у різних сферах (картографії, геоінформатики і комп'ютерних наук);
- окреслити існуючі технології для вирішення поставленої задачі;
- реалізація штучної згорткової нейронної мережі та її налаштування для формування контуру будівлі з використанням популярних фреймворків;
- тренування створеної нейронної мережі, використовуючи власний датасет;
- порівняльний аналіз роботи розроблених нейронних мереж різних програмних бібліотек.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ НА ПОСТАНОВКА ЗАДАЧІ

1.1 Аналіз предметної області

Нейронні мережі стали невід'ємною частиною нашого життя. Вони є частиною майже кожної інтелектуальної системи. Розвиток досліджень у галузі нейронних мереж уможливив їх застосування у складних системах. Поточні напрямки досліджень у галузі нейронних мереж включають еволюційні нейронні мережі, грубі та пікові нейронні мережі та глибокі нейронні мережі. Існує безліч практичних програм з використанням нейронних мереж.

Штучні нейронні мережі, зазвичай звані просто нейронними мережами, є обчислювальні системи, натхненні біологічними нейронними мережами, що становлять мозок тварин.

Автоматизація побудови карт останніми роками вивчалася багатьма дослідниками, головним чином області картографії, геоінформатики і комп'ютерних наук, особливо обчислювальної геометрії [1]. Методи часто ґрунтуються на геометричних міркуваннях; інтеграція операторів спирається на оптимізацію, набори правил або методи на основі агентів.

Підходи переважно застосовуються у векторному просторі, проте існують і методи, що використовують растрові уявлення просторової сцени. Прикладами таких операцій є агрегація, типізація[2], усунення наприклад, а також узагальнення побудови.

Узагальнення будівлі включає різні елементи, такі як вибір (відповідно до розміру, типу або використання), агрегування (для закриття невеликих прогалин) і спрощення контуру будівлі. У ході цього процесу типова форма будівлі має бути збережена або навіть покращена. Це часто включає випрямлення майже прямих кутів і забезпечення паралельних ліній в контурі будівлі, що, наприклад, може бути досягнуто за допомогою набору правил .Чинники, управляючі генералізацією, залежить від масштабу і площі будівлі, і навіть малих конструкцій (довжини фасаду). При переході до ще дрібніших масштабів ці параметри призвели б до

виключення більшої частини будівель, тому застосовується типізація, тобто заміна будівель будівельним шаблоном (переважно квадратним або прямокутним) із збереженням їхнього просторового розташування.

Були ранні спроби використовувати машинне навчання для отримання картографічних правил із заданих прикладів. Однією з цілей було дізнатися відповідні параметри операцій. У своїй роботі автори спостерігали за людиною-картографом, щоби вивчити її дії. Аналогічно Мустьєр [3] прагнув визначити оптимальні послідовності операторів з допомогою машинного навчання. Сестер намагався здобути просторові знання із заданих просторових даних. Хоча ці підходи були дуже цікавими, переважно вони залишалися доказами концепцій.

Глибоке навчання як ефективний знову з'явилося останніми роками завдяки доступній зараз обчислювальній потужності (особливо з використанням графічних процесорів), що дозволяє нам також проектувати дуже глибокі (багаторівневі) та складні мережі, а також велику кількість доступних навчальних даних. На успіх інтерпретації зображень багато в чому вплинула розробка нових схем моделювання, таких як CNN[4]. Вони обмежують кількість з'єднань у мережі з локальними середовищами, таким чином імітуючи зорову систему людини з її полями сприйняття, але також обмежуючи відносини нейронів на основі принципів сусідства. Такі моделі застосовувалися в різних дисциплінах для класифікації зображень, наприклад, при розпізнаванні цифр або символів, пошуку інформації про певні сцени. Крім того, повністю згорткові мережі та їх подальші мережеві архітектури досягли призначення мітки класу кожному пікселю, що називається семантичною сегментацією. Такі методи застосовуються у дуже актуальних завданнях, таких як інтелектуальні системи безпеки, автономне водіння та охорона здоров'я, а також у топографічному картографуванні, наприклад семантична сегментація аерофотознімків вилучення доріг із віддалених сприйняття зображен.

Моделі вивчають не тільки спрощення контурів будівель, а й одночасне об'єднання та вибір будівель в одній моделі. Це перевага порівняно з більшістю існуючих рішень, які часто орієнтовані на автоматизацію однієї операції

(наприклад, спрощення контуру); інші підходи вимагають спеціальних, явних правил взаємодії опису взаємодії між різними операціями (наприклад, підходи з урахуванням агентів). Наші моделі не вивчають вибір операцій, наприклад, виключення, спрощення або агрегації, а генерують результати узагальнення безпосередньо, тому немає необхідності враховувати взаємодію між операторами.

Нещодавно згорткові нейронні мережі (CNN) перевершили традиційні методи у багатьох дистанційних вимірах завдання. CNN можуть безпосередньо вивчати уявлення функцій з необроблених даних; таким чином вони забезпечують наскрізне рішення створення контурів будівель на основі даних дистанційного зондування. Самий досліджень у цій галузі надають ярлик «будівля» або «не будівля» до кожного пікселя зображення, що дає семантичне маски будівель.

1.2 Останні дослідження та публікації

Для методів, заснованих на класифікації, будівельні маски вилучаються класифікаторами з машинним навчанням, які приймають спектральну інформацію та/або просторові характеристики як вхідні дані для прогнозування кожного пікселя. Наприклад, Туркер та Кок-Сан [5] використовують машину опорних векторів (SVM) для визначення областей забудови на основі спектральних смуг та нормалізованого різницевого вегетаційного індексу (NDVI). Мета підходів, що ґрунтуються на індексах, полягає в тому, щоб розробити індекс ознак, який може бути безпосередньо застосований для отримання областей будівель без будь-якої класифікації чи процесу сегментації. Морфологічний будівельний індекс (МБІ) є широко використовуваним, і цей індекс поєднує багатомасштабні та різноспрямовані морфологічні оператори. Однак загальним обмеженням цих ранніх робіт є використання функцій, створених вручну, та розробка складних функцій, що призводить до поганого узагальнення.

Замість евристичного проектування функцій CNN можуть запропонувати найкращі можливості узагальнення. Завдяки нещодавнім досягненням у галузі мереж семантичної сегментації результати генерації контурів будівель були значно

покращені. Ці мережі зазвичай є повністю згорткові мережі (FCN) та архітектури кодер-декодер, такі як U-Net [6], SegNet [7] та FC-DenseNet [8], було показано, що FCN ефективний при обробці великих обсягів даних дистанційного зондування і забезпечує надійні результати сегментації будівель. SegNet використовується для створення першої цільної карти площі будівлі для США. Щоб підвищити точність сегментації великих будівель, запропонована архітектура на основі U-Net, де вихідні зображення та їх аналоги зі зниженою частотою дискретизації беруться як вхідні дані двох гілок, що мають однакові ваги, була запропонована стратегія змагального навчання для побудови вилучення з зображень дистанційного зондування, а FC-DenseNet використовується як базова мережа семантичної сегментації для створення точних слідів будівель. Однак багато експериментів показують, що передбачені семантичні маски будівель із CNN все ще не настільки задовільні, де межі будівель розмиті.

Згорткові нейронні мережі показали відмінні результати в порівнянні з традиційними методами вилучення будівель через їхню здатність витягувати абстрактні функції високого рівня із зображень. Проте, важко повністю використати численні можливості сучасних методів вилучення будівель; отже, внаслідок межі будівлі нерегулярні. Щоб подолати ці обмеження, Якун Сьє та Джун Джо у своїй статті запропонували метод виділення будівель із зображень високої роздільної здатності з використанням багатофункціональної згорткової нейронної мережі (MFCNN) та морфологічної фільтрації. Їх метод складається із двох кроків. По-перше, MFCNN, який складається з залишкового підключеного блоку, блоку розширеного сприйняття та блоку агрегації піраміди, використовується для сегментації будівель на рівні пікселів. По-друге, морфологічна фільтрація використовується для оптимізації меж будівель, покращення регулярності кордонів та отримання уточнених меж будівель. Набори даних Massachusetts[9] та Inria[10] були вибрані для експериментального аналізу. У тих самих експериментальних умовах результати вилучення, отримані з допомогою запропонованої MFCNN, порівнюються з іншими моделей глибокого

навчання, які широко використовувалися останніми роками: FCN-8, SegNet і U-Net. Результати обох наборів даних показали, що запропонована модель кун Сьє та Джун Джо покращує показник F1 на 3,31–5,99 %, підвищує загальну точність (OA) на 1,85–3,07 % та збільшує перетин над об'єднанням (IOU) на 3,47 -8,82%. %. Ці результати показують, що запропонований метод ефективний при витяганні будівель зі складних сцен.

Інший метод для вирішення проблеми розмиття запропонували Джінджу Лі та Лічао Моу. Їх метод полягає у вивчинні репрезентації поля тяжіння для побудови кордонів, яка може забезпечити підвищену репрезентативність. Їх метод складається з двох елементарних модулів: модуля Img2AFM[11] та модуля AFM2Mask[12]. Зокрема, перший спрямований на вивчення уявлення поля тяжіння, зумовленого вхідним зображенням, яке здатне посилювати межі побудови та пригнічувати фон. Останній модуль прогнозує маски сегментації будівель, використовуючи вивчену карту поля тяжіння. Пропонований метод оцінюється на трьох наборах даних з різною просторовою роздільною здатністю: набором даних ISPRS[13], набором даних INRIA та набором даних Planet. З експериментальних результатів дійшли висновку, що запропонований каркас може добре зберігати геометричні форми і чіткі межі будівель, що дає значні переваги проти іншими конкурентами.

Методи глибокого навчання, такі як згорткові нейронні мережі значно покращили продуктивність побудови сегментації зображень дистанційного зондування. Однак зображення для сегментації будівель часто мають форму традиційних ортофотознімків, де зсув рельєфу може призвести до значного зміщення між контуром даху та контуром будівлі; таке зсув створює серйозні проблеми для отримання точних контурів будівель, особливо для висотних будівель. Щоб вирішити цю проблему, Кі Чен та Юаній Жанг запропонували новий робочий процес створення виправлених контурів будинків з урахуванням традиційних ортофотопланів. Спочатку вони використовуємо мітки фасадів, які ефективно та недорого готуються, разом із мітками дахів для навчання мережі

семантичної сегментації. Потім добре навчена мережа, яка використовує як магістраль сучасну версію EfficientNet, витягує сегменти дахів та фасадів будівель із вхідного зображення. Нарешті, після кластеризації класифікованих пікселів в об'єкти будівлі на рівні екземпляра та відстеження контурів даху пропонується функція енергії, щоб максимально поєднати контур даху з контуром будівлі; таким чином можуть бути згенеровані виправлені сліди. Експерименти з аерофотознімками, що покривають житловий район з високою щільністю населення Шанхаї, показують, що запропонований робочий процес може генерувати більш точні контури будівель, ніж базові методи, особливо для висотних будівель.

Автоматичне вилучення будівель має важливі соціально-економічні застосування, такі як оцінка чисельності населення, міське планування, швидке реагування на стихійні лиха, моніторинг незаконного відведення землі та виявлення змін. Традиційні методи не можуть впоратися з проблемами, повністю пов'язаними з вилученням будівель, тобто різною формою, розміром, текстурою будівель, відсутніми та незавершеними будинками через оклюзію та високу внутрішньокласову мінливість. Існуючі підходи на основі CNN не здатні відновити інформацію про межі, особливо коли будівельні конструкції невеликі та складні. Щоб полегшити проблеми, з якими стикаються сучасні методи, Сатіш Чанд та Пратіва Дас запропонували спрощену модель, засновану на механізмі уваги, - удосконалену нейронну мережу перехресної уваги (RCA-Net) для точного отримання ознак побудови від грубої до точної. На відміну від недавніх підходів, заснованих на механізмі уваги, RCA-Net використовує просторову та канальну увагу для захоплення довгострокового багатомасштабного контексту. Потім вони представили ефективний модуль уваги, модуль Global Attention Fuse (GAF), який поєднує локальні та глобальні міжканальні відносини для захоплення основних функцій без збільшення обчислювальної складності. Також представлено функцію втрат, об'єднану втрату, яка поєднує в собі втрату BCE та втрату кубиків, щоб полегшити проблему незбалансованого розподілу класів. Експериментальні

результати показують, що пропонований нами метод перевершує новітній метод DSNet на 2,06% і 1,47% по IoU і на 2,11% і 1,27% за оцінкою F1 в загальнодоступних наборах даних: наборі даних про будівлі Массачусетса і наборі даних Inria Aerial Image Labeling відповідно.

Також, дослідження сфери картографії виявило, що згорткові нейронні мережі жваво використовуються у цій сфері. Картографічне узагальнення – це процес створення уявлень меншого масштабу з великомасштабних просторових даних. Цей процес здійснюють картографи, застосовуючи різні оператори, такі як вибір, спрощення або зсув, які в сумі призводять до більш простого та чіткого представлення просторової сцени в меншому масштабі. Існує безліч потужних автоматичних рішень окремих операторів. Основна проблема сьогодні полягає в освоєнні взаємодії між різними операторами, для чого використовуються, наприклад, підходи до оптимізації, підходи, що ґрунтуються на правилах, або підходи, що ґрунтуються на агентах. Однак тут еталоном, як і раніше, залишається людина-оператор, здатний спроектувати естетичне та правильне уявлення фізичної реальності шляхом ретельної координації кількох операторів. Методи глибокого навчання продемонстрували вражаючий успіх у завданнях інтерпретації, для яких складні алгоритмічні методи. Яскравим прикладом є класифікація та інтерпретація зображень, де підходи глибокого навчання перевершують традиційні методи комп'ютерного зору. В обох областях — комп'ютерний зір та картографія — люди дуже добре здатні створювати хороші рішення.

Дуже цікавий метод представили Вуфан Жао та Альфред Стеін. Цей метод може прогнозувати регуляризовані контури будівель у векторному форматі в рамках наскрізної структури глибокого навчання. Основна ідея їхнього фреймворку – навчитися передбачати розташування ключових вершин будівель та послідовно з'єднувати їх. Пропонований метод заснований на PolyMapper. Вони модернізуємо витяг ознак, ввівши блоки уточнення глобального контексту та кордонів, а також додали модулі уваги до каналів та простору, щоб підвищити ефективність модуля виявлення. Крім того, ми вводимо складовий шар для

подальшого збереження геометричних відносин між вершинами та прискорення логічного виведення. Вони протестували метод на двох великомасштабних наборах даних вилучення будівель VHR-RS. Також якісне порівняння показує, що метод значно покращує екземплярну сегментацію будівель різної форми.

1.3 Постановка задачі

Тому згідно з актуальністю теми та штучного інтелекту об'єктом дослідження обрано процеси формування контуру будівлі. Згідно з цим предметом дослідження є згорткові нейронні мережі для формування контуру будівлі.

Для досягнення поставленої мети необхідно вирішити наступні завдання:

– проаналізувати сучасний стан задачі побудови карт у різних сферах(картографії, геоінформатики і комп'ютерних наук);

– окреслити існуючі технології для вирішення поставленої задачі;

– реалізація штучної згорткової нейронної мережі та її налаштування для формування контуру будівлі з використанням популярних фреймворків;

– тренування створеної нейронної мережі, використовуючи власний датасет;

– порівняльний аналіз роботи розроблених нейронних мереж різних програмних бібліотек.

2 ТЕХНОЛОГІЇ, МЕТОДИ, ПІДХОДИ, АЛГОРИТМИ ДЛЯ ВИРІШЕННЯ ЗАДАЧІ ФОРМУВАННЯ КОНТУРУ БУДІВЕЛЬ

2.1 Комп'ютерні технології в картографії

Будівлі, як важливий компонент базової міської географічної інформації, відіграють важливу роль в оцінці чисельності населення, моніторингу змін, міському плануванні та будівництві міст. Отже, автоматичне вилучення контурів будівель із зображень високої роздільної здатності завжди було фундаментальним завданням у галузі досліджень дистанційного зондування . В останні роки, коли просторове дозвіл зображень дистанційного зондування значно збільшилося, додатки на основі дистанційного зондування, що включають методи вилучення будівель, набули значного розвитку. Багато алгоритмів вилучення будинків було запропоновано вченими.

Залежно від джерел даних, що використовуються, існуючі методи можна розділити на наступні три категорії.

1) методи оптичного зображення;

2) світлове виявлення та методи хмари точок дальності;

3) комбінації методів оптичного зображення та хмари точок.

Методи оптичного зображення засновані на просторових та спектральних характеристиках оптичних зображень, таких як текстурні особливості , геометричні особливості , крайові особливості , мультиспектральні особливості та тіньові особливості. Методи хмари точок LiDAR в основному ґрунтуються на інформації, витягнутій з даних хмари точок, такої як інформація про висоту, яка може бути чітко зафіксована в даних хмари точок та використана для ідентифікації будівель. Методи третього типу використовують мультисенсорні дані для вилучення будівель. Такі методи можуть досягати кращих результатів за рахунок комбінованого обліку додаткових характеристик, таких як спектральна інформація, просторові характеристики та висота над рівнем моря. Ці комбіновані підходи досягли певного успіху у витягуванні контурів будівель; однак зображення

дистанційного зондування зазвичай демонструють неоднорідні області, великі внутрішньокласові варіації та низькі міжкласові варіації, що унеможливлює створення відповідної зумовленої моделі для вилучення об'єктів. Крім того, типи об'єктів, захоплених на таких зображеннях, різноманітні та складні, і, отже, традиційні алгоритми сегментації, класифікації та виділення кордонів не можуть виконувати глибоке вилучення семантичних ознак. Проте, можна автоматично витягувати деталі глибокого зображення за допомогою методів машинного навчання шляхом створення глибоких нейронних мереж. Теорія глибокого навчання була спочатку запропонована в 2006 Хінтоном. Глибоке навчання є процес отримання абстрактних ознак високого рівня з даних шляхом побудови математичних моделей для підвищення точності класифікації та точності виявлення. За останні роки було запропоновано безліч моделей нейронних мереж, включаючи згорткові нейронні мережі (CNN) , рекурентні нейронні мережі та мережі глибокої довіри. Ці мережі використовувалися для різних високопродуктивних завдань комп'ютерного зору, таких як класифікація зображень , обробка природної мови , розпізнавання мови , обробка зображень дистанційного зондування . Серед цих мереж CNN досягли чудових результатів у завданнях класифікації зображень. Отже, багато вчених розробили безліч покращених алгоритмів на основі CNN.

2.2 Машине навчання

Машинне навчання - це область штучного інтелекту, яка у широкому сенсі визначається як здатність машини імітувати розумну поведінку людини. Системи штучного інтелекту використовуються для виконання складних завдань так само, як люди вирішують проблеми.

За словами Бориса Каца, головного наукового співробітника та голови групи InfoLab у CSAIL, метою II є створення комп'ютерних моделей, що демонструють «розумну поведінку», як у людей. Це означає, що машини можуть розпізнавати

візуальну сцену, розуміти текст, написаний природною мовою, або виконувати дії у фізичному світі.

Машинне навчання - один із способів використання ІІ. У 1950-х роках піонер ІІ Артур Семюел визначив його як область дослідження, яка дає комп'ютерам можливість вчитися без явного програмування.

За словами Майки Шульмана, викладача MIT Sloan та керівника відділу машинного навчання в Kensho, яка спеціалізується на штучному інтелекті для фінансової та розвідувальної спільнот США, це визначення вірне. Він порівняв традиційний спосіб програмування комп'ютерів, або програмне забезпечення 1.0, з випічкою, де рецепт вимагає точної кількості інгредієнтів і каже пекарю змішувати протягом точного часу. Традиційне програмування також потребує створення докладних інструкцій для комп'ютера.

Але в деяких випадках написання програми для машини вимагає багато часу або неможливо, наприклад, навчання комп'ютера розпізнавання зображень різних людей. У той час як люди можуть легко виконати це завдання, важко сказати комп'ютера, як це зробити. Машинне навчання використовує підхід, що дозволяє комп'ютерам навчатися програмувати себе на основі досвіду.

Машинне навчання починається з даних - чисел, фотографій або тексту, наприклад банківських транзакцій, зображень людей або навіть хлібобулочних виробів, записів про ремонт, даних тимчасових рядів із датчиків або звітів про продаж. Дані збираються та готуються для використання як навчальних даних або інформації, на якій навчатиметься модель машинного навчання. Чим більше даних, тим краще програма.

Звідти програмісти вибирають модель машинного навчання для використання, надають дані та дозволяють комп'ютерній моделі навчатися, щоб знаходити закономірності або робити прогнози. Згодом програміст-людина також може налаштувати модель, у тому числі змінити її параметри, щоб досягти більш точних результатів. (Веб-сайт AI Weirdness вченого-дослідника Джанелл Шейн є цікавим поглядом на те, як алгоритми машинного навчання вчаться і як вони

можуть помилятися - як це сталося, коли алгоритм спробував згенерувати рецепти і створив шоколадно-курячий пиріг з куркою.)

Деякі дані зберігаються з навчальних даних для використання як оціночні дані, які перевіряють, наскільки точна модель машинного навчання, коли їй показуються нові дані. В результаті виходить модель, яку можна використовувати у майбутньому з різними наборами даних.

Успішні алгоритми машинного навчання можуть робити різні речі, написав Мелоун у нещодавньому дослідницькому огляді про штучний інтелект і майбутнє роботи, який був написаний у співавторстві з професором Массачусетського технологічного інституту та директором CSAIL Даніелою Рус та Робертом Лаубахером, заступником директора Центру. .

«Функція системи машинного навчання може бути описовою, тобто система використовує дані для пояснення того, що сталося; прогностичний, тобто система використовує дані, щоб передбачити, що станеться; або приписувач, що означає, що система використовуватиме дані, щоб робити пропозиції про те, які дії слід вжити», - пишуть дослідники.

З зростаючим поширенням машинного навчання кожен у бізнесі, ймовірно, зіткнеться з ним, і йому знадобляться деякі практичні знання у цій галузі. Опитування Deloitte 2020 показало, що 67% компаній використовують машинне навчання, а 97% використовують або планують використовувати його в наступному році.

Від виробництва до роздрібної торгівлі, банківської справи та пекарень навіть застарілі компанії використовують машинне навчання, щоб відкривати нові цінності або підвищувати ефективність. «Машинне навчання змінює або змінить кожну галузь, і лідери повинні розуміти основні принципи, потенціал та обмеження», — сказав професор комп'ютерних наук Массачусетського технологічного інституту Олександр Мадрі, директор Центру машинного навчання Массачусетського технологічного інституту, що розгортається.

Мадрі додав, що хоч не всім потрібно знати технічні деталі, вони повинні розуміти, що робить технологія, що вона може і чого не може. «Я не думаю, що хтось може дозволити собі не знати, що відбувається».

Три підкатегорії машинного навчання:

Моделі **контрольованого** машинного навчання навчаються за допомогою помічених наборів даних, що дозволяє моделям навчатися та ставати більш точними з часом. Наприклад, алгоритм навчатиметься на зображеннях собак та інших об'єктів, помічених людьми, а машина вчитиметься самостійно визначати зображення собак. Машинне навчання з учителем є найпоширенішим типом, що використовується сьогодні.

При **неконтрольованому** машинному навчанні програма шукає закономірності у нерозмічених даних. Неконтрольоване машинне навчання може знаходити закономірності чи тенденції, які явно не шукають. Наприклад, неконтрольована програма машинного навчання може переглядати дані онлайн-продажів та визначати різні типи клієнтів, які здійснюють покупки.

Машинне навчання **з підкріпленням** навчає машини шляхом проб і помилок робити кращі події, створюючи систему винагород. Навчання з підкріпленням може навчати моделі грати в ігри або навчати автономні транспортні засоби водінню, повідомляючи машині, коли вона прийняла правильне рішення, що допомагає їй з часом дізнатися, які дії вона має вжити.

2.3 Штучні нейронні мережі

Що таке штучні нейронні мережі? Нейронні мережі, також відомі як штучні нейронні мережі (ШНМ) або змодельовані нейронні мережі (ЗНМ), є підмножиною машинного навчання і лежать в основі алгоритмів глибокого навчання. Їхня назва та структура надихнуті людським мозком, імітуючи те, як біологічні нейрони подають сигнали один одному.

Штучні нейронні мережі (ШНМ) складаються з шарів вузлів, що містять вхідний шар, один або кілька прихованих шарів та вихідний шар. Кожен вузол або

штучний нейрон з'єднується з іншим і має відповідну вагу та поріг. Якщо вихід будь-якого окремого вузла перевищує вказане граничне значення, цей вузол активується, надсилаючи дані на наступний рівень мережі. В іншому випадку дані не передаються на наступний рівень мережі.

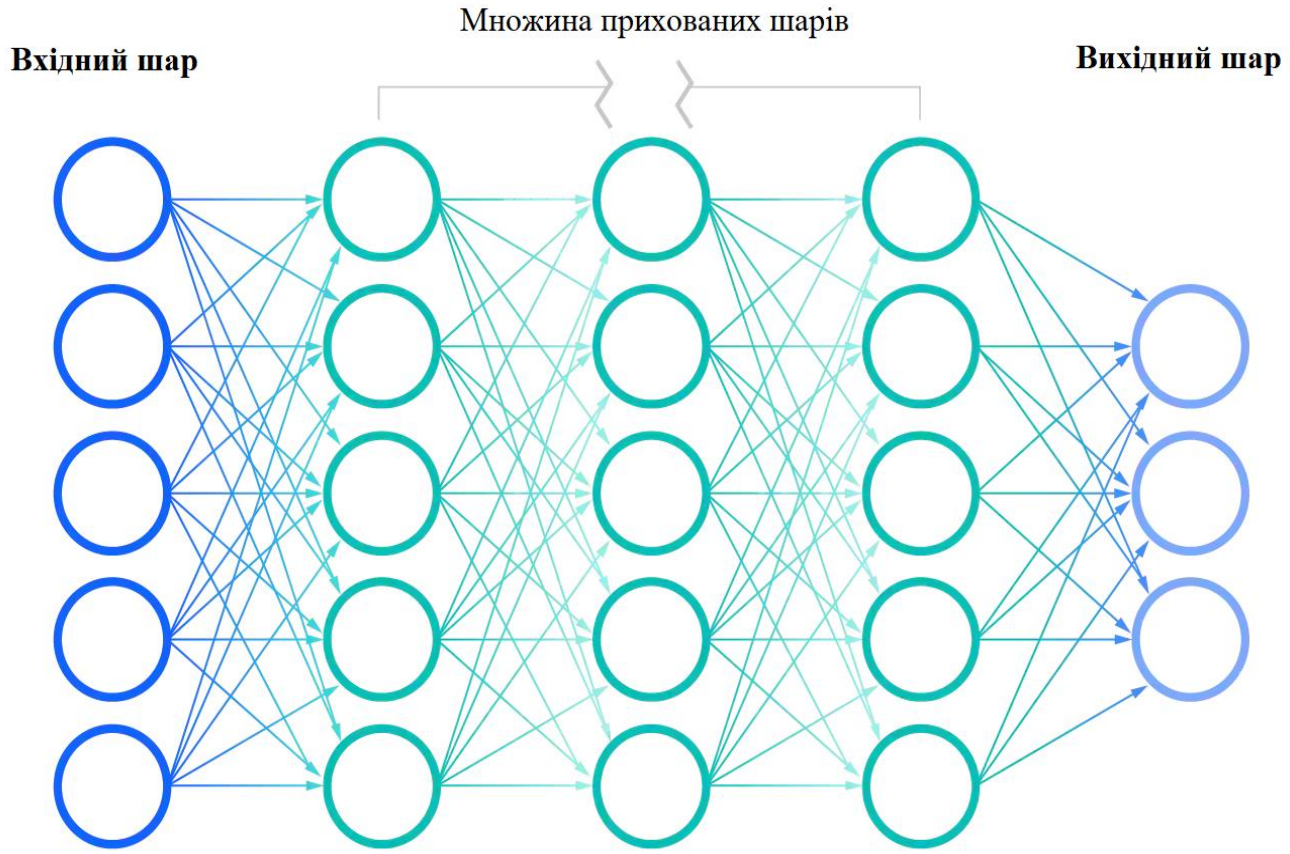


Рисунок 2.1 – Схема штучної нейронної мережі

Нейронні мережі покладаються на навчальні дані, щоб навчатися та покращувати свою точність з часом. Однак, як тільки ці алгоритми навчання будуть налаштовані на точність, вони стануть потужними інструментами в галузі комп'ютерних наук та штучного інтелекту, що дозволяють класифікувати та групувати дані з високою швидкістю. Завдання розпізнавання мови або зображень можуть займати хвилини, а не години, в порівнянні з ідентифікацією вручну

експертами-людьми. Однією з найвідоміших нейронних мереж є пошуковий алгоритм Google.

Як працюють нейронні мережі? Думайте про кожен окремий вузол, як про власну модель лінійної регресії, що складається з вхідних даних, ваг, зміщення (або порога) і вихідних даних. Формула виглядатиме приблизно так:

$$\sum_{i=1}^{m} w_i x_i + bias = w_1 x_1 + w_2 x_2 + w_3 x_3 + bias$$

Рисунок 2.2 – Загальний вигляд формули вузла

$$output = f(x) = \begin{cases} 1 \text{ if } \sum w_1 x_1 + b \geq 0 \\ 0 \text{ if } \sum w_1 x_1 + b < 0 \end{cases}$$

Рисунок 2.3 – Загальний вигляд формули вихідного сигналу

Після визначення вхідного шару йому надаються ваги. Ці ваги допомагають визначити важливість тієї чи іншої змінної, при цьому більші з них роблять більш значний внесок у результат порівняно з іншими вхідними даними. Потім всі вхідні дані множаться на відповідні ваги, а потім підсумовуються. Після цього вихідні дані проходять через активацію, яка визначає вихідні дані. Якщо цей вихід перевищує заданий поріг, він запускає (або активує) вузол, передаючи дані на наступний рівень в мережі. Це призводить до того, що вихід одного вузла стає входом наступного вузла. Цей процес передачі з одного рівня на наступний визначає цю нейронну мережу як мережа з прямим зв'язком.

Давайте розберемо, як може виглядати один вузол, використовуючи двійкові значення. Ми можемо застосувати цю концепцію до більшого прикладу, наприклад, чи варто вам зайнятися серфінгом (Так: 1, Ні: 0). Рішення йти чи не йти – це наш прогнозований результат, чи т-хет. Припустимо, що на ваше рішення впливають три фактори:

1. Хвилі добрі? (Так: 1, Ні: 0);
2. Порожній склад? (Так: 1, Ні: 0);
3. Чи був останнім часом напад акули? (Так: 0, Ні: 1).

Потім припустимо наступне, надавши нам наступні вхідні дані:

- X1 = 1, тому що хвилі накачують;
- X2 = 0, тому що натовпу немає;
- X3 = 1, тому що недавнього нападу акули не було.

Тепер нам потрібно привласнити деякі ваги, щоб визначити важливість. Великі ваги означають, що конкретні змінні мають значення для вирішення чи результату.

- W1 = 5, тому що великі хвилі бувають не часто;
- W2 = 2, тому що ви звикли до натовпу;
- W3 = 4, тому що ви боїтеся акул.

Нарешті, ми також приймемо граничне значення 3, яке буде переведено значення зміщення –3. З усіма різними вхідними даними ми можемо почати підставляти значення формулу, щоб отримати бажаний результат.

$$Y\text{-}= (1*5) + (0*2) + (1*4) – 3 = 6.$$

Якщо ми використовуємо функцію активації з початку цього розділу, ми можемо визначити, що вихід цього вузла дорівнює 1, оскільки 6 більше, ніж 0. У цьому випадку ви відправитеся в серфінг; але якщо ми скоригуємо ваги чи поріг, ми можемо отримати різні результати від моделі. Коли ми спостерігаємо за одним рішенням, як у наведеному вище прикладі, ми бачимо, як нейронна мережа може приймати більш складні рішення в залежності від результатів попередніх рішень або шарів.

У наведеному вище прикладі ми використовували персептрони, щоб проілюструвати деякі математичні операції, але нейронні мережі використовують сигмовидні нейрони, які відрізняються тим, що мають значення від 0 до 1. Оскільки нейронні мережі ведуть себе аналогічно до дерев рішень, каскадування даних з одного вузла до іншого, маючи значення x від 0 до 1, зменшить вплив будь-якої заданої зміни однієї змінної на виведення будь-якого заданого вузла, а потім і виведення нейронної мережі.

Коли ми почнемо думати про більш практичні варіанти використання нейронних мереж, таких як розпізнавання зображень або класифікація, ми будемо використовувати контрольоване навчання або позначені набори даних для навчання алгоритму. Принаймні навчання моделі хочемо оцінити її точність з допомогою функції витрат (чи втрат). Це також називають середньоквадратичною помилкою (MSE). У наведеному нижче рівнянні:

- i – індекс вибірки;
- y з позначкою – це прогнозований результат;
- y – фактичне значення;
- m – кількість зразків.

$$Cost\ Function = MSE = \frac{1}{2m} \sum_{i=1}^{m} (\hat{y} - y)^2$$

Рисунок 2.4 – Загальний вигляд формули MSE

Зрештою, мета полягає в тому, щоб мінімізувати нашу функцію вартості, щоб гарантувати правильність припасування для будь-якого даного спостереження.

Оскільки модель коригує свої ваги та зсуву, вона використовує функцію вартості та навчання з підкріпленням, щоб досягти точки збіжності або локального мінімуму. Процес, у якому алгоритм регулює свої ваги, є градієнтним спуском, що дозволяє моделі визначити напрямок, в якому потрібно зменшити помилки (або мінімізувати функцію вартості). З кожним прикладом, що навчає, параметри моделі коригуються, щоб поступово сходитися до мінімуму.

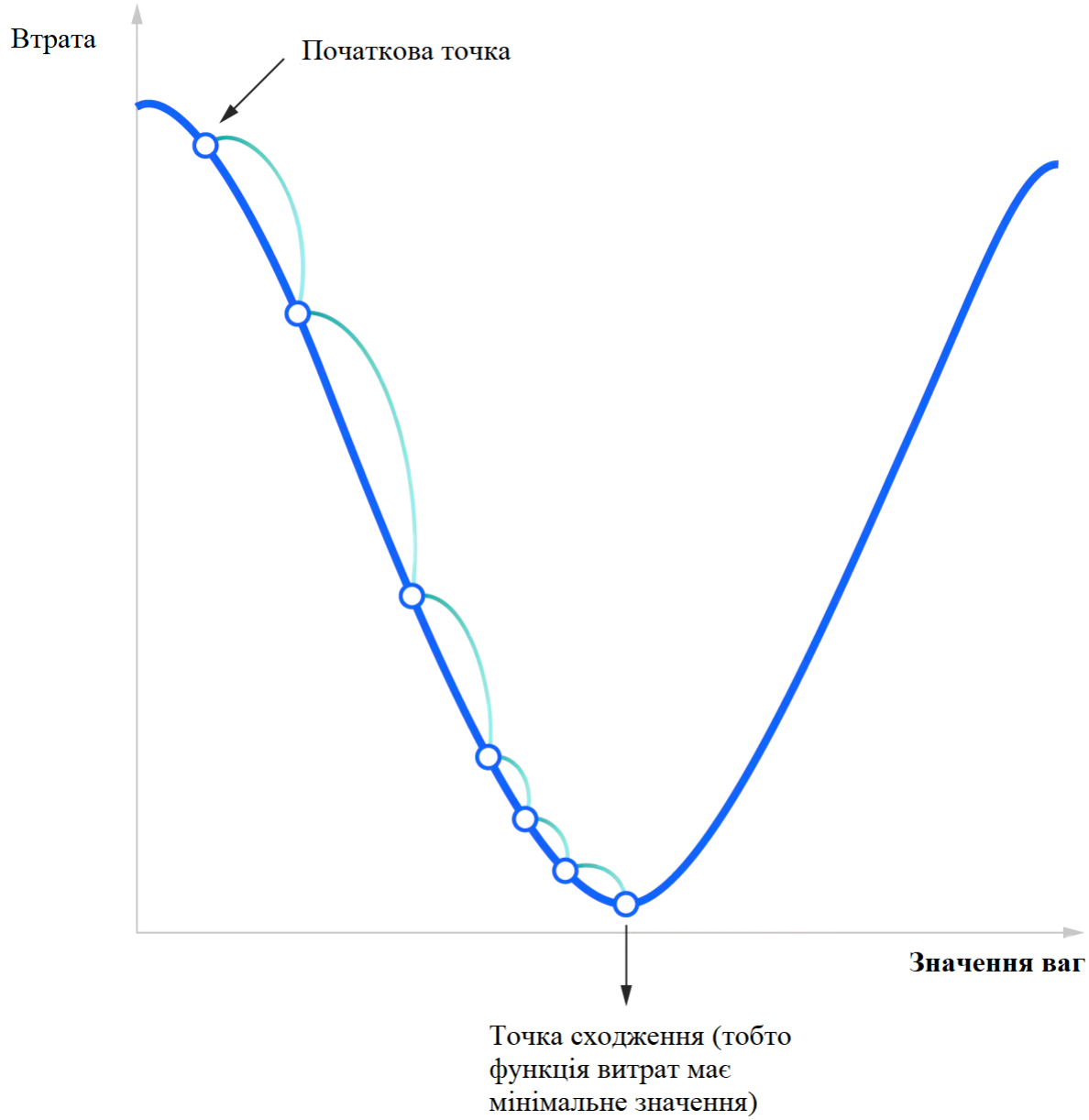


Рисунок 2.5 – Корегування помилки для тренування нейронної мережі

Більшість глибоких нейронних мереж мають прямий зв'язок, тобто працюють тільки в одному напрямку, від входу до виходу. Однак ви можете навчити свою модель за допомогою зворотного поширення; тобто рухатися у протилежному напрямку від виходу до входу. Зворотне поширення дозволяє нам розрахувати та атрибутувати помилку, пов'язану з кожним нейроном, що дозволяє нам відповідним чином налаштувати та підігнати параметри моделі (моделей).

Типи нейронних мереж:

Нейронні мережі можна розділити на різні типи, які використовуються для різних цілей. Хоча це не вичерпний список типів, нижче представлені найпоширеніші типи нейронних мереж, з якими ви зіткнетеся у звичайних випадках їх використання:

Перцептрон - найстаріша нейронна мережа, створена Френком Розенблаттом у 1958 році. Вона складається з одного нейрона і є найпростішою формою нейронної мережі:

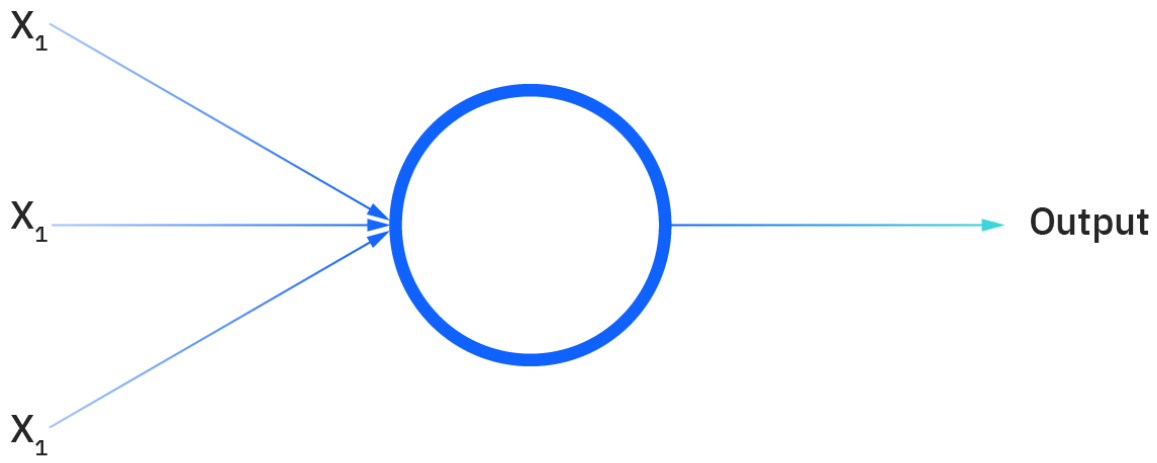


Рисунок 2.6 – Перцептрон

Нейронні мережі з прямим зв'язком, або багатошарові перцептрони (MLP), це те, на чому ми в першу чергу зосередилися в цій статті. Вони складаються з вхідного шару, прихованого шару або шарів та вихідного шару. Хоча ці нейронні мережі також зазвичай називають MLP, важливо відзначити, що насправді вони складаються з сигмоподібних нейронів, а не перцептронів, оскільки більшість реальних завдань є нелінійними. Дані зазвичай вводяться в ці моделі для навчання, і вони є основою для комп'ютерного зору, обробки природної мови та інших нейронних мереж.

Згорткові нейронні мережі (CNN) схожі на мережі прямого зв'язку, але вони зазвичай використовуються для розпізнавання зображень, розпізнавання образів та комп'ютерного зору. Ці мережі використовують принципи лінійної алгебри, зокрема матричне множення, виявлення закономірностей в зображенні.

Рекурентні нейронні мережі (RNN) ідентифікуються за їх петлями зворотного зв'язку. Ці алгоритми навчання в основному використовуються при використанні даних часових рядів для прогнозування майбутніх результатів, таких як прогнози ринку або прогнозування продажів.

2.3 Згорткові нейронні мережі

Штучний інтелект став свідком монументального зростання у подоланні розриву між можливостями людей та машин. Дослідники та ентузіасти однаково працюють над численними аспектами цієї галузі, щоб досягти дивовижних результатів. Однією з багатьох областей є область комп'ютерного зору.

Порядок денний у цій галузі полягає в тому, щоб дати машинам можливість бачити світ так само, як і люди, сприймати його аналогічним чином і навіть використовувати знання для безлічі завдань, таких як розпізнавання зображень та відео, аналіз та класифікація зображень, відтворення медіа, системи рекомендацій. , обробка природної мови і т. д. Досягнення в області комп'ютерного зору з глибоким навчанням були створені та вдосконалені з часом, в основному на одному конкретному алгоритмі - **згорткової нейронної мережі**.

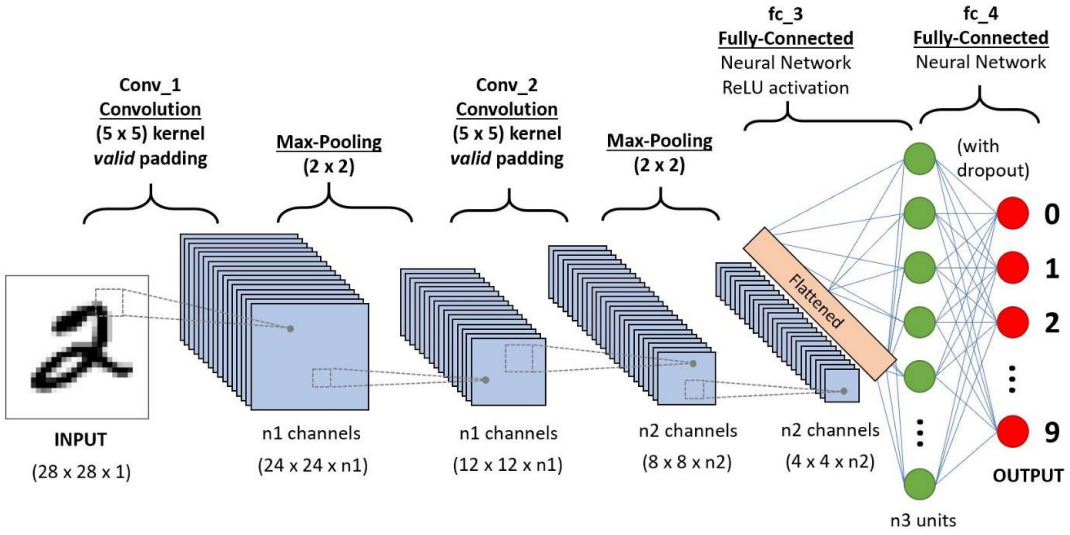


Рисунок 2.7 – Послідовність CNN для класифікації рукописних цифр

Згорткова нейронна мережа (ConvNet/CNN) — це алгоритм глибокого навчання, який може приймати вхідне зображення, привласнювати важливість (навчені ваги та зміщення) різним аспектам/об'єктам на зображенні та мати можливість відрізняти один від одного. Попередня обробка, необхідна ConvNet, набагато нижче порівняно з іншими алгоритмами класифікації. У той час, як у примітивних методах фільтри розробляються вручну, при достатньому навчанні ConvNets мають можливість вивчати ці фільтри/характеристики.

Архітектура ConvNet аналогічна схемі підключення нейронів у людському мозку і була натхненна організацією зорової кори. Окремі нейрони реагують на подразники лише обмеженої області поля зору, відомої як рецептивне поле. Набір таких полів перекривається, щоб покрити усю візуальну область.

Зображення - це не що інше, як матриця значень пікселів, чи не так? Так чому б просто не згладити зображення (наприклад, матрицю зображення 3x3 вектор 9x1) і передати його в багаторівневий перцептрон для цілей класифікації?

У разі надзвичайно простих бінарних зображень метод може показувати середню оцінку точності при прогнозуванні класів, але не матиме точності, коли йдеться про складні зображення, що мають піксельні залежності всюди.

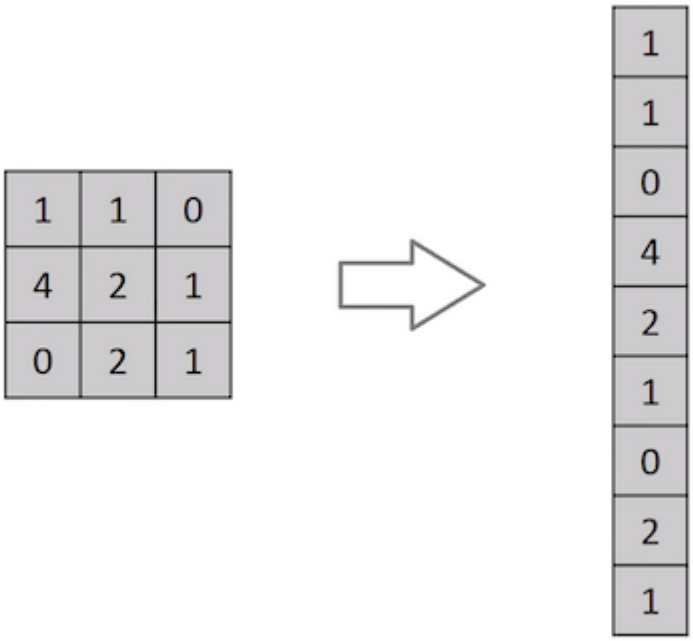


Рисунок 2.8 – Зведення матриці зображення 3x3 у вектор 9x1

CNN може успішно фіксувати просторові та часові залежності у зображенні за допомогою відповідних фільтрів. Архітектура найкраще підходить для набору даних зображень завдяки зменшенню кількості задіяних параметрів та можливості повторного використання ваг. Іншими словами, мережу можна навчити краще розуміти складність зображення.

На малюнку нижчеми маємо зображення RGB, розділене трьома кольоровими площинами — червоною, зеленою та синьою. Існує низка таких колірних просторів, в яких існують зображення - відтінки сірого, RGB, HSV, CMYK і т.д.

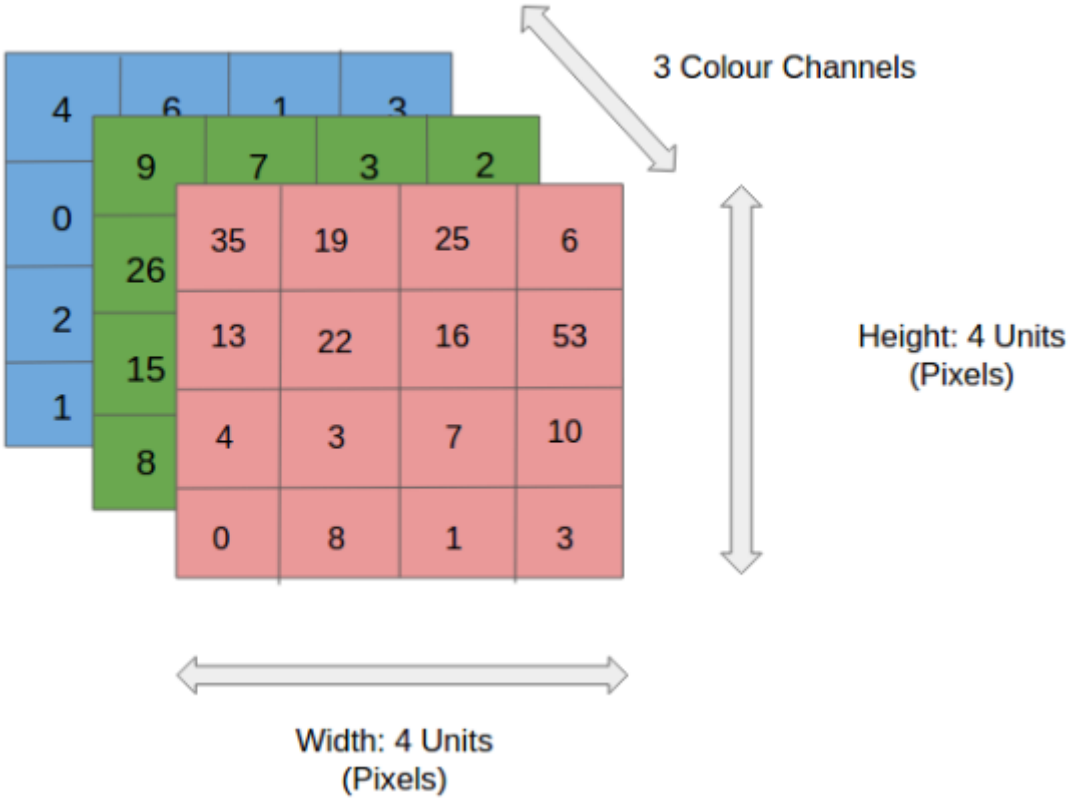


Рисунок 2.9 – RGB-зображення 4x4x3

Ви можете уявити, наскільки інтенсивними будуть обчислення, коли зображення досягнуть розмірів, скажімо, 8K (7680×4320). Роль CNN полягає в тому, щоб перетворити зображення на форму, яку легше обробляти, без втрати функцій, які мають вирішальне значення для отримання хорошого прогнозу. Це важливо, коли ми повинні розробити архітектуру, яка не тільки хороша для вивчення функцій, але й масштабується для великих наборів даних.

Шар згортки – ядро:

Розміри зображення = 5 (висота) x 5 (ширина) x 1 (кількість каналів, наприклад, RGB)

У наведеній нижче демонстрації зелена секція нагадує наше вхідне зображення 5x5x1, I. Елемент, що бере участь у виконанні операції згортки в першій частині шару згортки, називається ядром/фільтром, K, і представлений жовтим кольором. Ми вибрали K як матрицю 3x3x1.

Далі буде наведено два етапи згортки зображення:

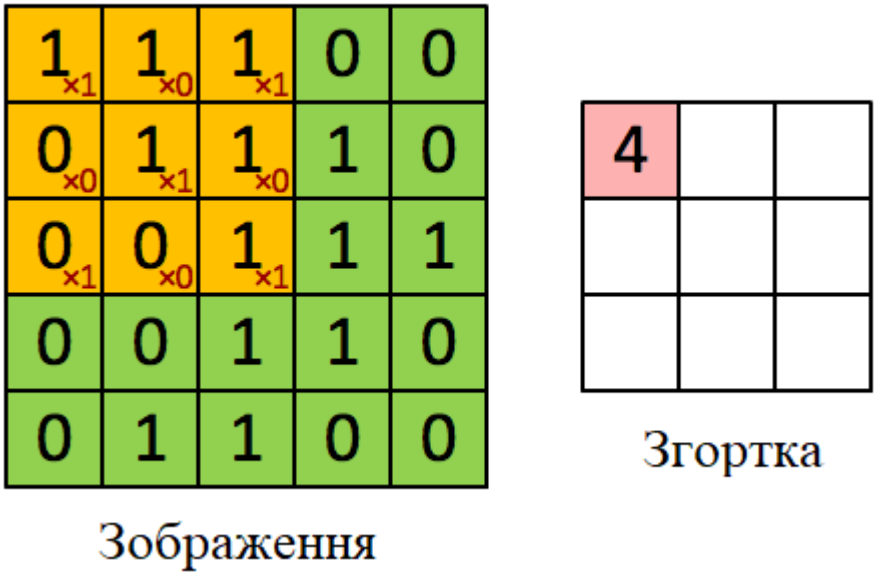


Рисунок 2.10 – Перший крок згортки

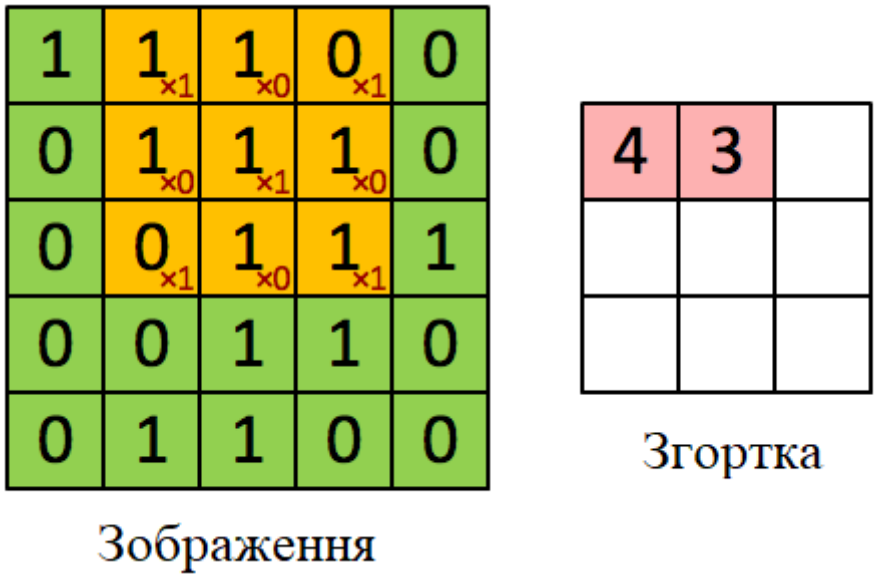


Рисунок 2.11 – Другий крок згортки

Таким чином продовжується доки матриця 3x3 не встане у ліву крайню частину вхідного зображення.

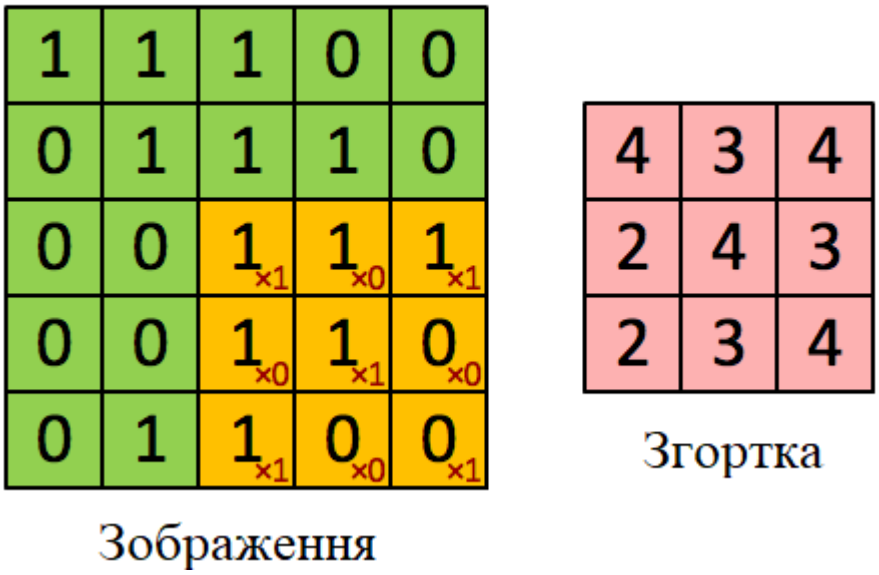


Рисунок 2.12– Останній крок згортки

Ядро зміщується 9 разів через довжину кроку = 1 (без кроку), щоразу виконуючи операцію матричного множення між K та частиною P зображення, над якою зависає ядро.

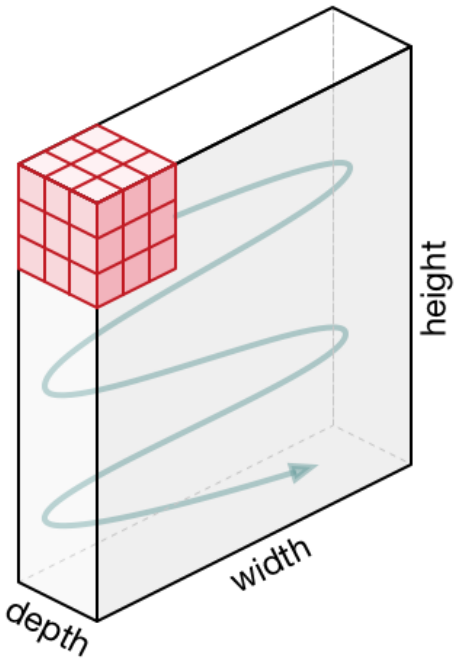


Рисунок 2.13 – Переміщення ядра

Як ми можемо побачити на рисунку вище, фільтр переміщається вправо з певним значенням кроку, доки не проаналізує всю ширину. Рухаючись далі, він переходить до початку (ліворуч) зображення з тим самим значенням кроку і повторює процес, доки не буде пройдено все зображення.

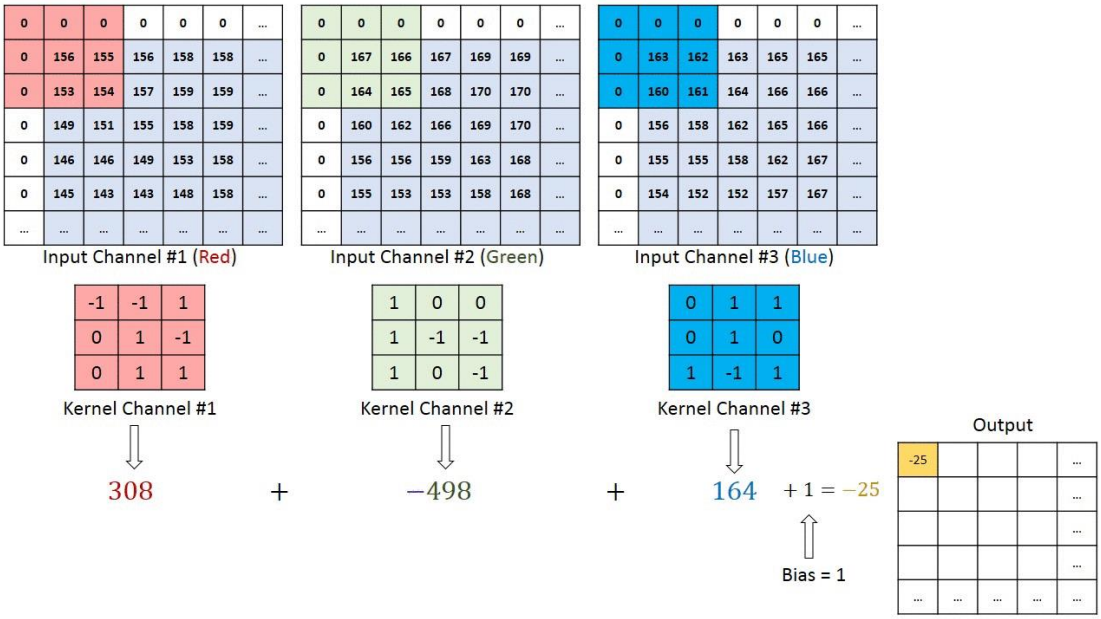


Рисунок 2.14 – Операція згортки на матриці зображення MxNx3 із ядром 3x3x3

У разі зображень з кількома каналами (наприклад, RGB) ядро має таку саму глибину, як і вхідне зображення. Розмноження матриць виконується між стеком Kn та In ([K1, I1]; [K2, I2]; [K3, I3]), і всі результати сумуються зі зміщенням, щоб дати нам стислий вихід згорткового каналу з однією глибиною.

Метою операції згортки є вилучення високорівневих функцій, таких як краї з вхідного зображення. ConvNets не потрібно обмежувати лише одним згортковим шаром. Зазвичай перший ConvLayer відповідає за захоплення функцій низького рівня, таких як краї, колір, орієнтація градієнта тощо. З додаванням шарів архітектура також адаптується до функцій високого рівня, даючи нам мережу, яка має корисне розуміння. зображень у наборі даних, як і ми. Нижче буде наведено два кроки операції згортки з кроком 2:

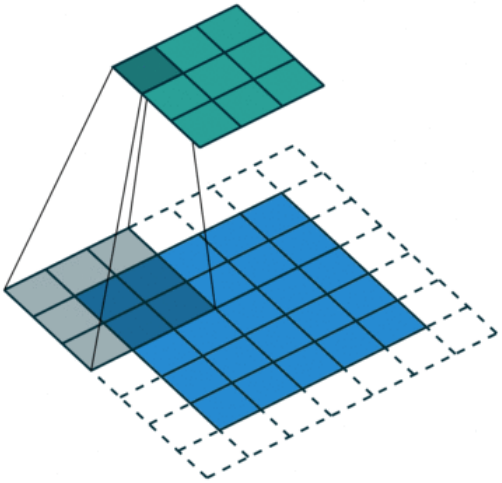


Рисунок 2.15 – Перший крок операції згортки із довжиною кроку = 2

Як ми можемо побачити на малюнку нижче, ядро яке спочатку на першому кроці знаходилось на координаті [1,1], змістилося на координату [1,3]

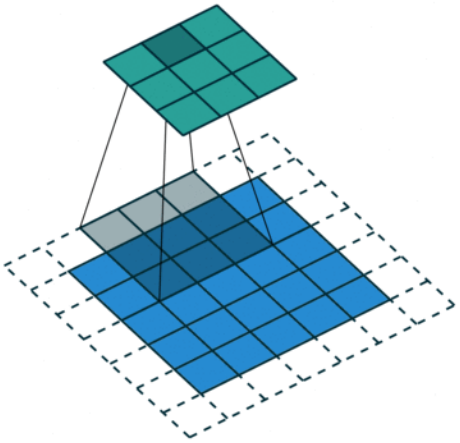


Рисунок 2.16 – Другий крок операції згортки із довжиною кроку = 2

Є два типи результатів операції - один, у якому згорнутий ознака зменшується у розмірності порівняно з вхідними даними, а інший, у якому розмірність або збільшується, або залишається незмінною. Це робиться шляхом застосування **Valid Padding** у разі першого або **Same Padding** у разі останнього.

Коли ми збільшуємо зображення 5x5x1 до зображення 6x6x1, а потім застосовуємо до нього ядро 3x3x1, виявляємо, що згорнута матриця виявляється розміром 5x5x1. Звідси і назва – **Same Padding.**

З іншого боку, якщо ми виконаємо ту саму операцію без заповнення, нам буде представлена матриця, яка має розміри самого ядра (3x3x1) **Valid Padding**.

Подібно до згортального шару, **шар пулінгу** відповідає за зменшення просторового розміру згорнутого об'єкта. Це необхідно зменшення обчислювальної потужності, необхідної для обробки даних за рахунок зменшення розмірності. Крім того, це корисно для отримання домінуючих ознак, які є інваріантними до обертання та положення, таким чином підтримуючи процес ефективного навчання моделі.

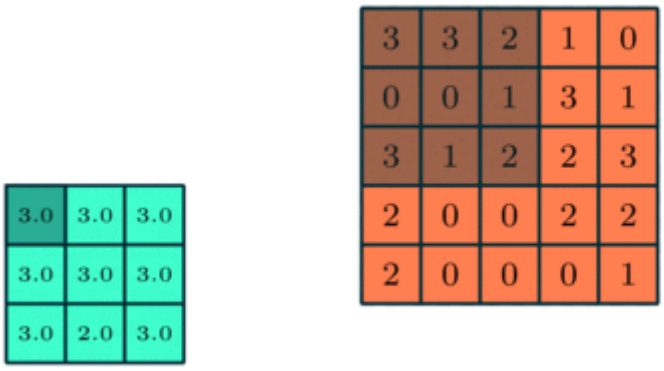


Рисунок 2.17 – Пулінг 3x3 за згорнутою функцією 5x5

Існує два типи пулінгу: максимальний пулінг та середній пулінг. Максимальний пулінг повертає максимальне значення частини зображення, що охоплюється ядром. З іншого боку, середній пулінг повертає середнє значення всіх значень частини зображення, охоплюваної ядром.

Максимальний пулінг також працює як шумоглушник. Він повністю відкидає шумові активації, а також виконує шумозаглушення разом із зменшенням розмірності. З іншого боку, середній пулінг просто зменшує розмірність як

механізм придушення шуму. Отже, ми можемо сказати, що максимальний пулінг працює набагато краще ніж середній пулінг.

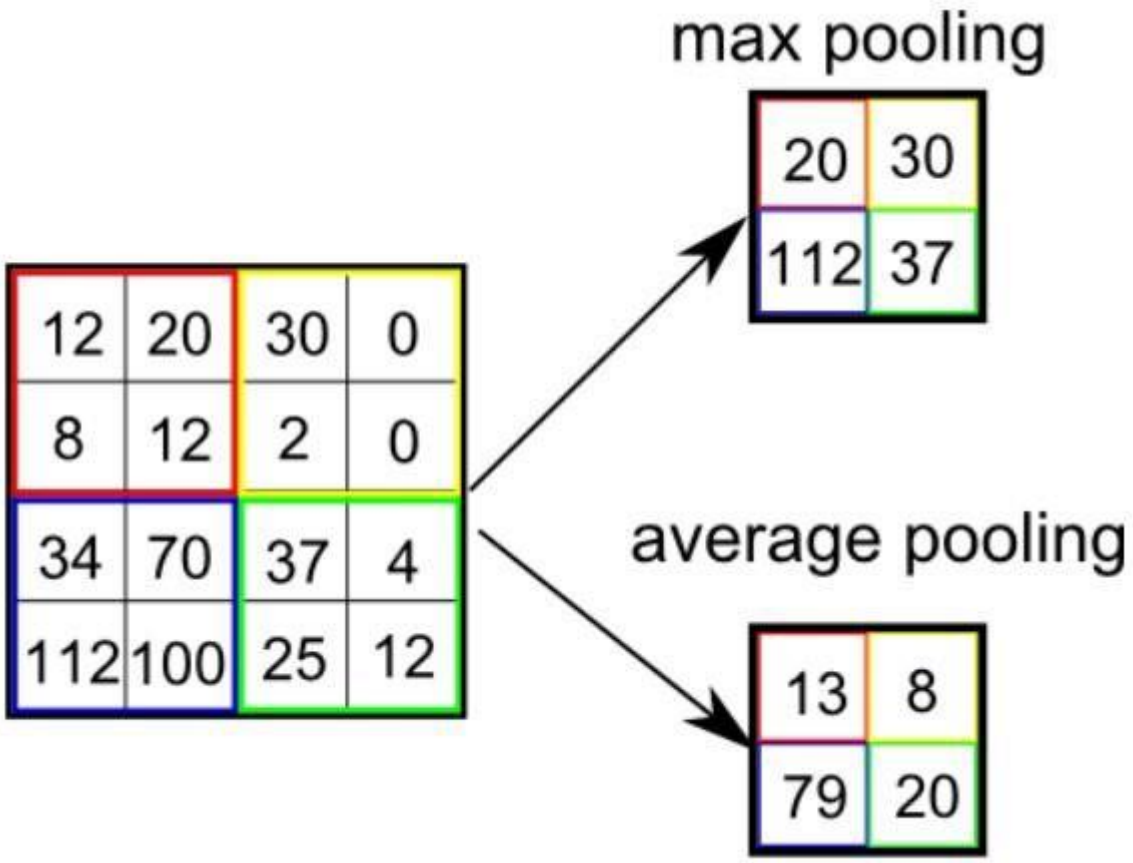


Рисунок 2.18 – Типи пулінгу

Згортковий шар і об'єднуючий шар разом утворюють і-й шар згорткової нейронної мережі. Залежно від складності зображень кількість таких шарів може бути збільшено ще для більшого захоплення деталей низького рівня, але за рахунок більшої обчислювальної потужності.

Додавання повнозв'язного шару - це (зазвичай) дешевий спосіб вивчення нелінійних комбінацій високорівневих функцій, представлених вихідними даними згорткового шару. Повнозв'язний шар вивчає можливо нелінійну функцію в цьому просторі.

Тепер, коли ми перетворили наше вхідне зображення у відповідну форму для нашого багаторівневого перцептрону, ми згладимо зображення у вектор-стовпець. Згладжений висновок подається в нейронну мережу з прямим зв'язком, а зворотне

поширення застосовується до кожної ітерації навчання. Протягом низки епох модель здатна розрізняти домінуючі та деякі низькорівневі ознаки на зображеннях та класифікувати їх за допомогою методу класифікації Softmax.

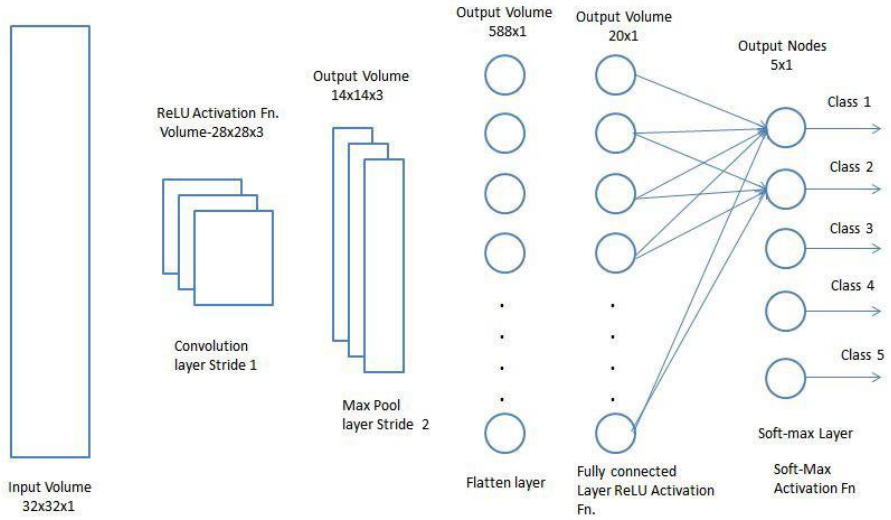


Рисунок 2.19 – Схема методу класифікації для згорткової нейронної мережі

Доступні різні архітектури CNN, які відіграли ключову роль у побудові алгоритмів, які забезпечують та підтримуватимуть ІІ в цілому в найближчому майбутньому. Деякі з них перераховані нижче:

1. LeNet;
2. AlexNet;
3. VGGNet;
4. GoogLeNet;
5. ResNet;
6. ZFNet.

3 РЕАЛІЗАЦІЯ ЗГОРТКОВОЇ НЕЙРОННОЇ МЕРЕЖІ ДЛЯ ФОРМУВАННЯ КОНТУРУ БУДІВЛІ

3.1 Опис початкового набору даних

Результат роботи нейронних мереж напряму залежить від початкового набору даних, а тому необхідно зробити якісний датасет, для розроблюваної згорткової нейронної мережі. В Інтернеті можна знайти дуже багато супутникових знімків, але є один нюанс.

Коли людина шукає будівлі на супутниковому знімку, то в першу чергу вона помічає дахи. Але висота будівель відрізняється, супутник виконуючи знімки, може зняти одну й ту саму місцевість з різних кутів. З цього випливає, що якщо ми помістимо на векторну карту прямокутник, який відноситься до даху, то в нас не має впевненості та гарантій що при обновленні знімку з супутника дах не «з'їде». З фундаментом вже інша річ. Фундамент заритий у землю, я з якого кута супутник не буде знімати, він весь час буде залишатись на одному місці.

Проблема різних територій вагомо впливає на результативність розроблюваної згорткової нейронної мережі. В Інтернеті можна знайти дуже багато цікавих територій та країн. Але навіть якщо ми будемо тренувати нашу нейронну мережу на одній країні то можна побачити, що навіть у одному місці існують дуже різноманітні будівлі та типи забудувань.

З цього виплило немало питань. Зібрати величезний датасет а потім тренувати? Робити датасет тільки для певних типів забудувань? Чи зробити базову мережу а потім дотреновувати її під різні типи забудувань?

На малюнку нижче, можна побачити різні будівлі та різні типу забудувань одного міста.

Рисунок 3.1 – Типи забудувань одного міста

Зробивши певний аналіз, було з’ясовано, що:

1. Безсумнівно треба розширяти датасет під різні типи забудувань, який буде використовуватись для тренування нашої згорткової нейронної мережі. Мережа, тренована на одному типі забудувань, спроможна виділяти будівлі іншого типу забудувань, хоч і досить не якісно;

2. Краще тренувати одну велику мережу на великому наборі данних. Шляхом тестувань було виявлено, що такий спосіб гарно узагальнюється на різні території. Якщо тренувати окремі мережі для кожного типу забудувань, то якість або залишиться як тренування на великому наборі даних, або покращиться непомітно. Таким чином, використання різних мереж для різних територій немає сенсу. Ще треба додати, що це потребує більшої кількості даних і додаткового класифікатора типу забудування;

3. Якщо використовувати старі мережі при додаванні нових територій у дані, мережі будуть тренуватися в рази краще. Дотренування старих мереж, задля розширення результатних даних приводить до аналогічного результату, що і тренування мережі з нуля, однак потребує мінімального часу.

Виходячи з всього вищесказаного було обран Inria Aerial Image Labeling Dataset.

Маркування аерофотознімків Inria торкається основної теми дистанційного зондування: автоматичне попіксельне маркування аерофотознімків (посилання на документ).

Особливості набору даних:

– покриття 810 км² (405 км² для навчання та 405 км² для тестування);

– ортотрансформовані аерофотознімки з просторовою роздільною здатністю 0,3 м;

– наземні справжні дані для двох семантичних класів: побудова і не побудова (публічно розкриваються тільки для навчального підмножини);

– зображення охоплюють різні міські поселення, починаючи від густонаселених районів (наприклад, фінансовий район Сан-Франциско) і до альпійських міст (наприклад, Лієнц в австрійському Тиролі).

Замість того, щоб розбивати сусідні частини тих самих зображень на навчальну та тестову підгрупи, до кожної з підгруп включаються різні міста. Наприклад, зображення над Чикаго включені в набір для навчання (а не в набір для тестування), а зображення над Сан-Франциско включені в набір для тестування (а не в набір для навчання).

Кінцевою метою цього набору даних є оцінка здатності методів до узагальнення: хоча зображення Чикаго можуть використовуватися для навчання, система повинна маркувати аерофотознімки інших регіонів з різними умовами освітлення, міським ландшафтом та часом року.

Набір даних було створено шляхом об'єднання загальнодоступних зображень та офіційних слідів будівлі, які є суспільним надбанням. Нижче буде наведено приклад зображень з датасету.

Рисунок 3.2 – приклад зображень з датасету

Після того, як було обрано зображення, їх потрібно підготувати для подальшого використання.

Далі буде реалізовано дві нейронні мережі на базі двох архітектур Q-Net та MaskRCNN. Для них використовується різне анотування.

Для анотування зображень для архітектури MaskRCNN було використано веб застосунок Makesense.ai. Makesense.ai – безкоштовний онлайн-інструмент для маркування фотографій з відкритим кодом, який підтримує таки типи маркування як : прямокутники, лінії, крапки, полігони. Що до вихідного формату то

Makesense.ai забезпечує такі вихідні формати як YOLO, VOC XML, VGG, JSON, CSV. Нижче розглянемо процес анотування зображення.

Першим кроком зайдемо у Makesense.ai за посиланням https://www.makesense.ai/.

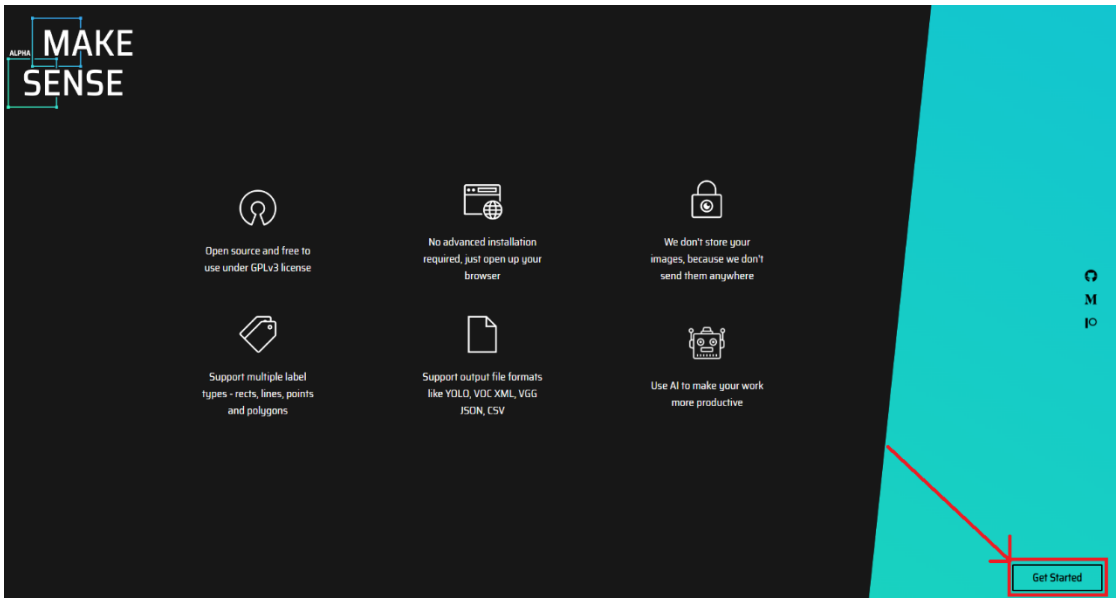


Рисунок 3.3 – Головна сторінка Makesense.ai

Після переходу за посиланням, перед нами з’явиться головна сторінка веб застосунку. Для того, щоб продовжити роботу, натискаємо на кнопку у правому нижньому куту (виділено червоним прямокутником, та стрілкою вказаною на нього) Get Started.

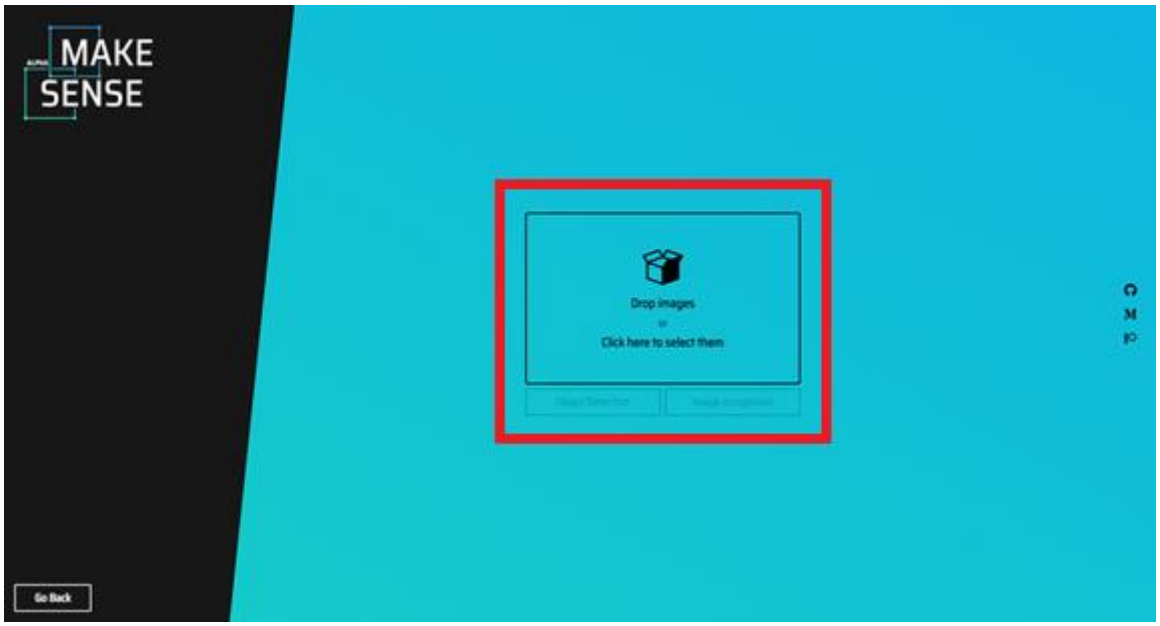


Рисунок 3.4 – Сторінка завантаження зображень

Після натискання кнопки Get Started, відкриється сторінка завантаження зображень. Натискаємо на кнопку Drag image or Click here to select them ,та вибираємо зображення у діалоговому вікні, що відкрилося, або перетаскуємо зображення у виділену зону Drag and drop.

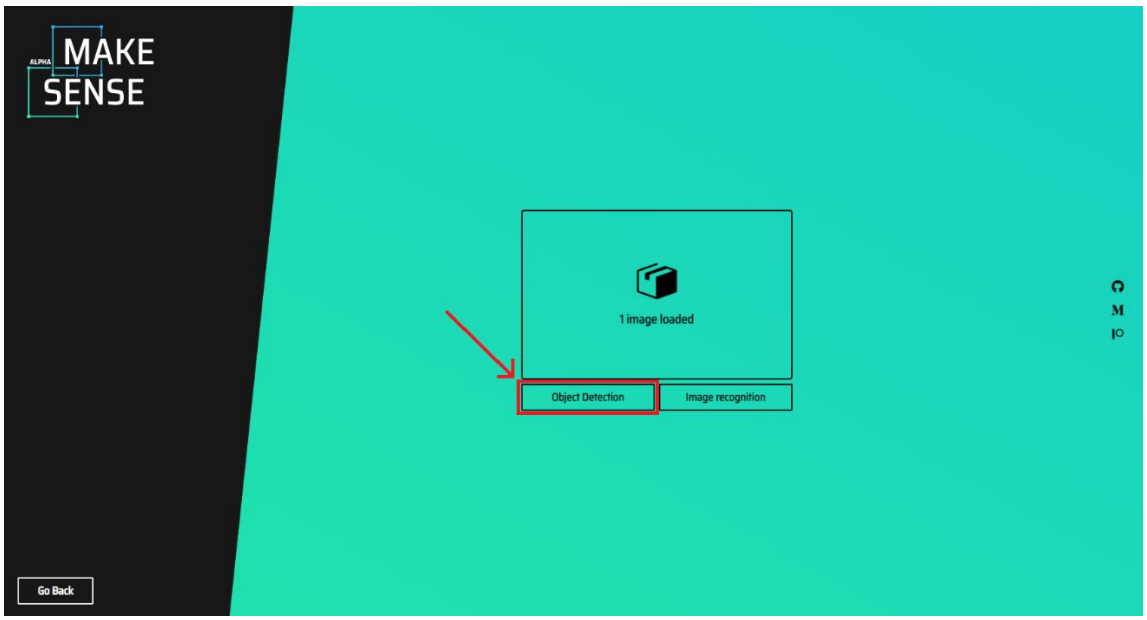


Рисунок 3.5 – Сторінка завантаження зображень

Як ми бачимо на рисунку вище, після обирання або перетаскування зображень для анотування, стануть доступні для натискання наступні кнопки Object Detection та Image recognition – вибираємо Object Detection.

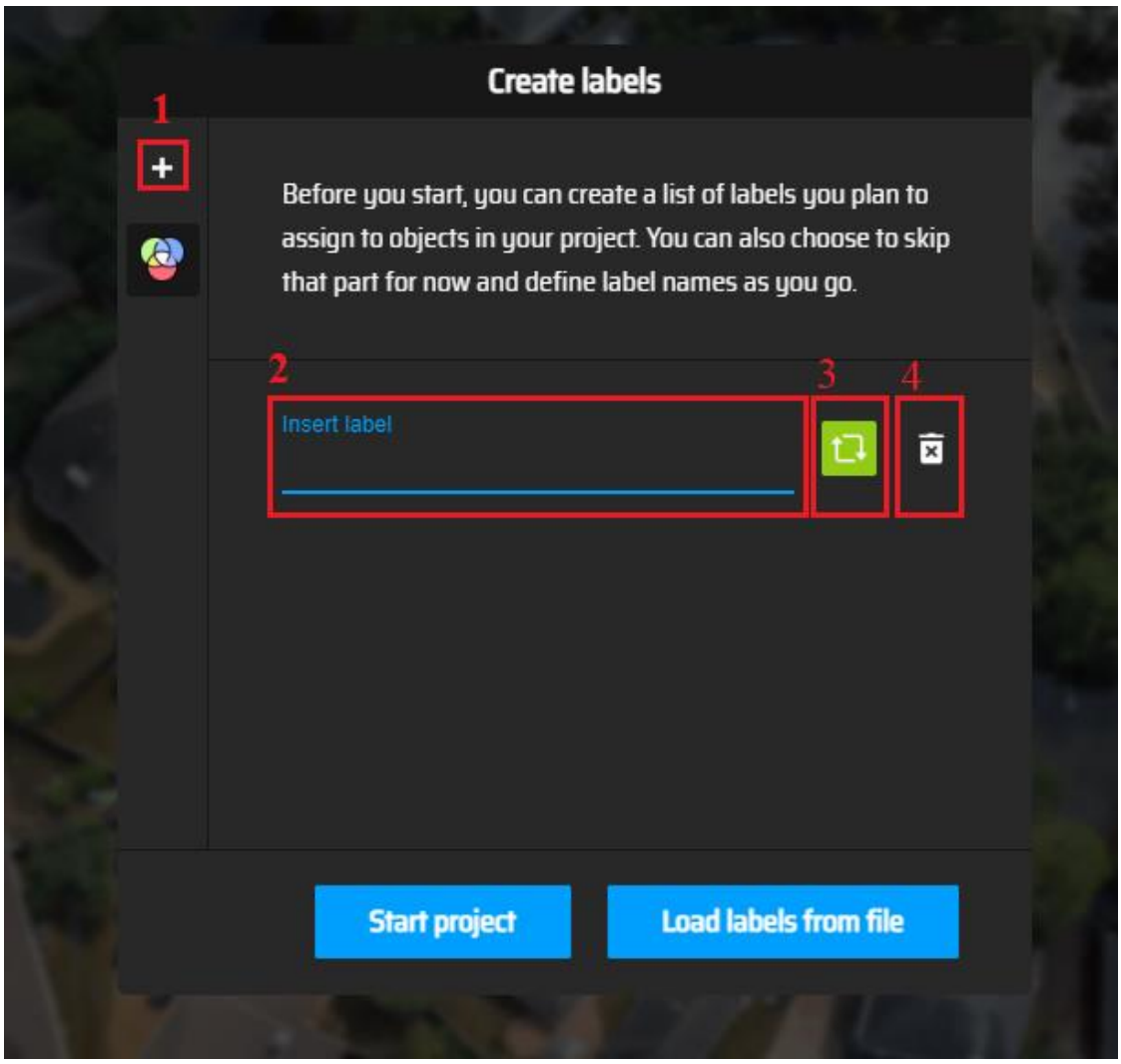


Рисунок 3.6 – Вибір назви маркувань

Після натискання кнопки Object Detection з’явиться вікно створення маркувань (класів). Нижче будуть описані дії кнопок під цифрами.

1. Додавання нового класу;
2. Назва класу;
3. Колір класу;
4. Видалення класу.

Після закінчення формування класу натискаємо на кнопку Start Project.

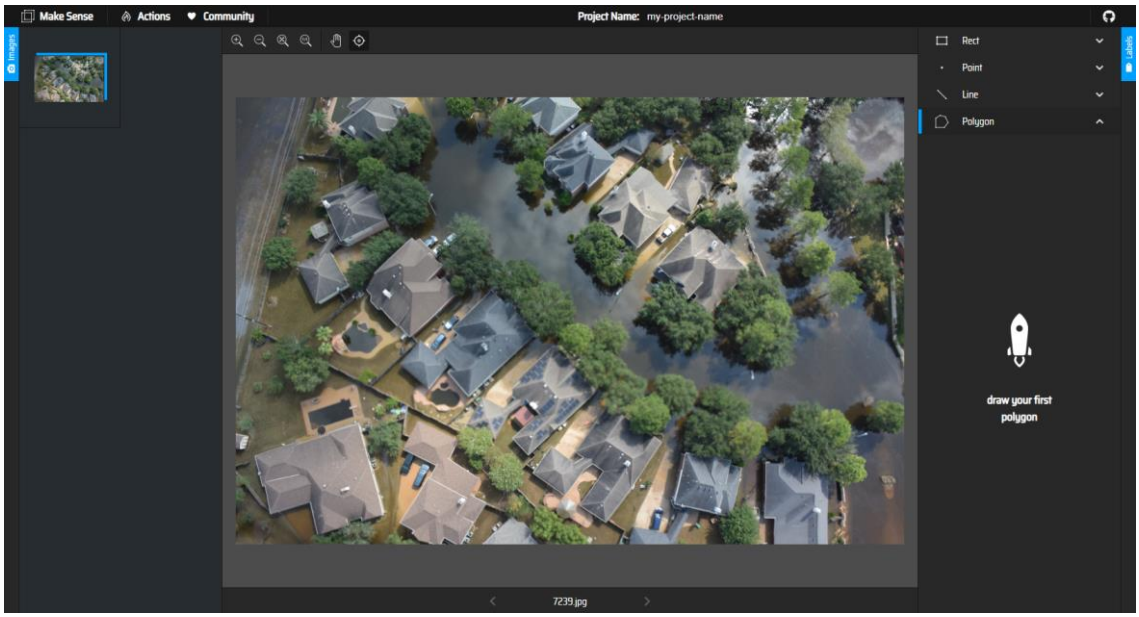


Рисунок 3.7 – Головний інтерфейс веб застосунку

Отже, ми можемо приступити до анотування зображень. Вибираємо метод маркування Poligon та починаємо обводити будівлі.

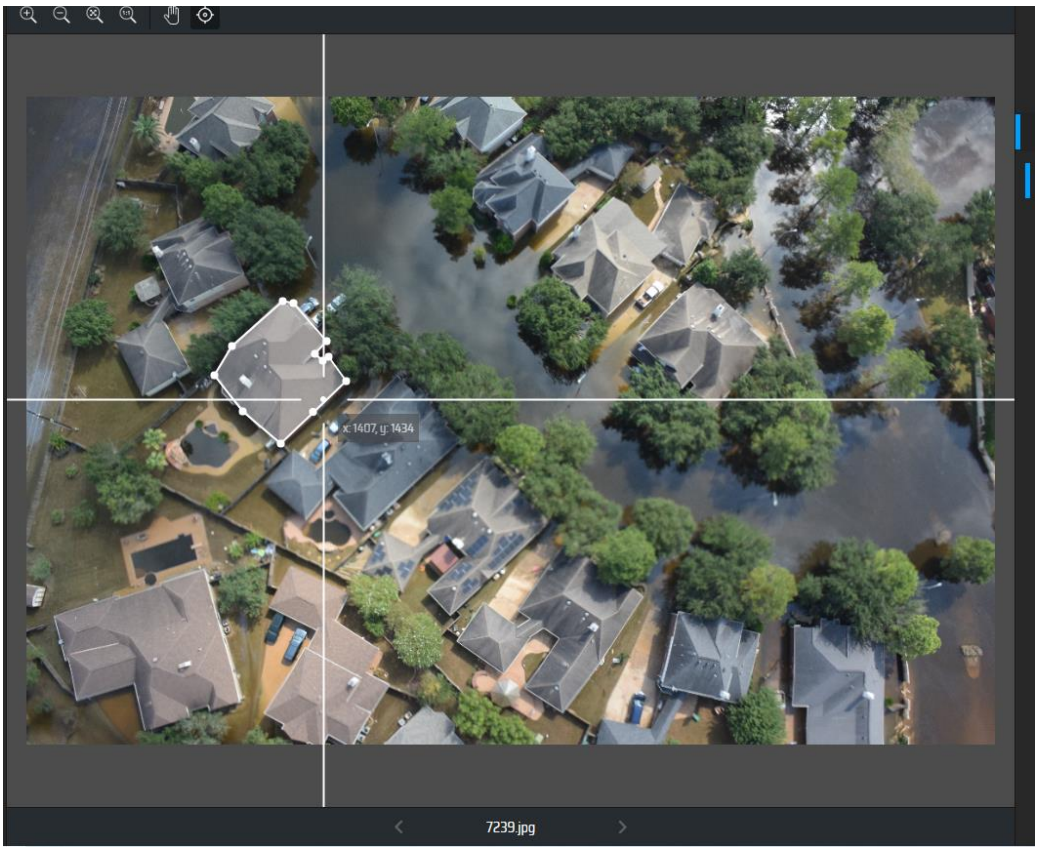


Рисунок 3.8 – Приклад маркування однієї будівлі

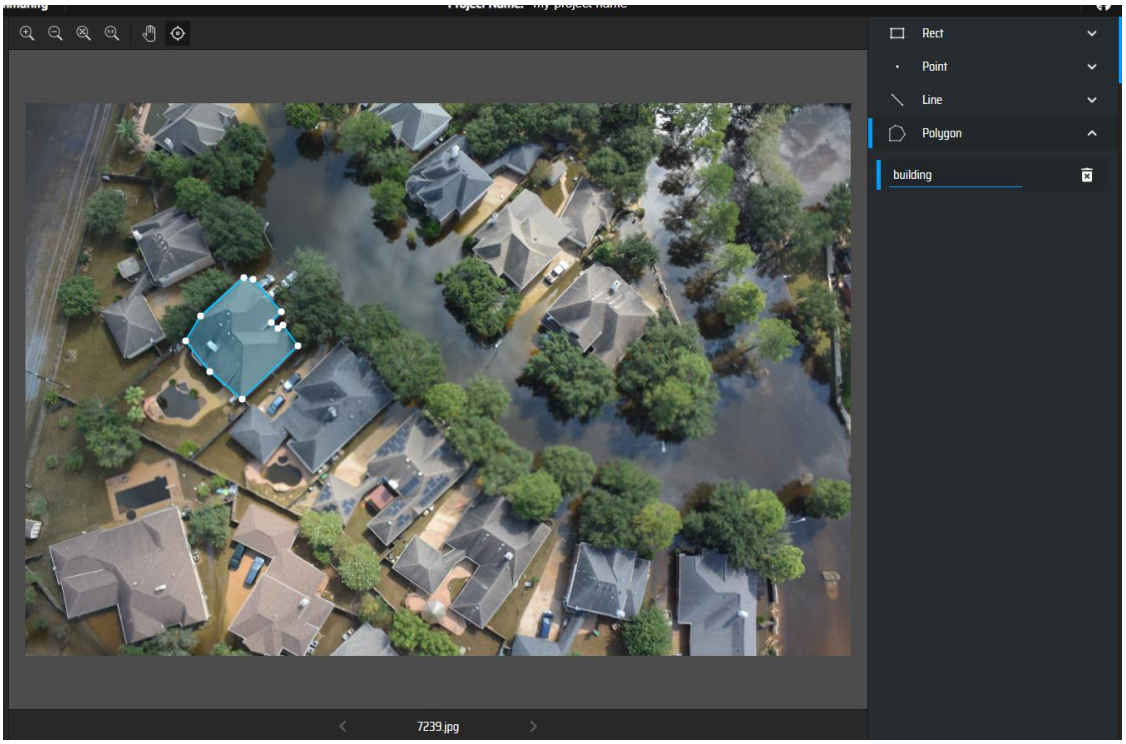


Рисунок 3.9 – Приклад маркування однієї будівлі

Як ми можемо побачити на малюнку вище, маркована нами будівля має клас building – це можна побачити справа. Таким чином продовжуємо маркувати всі будівлі. Важливо робити точні маркування, тому не слід знехтувати кількістю точок.

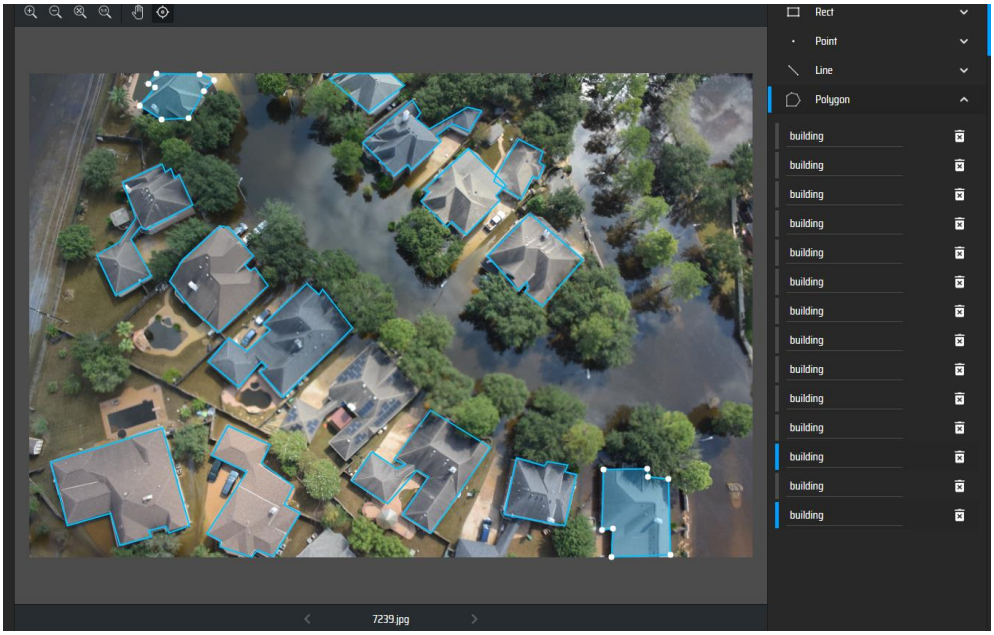


Рисунок 3.10 – Приклад маркування всіх будівель

Після маркування всіх будівель, треба експортувати дані у форматі VGG.json. Для цього натискаємо на Actions - Export Annotations.

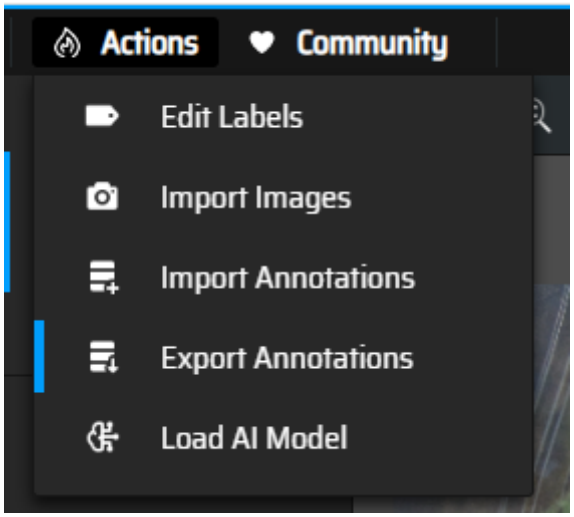


Рисунок 3.11 – Вкладка Actions

Потім з’явиться вікно експорту, натискаємо Polygon – Single file in VGG JSON format - Export. Згодом після натискання завантажиться файл анотацій у вибраному форматі.

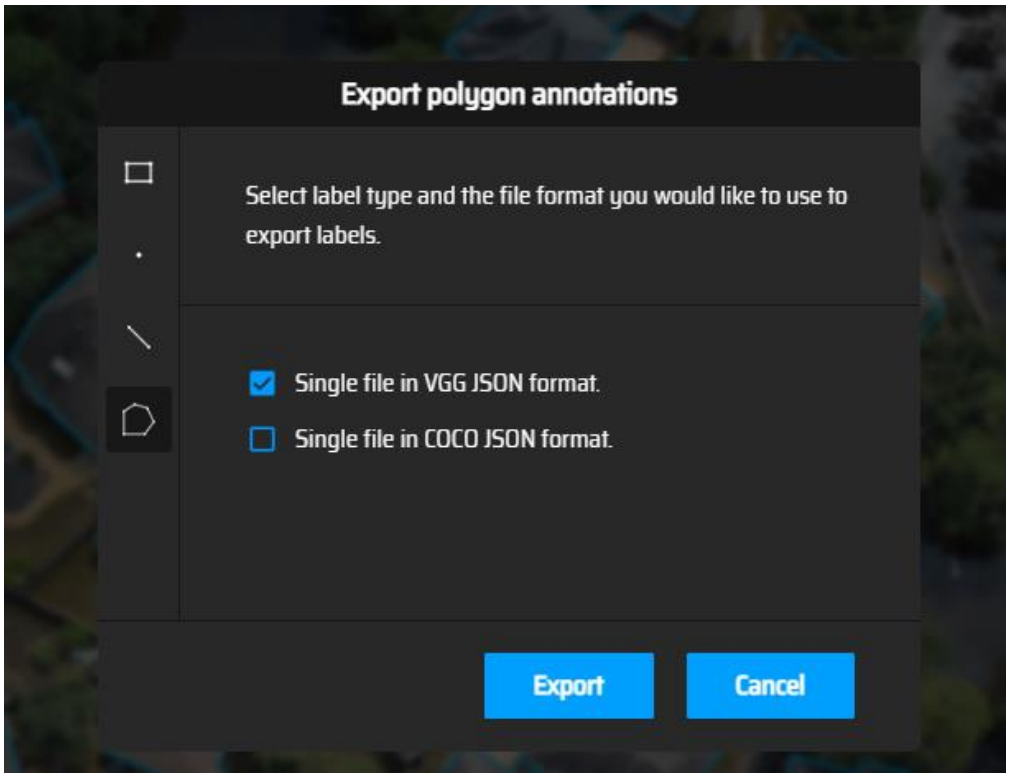


Рисунок 3.12 – Експортування анотацій

Для кращої видимості змісту та структури json файлу використаємо JSON Reader.

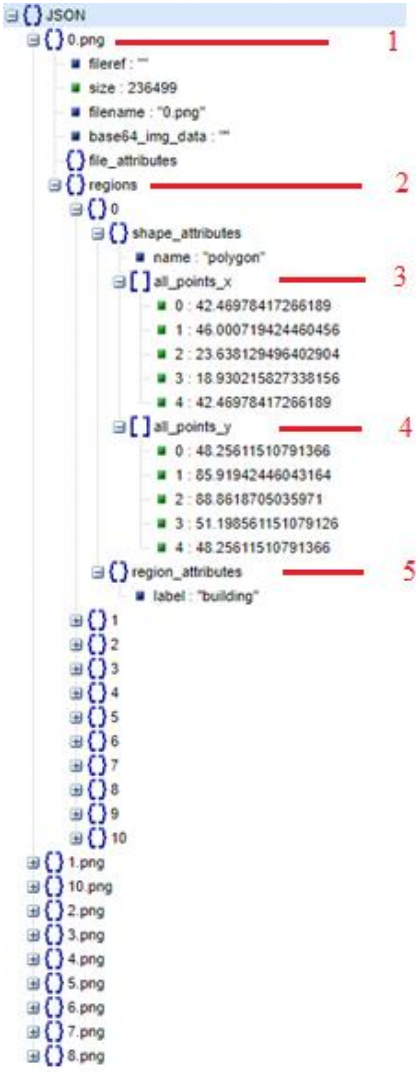


Рисунок 3.13 – Структура JSON файлу

Виходячи с цього, можна побачити, що JSON файл складається з:

1. Назва зображення;
2. Регіони (будівлі);
3. Точки по координаті X;
4. Точки по координаті Y;
5. Маркування регіонів (класи).

Для U-Net потрібно було підготувати маски для класів. Якщо для MaskRCNN в процесі анотування кінцевим результатом буде один JSON файл, то для масок U-NET потрібно для кожного зображення повинен буди файл з маскою. Для формування масок було використано Adobe Photoshop CS6.

Adobe Photoshop - багатофункціональний графічний редактор, що розробляється і розповсюджується компанією Adobe Systems. Здебільшого працює з растровими зображеннями, однак має деякі векторні інструменти. Продукт є лідером ринку в галузі комерційних засобів редагування растрових зображень та найвідомішою програмою розробника.

В даний час Photoshop доступний на платформах MacOS, Windows та iPadOS. Для Windows Phone та Android доступна спрощена версія програми під назвою Adobe Photoshop Touch. Також існує версія Photoshop Express для Windows Phone 8 та 8.1. Приклад зображення та його маски можна побачити на зображенні нижче.

Рисунок 3.14 – Зображення для тренування та його маска

3.2 Реалізація Mask R-CNN архітектури на Python

Mask R-CNN - це згорткова нейронна мережа (CNN), сучасна з погляду сегментації зображень. Цей варіант глибокої нейронної мережі виявляє об'єкти на зображенні та створює високоякісну маску сегментації для кожного екземпляра.

Рисунок 3.15 – Приклад результату роботи Mask R-CNN

Mask R-CNN складається з декількох компонентів:

1. Згорткові нейронні мережі (CNN);
2. Регіональні згорткові нейронні мережі (R-CNN);
3. Швидший R-CNN з регіональними мережами пропозицій (RPN);
4. Маска R-CNN і як вона працює.

Так як у другій частині було розглянуто принцип роботи згорткових нейронних мереж, то продовжимо з пояснення регіональних згорткових мереж.

R-CNN або RCNN розшифровується як Region-Based Convolutional Neural Network, це тип моделі машинного навчання, який використовується для завдань комп'ютерного зору, зокрема, для виявлення об'єктів.

Щоб зрозуміти, що таке RCNN, ми розглянемо архітектуру RCNN. На наступному зображенні зображено концепцію регіональних CNN (R-CNN).

У цьому підході використовуються рамки, що обмежують, в областях об'єкта, які потім оцінюють згорткові мережі незалежно для всіх областей інтересу (ROI), щоб класифікувати кілька областей зображення в пропонований клас.

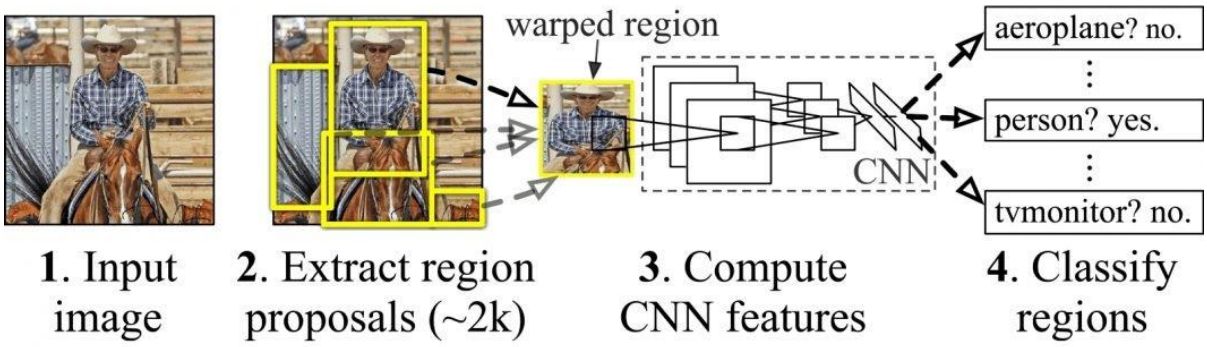


Рисунок 3.16 – Концепція R-CNN

Архітектура RCNN була розроблена для вирішення задач виявлення зображень. Крім того, архітектура R-CNN складає основу Mask R-CNN, і вона була покращена до того, що ми знаємо як швидший R-CNN.

Що таке Faster R-CNN? Faster R-CNN - це покращена версія архітектури R-CNN з двома етапами:

– регіональна мережа пропозицій (RPN). RPN – це просто нейронна мережа, яка пропонує кілька об'єктів, доступних на певному зображенні;

– Faster R-CNN. Це витягує функції з використанням RoIPool (об'єднання областей інтересу) з кожного поля-кандидата та виконує класифікацію та регресію рамки, що обмежує. RoIPool - це операція з вилучення невеликої карти ознак з кожної області виявлення при виявленні.

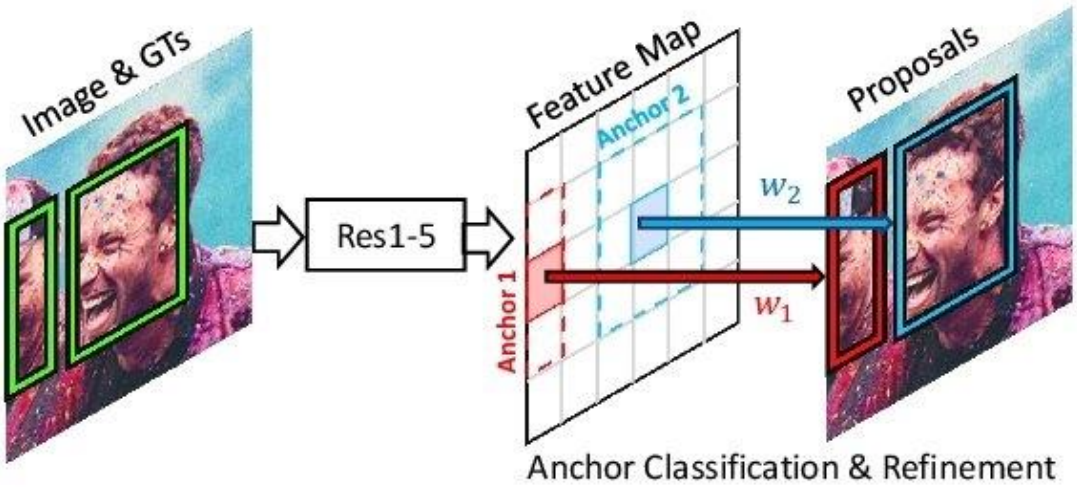


Рисунок 3.17 – Концепція мереж регіональних пропозицій (RPN)

Faster R-CNN просуває цей потік, вивчаючи механізм уваги за допомогою мережі регіональних пропозицій та архітектури Fast R-CNN. Причина, через яку "Fast R-CNN" швидше, ніж R-CNN, полягає в тому, що вам не потрібно щоразу передавати 2000 пропозицій по регіонах згорткової нейронної мережі. Натомість операція згортки виконується лише один раз для кожного зображення, і з нього створюється картка об'єктів.

Крім того, Faster R-CNN це оптимізована форма R-CNN, оскільки вона створена для підвищення швидкості обчислень (запуск R-CNN набагато швидше).

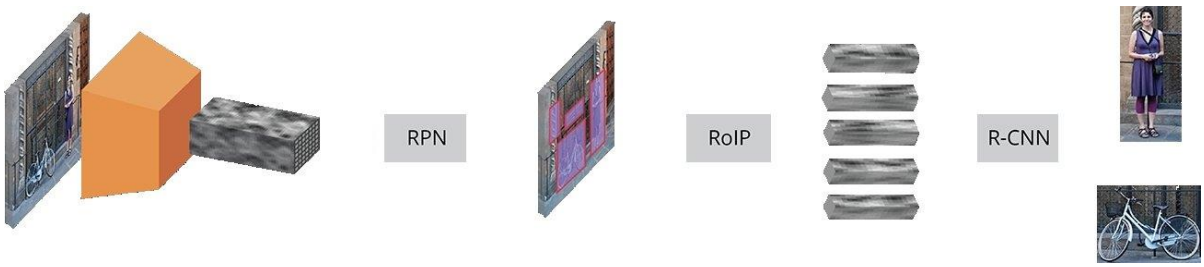


Рисунок 3.18 – Концепція Faster R-CNN

Mask R-CNN або Mask RCNN - це згорткова нейронна мережа (CNN), сучасна з точки зору сегментації зображень та сегментації екземплярів. Mask R-CNN був розроблений на основі Faster R-CNN, регіональної згорткової нейронної мережі.

Перший крок до розуміння того, як працює Mask R-CNN, вимагає розуміння концепції сегментації зображення.

Завдання комп'ютерного зору "Сегментація зображення" - це процес поділу цифрового зображення на кілька сегментів (наборів пікселів, також відомих як об'єкти зображення). Ця сегментація використовується для виявлення об'єктів та меж (ліній, кривих тощо).

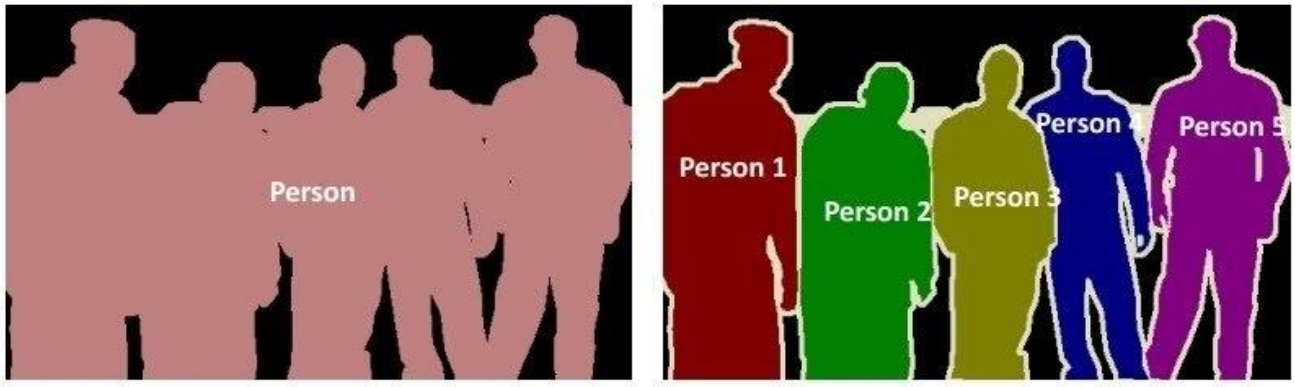
Є 2 основних типи сегментації зображення, які підпадають під дію Mask R-CNN:

1. Семантична сегментація;
2. Сегментація екземпляра.

Семантична сегментація

Семантична сегментація класифікує кожен піксель за фіксованим набором категорій без диференціації екземплярів об'єкта. Іншими словами, семантична сегментація має справу з ідентифікацією/класифікацією схожих об'єктів як єдиного класу на рівні пікселів.

Як показано на зображенні нижче, всі об'єкти були класифіковані як єдине ціле (людина). Семантична сегментація інакше відома як сегментація фону, оскільки відокремлює об'єкти зображення від фону.



Семантична сегментація Сегментація екземпляру

Рисунок 3.19 – Концепція Faster R-CNN

Сегментація екземплярів або розпізнавання екземплярів займається правильним виявленням всіх об'єктів на зображенні, а також точним сегментуванням кожного екземпляра. Таким чином, це комбінація виявлення об'єктів, локалізації об'єктів та класифікації об'єктів. Іншими словами, цей тип сегментації йде далі, щоб дати чітку різницю між кожним об'єктом, що класифікується як аналогічні екземпляри.

Як показано у наведеному вище прикладі зображення, для сегментації екземплярів всі об'єкти є людьми, але цей процес сегментації поділяє кожну людину як єдине ціле. Семантична сегментація інакше відома як сегментація переднього плану, тому що вона акцентує увагу на об'єктах зображення, а не на тлі.

Як працює Mask R-CNN?

Маска R-CNN була побудована з використанням Faster R-CNN. У той час як Faster R-CNN має 2 виходи для кожного об'єкта-кандидата, мітку класу та усунення

обмежувальної рамки, Mask R-CNN – це додавання третьої гілки, яка виводить маску об'єкта. Додатковий висновок маски відрізняється від виведення класу та блоку, вимагаючи вилучення набагато більш тонкого просторового компонування об'єкта.

Mask R-CNN є розширенням Faster R-CNN і працює шляхом додавання гілки для прогнозування маски об'єкта (області інтересу) паралельно з існуючою гілкою для розпізнавання рамки, що обмежує.

Переваги Маски R-CNN:

– простота: Mask R-CNN простий у навчанні;

– продуктивність: Mask R-CNN перевершує всі існуючі записи з однією моделлю в кожному завданні;

– ефективність: цей метод є дуже ефективним і додає лише невеликі накладні витрати Faster R-CNN;

– гнучкість: Mask R-CNN легко узагальнити інші завдання. Наприклад, у тій самій структурі можна використовувати Mask R-CNN для оцінки пози людини.

Мовою програмування було обрано Python як одну з найсучасніших та перспективних мов програмування для реалізації нейронних мереж. Для реалізації та тренування Mask R-CNN було встановлено необхідні залежності:

– numpy;
– scipy;
– pillow;
– cython;
– matplotlib;
– scikit-image;
– tensorflow>=1.3.0;
– keras>=2.0.8;
– opencv-python;
– h5py;
– imgaug;

– IPython.

Згідно з проведеної оптимізації та пошуку найкращих результатів було налаштовано згорткову нейрону мережу (див. додаток А).

Створений застосунок має 2 режими, а саме : навчання та основний. Якщо запущений режим навчання, то застосунок буде тренувати ваги, які знаходяться у директорії застосунку, згідно з налаштуваннями нейронної мережі. Такий режим викликається командою: python custom.py train /шлях до датасету/ --weights = coco.

Приклад тренування буде наведений на зображеннях нижче. За кожної епохи показує такі дані:

Epoch 2/30

8/8 [==============================] - 4s 474ms/step

– loss: 3.3554;
– rpn_class_loss: 0.5085;
– rpn_bbox_loss: 0.9771;
– mrcnn_class_loss: 0.1544;
– mrcnn_bbox_loss: 1.2109;
– mrcnn_mask_loss: 0.5046;
– val_loss: 3.3560;
– val_rpn_class_loss: 0.3462;
– val_rpn_bbox_loss: 0.9019;
– val_mrcnn_class_loss: 0.1871;
– val_mrcnn_bbox_loss: 1.2964;
– val_mrcnn_mask_loss: 0.6245.

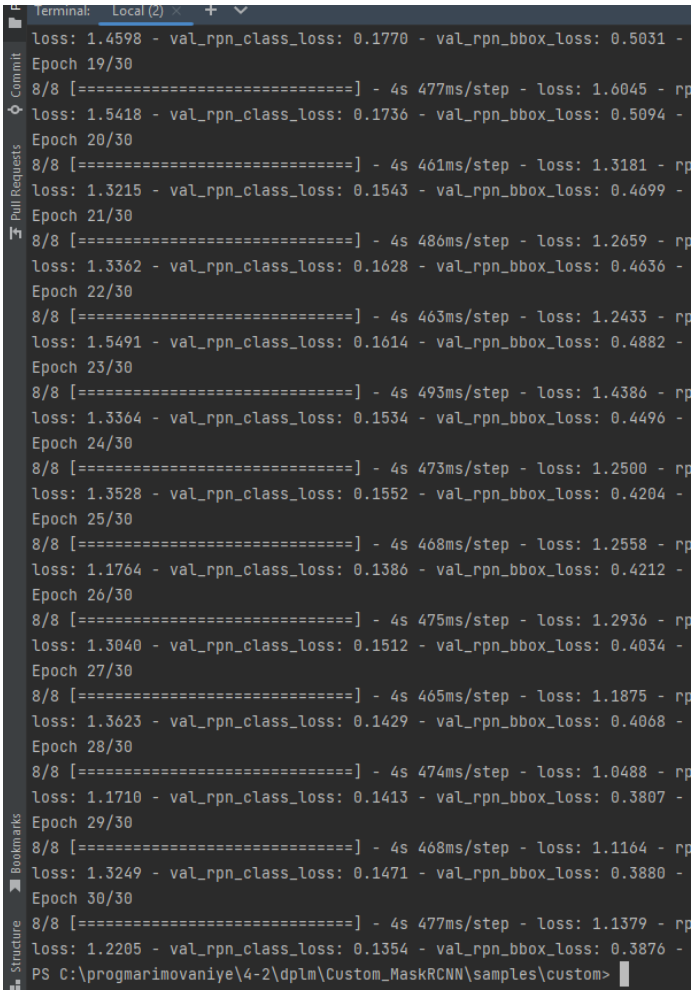


Рисунок 3.20 – Тренування розробленої нейронної мережі

Як можна побачити, кінцева точність складає 89%, ця точність буде згодом покращена.

Після успішного тренування ваг можна вибирати основний режим роботи, який запускається за допомогою файлу main.py. Цей режим буде обробляти всі зображення, які знаходяться у папці test в директорії застосунку. Приклади виконання програми у цьому режимі буде наведено нижче.

Рисунок 3.21 – Приклад роботи застосунку

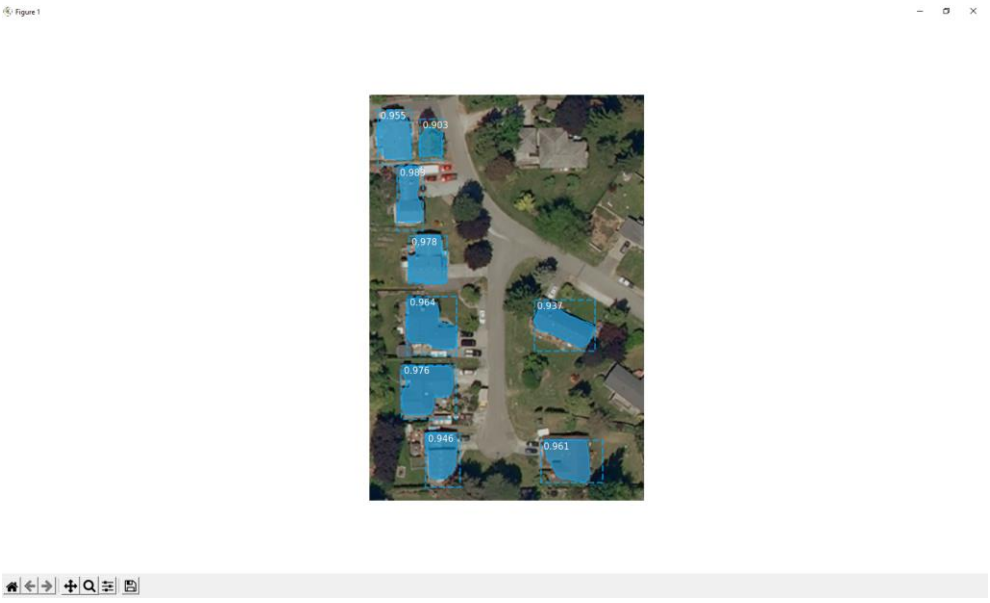


Рисунок 3.22 – Приклад роботи застосунку

3.3 Реалізація U-Net архітектури на Python

U-Net – це архітектура для семантичної сегментації. Він складається з шляху, що звужується і розширюється. Контрактний шлях відповідає типовій архітектурі згорткової мережі. Він складається з повторного застосування двох згорток 3x3 (згорток без доповнень), за кожною слідує випрямлена лінійна одиниця (ReLU) та операція максимального об'єднання 2x2 з кроком 2 для зниження дискретизації. На

кожному етапі зниження дискретизації ми подвоюємо кількість функціональних каналів.

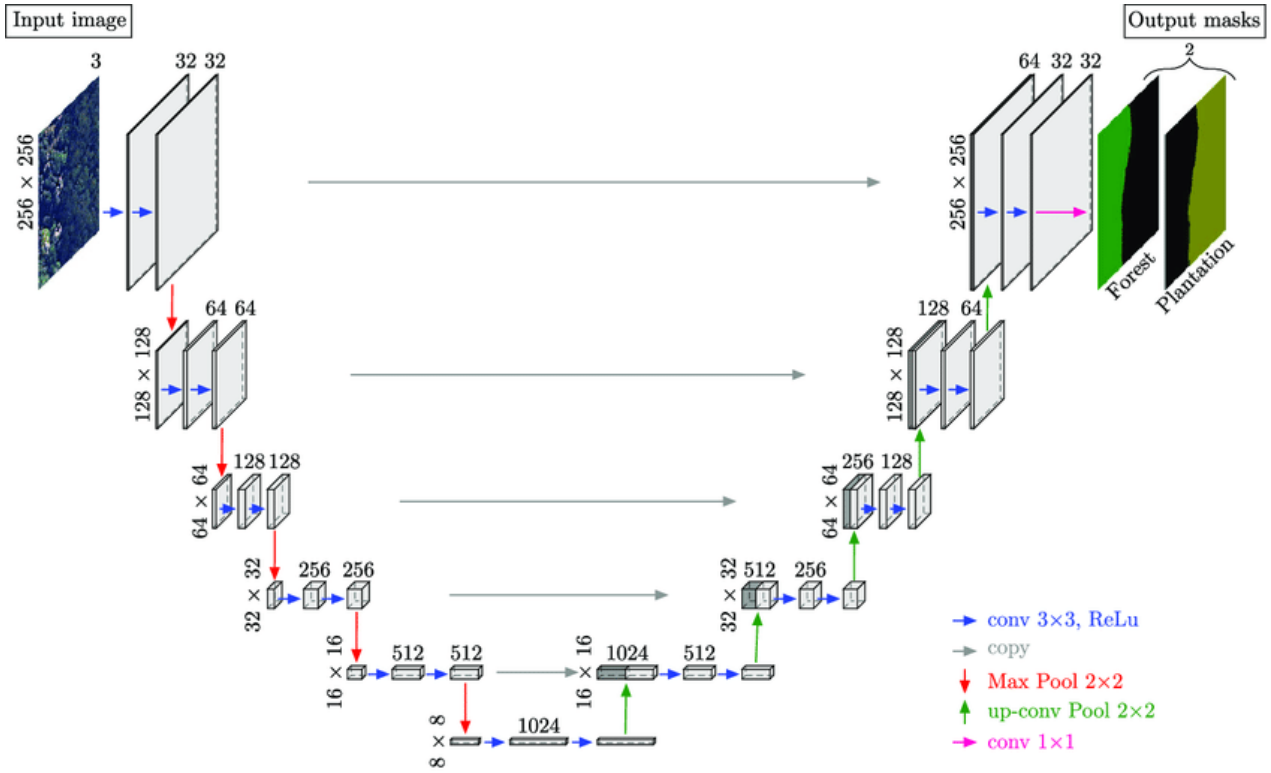


Рисунок 3.23 – Приклад U-Net архітектури

Кожен крок розширеного шляху складається з підвищує дискретизації карти ознак, за якою слідує згортка 2x2 («висхідна згортка»), яка вдвічі зменшує кількість каналів ознак, конкатенація з відповідним чином обрізаною картою ознак з дороги, що скорочується, і два 3x3. згорток, за кожним з яких слідує ReLU. Обрізка необхідна через втрату крайових пікселів у кожному згортці. На останньому рівні згортка 1x1 використовується для порівняння кожного 64-компонентного вектора ознак з бажаною кількістю класів. Усього мережа має 23 згорткові шари.

Переваги використання U-Net:

– обчислювально ефективний;
– навчальний з невеликим набором даних;
– навчені від початку до кінця;
– переважно для біомедичних додатків.

Що таке семантична сегментація та локалізація? Простіше кажучи, вони відносяться до маркування на рівні пікселів, тобто кожен піксель у зображенні має позначку класу. Ви отримуєте карти сегментації, що виглядають як на рис.1. До цього біомедичні дослідники дотримувалися двох підходів:

1. Класифікація зображення загалом (злоякісне чи доброякісне);
2. Поділ зображень на ділянки та їх класифікація.

Через збільшений розмір набору даних виправлення, безумовно, було краще, ніж класифікація всього зображення, однак у нього було кілька недоліків, пов'язаних з ним. Менші кроки або виправлення з великою кількістю перекриттів вимагають великих обчислювальних ресурсів і призводять до надмірної інформації, що повторюється. По-друге, життєво важливим є хороший компроміс між контекстною інформацією та локалізацією. Невеликі виправлення призводять до втрати контекстуальної інформації, тоді як виправлення спотворюють результати локалізації. Нарешті, патчі, що не перекриваються, призводять до втрати контекстної інформації. На основі попередніх спостережень, архітектура кодер-декодер дає набагато вищі значення перетину по об'єднанню (IoU), ніж передача кожного пікселя CNN для класифікації.

Мовою програмування було обрано Python як одну з найсучасніших та перспективних мов програмування для реалізації нейронних мереж. Для реалізації та тренування Mask R-CNN було встановлено необхідні залежності:

- Tensorflow;
- Keras >= 1.0;
- Libtiff;
- OpenCV 3.

Створений застосунок має 2 режими: навчання та основний. Якщо запущений режим навчання, то застосунок буде тренувати ваги, які знаходяться у директорії застосунку, згідно з налаштуваннями нейронної мережі.

Процес тренування буде наведений на зображеннях нижче.

```
Epoch 1/30
90/90 [==============================] - 90s 1s/step - loss: 0.7482 - val_loss: 0.7945
Epoch 2/30
90/90 [==============================] - 92s 1.04ms/step - loss: 0.4964 - val_loss: 0.5646
Epoch 3/30
90/90 [==============================] - 90s 1s/step - loss: 0.4039 - val_loss: 0.3900
Epoch 4/30
90/90 [==============================] - 89s 0.967ms/step - loss: 0.3582 - val_loss: 0.3574
Epoch 5/30
90/90 [==============================] - 89s 0.988ms/step - loss: 0.3335 - val_loss: 0.3607
Epoch 6/30
```

Рисунок 3.24 – Процес тренування нейронної мережі

Після успішного тренування ваг можна вибирати основний режим роботи, який запускається за допомогою файлу main.py. Цей режим буде обробляти всі зображення, які знаходяться у папці test в директорії застосунку. Приклади виконання програми у цьому режимі буде наведено нижче.

Рисунок 3.25 – Приклад роботи застосунку

Спеціальний розділ

ОХОРОНА ПРАЦІ

до кваліфікаційної роботи

на тему:

«Застосування згорткових нейронних мереж для формування контуру будівлі»

Спеціальність 122 «Комп'ютерні науки»

122 – БКР – 401.21810102

Виконав студент 4-го курсу, групи 401
В. О. Балутін
(підпис, ініціали та прізвище)
«___» _____ 2022 р.

Консультант *ст. викладач*
(наук. ступінь, вчене звання)
О. В. Макарова
(підпис, ініціали та прізвище)
«___» _____ 2022 р.

Миколаїв – 2022

4 ОХОРОНА ПРАЦІ

Для підтримання максимальної працездатності та забезпечення необхідних умов праці необхідно притримуватись відповідних нормативно-правових актів. Однією із основних систем, нормативно правова база яких спрямована на регулювання правових, санітарно-гігієнічних, організаційних та технічних заходів та засобів для збереження життя та здоров’я та забезпечення високого рівня працездатності людини у процесі трудової діяльності є охорона праці. На території України дана система врегульована законом «Про охорону праці» [20]. Закон встановлює загальні вимоги з охорони праці для всіх підприємств та суб’єктів підприємницької діяльності незалежно від форм власності та видів діяльності. Тобто, положення даного закону так само впливають на підприємства, сфера діяльності яких спрямована інформаційні технології.

Досить багато проблем зі здоров'ям можуть виникнути через використання комп'ютера. Наприклад, проблеми із зором, проблеми зі спиною, також від комп'ютера надходить електромагнітне випромінювання.

При роботі з комп'ютером людина повністю залежить від положення дисплея. Крім того, зображення на екрані динамічно оновлюється, а низька частота оновлення викликає його мерехтіння. При цьому очні і внутрішньоочні м'язи, фокусують погляд, втомлюються від надмірного навантаження. Розвивається зорове стомлення, що сприяє виникненню короткозорості. Тривала робота з комп'ютером вимагає також підвищеної зосередженості, що призводить до появи головного болю, дратівливості, нервової напруги і стресу.

4.1 Нормативна документація щодо забезпечення охорони праці під час використання екранними пристроями

Існують правила, які працівники підприємств, установ повинні дотримуватися під час використання електронно-обчислювальних машин. Дотримування цих правил знизить наслідки несприятливої дії на працівників шкідливих та небезпечних факторів, які можуть супроводжуватися під час роботи з електронно-обчислювальними машинами. По перше, це стосується зорових та нервово-емоційних перевантажень, серцево-судинних захворювань та психічної складової працівника.

У правилах охорони праці під час експлуатації електронно-обчислювальних машин викладені гігієнічні та ергономічні вимоги до робочих приміщень та місць, параметрів робочого середовища, і якщо дотримуватися усіх цих правил, дає змогу уникнути порушення стану здоров'я користувача комп'ютера.

Відповідальність за виконання усіх цих правил покладається на посадових осіб, фізичних осіб , які займаються підприємницькою діяльністю та здійснюють застосування електронно-обчислювальних машин в адміністративних та промислових приміщеннях.

Державний санітарний нагляд за дотримання усіх цих правил державними органами, підприємствами, установами, організаціями незалежно від форми власності, а також фізичними особами, які займаються підприємницькою діяльністю, покладається на органи і установи санітарно – епідеміологічного профілю Міністерства охорони здоров'я України, відповідні установи, організації, частини та підрозділи Міністерства оборони України, закони України (ст. 31 Закону України «Про забезпечення санітарного та епідеміологічного благополуччя населення») [25].

4.2 Вимоги до приміщення з використанням екранних пристроїв

Усі приміщення, де робітники використовують персональні комп’ютери, повинні відповідати ряду вимог. Ці вимоги перевіряються лише спеціальними службами, які призначені для нагляду за дотриманням вимог охорони праці на підприємствах. Нижче наведено частину вимог з документу про правила охорони праці під час роботи на підприємствах:

− об'ємно-планувальні рішення будівель та приміщень для роботи з електронно-обчислювальними машинами мають відповідати вимогам ДСанПІН 3.3.2.007-98 [9];

− розміщення робочих місць з ВДТ ЕОМ і ПЕОМ у підвальних приміщеннях, на цокольних поверхах заборонено;

− площа на одне робоче місце становить не менше ніж 6,0 м, а об'єм - не менше ніж 20,0.м;

− приміщення для роботи з комп’ютерами повинні мати природне та штучне освітлення відповідно до СНиП II-4-79 [18];

− природне освітлення має здійснюватися через світлові прорізи, орієнтовані переважно на північ чи північний схід, і забезпечувати КПО не нижче, ніж 1,5%:

− виробничі приміщення повинні обладнуватись шафами для зберігання документів, магнітних дисків, полицями, стелажами, тумбами тощо, з урахуванням вимог до площі приміщень;

− у приміщеннях з електронно-обчислювальними машинами слід щоденно робити вологе прибирання;

− приміщення з електронно-обчислювальними машинами мають бути забезпечені аптечками першої медичної допомоги;

− на підприємствах, де є комп’ютери, мають бути обладнані побутові приміщення для відпочинку під час роботи, кімната психологічного розвантаження. В кімнаті психологічного розвантаження слід передбачити

встановлення пристроїв для приготування й роздачі тонізуючих напоїв, а також місця для занять фізичною культурою (СНиП 2.09.04.-87) [17].

Нижче наведено частину вимоги до безпеки робочих місць працівників з екранними пристроями з наказу Міністерством соціальної політики України «Про затвердження Вимог щодо безпеки та захисту здоров'я працівників під час роботи з екранними пристроями» на стан 18 травня 2018 року [15]:

- робочі місця працівників з екранними пристроями повинні бути спроектовані так і мати такі розміри, щоб працівники мали простір для зміни робочого положення і рухів;
- всі випромінювання від екранних пристроїв повинне бути зведене до гранично допустимого рівня з точки зору безпеки і охорони здоров'я працівників;
- організація робочого місця працівника з екранними пристроями повинна забезпечувати відповідність усіх елементів робочого місця та їх розташування ергономічним, антропологічними, психофізіологічним вимогам, а також характеру виконуваних робіт;
- освітлення робочого місця працівника з екранними пристроями повинно створювати відповідний контраст між екраном і навколишнім середовищем (з урахуванням виду роботи) і відповідати вимогам ДСанПІН 3.3.2.007-98 [21];
- мікроклімат виробничих приміщень з робочими місцями працівників з екранними пристроями повинен підтримуватися на постійному рівні і відповідати вимогам Санітарних норм мікроклімату виробничих приміщень ДСН 3.3.6.042-99 [22];
- робочий стіл або робочу поверхню повинні бути достатнього розміру і мати поверхню з низькою відбивною здатністю, допускати гнучкість при розміщенні екрана, клавіатури, документів і відповідного обладнання;
- робоче крісло має бути стійким та дозволяти працівникові з екранними пристроями легко рухатися і займати зручне положення. Сидіння повинна регулюватися по висоті, спинка сидіння - як по висоті, так і по нахилу;
- слід передбачати підставку для тих, кому це необхідно для зручності.

Крім тих вимог, які повинні дотримуватися усі робочі місця на підприємствах, є ще правила електронебезпеки. Нижче наведено частину вимог з документу про правила охорони праці під час експлуатації електронно-обчислювальної машини, розділ 2, пункт 2.3 – Вимоги електронебезпеки [23]:

− під час монтажу необхідно унеможливити виникнення електричного джерела загоряння через коротке замикання та перевантаження проводів;

− обов’язково встановити аварійний резервний вимикач, якщо у приміщення працює понад п’ять комп’ютерів;

− електромережу штепсельних розеток живлення для комп’ютерів та іншого обладнання прокладати у каналах, при цьому не допускається використання деяких матеріалів, що швидко загоряються.

Також існує вимоги до виробничого приміщення щодо параметрів мікроклімату, освітлення, шуму та вібрації, рівні електромагнітного та іонізуючого випромінювання. Це потрібно для зменшення впливу негативних факторів на праці працівника.

У виробничих приміщеннях на усіх робочих місцях з персональними комп’ютерами мають обов’язково забезпечуватися оптимальні значення параметрів мікроклімату, а саме температури відносної вологості й рухливості повітря (ГОСТ 12.1.005-88, СН 4088-86) [18, 19] (табл. 1).

Таблиця 4.1 - Норми мікроклімату для приміщень з ВТД ЕОМ та ПЕМ

Пора року	Категорія робіт	Температура повітря, °С, не більше	Відносна вологість повітря, %	Швидкість руху повітря, м/с
Холодна	Легка - 1а	22-24	40-60	0,1
	Легка - 16	21-23	40-60	0,1
Тепла	Легка - 1а	23-25	40-60	0,1
	Легка - 16	22-24	40-60	0,2

До категорії робіт 1а належать, що виконується робота сидячі і не потребує фізичного напруження людини, до категорії робіт 1б належать, що робота виконується сидячі, стоячи або пов'язані з ходінням та потребує деякого фізичного напруження.

Рівні позитивних та негативних іонів в повітрі приміщень з ВДТ мають відповідати санітарно – гігієнічним нормам №2152-80 [20] (табл. 2).

Таблиця 4.2 - Рівні іонізації повітря приміщень при роботі на ВДТ

Рівні	Кількість іонів в 1 см повітря	
	n+	n-
Мінімально необхідні	400	600
Оптимальні	1500-3000	3000-5000
Максимально допустимі	50000	50000

Штучне освітлення в приміщеннях з робочими місцями на підприємствах повинно бути обладнане ВДТ та має здійснюватися системою загального рівномірного освітлення. У виробничих та адміністративних приміщеннях, у разі роботою з документами, допускається застосування системи комбінованого освітлення.

Значення освітленості на поверхні робочого стола має становити 300-500лк. Якщо не має можливості забезпечити системою загального освітлення, допускається використання місцевим освітленням. Місцеве освітлення слід встановлювати так, щоб не створювати відблисків на поверхні екрану, а освітленість екрану персонального комп'ютера має не перевищувати 300лк.

Як джерело світу у приміщеннях для штучного освітлення мають використовуватися люмінесцентні лампи типу ЛБ. При використання відбитого освітлення у виробничих та адміністративних приміщеннях допускається використання метало-галогенних ламп, які мають потужність 250Вт.

Допускається використання ламп розжарювання у світильниках місцевого освітлення.

Інтенсивність потоків інфрачервоного випромінювання має не перевищувати допустимих значень, які зазначені у ДСН 3.3.6.042-99 [24].

Інтенсивність потоків ультрафіолетового випромінювання має не перевищувати допустимих значень, які зазначені у СН 4557-88 [19].

Потужність експозиційної дози рентгенівського випромінювання на відстані 0,05м від екрану та корпусу комп'ютера не повинна перевищувати 0,1мбер/рік (100мкР/рік).

4.3 Вимоги до роботи з екранними пристроями

На сьогодні існує ряд вимог, які повинні використовувати працівники на підприємствах, основним місцем роботи яких є місце за персональним комп'ютером. Нижче наведено частину вимог з наказу Міністерством соціальної політики України «Про затвердження Вимог щодо безпеки та захисту здоров'я працівників під час роботи з екранними пристроями» на стан 18 травня 2018 року [3]:

Мінімальні вимоги безпеки при роботі з екранними пристроями:

− щодня перед початком роботи необхідно очищати екран пристрою від пилу та інших забруднень;

− після закінчення роботи екранні пристрої слід відключати від електричної мережі;

− у разі виникнення аварійної ситуації необхідно негайно відключити екранний пристрій від електричної мережі.

Не допускається:

− виконувати технічне обслуговування, ремонт і налагодження екранних пристроїв безпосередньо на робочому місці працівника при роботі з екранними пристроями;

− відключати захисні пристрої, самочинно проводити зміни у конструкції та складі екранних пристроїв або їх технічне налагодження;

– працювати з екранними пристроями, в яких під час роботи з'являються нехарактерні сигнали, нестабільне зображення на екрані та інші несправності;

– при виконанні робіт операторського типу, пов'язаних з нервово-емоційним напругою, у приміщеннях при роботі з екранними пристроями, на пультах і постах керування технологічними процесами і в інших приміщеннях повинні дотримуватися оптимальні умови мікроклімату відповідно до вимог ДСН 3.3.6.042-99 [16].

Мінімальні вимоги безпеки до екранних пристроїв:

– екранні пристрої не повинні бути джерелом ризику для працівників.

– всі випромінювання, за винятком видимої частини електромагнітного спектра, повинні бути зведені до незначного рівня з точки зору безпеки і охорони здоров'я працівників.

– символи на екранних пристроях повинні бути чіткими, відповідного розміру. Між символами і рядками символів має бути належну відстань.

– зображення на екрані повинно бути стабільним, без спалахів або інших видів нестабільності.

– яскравість і/або контрастність символів має легко регулюватися працівником при роботі з екранними пристроями, а також швидко адаптуватися до навколишніх умов.

– вибираючи екрани, слід віддавати перевагу таким екранів, які легко і вільно повертаються і нахиляються у відповідності з потребою працівника.

– при необхідності може використовуватися окрема підставка або регульований стіл для розміщення екрану.

– екран не повинен відсвічувати або відбивати світло, щоб не викликати дискомфорту у працівника при роботі з екранними пристроями.

– вибираючи клавіатуру, слід віддавати перевагу такій клавіатурі, яка відкидається і є автономною (відокремленої від екрана), щоб працівник міг вибрати зручну робочу позу і уникнути втоми рук (кисті та верхньої частини руки).

– поверхня клавіатури має бути матовою, щоб уникнути відображення. Розташування клавіш і самі клавіші повинні полегшувати роботу з клавіатурою. Позначення клавіш має бути досить контрастним і розбірливим.

– при розробці, виборі, замовленні та модифікації програмного забезпечення, а також при розробці завдань, які передбачають використання обладнання з екранними пристроями, роботодавець повинен керуватися таким програмним забезпеченням, яке відповідає завданням і є простим у використанні, а де необхідно адаптованим до рівня знань і досвіду працівника.

4.4 Аналіз освітленості робочого приміщення

Доволі важливим фактором впливу на здоров'я та працездатність людини є освітленість приміщення. Для визначення того, чи є достатнім забезпечення природного освітлення в приміщенні порівнюють значення фактичної загальної площі вікон в приміщенні до мінімальної [23][24]. Мінімальна площа вікон в приміщенні визначається за формулою 4.1:

$$S_в = \frac{e_N \cdot k_з \cdot \eta_в \cdot S_{підл} \cdot k_{буд}}{\tau_{заг} \cdot r_1 \cdot 100} \qquad (4.1)$$

де e_N - нормований коефіцієнт природного освітлення для розглянутих умов праці,

$\kappa_з$ - коефіцієнт запасу, що приймається при розрахунках природного освітлення,

$\eta_в$ - світлова характеристика вікна,

$S_{підл}$- площа підлоги виробничого приміщення ,

$\tau_{заг}$ - загальний коефіцієнт пропускання світла,

r_1 - коефіцієнт, що враховує підвищення коефіцієнта природного освітлення за рахунок світла, яке відбивається від внутрішніх поверхонь приміщення,

$K_{буд}$ - коефіцієнт, що враховує вплив протилежної будівлі на освітленість у виробничому приміщенні.

Вхідні дані для розрахунку мінімальної площі вікна для забезпечення природної освітленості наведені в таблиці 4.1.

Таблиця 4.3 – Вхідні дані для розрахунку необхідної площі вікна.

№	**Параметр**	**Позначення**	**Розмірність**	**Значення**
1	Нормований коефіцієнт природного освітлення	e_N	%	0,9
2	Коефіцієнт запасу	$\kappa_з$	1	1,2
3	Світлова характеристика вікна	$\eta_в$	1	9.87
4	Площа підлоги	$S_{підл}$	м^2	8.23
5	Загальний коефіцієнт пропускання світла	$\tau_{заг}$	1	2,4
6	Коефіцієнт, що враховує підвищення коефіцієнта природного освітлення	r_1	1	0,621
7	Коефіцієнт, що враховує вплив протилежної будівлі на освітленість	$K_{буд}$	м	1.01

Визначення площі вікна, яке наявне в приміщенні за формулою 4.2:

$$S_{нв} = h * l = 1.6 * 1 = 1.6(м^2) \quad (4.2)$$

Наступним кроком необхідно обчислити значення площі вікна для забезпечення природної освітленості приміщення за формулою 1.1:

$$S_в = \frac{0.9 \cdot 1{,}2 \cdot 9{,}87 \cdot 8{,}32 \cdot 1.01}{2{,}4 \cdot 0{,}621 \cdot 100} = 0{,}6(\text{м}^2)$$

Згідно попередніх обчислень, можна зробити висновки, що того вікна, яке наявне в приміщені більш ніж достатньо для забезпечення природного освітлення в робочому приміщенні для робіт за комп'ютером.

4.5 Виробничий шум та вібрація

Відомо, що шум несприятливо діє на слуховий аналізатор та інші органи та системи організму людини. Визначальне значення щодо такої дії має інтенсивність шуму, його частотний склад, тривалість щоденного впливу, індивідуальні особливості людини, а також специфіка виробничої діяльності.

Рівні звукового тиску в октавних смугах частот, рівні звуку та еквівалентні рівні звуку на робочих місцях, обладнаних ВДТ і ПК визначені ДСанПіН 3.3.2-007-98 [21].

Основними заходами та засобами боротьби з шумом є:

− зниження рівнів шуму в джерелі його утворення (застосовується, як правило, в процесі проектування);

− використання звукопоглинаючих та звукоізолюючих засобів;

− раціональне планування виробничих приміщень та робочих місць.

На комп'ютеризованих робочих місцях основними джерелами шуму є вентилятори системного блоку, накопичувані, принтери ударної дії. Для зниження рівнів шуму на робочих місцях рекомендується розмістити друкувальні пристрої ударної дії (матричні, шрифтові принтери тощо) в іншому приміщенні, або огородити їх звукоізолюючими екранами.

Оскільки зовнішні шуми (вулиця, суміжні приміщення) також можуть негативно впливати на функціональний стан операторів ВДТ, то стіни приміщень, в яких розташовані комп'ютеризовані робочі місця бажано облицювати звукопоглинаючими матеріалами. Однак доцільність їх застосування повинна бути обґрунтована спеціальними інженерно-акустичними розрахунками. Звукопоглинаюче облицювання стін (іноді й стелі) необхідно здійснювати матеріалами, що мають максимальний коефіцієнт звукопоглинання в межах частот 31,5-8000 Гц і дозволені для оздоблення приміщень органами державного санітарно-епідеміологічного нагляду.

Для зниження вібрації обладнання, пристрої, пристосування необхідно встановлювати на спеціальні амортизуючи прокладки, передбачені нормативними документами.

ВИСНОВКИ

Під час написання бакалаврської кваліфікаційної роботи проведено аналіз використання згорткових нейронних мереж в картографії. Виявлено поширення та популярність нейронних мереж, як одного із методів машинного навчання. Крім того, в розділі проведено аналіз існуючих аналогів та останніх публікацій подібних систем, виділено їх сильні та слабкі сторони для уникнення помилок при розробці згоркової нейронної мережі для формування контуру будівлі. Проведено загальний огляд на сучасні підходи реалізації згоркових нейронних мереж, визначено, які технології користуються високою популярністю серед розробників. Досліджено принципи роботи згорткових нейронних мереж на прикладі. Проведено аналіз різних датасетів для тренування та виявлено кращий. Досліджено такі архітектури згорткових мереж як U-Net та Mask R-CNN. На мові програмування Python створено та налаштовано згоркові нейронні мережі на обраних архітектурах для формування контуру будівель.

Отже згідно з поставленим завданням та метою виконано усі пункти. Створено декілька реалізацій згоркових нейронних мереж на різних архітектурах та налаштовано на роботу з обраним дата сетом. Кожен створений застосунок має точність 89%, тому їх можливо використовувати з повною довірою до результатів.

Створені системи можливо інтегрувати в будь-який робочій термінал. А завдяки цьому можливо надати експертам програмне забезпечення для формування контуру будівель. Це допоможе зекономити багато часу да досвіду на ручному анотуванні, тому як була розроблена ефективну та надійну схему автоматичного вилучення полігонів контурів будівель із зображень дистанційного зондування.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Обчислювальна геометрія – *Wikipedia: веб-сайт.* URL: https://en.wikipedia.org/wiki/Computational_geometry (дата звернення: 05.05.2022).
2. Типізація – *Wikipedia: веб-сайт.* URL: https://en.wikipedia.org/wiki/Systems_thinking (дата звернення: 05.05.2022).
3. Мустьер – *Wikipedia: веб-сайт.* URL: https://en.wikipedia.org/wiki/Mousterian (дата звернення: 05.05.2022).
4. Convolutional neural network – *Wikipedia: веб-сайт.* URL: https://en.wikipedia.org/wiki/Convolutional_neural_network (дата звернення: 05.05.2022).
5. Turker MBC – *веб-сайт.* URL: https://www.researchgate.net/scientific-contributions/Mustafa-Turker-2120549925 (дата звернення: 05.05.2022).
6. U-Net Architecture – *веб-сайт.* URL: https://lmb.informatik.uni-freiburg.de/people/ronneber/u-net/ (дата звернення: 06.05.2022).
7. Seg-Net Architecture – *веб-сайт.* URL: https://arxiv.org/abs/1511.00561 (дата звернення: 06.05.2022).
8. FC-DenseNet Architecture – *веб-сайт.* URL: https://www.researchgate.net/figure/FC-DenseNet-architecture_fig1_337761543 (дата звернення: 06.05.2022).
9. Inria Aerial Image*: веб-сайт.* URL: https://project.inria.fr/aerialimagelabeling/ (дата звернення: 06.05.2022).
10. Massachusetts Aerial Image – *веб-сайт.* URL: https://www.mass.gov/info-details/massgis-data-2021-aerial-imagery (дата звернення: 05.05.2022).
11. Using Img2AFM in footprint generating – *веб-сайт.* URL: https://www.researchgate.net/publication/360667786_Semi-Supervised_Building_Footprint_Generation_with_Feature_and_Output_Consistency_Training (дата звернення: 06.05.2022).

12. Using AFM2Mask in footprint generating – *веб-сайт.* URL: https://www.researchgate.net/publication/354639709_Building_Footprint_Generation_Through_Convolutional_Neural_Networks_With_Attraction_Field_Representation (дата звернення: 06.05.2022).
13. ISPRS Dataset – *веб-сайт.* URL: https://www.isprs.org/ (дата звернення: 06.05.2022).
14. RCA-Net explained – *веб-сайт.* URL: https://www.ibm.com/docs/en/networkmanager/4.2.0?topic=layer-about-root-cause-analysis-event-enrichment (дата звернення: 06.05.2022).
15. Про затвердження Вимог щодо безпеки та захисту здоров'я працівників під час роботи з екранними пристроями – веб-сайт. URL: http://zakon3.rada.gov.ua/laws/show/z0508-18 (дата звернення: 05.05.2022).
16. Про затвердження Правил охорони праці під час експлуатації електронно-обчислювальних машин – веб-сайт. URL: http://zakon3.rada.gov.ua/laws/show/z0382-99 (дата звернення: 05.05.2022).
17. Державні санітарні правила і норми роботи з ВДТ ЕОМ ДСанПІН 3.3.2.007-98 – веб-сайт. URL: http://mozdocs.kiev.ua/view.php?id=2445 (дата звернення: 05.05.2022).
18. СНиП II-4-79. Природне і штучне освітлення – веб-сайт. URL: https://dnaop.com/html/45036/doc- СНиП_II-4-79_ (дата звернення: 05.05.2022).
19. СНиП 2.09.04.-87. Адміністративні і побутові будівлі – веб-сайт. URL: https://dnaop.com/html/54074/doc -СНиП_2.09.04-87_ (дата звернення: 05.05.2022).
20. ДСН 3.3.6.042-99. Санітарні норми мікроклімату виробничих приміщень – веб-сайт. URL: https://dnaop.com/html/34094/doc -ДСН_3.3.6.042-99 (дата звернення: 05.05.2022).
21. ГОСТ 12.1.005-88.ССБП – веб-сайт. URL: http://docs.cntd.ru/document/1200003608 (дата звернення: 05.05.2022).

22. СН 4088-86. Санітарні норми мікроклімату виробничих приміщень – веб-сайт. URL: http://docs.cntd.ru/document/901710059 (дата звернення: 05.05.2022).
23. Санітарно – гігієнічним нормам №2152-80 – веб-сайт. URL: https://dnaop.com/html/2296/doc -ГН_2152-80 (дата звернення: 05.05.2022).
24. СН 4557-88. Санітарні норми ультрафіолетового випромінювання – веб-сайт. URL: https://dnaop.com/html/2299/doc -СН_4557-88 (дата звернення: 05.05.2022).
25. Про затвердження санітарного та епідемічного благополуччя населення – веб-сайт. URL: http://zakon2.rada.gov.ua/laws/show/4004-12 (дата звернення: 05.05.2022).

ДОДАТОК А

Файл класу з налаштуванням Mask R-CNN

```
import os
import sys
import json
import datetime
import numpy as np
import skimage.draw

# Root directory of the project
ROOT_DIR = os.path.abspath("../../")

# Import Mask RCNN
sys.path.append(ROOT_DIR)  # To find local version of the library
from mrcnn.config import Config
from mrcnn import model as modellib, utils

# Path to trained weights file
COCO_WEIGHTS_PATH = os.path.join(ROOT_DIR, "mask_rcnn_coco.h5")

# Directory to save logs and model checkpoints, if not provided
# through the command line argument --logs
DEFAULT_LOGS_DIR = os.path.join(ROOT_DIR, "logs")

############################################################
#  Configurations
############################################################


class CustomConfig(Config):
    """Configuration for training on the custom dataset.
    Derives from the base Config class and overrides some values.
    """
    # Give the configuration a recognizable name
    NAME = "custom"
    BACKBONE_STRIDES = [4, 8, 16, 32, 64]
    IMAGE_MIN_DIM = 128
    IMAGE_MAX_DIM = 128
    DETECTION_MAX_INSTANCES = 100
    BBOX_STD_DEV = [0.1, 0.1, 0.2, 0.2]
    LEARNING_MOMENTUM = 0.9
    LEARNING_RATE = 0.001
    MASK_POOL_SIZE = 14
    MASK_SHAPE = [28,28]
    POOL_SIZE = 7
    VALIDATION_STEPS = 5

    # We use a GPU with 12GB memory, which can fit two images.
    # Adjust down if you use a smaller GPU.
    IMAGES_PER_GPU = 1
    # Number of classes (including background)
    NUM_CLASSES = 1 + 1  # Background + number of classes (Here, 2)

    # Number of training steps per epoch
    STEPS_PER_EPOCH = 8
    ROI_POSITIVE_RATIO = 0.33
    RPN_ANCHOR_RATIOS = [0.5, 1, 2]
    RPN_ANCHOR_SCALES = [8,16,32,64,128]
    RPN_ANCHOR_STRIDE = 1
```

```
    RPN_NMS_THRESHOLD = 0.7
    RPN_BBOX_STD_DEV = [0.1, 0.1, 0.2, 0.2]
    RPN_TRAIN_ANCHORS_PER_IMAGE = 128
    # Skip detections with < 90% confidence
    USE_MINI_MASK = True
    USE_RPN_ROIS = True
    DETECTION_MIN_CONFIDENCE = 0.7
    WEIGHT_DECAY = 0.0001

############################################################
#  Dataset
############################################################

class CustomDataset(utils.Dataset):

    def load_custom(self, dataset_dir, subset):
        """Load a subset of the custom dataset.
        dataset_dir: Root directory of the dataset.
        subset: Subset to load: train or val
        """
        # Add classes according to the numbe of classes required to detect
        self.add_class("custom", 1, "building")

        # Train or validation dataset?
        assert subset in ["train", "val"]
        dataset_dir = os.path.join(dataset_dir, subset)

        # Load annotations
        # VGG Image Annotator (up to version 1.6) saves each image in the form:
        # { 'filename': '28503151_5b5b7ec140_b.jpg',
        #   'regions': {
        #       '0': {
        #           'region_attributes': {},
        #           'shape_attributes': {
        #               'all_points_x': [...],
        #               'all_points_y': [...],
        #               'name': 'polygon'}},
        #       ... more regions ...
        #   },
        #   'size': 100202
        # }
        # We mostly care about the x and y coordinates of each region
        # Note: In VIA 2.0, regions was changed from a dict to a list.
        annotations = json.load(open(os.path.join(dataset_dir,
"via_region_data.json")))
        annotations = list(annotations.values())  # don't need the dict keys

        # The VIA tool saves images in the JSON even if they don't have any
        # annotations. Skip unannotated images.
        annotations = [a for a in annotations if a['regions']]

        # Add images
        for a in annotations:
            # Get the x, y coordinaets of points of the polygons that make up
            # the outline of each object instance. These are stores in the
            # shape_attributes (see json format above)
            # The if condition is needed to support VIA versions 1.x and 2.x.
            polygons = [r['shape_attributes'] for r in a['regions'].values()]
            #labelling each class in the given image to a number

            custom = [s['region_attributes'] for s in a['regions'].values()]
```

```
            num_ids=
            #Add the classes according to the requirement
            for n in custom:
                try:
                    if n['label']=='building':
                        num_ids.append(1)
                except:
                    pass

            # load_mask() needs the image size to convert polygons to masks.
            # Unfortunately, VIA doesn't include it in JSON, so we must read
            # the image. This is only managable since the dataset is tiny.
            image_path = os.path.join(dataset_dir, a['filename'])
            image = skimage.io.imread(image_path)
            height, width = image.shape[:2]

            self.add_image(
                "custom",
                image_id=a['filename'],  # use file name as a unique image id
                path=image_path,
                width=width, height=height,
                polygons=polygons,
                num_ids=num_ids)

    def load_mask(self, image_id):
        """Generate instance masks for an image.
       Returns:
        masks: A bool array of shape [height, width, instance count] with
            one mask per instance.
        class_ids: a 1D array of class IDs of the instance masks.
        """
        # If not a custom dataset image, delegate to parent class.
        image_info = self.image_info[image_id]
        if image_info["source"] != "custom":
            return super(self.__class__, self).load_mask(image_id)
        num_ids = image_info['num_ids']
        #print("Here is the numID",num_ids)

        # Convert polygons to a bitmap mask of shape
        # [height, width, instance_count]
        info = self.image_info[image_id]
        mask = np.zeros([info["height"], info["width"], len(info["polygons"])],
                        dtype=np.uint8)
        for i, p in enumerate(info["polygons"]):
            # Get indexes of pixels inside the polygon and set them to 1
            rr, cc = skimage.draw.polygon(p['all_points_y'], p['all_points_x'])
            mask[rr, cc, i] = 1

        # Return mask, and array of class IDs of each instance. Since we have
        # one class ID only, we return an array of 1s
        num_ids = np.array(num_ids, dtype=np.int32)
        return mask, num_ids#.astype(np.bool), np.ones([mask.shape[-1]],
dtype=np.int32),

    def image_reference(self, image_id):
        """Return the path of the image."""
        info = self.image_info[image_id]
        if info["source"] == "custom":
            return info["path"]
        else:
            super(self.__class__, self).image_reference(image_id)
```

```
def train(model):
    """Train the model."""
    # Training dataset.
    dataset_train = CustomDataset()
    dataset_train.load_custom(args.dataset, "train")
    dataset_train.prepare()

    # Validation dataset
    dataset_val = CustomDataset()
    dataset_val.load_custom(args.dataset, "val")
    dataset_val.prepare()

    # *** This training schedule is an example. Update to your needs ***
    # Since we're using a very small dataset, and starting from
    # COCO trained weights, we don't need to train too long. Also,
    # no need to train all layers, just the heads should do it.
    print("Training network heads")
    model.train(dataset_train, dataset_val,
                learning_rate=config.LEARNING_RATE,
                epochs=30,
                layers='heads')


def color_splash(image, mask):
    """Apply color splash effect.
    image: RGB image [height, width, 3]
    mask: instance segmentation mask [height, width, instance count]

    Returns result image.
    """
    # Make a grayscale copy of the image. The grayscale copy still
    # has 3 RGB channels, though.
    gray = skimage.color.gray2rgb(skimage.color.rgb2gray(image)) * 255
    # Copy color pixels from the original color image where mask is set
    if mask.shape[-1] > 0:
        # We're treating all instances as one, so collapse the mask into one layer
        mask = (np.sum(mask, -1, keepdims=True) >= 1)
        splash = np.where(mask, image, gray).astype(np.uint8)
    else:
        splash = gray.astype(np.uint8)
    return splash


def detect_and_color_splash(model, image_path=None, video_path=None):
    assert image_path or video_path

    # Image or video?
    if image_path:
        # Run model detection and generate the color splash effect
        print("Running on {}".format(args.image))
        # Read image
        image = skimage.io.imread(args.image)
        # Detect objects
        r = model.detect([image], verbose=1)[0]
        # Color splash
        splash = color_splash(image, r['masks'])
        # Save output
        file_name = "splash_{:%Y%m%dT%H%M%S}.png".format(datetime.datetime.now())
        skimage.io.imsave(file_name, splash)
    elif video_path:
        import cv2
```

```
        # Video capture
        vcapture = cv2.VideoCapture(video_path)
        width = int(vcapture.get(cv2.CAP_PROP_FRAME_WIDTH))
        height = int(vcapture.get(cv2.CAP_PROP_FRAME_HEIGHT))
        fps = vcapture.get(cv2.CAP_PROP_FPS)

        # Define codec and create video writer
        file_name = "splash_{:%Y%m%dT%H%M%S}.avi".format(datetime.datetime.now())
        vwriter = cv2.VideoWriter(file_name,
                                  cv2.VideoWriter_fourcc(*'MJPG'),
                                  fps, (width, height))

        count = 0
        success = True
        while success:
            print("frame: ", count)
            # Read next image
            success, image = vcapture.read()
            if success:
                # OpenCV returns images as BGR, convert to RGB
                image = image[..., ::-1]
                # Detect objects
                r = model.detect([image], verbose=0)[0]
                # Color splash
                splash = color_splash(image, r['masks'])
                # RGB -> BGR to save image to video
                splash = splash[..., ::-1]
                # Add image to video writer
                vwriter.write(splash)
                count += 1
        vwriter.release()
    print("Saved to ", file_name)


############################################################
#  Training
############################################################

if __name__ == '__main__':
    import argparse

    # Parse command line arguments
    parser = argparse.ArgumentParser(
        description='Train Mask R-CNN to detect custom objects.')
    parser.add_argument("command",
                        metavar="<command>",
                        help="'train' or 'splash'")
    parser.add_argument('--dataset', required=False,
                        metavar="/path/to/custom/dataset/",
                        help='Directory of the custom dataset')
    parser.add_argument('--weights', required=True,
                        metavar="/path/to/weights.h5",
                        help="Path to weights .h5 file or 'coco'")
    parser.add_argument('--logs', required=False,
                        default=DEFAULT_LOGS_DIR,
                        metavar="/path/to/logs/",
                        help='Logs and checkpoints directory (default=logs/)')
    parser.add_argument('--image', required=False,
                        metavar="path or URL to image",
                        help='Image to apply the color splash effect on')
    parser.add_argument('--video', required=False,
                        metavar="path or URL to video",
```

```
                        help='Video to apply the color splash effect on')
    args = parser.parse_args()

    # Validate arguments
    if args.command == "train":
        assert args.dataset, "Argument --dataset is required for training"
    elif args.command == "splash":
        assert args.image or args.video,\
               "Provide --image or --video to apply color splash"

    print("Weights: ", args.weights)
    print("Dataset: ", args.dataset)
    print("Logs: ", args.logs)

    # Configurations
    if args.command == "train":
        config = CustomConfig()
    else:
        class InferenceConfig(CustomConfig):
            # Set batch size to 1 since we'll be running inference on
            # one image at a time. Batch size = GPU COUNT * IMAGES PER GPU
            GPU_COUNT = 1
            IMAGES_PER_GPU = 1
        config = InferenceConfig()
    config.display()

    # Create model
    if args.command == "train":
        model = modellib.MaskRCNN(mode="training", config=config,
                                  model_dir=args.logs)
    else:
        model = modellib.MaskRCNN(mode="inference", config=config,
                                  model_dir=args.logs)

    # Select weights file to load
    if args.weights.lower() == "new":
        print("weight path entered")
        print(NEW_WEIGHTS_PATH)
        weights_path = NEW_WEIGHTS_PATH
    if args.weights.lower() == "coco":
        weights_path = COCO_WEIGHTS_PATH
        # Download weights file
        if not os.path.exists(weights_path):
            utils.download_trained_weights(weights_path)
    elif args.weights.lower() == "last":
        # Find last trained weights
        weights_path = model.find_last()
    elif args.weights.lower() == "imagenet":
        # Start from ImageNet trained weights
        weights_path = model.get_imagenet_weights()
    else:
        weights_path = args.weights

    # Load weights
    print("Loading weights ", weights_path)
    if args.weights.lower() == "coco":
        # Exclude the last layers because they require a matching
        # number of classes
        model.load_weights(weights_path, by_name=True, exclude=[
"mrcnn_class_logits", "mrcnn_bbox_fc", "mrcnn_bbox", "mrcnn_mask",
"mrcnn_class_bn1", "mrcnn_class_bn2", "mrcnn_class_conv2", "mrcnn_class_conv1"])
    else:
```

```
    model.load_weights(weights_path, by_name=True)

# Train or evaluate
if args.command == "train":
    train(model)
elif args.command == "splash":
    detect_and_color_splash(model, image_path=args.image,
                            video_path=args.video)
else:
    print("'{}' is not recognized. "
          "Use 'train' or 'splash'".format(args.command))
```

ДОДАТОК Б

Файл запуску застосунку на архітектурі Mask R-CNN

```
import os
import sys
import random
import math

import config as config
import numpy as np
import skimage.io
import matplotlib
import matplotlib.pyplot as plt

# Root directory of the project
from samples.custom import custom

ROOT_DIR = "C:/progmarimovaniye/4-2/dplm/Custom_MaskRCNN"

# Import Mask RCNN
sys.path.append(ROOT_DIR)  # To find local version of the library
from mrcnn import utils

import mrcnn.model as modellib
from mrcnn import visualize

# Import COCO config
sys.path.append(os.path.join(ROOT_DIR, "samples/coco/"))  # To find local version

# Directory to save logs and trained model
MODEL_DIR = os.path.join(ROOT_DIR, "logs")
CUSTOM_MODEL_PATH = "C:/progmarimovaniye/4-
2/dplm/Custom_MaskRCNN/logs/custom/mask_rcnn_custom_0030.h5"
# Local path to trained weights file
COCO_MODEL_PATH = os.path.join(ROOT_DIR, "mask_rcnn_coco.h5")
# Download COCO trained weights from Releases if needed
if not os.path.exists(COCO_MODEL_PATH):
    utils.download_trained_weights(COCO_MODEL_PATH)

# Directory of images to run detection on
IMAGE_DIR = os.path.join(ROOT_DIR, "images")


class InferenceConfig(custom.CustomConfig):
    # Set batch size to 1 since we'll be running inference on
    # one image at a time. Batch size = GPU_COUNT * IMAGES_PER_GPU
    GPU_COUNT = 1
    IMAGES_PER_GPU = 1
    IMAGE_MAX_DIM = 256
    IMAGE_MIN_DIM = 256
    DETECTION_MIN_CONFIDENCE = 0.8


config = InferenceConfig()
config.display()
```

ДОДАТОК В

Файл класу з налаштуванням U-Net

```
import numpy as np
import os
import skimage.io as io
import skimage.transform as trans
import numpy as np
from keras.models import *
from keras.layers import *
from keras.optimizers import *
from keras.callbacks import ModelCheckpoint, LearningRateScheduler
from keras import backend as keras


def unet(pretrained_weights = None,input_size = (256,256,1)):
    inputs = Input(input_size)
    conv1 = Conv2D(64, 3, activation = 'relu', padding = 'same',
kernel_initializer = 'he_normal')(inputs)
    conv1 = Conv2D(64, 3, activation = 'relu', padding = 'same',
kernel_initializer = 'he_normal')(conv1)
    pool1 = MaxPooling2D(pool_size=(2, 2))(conv1)
    conv2 = Conv2D(128, 3, activation = 'relu', padding = 'same',
kernel_initializer = 'he_normal')(pool1)
    conv2 = Conv2D(128, 3, activation = 'relu', padding = 'same',
kernel_initializer = 'he_normal')(conv2)
    pool2 = MaxPooling2D(pool_size=(2, 2))(conv2)
    conv3 = Conv2D(256, 3, activation = 'relu', padding = 'same',
kernel_initializer = 'he_normal')(pool2)
    conv3 = Conv2D(256, 3, activation = 'relu', padding = 'same',
kernel_initializer = 'he_normal')(conv3)
    pool3 = MaxPooling2D(pool_size=(2, 2))(conv3)
    conv4 = Conv2D(512, 3, activation = 'relu', padding = 'same',
kernel_initializer = 'he_normal')(pool3)
    conv4 = Conv2D(512, 3, activation = 'relu', padding = 'same',
kernel_initializer = 'he_normal')(conv4)
    drop4 = Dropout(0.5)(conv4)
    pool4 = MaxPooling2D(pool_size=(2, 2))(drop4)

    conv5 = Conv2D(1024, 3, activation = 'relu', padding = 'same',
kernel_initializer = 'he_normal')(pool4)
    conv5 = Conv2D(1024, 3, activation = 'relu', padding = 'same',
kernel_initializer = 'he_normal')(conv5)
    drop5 = Dropout(0.5)(conv5)

    up6 = Conv2D(512, 2, activation = 'relu', padding = 'same', kernel_initializer
= 'he_normal')(UpSampling2D(size = (2,2))(drop5))
    merge6 = concatenate([drop4,up6], axis = 3)
    conv6 = Conv2D(512, 3, activation = 'relu', padding = 'same',
kernel_initializer = 'he_normal')(merge6)
    conv6 = Conv2D(512, 3, activation = 'relu', padding = 'same',
kernel_initializer = 'he_normal')(conv6)

    up7 = Conv2D(256, 2, activation = 'relu', padding = 'same', kernel_initializer
= 'he_normal')(UpSampling2D(size = (2,2))(conv6))
    merge7 = concatenate([conv3,up7], axis = 3)
    conv7 = Conv2D(256, 3, activation = 'relu', padding = 'same',
kernel_initializer = 'he_normal')(merge7)
    conv7 = Conv2D(256, 3, activation = 'relu', padding = 'same',
kernel_initializer = 'he_normal')(conv7)
```

```
    up8 = Conv2D(128, 2, activation = 'relu', padding = 'same', kernel_initializer
= 'he_normal')(UpSampling2D(size = (2,2))(conv7))
    merge8 = concatenate([conv2,up8], axis = 3)
    conv8 = Conv2D(128, 3, activation = 'relu', padding = 'same',
kernel_initializer = 'he_normal')(merge8)
    conv8 = Conv2D(128, 3, activation = 'relu', padding = 'same',
kernel_initializer = 'he_normal')(conv8)

    up9 = Conv2D(64, 2, activation = 'relu', padding = 'same', kernel_initializer
= 'he_normal')(UpSampling2D(size = (2,2))(conv8))
    merge9 = concatenate([conv1,up9], axis = 3)
    conv9 = Conv2D(64, 3, activation = 'relu', padding = 'same',
kernel_initializer = 'he_normal')(merge9)
    conv9 = Conv2D(64, 3, activation = 'relu', padding = 'same',
kernel_initializer = 'he_normal')(conv9)
    conv9 = Conv2D(2, 3, activation = 'relu', padding = 'same', kernel_initializer
= 'he_normal')(conv9)
    conv10 = Conv2D(1, 1, activation = 'sigmoid')(conv9)

    model = Model(input = inputs, output = conv10)

    model.compile(optimizer = Adam(lr = 1e-4), loss = 'binary_crossentropy',
metrics = ['accuracy'])

    #model.summary()

    if(pretrained_weights):
       model.load_weights(pretrained_weights)

    return model
```

ДОДАТОК Г

Файл запуску застосунку на архітектурі U-Net

```
from model import *
from data import *

#os.environ["CUDA_VISIBLE_DEVICES"] = "0"


data_gen_args = dict(rotation_range=0.2,
                     width_shift_range=0.05,
                     height_shift_range=0.05,
                     shear_range=0.05,
                     zoom_range=0.05,
                     horizontal_flip=True,
                     fill_mode='nearest')
myGene =
trainGenerator(2,'data/membrane/train','image','label',data_gen_args,save_to_dir =
None)

model = unet()
model_checkpoint = ModelCheckpoint('unet_membrane.hdf5', monitor='loss',verbose=1,
save_best_only=True)
model.fit_generator(myGene,steps_per_epoch=300,epochs=1,callbacks=[model_checkpoin
t])

testGene = testGenerator("data/membrane/test")
results = model.predict_generator(testGene,30,verbose=1)
saveResult("data/membrane/test",results)
```