Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет Інформаційно-аналітичних технологій та менеджменту
(повна назва)
Кафедра Інформатики
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти перший (бакалаврський)

СИСТЕМА СПОСТЕРЕЖЕННЯ ЗА ДОВКОЛИШНІМ СЕРЕДОВИЩЕМ НА ОСНОВІ АНАЛІЗУ НАБОРІВ ЗОБРАЖЕНЬ
(тема)

Виконав:
студент 4 курсу, групи ІТІНФ-18-1

Лопатінський А.А
(прізвище, ініціали)

Спеціальності 122 Комп'ютерні науки
(код і повна назва спеціальності)

Тип програми освітньо-професійна

Освітня програма Інформатика
(повна назва освітньої програми)

Керівник проф. Кіріченко Л.О.
(посада, прізвище, ініціали)

Допускається до захисту

Зав. кафедри _____ Кобилін О.А.

(підпис) (прізвище, ініціали)

2022 р.

Харківський національний університет радіоелектроніки

Факультет Інформаційно-аналітичних технологій та менеджменту
(повна назва)

Кафедра Інформатики
(повна назва)

Рівень вищої освіти перший (бакалаврський)

Спеціальність 122 Комп'ютерні науки
(код і повна назва)

Тип програми освітньо-професійна

Освітня програма Інформатика
(повна назва освітньої програми)

ЗАТВЕРДЖУЮ:

Зав. кафедри _____
(підпис)
«____»_____ 2022 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

студентові Лопатінському Андрію Анатолійовичу
(прізвище, ім'я, по батькові)

1. Тема роботи Система спостереження за довколишнім середовищем на основі аналізу наборів зображень
затверджена наказом університету від 16 травня 2022 року № 541Ст

2. Термін подання студентом роботи до екзаменаційної комісії 28 травня 2022 р.

3. Вихідні дані до роботи Науково-методична та науково-технічна література, дані інтернет-мережі, відкрита програмна бібліотека для машинного навчання TensorFlow, фреймворк Keras, високорівнева мова програмування Python, web-фреймворк Django, бібліотеки NumPy та SciPy, бібліотека ImageAI для розпізнавання образів на зображенні, тренувальний набор зображень для навчання нейронної мережі, відкрита модель згорткової нейронної мережі, JavaScript-фреймворк Angular для створення веб-інтерфейсу користувача

4. Перелік питань, що потрібно опрацювати в роботі
1.Зробити попереднє завантаження фото на сервер.
2.Визначити набор даних для ідентифікації об'єкта.
3.Реалізувати ідентифікацію об'єкта на зображенні за набором даних
4.Реалізувати виявлення пошкоджень об'єкту за фото
5.Виділити об'єкт та його пошкодження
6.Реалізувати вивод результату користувачу

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (п.5 включається до завдання за рішенням випускової кафедри) візуалізація алгоритмів, приклади роботи алгоритмів, візуалізація згорткової нейронної мережі, програмний інтерфейс користувача

6. Консультанти розділів роботи (п.6 включається до завдання за наявності консультантів згідно з наказом, зазначеним у п.1)

Найменування розділу	Консультант (посада, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата
Консультант з дотримання діючих стандартів та норм	доцент Бєлова Н.В.		

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Терміни виконання етапів роботи	Примітка
1	Отримання завдання на кваліфікаційну роботу	12.04.2022	
2	Аналіз завдання, підбір літератури	13.04.22-15.04.22	
3	Аналіз літератури з досліджуваної проблеми	16.04.22-17.04.22	
4	Аналіз підходів до підходів розпізнавання пошкоджені на об'єкті	18.04.22-28.04.22	
5	Розробка методу розпізнавання пошкоджень об'єкта на зображенні	29.04.22-14.05.22	
6	Програмна реалізація	15.05.22-23.05.22	
7	Оформлення пояснювальної записки	24.05.22-26.05.22	
8	Перевірка на плагіат	06.06.22	
9	Рецензування	06.06.22	
10	Підготовка презентації та доповіді	06.06.22-14.06.22	
11	Занесення роботи в електронний архів	06.06.22	
12	Попередній захист кваліфікаційної роботи	14.06.22	

Дата видачі завдання 18 квітня 2022 р.

Студент _____
(підпис)

Керівник роботи _____ ст. викл. Кіріченко Л.О.
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ/ABSTRACT

Пояснювальна записка до кваліфікаційної роботи: 65 с., 33 рис., 26 джерело.

РОЗПІЗНАВАННЯ ОБРАЗІВ, ЗГОРТКОВА НЕЙРОННА МЕРЕЖА, СЕГМЕНТАЦІЯ ЗОБРАЖЕНЬ, ЗГОРТКОВИЙ ШАР, МАРКУВАННЯ ОБ'ЄКТІВ ЗОБРАЖЕННЯ, АРХІТЕКТУРА MOBILENET

Об'єктом роботи є зображення з об'єктами для моніторингу та набор тренувальних даних.

Метою роботи є розробка системи спостереження за навколишнім середовищем за аналізом набору зображень (потенційно знятих на камеру спостереження) на прикладі завантаженого фото та розпізнавання на ньому об`єктів та їх пошкоджень.

Система реалізована у вигляді web-додатку, який проводить ідентифікацію об'єктів на фото за допомогою побудованої моделі та набору тренувальних даних. Далі виконує розпізнавання дефектів з подальшим їх маркуванням. Інтерфейс програми дозволяє додати об'єкти для моніторингу та тренувальні набори даних.

Програма реалізує сучасні алгоритми комп'ютерного зору. Результат роботи програми − це марковане вхідне зображення з виявленими дефектами.

PATTERN RECOGNITION, CONVOLUTIONAL NEURAL NETWORK, IMAGE SEGMENTATION, CONVOLUTIONAL LAYER, MARKING IMAGE OBJECTS, MOBILENET ARCHITECTURE

The object of research is an image with objects to observe.

The aim of the research is to develop a system that would monitor surroundings by analyzing a group of pictures (potentially filmed on surveillance cameras) on the example of an uploaded image and recognition of objects and their defects.

The system is implemented as a web application that carries out identifications of objects on the picture by means of the developed model and the set of training data. The UI allows us to add other objects to be monitored and sets of training data.

The program implements modern computer vision algorithms. The result of program execution is the original image with defects highlighted.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

UI – інтерфейс користувача

ЗНМ − Згорткова нейронна мережа

ІНМ − Імпульсна нейронна мережа

SAAS – Software as a service

ВСТУП

Технологія комп’ютерного зору стає більш популярною та активно вдосконалюється останні 50 років. Перші спроби впровадження таких технологій беруть початок у середині 20-го сторіччя, але на тоді ще не було потужних систем, які б давали можливість ефективно інтегрувати комп’ютерний зір у маси. Проте зараз в нас є набагато більше ресурсів: більші обчислювальні потужності, оптимізоване ПО, а також людство досягло значного прогресу в розробці швидких алгоритмів аналізу зображень то штучного інтелекту.

Актуальність роботи полягає у необхідності використання технологій комп’ютерного зору у повсякденному ритмі. Аналіз з подальшою цифровою обробкою зображень використовується для поліпшення їх вигляду на мобільних пристроях. Однією з актуальних галузей є розпізнавання образів на зображенні: застосування комп’ютерного зору охоплює системи візуального контролю, системи безпеки, контроль промислових об’єктів, тощо.

Задачі розпізнавання об’єктів, а також їх пошкоджень, можуть розглядатися на прикладі аналізу різноманітних споруд. Наприклад, це можуть бути промислові об’єкти, пам’ятки архітектури, важкодоступні споруди що віддалені від наземних пунктів або які знаходяться в несприятливих для людини умовах. Також, має місце автоматизація нагляду за станом споруд та конструкцій.

Можна зробити висновок, що потрібна система, яка буде допомагати рядовому користувачу (SAAS), обробляти велику кількість даних та зосереджуватися на важливих деталях.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА СУЧАСНИХ РІШЕНЬ СЕГМЕНТАЦІЇ ЗОБРАЖЕНЬ

1.1 Поняття та застосування сегментації зображень

Зазвичай при аналізі зображень ставиться завдання розділити пікселі зображень на групи різних полів. Цей процес групування називається сегментацією. Сегментація використовується в багатьох сферах, наприклад у виробництві для виявлення дефектів збірки деталей, у медицині для первинної обробки зображень, а також для відображення областей супутникових зображень. Оскільки використання інтелектуальних технологій виробництва або дослідницьких технологій набирає обертів у всьому світі, методи аналізу зображень є надзвичайно важливими.

Тому виникає потреба у впровадженні більш складних методів поділу, які можна застосувати до багатьох конкретних завдань. В даний час у багатьох областях, де вивчається відеозображення (геологія, мікробіологія, астрономія тощо), використовуються методи сегментації, які підходять саме для цієї сфери діяльності. Хоча ці методи рідко важливі, оскільки мають схожий принцип дії. Але в більшості їх алгоритмів є відмінності. Розбиття зображення — це процес розбиття зображення на компоненти, які містять корисну інформацію: об’єкти та їх межі, додаткову інформацію, геометричні елементи тощо.

Розщеплення зображень є однією з найскладніших завдань обробки зображень. Як правило, алгоритми сегментації зображень базуються на одному з двох базових властивостей сигналу яскравості: розривності й однорідності. В першому випадку підхід складається з розбиття зображення, керуючись різкими змінами сигналу, такими, як перепади яскравості на зображенні. Друга категорія методів використовує розбиття зображень на

області, однорідні в розумінні завчасно обраних критеріїв. Прикладами таких методів можуть слугувати порогова обробка, вирощування областей, злиття і розбиття областей.

Сегментація поділяє зображення на ділянки або об’єкти, які його складають. Але кількість деталей, створених при такому згортанні, залежить від поточного завдання. Сегментація повинна бути повною, коли об’єкти ми хочемо відокремити. Завданням автоматизованого контролю складу електронних пристроїв є, наприклад, аналіз зображень виготовленої продукції з метою виявлення конкретних дефектів, таких як втрата компонентів або переривання контакту. Тому немає необхідності розділяти нижній рівень деталізації, необхідний для виявлення таких дефектів. Так само і в будь-якому іншому завданні необхідно обрати прийнятний метод сегментації, з конкретним рівнем деталізації, достатнім для того, щоб давати якомога більше інформації при мінімальних затратах часу й технічного устаткування.

Кінцевий успіх комп’ютерних процедур аналізу зображень багато в чому визначається точністю сегментації, тому значна увага повинна бути приділена підвищенню її надійності. В деяких ситуаціях, наприклад у завданнях технічного контролю, можливо хоча б деякою мірою керувати умовами зйомки. Досвідчений проєктувальник системи обробки зображень незмінне зверне увагу на подібні можливості.

В інших прикладних завданнях, наприклад в автономних системах наведення на ціль, розробник не може контролювати зовнішні умови, тому звичайний підхід полягає в тому, щоб зосередитися на виборі сенсора такого виду, який, скоріш за все, буде підсилювати сигнал від об’єктів, які нас цікавлять, і одночасно послаблювати вплив не суттєвих деталей зображення. Хорошим прикладом такої методики є інфрачервона фотографія, яка може бути використана у військових цілях для виявлення об’єктів з високим тепловим випромінюванням, наприклад військової техніки чи підрозділів.

1.2 Розгляд існуючих алгоритмів сегментації зображень

Існують десятки методів сегментації зображень. До найбільш популярних відносяться такі алгоритмічні підходи: Watershed, Floodfill, Grabcut, MeanShift.

1.2.1 Алгоритм сегментації за вододолами (WaterShed)

Алгоритм Watershed працює з зображенням як з функцією від двох змінних [4]:

$$f = I(x, y), \tag{1.1}$$

де *x,y* — координати пікселів.

Значенням функції може бути інтенсивність або модуль градієнта. Для максимальної контрастності можна видалити перехід із зображення (рис. 1.1). Якщо абсолютне значення градієнта розділити по осі OZ, то на ділянках з різницею інтенсивності утворюються хребти, а в однорідних областях (рис. 1.2).

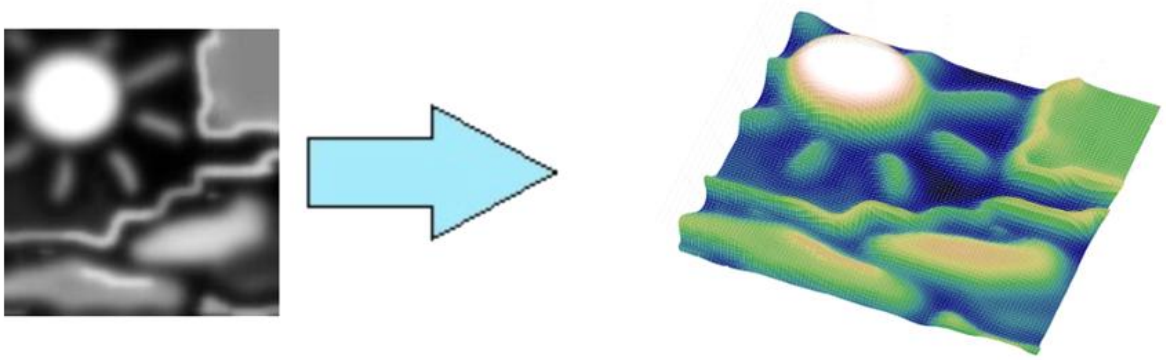

Рисунок 1.1 – Значення градієнта зображення по осі OZ

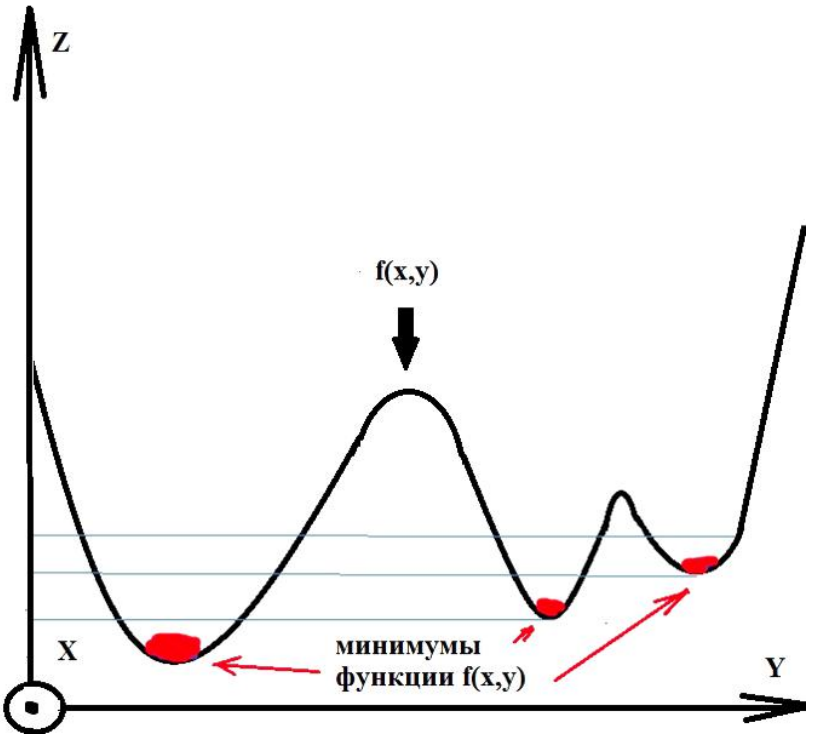

Рисунок 1.2 – Результат f за градієнтом зображення

Після знаходження мінімуму функції f відбувається процес наповнення водою, який починається з глобального мінімуму. Коли рівень води досягає значення найближчого локального мінімуму, її заповнюють водою (рис. 1.3).

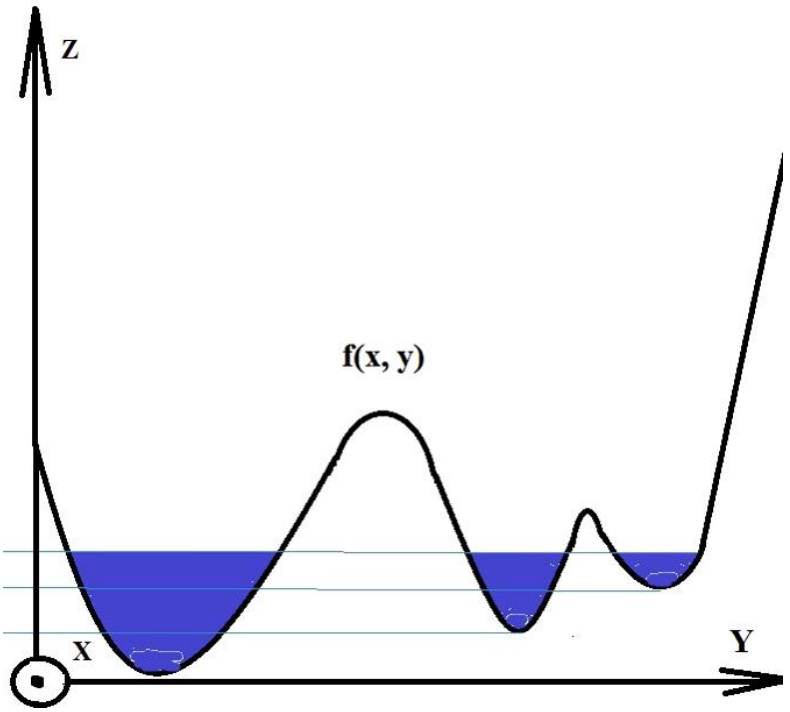

Рисунок 1.3 – Заповнення водою за мінімуми функції

Коли два регіони починають зливатися, будується стіна, щоб запобігти об’єднанню областей (рис. 1.4).

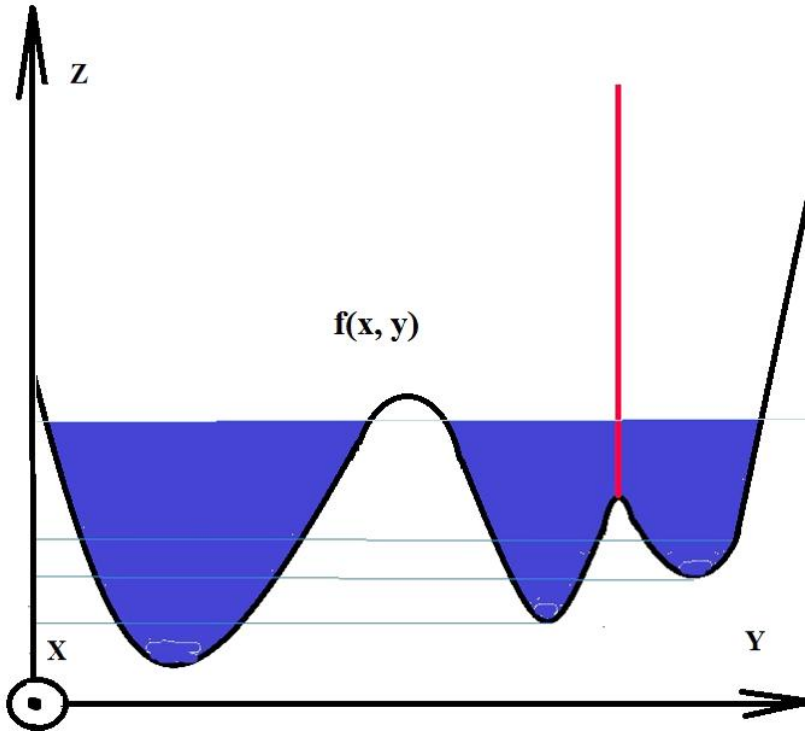

Рисунок 1.4 – Перегородка між регіонами, котрі зливаються

Вода продовжить підніматися до тих пір, поки регіони не відділятися тільки штучно збудованими перегородками. На рисунку 1.5 зображено поетапне заповнення водою на графіку.

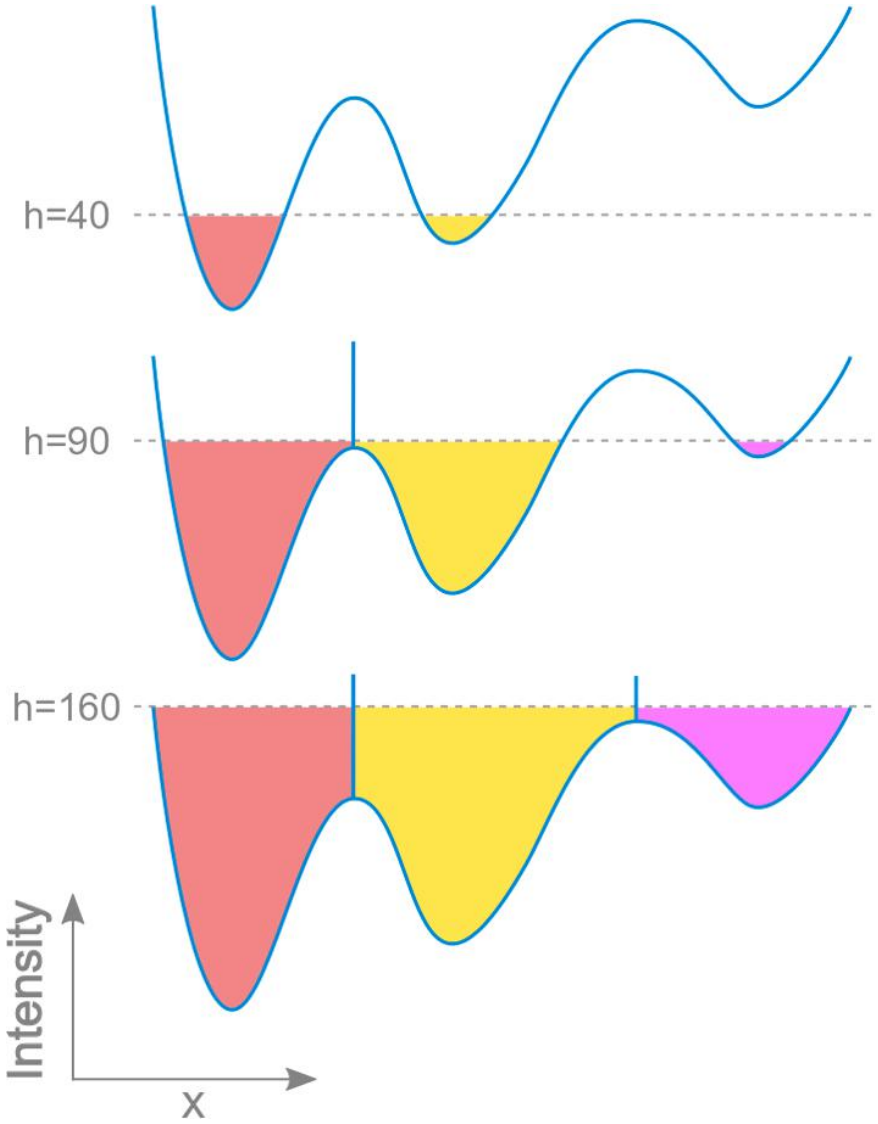

Рисунок 1.5 – Етапи алгоритму WaterShed

На рисунку 1.6 та наведений результати сегментації зображення за даним алгоритмом:

Рисунок 1.6 – Приклад роботи алгоритму WaterShed

Такий алгоритм може бути корисним, якщо на зображенні невелике число локальних мінімумів. Проте, в разі ж їх великої кількості виникає надлишкове розбиття на сегменти (рис 1.7).

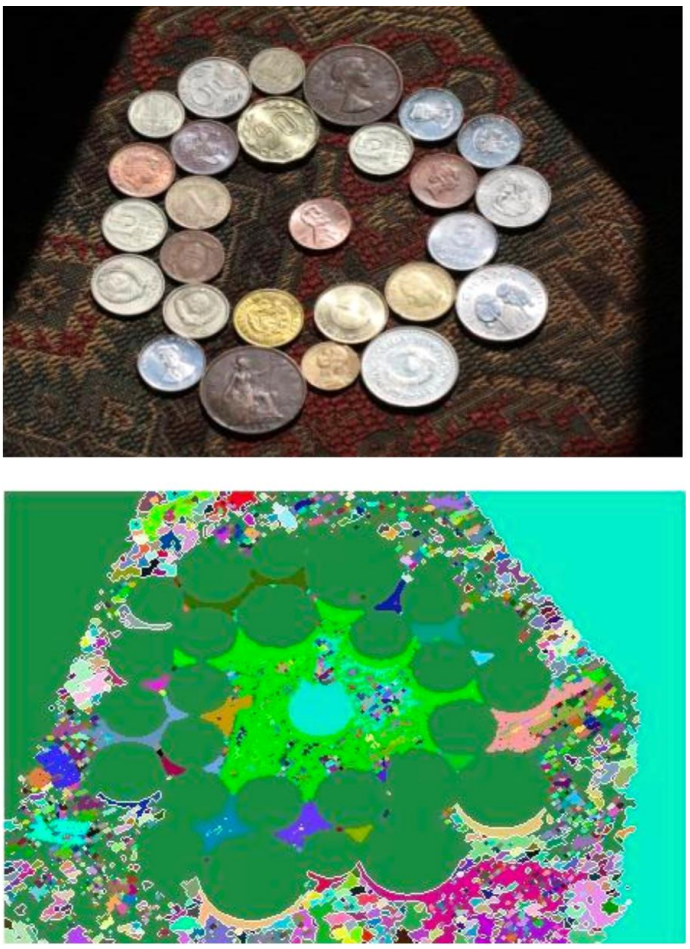

Рисунок 1.7 – Приклад надлишкової деталізації

Щоб зафіксувати зайві дрібні деталі, можна визначити ділянки, які потрібно прив’язати до найближчого низу. Розділ буде створено лише в тому випадку, якщо обидва регіони відповідають мітці, інакше ці функції збігатимуться. Цей метод усуває ефект надмірної фрагментації, але вимагає попередньої обробки зображення для позначення тегів, які можна інтерактивно позначити на зображенні (рисунок 1.8).

Рисунок 1.8 – Результат роботи алгоритму з попередньою обробкою зображення

1.2.2 Заливка (FloodFill)

Ви можете використовувати FloodFill, щоб позначити області, які мають однаковий колір. Для цього виберіть початковий піксель і встановіть інтервал зміни кольору сусідніх пікселів щодо вихідного. Інтервал може бути асиметричним. Алгоритм об’єднує пікселі в розділ (заповнює його кольором), якщо вони потрапляють у встановлений діапазон. Вихідним є частина, заповнена певним кольором, і її площа в пікселях. Цей алгоритм може бути корисним для заповнення ділянок зі слабкими колірними варіаціями одним і тим же фоном. Одним із способів використання FloodFill є виявлення країв пошкоджених об’єктів. Наприклад, якщо алгоритм заповнює суміжні області, заповнюючи однорідні області певним кольором, то цілісність кордону між цими областями порушується. На рисунку 1.9 видно, що цілісність меж затоплених територій збережена:

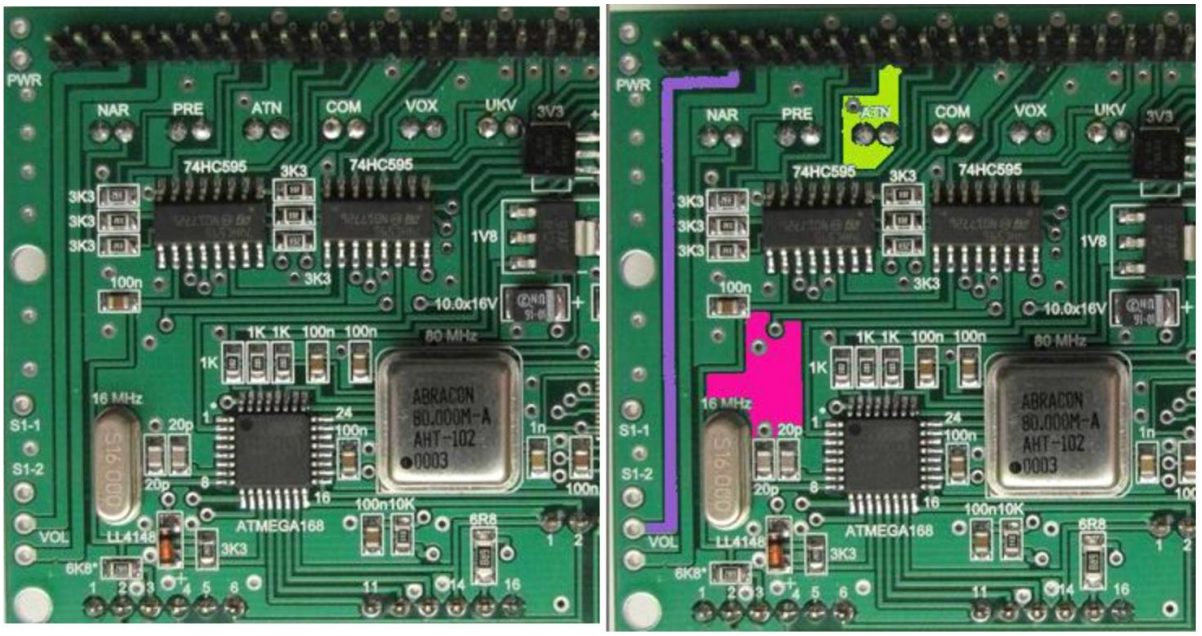

Рисунок 1.9 – Приклад роботи FloodFill

На наступному рисунку показаний варіант роботи FloodFill у разі пошкодження однієї з кордонів у попередньому зображенні (рис 1.10).

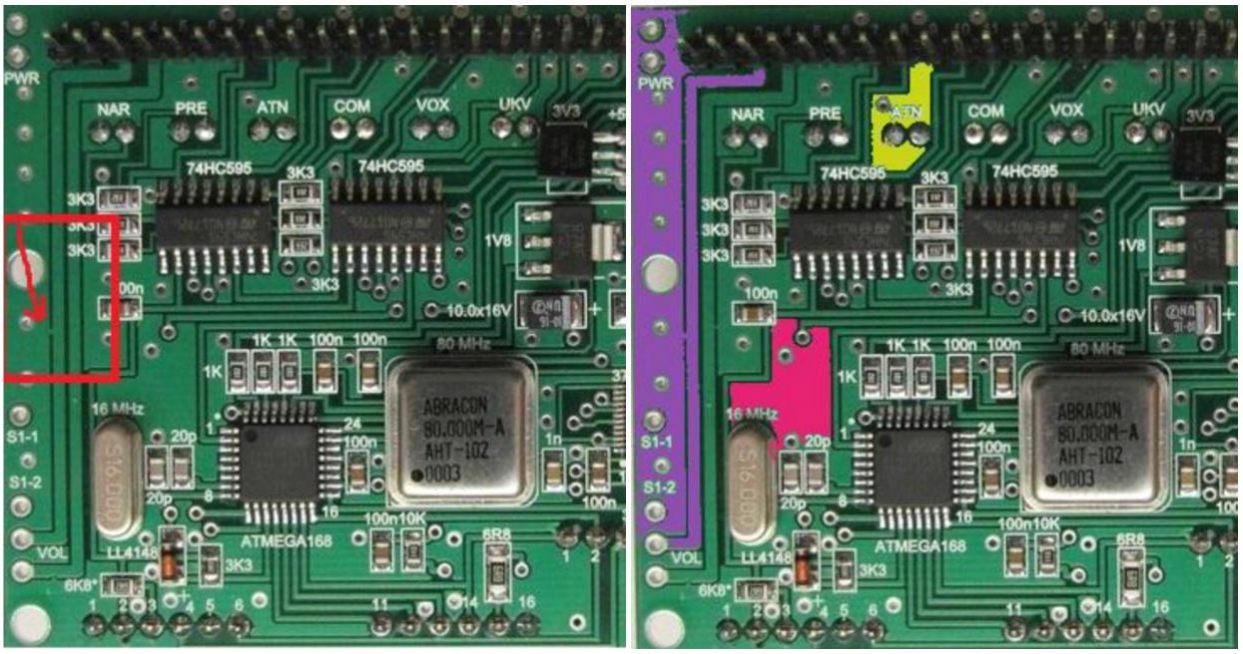

Рисунок 1.10 – Результат FloodFill у випадку пошкодження кордонів області

1.2.3 GrabCut

GrabCut – це інтерактивний алгоритм виділення об’єкта, розроблявся як більш зручна альтернатива магнітному ласо (щоб виділити об’єкт, користувачеві потрібно обвести його контур за допомогою миші). Для роботи алгоритму досить укласти об’єкт разом з частиною фону в прямокутник (grab). Сегментування об’єкту відбудеться автоматично (cut) як показано на рисунку 1.11.

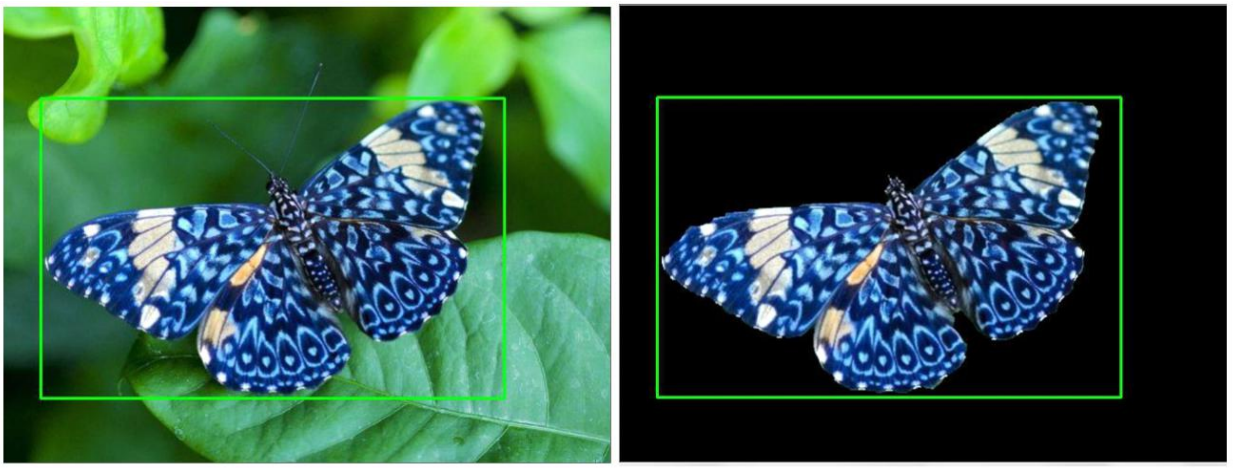

Рисунок 1.11 – Приклад роботи GrabCut

Сегментація може бути складною, якщо всередині межі є кольори, які є в достатку не тільки на об’єкті, але й на фоні. У цьому випадку можна нанести додаткові розмітки на об’єкт (червона лінія) і на фон (синя лінія) (рис. 1.12).

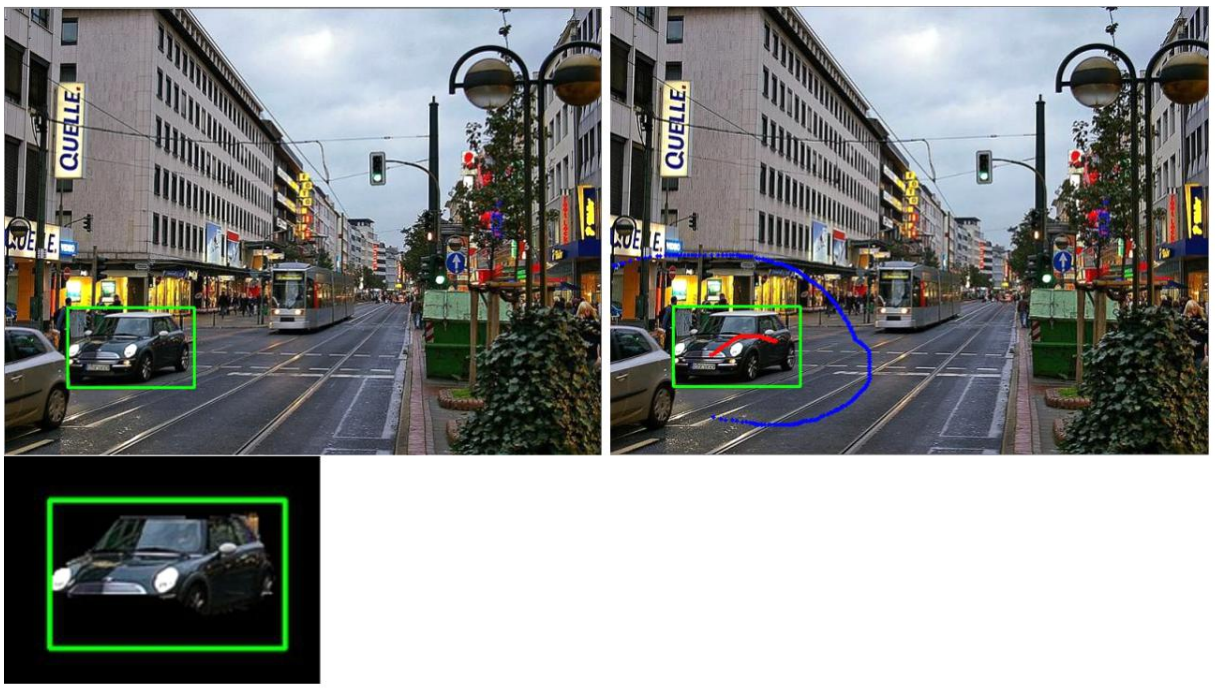

Рисунок 1.12 – Приклад роботи GrabCut з мітками об’єкта та фону

1.2.4. Алгоритм сегментації MeanShift

MeanShift групує об’єкти зі схожими характеристиками. Пікселі зі схожими характеристиками об’єднуються в один сегмент, на виході виходить зображення з однорідними ділянками (рис. 1.13) [14].

Рисунок 1.13 – Приклад роботи MeanShift

Ви можете вибрати піксельні координати (x, y) і компоненти пікселя RGB як координати у функціональному просторі. Відображаючи пікселі в просторі ознак, можна побачити конденсацію в певних місцях (рис. 1.14).

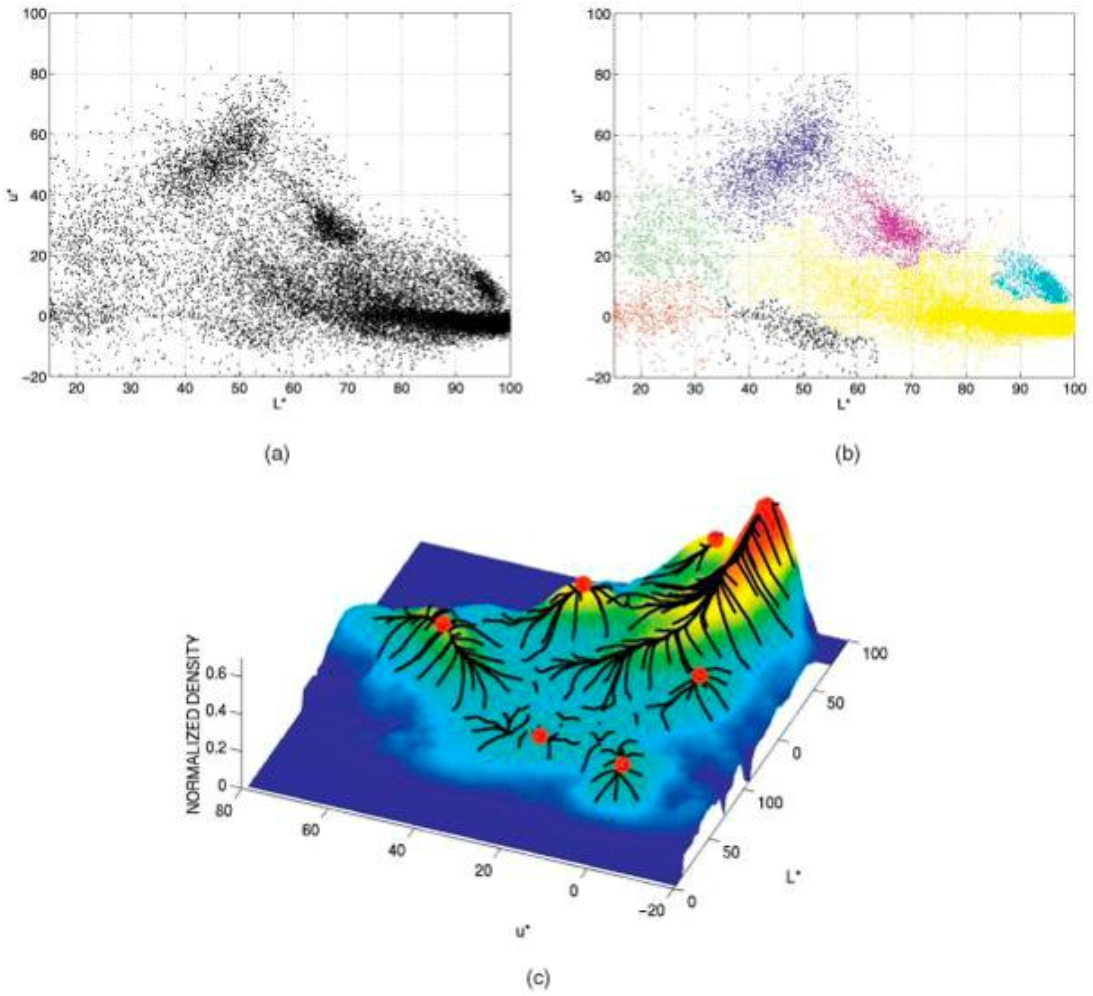

Рисунок 1.14 – Представлення MeanShift на системі координат

Коли ви вибираєте пікселі та інтенсивність кольору як знак, пікселі схожого кольору об’єднуються в один сегмент і розміщуються близько один до одного. Тому, якщо ви виберете інший вектор ознак, піксель об’єднається в сегменти. Наприклад, якщо видалити координати з символів, небо та озеро будуть розглядатися як один сегмент, оскільки пікселі цих об’єктів у просторі символів впали до одного локального максимуму.

Якщо об’єкт, який ми хочемо виділити, складається з областей, які сильно відрізняються за кольором, MeanShift не зможе об’єднати ці області в одну, і наш об’єкт буде складатися з кількох сегментів. Але добре мати справу

з об’єктом однотонного кольору на кольоровому фоні. MeanShift також використовується при реалізації алгоритму відстеження рухомих об’єктів.

1.2.5. Аналіз представлених алгоритмів сегментації зображень

Ми розглянули лише малу частину існуючих алгоритмів. FloodFill підходить для заливки об’єктів суцільного кольору. GrabCut добре відокремить певний об’єкт від фону. WaterShed підходить для зображень із простою текстурою. Тому алгоритм сегментації, звичайно, повинен бути обраний з конкретного завдання.

1.3 Аналіз існуючих рішень з сегментації зображень

Через об’ємність або відсутність рішень у відкритому доступі можна сказати про складність реалізації системи спостереження за набором зображень. Більшість з них розроблені у межах великих підприємств, що мають велику інфраструктуру та обсяги промисловості. Таке програмне забезпечення знаходиться у закритому вигляді.

Проте можна допустити, що попит у подібному, але легшому софті, достатній і його могли використовувати у повсякденному житті. Моніторингом певної місцевості люди займаються кожен день. Наприклад, це має місце в спостереженням за благоустроєм міста: чистота тротуарів, стан доріг, виявлення пошкоджень на пам’ятка архітектури, охайність стін будівель, стан промислових об’єктів та ін.

Отже, розробка відкритого універсального програмного рішення задачі моніторингу та виявлення пошкоджень на об’єктах може мати велику

цінність для декількох індустрій, включаючи індустрію малих підприємств та державних організацій охорони довкілля. Актуальним є питання розробки продукти для широкої аудиторії з інтегруванням з існуючими екосистемами за API.

1.4 Постановка задачі

Метою роботи є розробка web-додатку для спостереження за навколишнім середовищем на основі аналізу набору зображень. Будуть використані тестові зображення, проте можна вважати, що вони потенційно зняті на камеру спостереження. Користувач повинен завантажити фотографію у веб-інтерфейсі для подальшого розпізнавання на ньому об`єктів та їх пошкоджень.

Ідентифікація об’єктів на фото буде відбуватися за допомогою побудованої моделі та набору тренувальних даних. Далі буде виконуватись модуль розпізнавання дефектів з подальшим їх маркуванням. Буде розроблен інтерфейс програми з можливістю додати об’єкти для моніторингу та тренувальні набори даних.

В ході розробки веба планується використати Python, Django та Angular. Реалізація ЗМН буде виконана завдяки бібліотеки TensorFlow в тандемі з Keras. Для ідентифікації об’єктів буду користуватись бібліотекою ImageAI для Python.

2 АНАЛІЗ ЗАСОБІВ ДЛЯ РОЗПІЗНАВАННЯ ОБ’ЄКТІВ

2.1 Аргументація вибору нейронної мережі

Найбільш значущі результати в області розпізнавання об’єктів показує згорткова нейронна мережа, яка є логічним просуванням ідей у таких архітектурних нейронних мережах, як когнітрон і неокогнітрон. Його успіх пояснюється можливістю розглядати топологію двовимірного зображення, на відміну від багатошарового персептрона [1].

Згорткові нейронні мережі забезпечують часткову стійкість до змін розміру, переміщення, обертання, зміни кута та інших деформацій. Згорткові нейронні мережі поєднують три архітектурні ідеї, щоб забезпечити інваріантність для масштабування, обертального зміщення та просторової деформації:

– локальні рецепторні масиви, які забезпечують локальні двовимірні зв’язки з нейронами;

– загальні ваги синапсів, тобто ідентифікація певних елементів на всьому зображенні та зменшення загальної ваги;

– ієрархічна організація з просторовими підвибірками.

На даний момент згортка нейронна мережа та її модифікації вважаються найкращими за точністю та швидкістю алгоритмів визначення місця розташування об’єктів на зображеннях [3].

2.2 Структура згорткової нейронної мережі

Згорткова нейронна мережа складається з кількох типів шарів: шар згортки, шар підвибірки та «звичайний» шар нейронної мережі відповідно до рисунка 2.1:

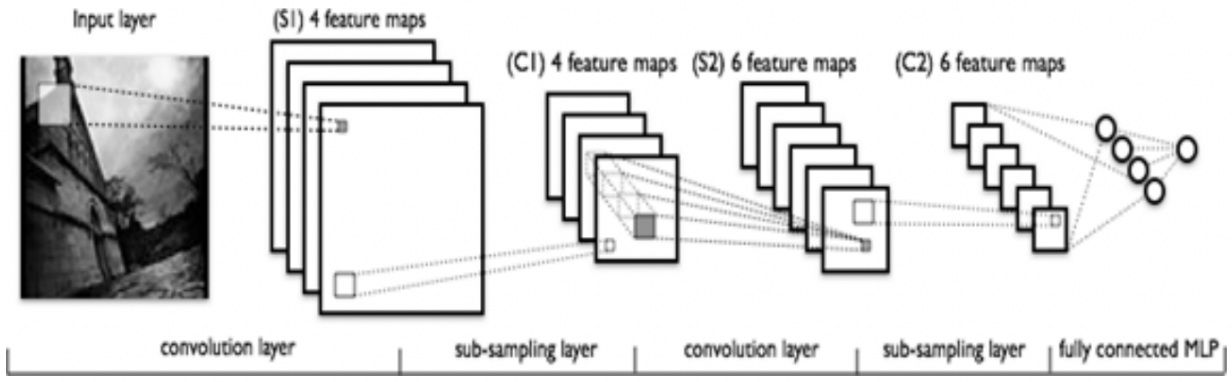

Рисунок 2.1 – Топологія ЗНМ

Перші два типи шарів, які чергуються один з одним, стають вхідними векторними функціями для багатошарового персептрона. Згортка мережа має свою назву відповідно до назви операції - згортка, суть якої описана нижче.

Згорткова мережа є хорошим центром між біологічно правдоподібною мережею і звичайним багатошаровим персептроном. З їх допомогою наразі досягнуто найкращих результатів для розпізнавання зображень. У середньому точність ідентифікації таких мереж перевищує звичайний ІНМ на 10-15%. ЗНМ є важливою технологією глибокого навчання.

Основною причиною успіху ЗНМ є концепція спільної ваги. Незважаючи на великий розмір, ці мережі мають невелику кількість параметрів, що налаштовуються, порівняно з їхніми неокогнітронними предком. Існують варіанти ЗНМ, схожі на неокогнітрон. У таких мережах відбувається деяка відмова від пов’язаних заходів, але алгоритм навчання залишається тим самим і базується на протилежних поширених помилках [26]. ЗНМ може негайно працювати на послідовній машині та миттєво

навчатися шляхом розпаралелювання процесу згортки на кожній карті, а також зворотної згортки, якщо помилка поширюється по мережі.

2.3 Топологія згорткової нейронної мережі

Визначення топології нейронної мережі орієнтовано на завдання, дані наукових статей та власний експериментальний досвід.

Ви можете вибрати наступні кроки, які впливають на вибір топології:

– визначити завдання за допомогою нейронних мереж (класифікація, передбачення, модифікація);

– визначити обмеження вирішення проблеми (швидкість, точність відповіді);

– вказати вхід (тип: зображення, звук) і вихід (кількість класів). Завдання моєї нейронної мережі — класифікувати зображення, особливо промислові об’єкти [2]. Заборона мережі має точність виявлення не менше 70%. Загальна топологія мережі згідно з рисунком 2.2.

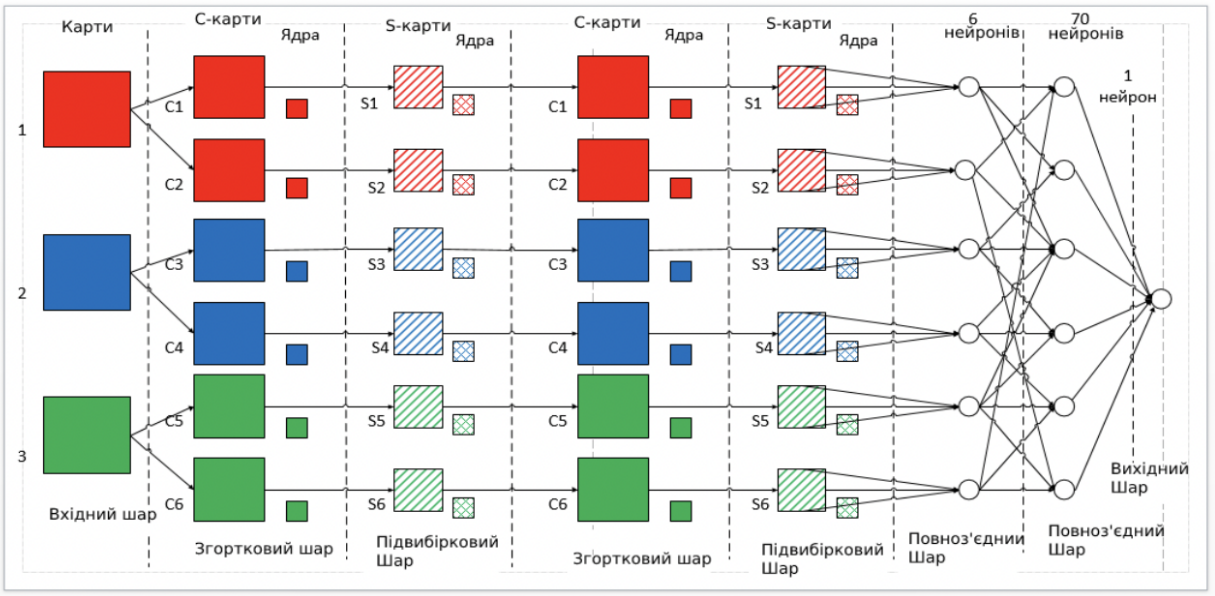

Рисунок 2.2 – Топологія нейронної мережі

2.4 Вхідний шар

Вхідні дані являють собою кольорові зображення типу JPEG, розміру 600x600 пікселів. Якщо розмір занадто великий, зростає складність обчислень або порушується межа швидкості відгуку, визначення розміру цієї задачі можна вирішити методом відбору. Якщо ви виберете занадто малий розмір, мережа не помітить ключових особливостей. Кожне зображення розділене на 3 канали: червоний, синій, зелений. Створюються 3 зображення розміром 600 x 600 пікселів [5]. Вхідний шар враховує двовимірну топологію зображень і складається з кількох карток (матриць), картка стає одноразовою, якщо зображення представлено відтінками сірого, якщо не 3, де кожній картці на зображенні відповідає певний канал (червоний, синій і зелений).

Вхідні дані кожного конкретного значення пікселя нормалізуються в діапазон від 0 до 1, за формулою [11]:

$$f\ (p,\ min,\ max)\ = \frac{p - min}{max - min}\ , \qquad (2.1)$$

де f — функція нормалізації;

p — значення кольору пікселя від 0 до 255;

min — мінімальне значення пікселя 0, max — максимальне значення пікселя 255.

2.5. Згортковий шар

Шар згортки — це набір карт (матриць), кожна карта має синаптичне ядро (фільтр).

Кількість карток визначається вимогами задачі. Якщо взяти велику кількість карток, то підвищиться якість розпізнавання, але підвищиться ефективність обчислення [15]. На основі аналізу наукових статей у більшості випадків пропонується брати співвідношення один до двох, тобто кожна карта попереднього шару (наприклад, перший шар згортки, перший є вхідним) підключено до двох шарів згортки карт, як показано на рисунку 2.3.

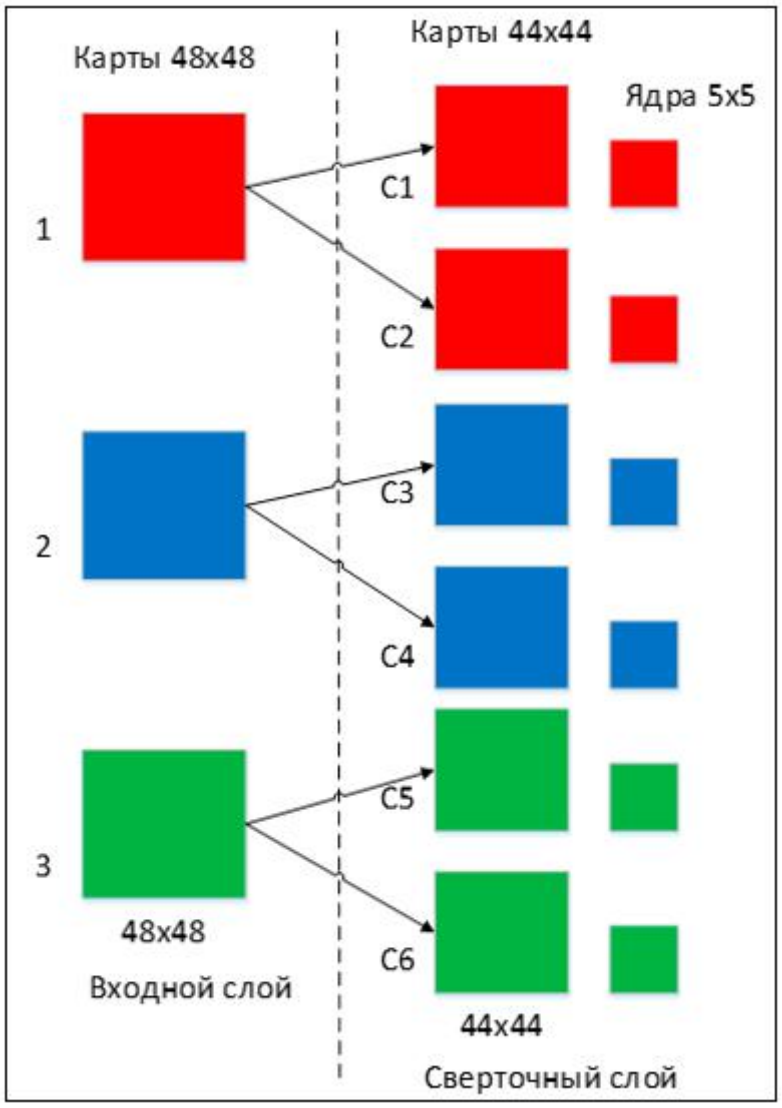

Рисунок 2.3 – Організація зв’язків між картами згорткового шару та попередніми

Розмір у всіх карт згорткового шару однаковий та обчислюється за формулою [8]:

$$(w,\ h)\ =\ (mW\ -\ kW\ +\ 1,\ mH\ -\ kH\ +\ 1), \qquad (2.2)$$

де *(w, h)* — розмір згорткової картки;

mW — ширина попередньої картки;

mH — висота попередньої картки;

kW — ширина ядра згортки;

kH — висота ядра згортки;

Ядро - це фільтр або вікно, яке ковзає по області попередньої карти і виявляє конкретні властивості об’єктів. Наприклад, якщо мережа навчена на основі багатьох людей, то одне з ядер може подавати найбільший сигнал у процесі навчання оку, роту, брів або носа, інше ядро буде виявляти інші сигнали. Розмір ядра зазвичай приймається в порядку від 3x3 до 7x7 [9]. Якщо розмір ядра невеликий, воно не може розпізнати мітки. Якщо ядро занадто велике, то кількість зв’язків між нейронами збільшується. Крім того, розмір ядра вибирається таким чином, щоб розмір карт шару згортки був однаковим, тому ви не можете втратити інформацію при зменшенні розмірності шару підвибірки, описаного нижче.

Ядро являє собою систему ваг або синапсів, що розділяються, це одна з головних особливостей згорткової нейромережі. У звичайній багатошаровій мережі дуже багато зв’язків між нейронами, тобто синапсами, що дуже уповільнює процес детектування [7]. У згортковій мережі – навпаки, загальні ваги дозволяє скоротити кількість зв’язків і дозволити знаходити ту саму ознаку по всій області зображення.

Спочатку значення кожної карти в шарі згортки дорівнюють 0. Значення ваг ядер розміщені випадковим чином у діапазоні від 0,5 до 0,5. Ядро видаляє попередню карту і виконує операцію згортки, яка часто використовується для обробки зображень. Формула операції згортки:

$$(f * g)[m,n] = \sum_{k,l} f[m - k,\ n - l] * g[k,\ l], \qquad (2.3)$$

де f – початкова матриця зображення;

g – ядро згортки;

[m, n] – розмірність поточного сектора на зображенні;

[k, l] – розмірність ядра згортки.

Неформально цю операцію можна описати так: вікном розміру ядра g проходимо із заданим кроком (зазвичай 1) все зображення f, кожному кроці поелементно множимо вміст вікна на ядро g, результат сумується і записується в матрицю результату [12], як у рисунку 2.4.

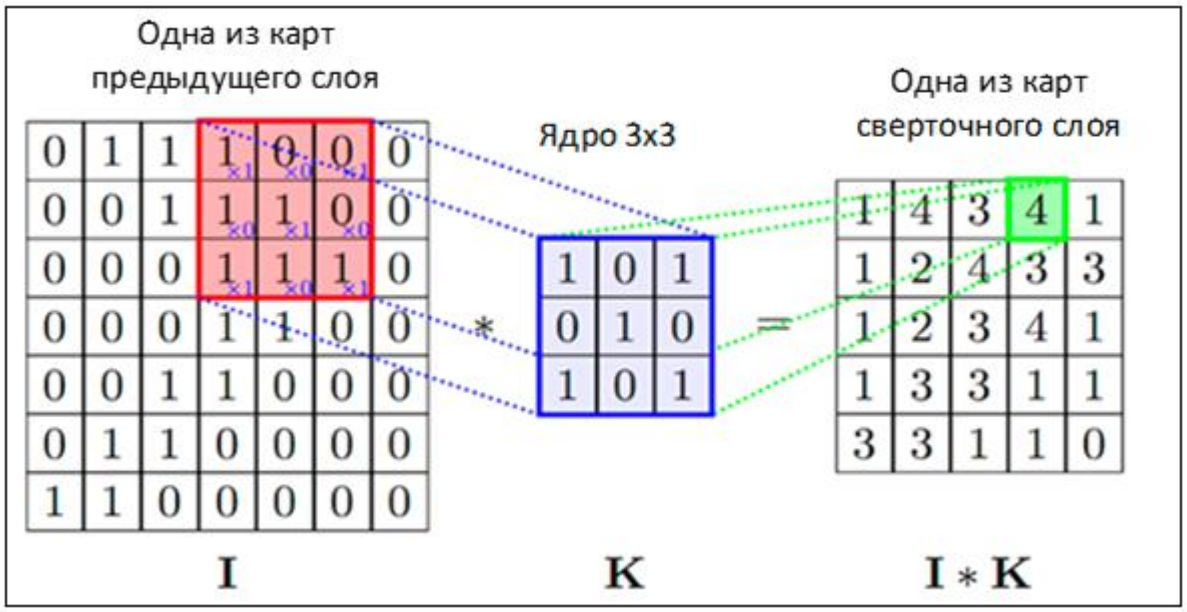

Рисунок 2.4 – Операція згортки та отримання значень згорткової карти

2.6. Підвиборчий шар

Шар підвибірки такий же, як і карта згортки, але їх кількість відповідає шару згортки, їх 6. Мета шару — зменшити розмір карт попереднього шару. Якщо попередня операція згортки вже виявила деякі особливості, то подальша обробка такого детального зображення не потрібна, і воно було складено менш детально. Крім того, фільтрація непотрібних функцій допомагає запобігти повторному навчанню.

У процесі сканування ядра шару підвибірки (фільтра) на карті попереднього шару ядро сканування не перетинається, на відміну від шару

згортки. Як правило, кожна карта має ядро 2x2, що зменшує попередні карти шару згортки в 2 рази. Уся карта функцій поділена на клітинки елементів розміром 2x2, включаючи максимальну, вибрану за значенням [13].

Зазвичай у підвиборному шарі застосовується функція активації RelU Операція підвибору, така як MaxPooling показана на рисунку 2.5.

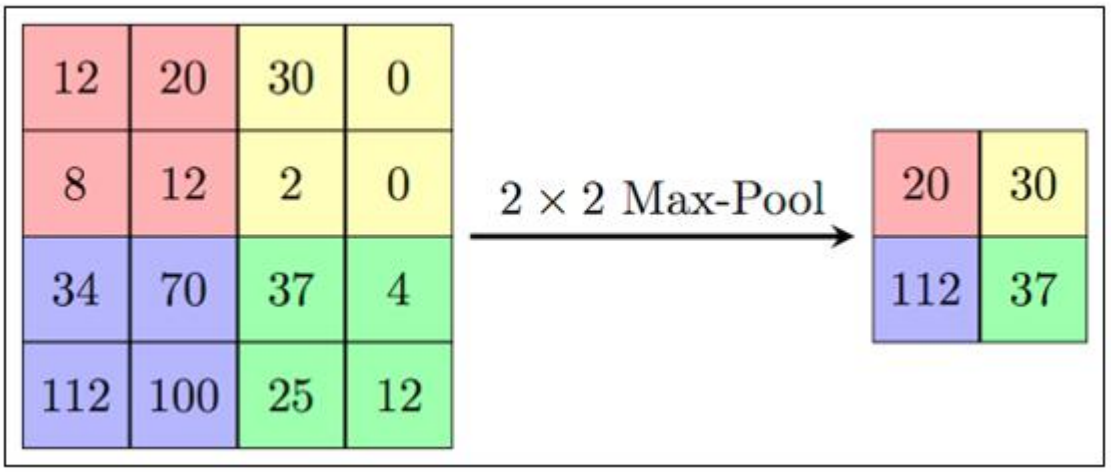

Рисунок 2.5 – Max Pooling (Операція вибірки). Формування нової карти підвиборчого шару на базі попередньої картки згорткового шару

2.7 Повнозв’язний шар

Останній із типів шарів є шаром звичайного багатошарового персептрона (рис. 2.6). Мета шару – класифікація, моделює складну нелінійну функцію. Завдяки її оптимізації, покращується якість розпізнавання [14].

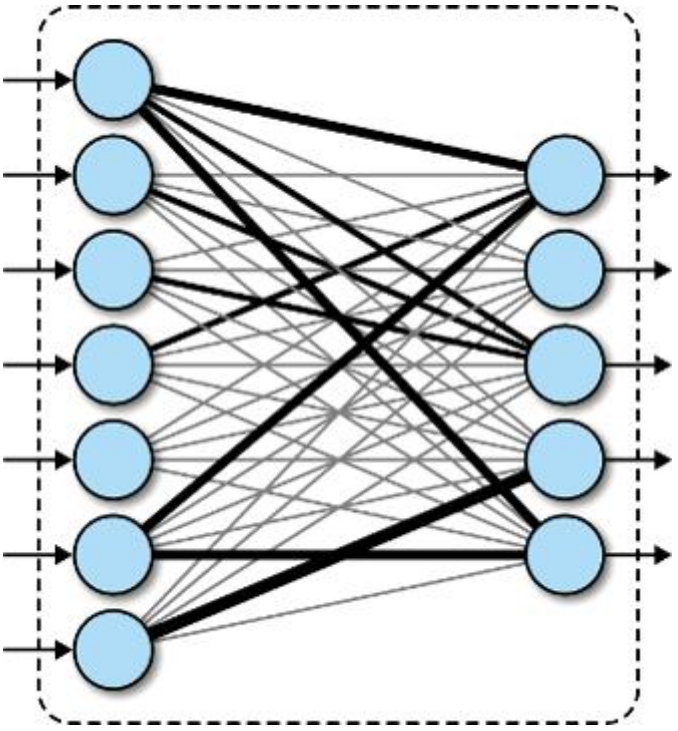

Рисунок 2.6 – Повнозв’язний шар

Нейрони кожної карти попереднього шару зв’язані з одним нейроном прихованого шару. Таким чином число нейронів прихованого шару дорівнює числу карт підвибіркового шару. Проте зв’язки можуть бути не обов’язково такими. Наприклад, тільки частина нейронів будь-якої з карт підвибіркового шару пов’язана з першим нейроном прихованого шару, а частина, що залишилася з другим, або всі нейрони першої карти пов’язані з нейронами першого та другого прихованого шару.

2.8 Вихідний шар

Вихідний шар пов’язаний із усіма нейронами попереднього шару. Кількість нейронів відповідає кількості класів, що розпізнаються, тобто 2 (C1, C2). Однак, для зменшення кількості зв’язків та обчислень для бінарного випадку ми можемо використовувати один нейрон. При використанні, у

якості функцію активації гіперболічний тангенс, вихід нейрона зі значенням -1 має на увазі приналежність до класу C1, а вихід нейрона зі значенням 1 означає приналежність до класу C2.

2.9 Функція активації

Одним із етапів розвитку нейронної мережі є вибір функції активації нейронів. Тип функції активації значною мірою визначає функціональність нейронної мережі та спосіб навчання мережі. Класичний алгоритм зворотного поширення помилок дуже добре працює на двошарових і тришарових нейронних мережах, але зі збільшенням глибини починають виникати проблеми [6]. Однією з причин є так зване ослаблення градієнтів. Оскільки помилка поширюється від вихідного шару до вхідного шару кожного шару, поточний результат виведення функції активації збільшується. Похідна у традиційної сигмоїдної функції активації менше одиниці по всій області визначення, тому після кількох шарів похибка може наблизитися до нуля. І навпаки, якщо функція активації має нескінченну похідну (наприклад, гіперболічний тангенс), її поширення може призвести до вибухового збільшення похибки, що призведе до нестабільності процесу навчання [10].

3. КОМП'ЮТЕРНА МОДЕЛЬ РОЗПІЗНАВАННЯ ПОШКОДЖЕНЬ НА ОБ'ЄКТАХ

3.1 Обґрунтування вибору середовища програмної реалізації

В ході кваліфікаційної роботи були використані бібліотека TensorFlow та фреймворк Keras на основі архітектури MobileNet. Обрана мова програмування - Python з фреймворком Django.

3.1.1 Бібліотека TensorFlow

Бібліотека – це файл або набір файлів, що містять підпрограми, функції, які використовуються для розробки програмного забезпечення. Різні мови програмування мають свій набір бібліотек [18].

Виділяють:

– стандартні бібліотеки – постачаються додатково з мовою програмування;

– призначені для користувача бібліотеки – розробляються програмістами.

Бібліотека TensorFlow значно спрощує інтеграцію використання елементів і функцій штучного інтелекту, призначених для розпізнавання мови, комп'ютерного зору або обробки природної мови.

TensorFlow – це не тільки найглибша бібліотека для навчання, але, як і пошукова система Google, вона вважається найкращою у своєму класі. Альтернативи включають програмне забезпечення Torch, розроблене

швейцарськими дослідниками та Каліфорнійським університетом. Остання версію Berkeley Caffe, Caffe2 була розроблена за допомогою Facebook.

Бібліотека підходить для широкого сімейства методів машинного навчання, а не тільки для нейронних мереж. Лінійна алгебра та інші принципи, засновані на даному фреймворку, добре спостерігаються. На додаток до основних функцій машинного навчання TensorFlow також включає власну систему журналів, власний інтерактивний візуалізатор журналів і навіть потужну архітектуру даних. Модель виводу TensorFlow відрізняється від Python scikit-learn і більшості R-інструментів.

3.1.2 Фреймворк Keras

Keras – це фреймворк, який доповнює бібліотеки для моделей нейронних мереж. Він має добре розроблений API, який дозволяє створювати модель шляхом збору блоків високого рівня. Ця структура написана на Python і може використовуватися як доповнення до таких бібліотек, як TensorFlow, CNTK або Theano [20]. Keras розроблено для спрощення моделей машинного навчання [19]. Основні особливості фреймворку, що визначають простоту використання розробки:

– дозволяє швидко та легко створювати прототипи з легкістю, модульністю та масштабованістю;

– підтримує згорткові мережі та мережі, що повторюються, а також їх комбінацію;

– можна використовувати процесор і відеокарту для розрахунків.

3.1.3 Фреймворк Django

Django – це безкоштовний фреймворк з відкритим вихідним кодом, написана на Python [17]. Архітектура Django подібна до шаблону Model View Controller (MVC), але “контролер” Django – це “подання”, а “view” – то “шаблон”. Ось чому розробники Django називають модель MVC MVT, тобто “Шаблон перегляду моделі” [16].

Цей каркас має наступні переваги:

– швидкість. Django розроблено, щоб допомогти розробникам створювати програми якомога швидше. Це включає генерацію ідей, розробку та випуск проекту, де Django може заощадити час і ресурси на кожному з цих етапів. Таким чином, його можна назвати ідеальним рішенням для розробників, для яких питання дедлайну варто в пріоритет;

– повна комплектація: Django працює з десятками додаткових функцій, які помітно допомагають з аутентифікацією користувача, картами сайту, адмініструванням вмісту, RSS тощо. Дані аспекти допомагають здійснити кожен етап веб-розробки;

– безпека: працюючи в Django, користувач отримує захист від помилок, пов’язаних з безпекою, які ставлять під загрозу проект. Це можуть бути ін’єкції SQL, крос-сайт підробки, і крос-сайтовий скриптинг. Для ефективного використання логінів і паролів, система користувальницької аутентифікації є ключем;

– масштабованість: фреймворк Django найкращим чином підходить для роботи з високими трафіками. Тому велика кількість сайтів використовують Django для задоволення вимог, пов’язаних з трафіком;

– різнобічність: менеджмент контенту, наукові обчислювальні платформи, навіть великі організації - з усім цим можна ефективно справлятися за допомогою Django.

3.1.4 Архітектура MobileNet

Архітектура MobileNet на даний момент є найефективнішою архітектурою нейронної мережі. Переваги включають байдужість, гнучкість та оптимізацію. Його можна використовувати навіть на мобільних пристроях і комп’ютерах з малою ємністю.

Звичайна згортка являє собою фільтр $D_k * D_k * C_{in}$, де D_k — це розмір ядра згортки, а C_{in} — кількість каналів на вході. Загальна важкість обчислення згорткового шару має $D_k * D_k * C_{in} * D_f * D_f * C_{out}$, де D_f — висота та ширина шару, а C_{out} — кількість каналів на виході. Вважаємо, що просторові розміри вхідного та вихідного тензора співпадають [21].

Ідея Depthwise Separable Convolution заключається в тому, щоб розкласти подібний шар на depthwise-згортку, яка вдає із себе поканально фільтр, та 1x1-згортку, відому як pointwise convolution. Сумарна кількість операцій для застосування такого шару дорівнює $(D_k * D_k + C_{out}) * C_{in} * D_f * D_f$.

Частина згортки, яка нас цікавить, складається із звичайного шару згортки зі згорткою 3x3 на початку та 13 блоків, показаних у правій частині рисунка 3.1, з поступовим збільшенням кількості фільтрів та зменшенням просторового розміру тензора [21]. Ліворуч буде звичайний мережевий блок згортки, а праворуч — базовий блок MobileNet.

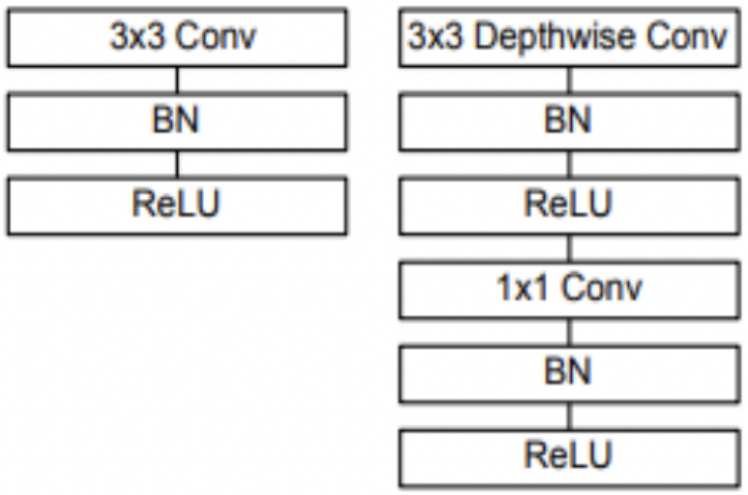

Рисунок 3.1 – Різниця між MobileNet та класичною згортковою мережею

Однією з особливостей цієї архітектури є відсутність максимальних шарів стовпів. Натомість для зменшення просторового виміру використовується згортка з параметром кроку 2. Коефіцієнт ширини відповідає за кількість каналів у кожному шарі. Наприклад, $\alpha = 1$ дає нам архітектуру, описану вище, а $\alpha = 0{,}25$ – архітектуру з чотириразовим зменшенням кількості вихідних каналів на блок.

Множник роздільної здатності відповідає за просторовий розмір вхідних датчиків. Наприклад, $\rho = 0{,}5$ означає, що довжина і ширина функціонального компонента, що надходить на вхід кожного шару, зменшуються вдвічі. Обидва параметри можуть змінювати розмір мережі: ми зменшуємо α і ρ, зменшуємо точність розпізнавання, але в той же час збільшуємо швидкість і зменшуємо споживання пам’яті.

3.2 Програмна реалізація

Дана система призначена для двох акторів: користувача, який хоче перевірити статус об’єкта, і адміністратора, який може додавати камери спостереження та створювати або редагувати користувачів і групи доступу. Користувач повинен увійти, щоб використовувати систему.

3.2.1 Структура програми

Програмне забезпечення для цього завдання розроблено відповідно до парадигм об’єктно-орієнтованого програмування та шаблону MVC. При розробці програмного продукту була реалізована трирівнева архітектура для гнучкої роботи веб-системи, на рисунку 3.2 показано клієнт-серверний додаток, де користувач спілкується з сервером через браузер. Результат запитів відображається на сторінках HTML.

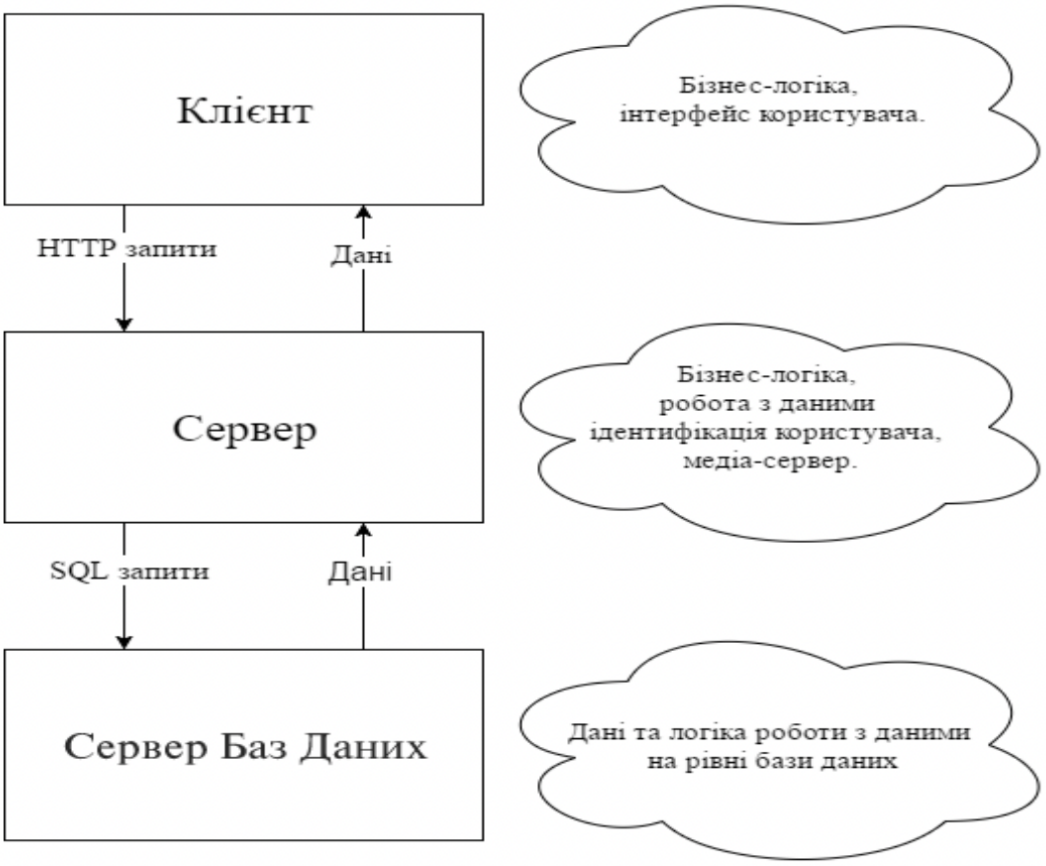

Рисунок 3.2 – Архітектура веб-системи

3.2.2 Опис розроблених алгоритмів

Такі технології, як Angular 6 і Angular Material, були використані для розробки цього веб-програмного забезпечення для обміну та обробки зображень для ідеального представлення інтерфейсу користувача.

Для реалізації серверної частини використовуються такі технології, як Python, Django REST Framework, MySQL, Django ORM. Для створення алгоритмів машинного навчання також використовуються такі технології, як TensorFlow, Keras, ImageAI, NumPy, SciPy. Вони реалізовані як частина серверної частини веб-системи.

Виконані кроки:

– додавання камери спостереження;

– додавання набору тренувальних даних для визначення об’єкту на зображенні;

– навчання системи для визначення зображень;

– реалізація захвату об’єкта з вхідного зображення;

– фіксування пошкоджень на об’єкті;

– обробка областей з пошкодженнями;

– візуалізація результатів;

Більш чітке представлення алгоритму сегментації та кроків реалізації зображене но рисунку 3.3 [22, 23, 25].

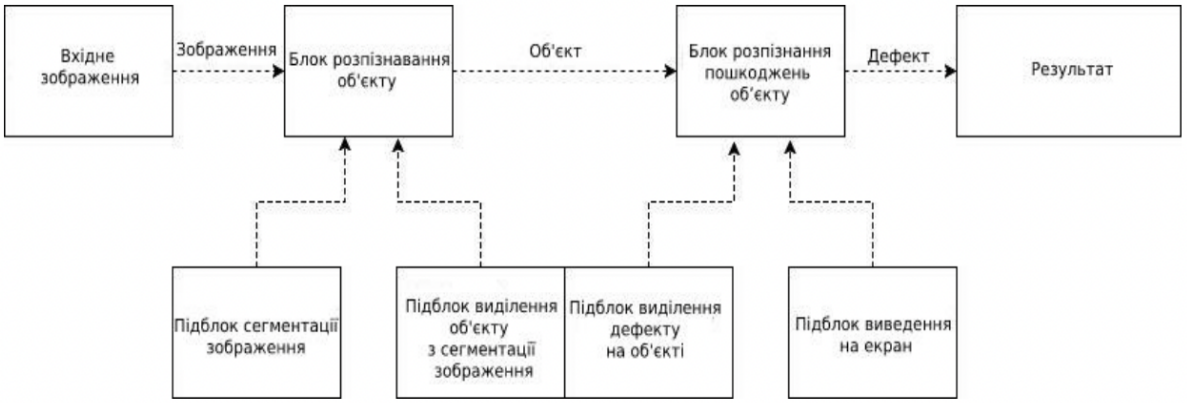

Рисунок 3.3 – Етапи роботи програми для розпізнавання пошкодження об’єкту

Також відображається діаграма вступу з основними діями що може зробити користувач (рис. 3.4).

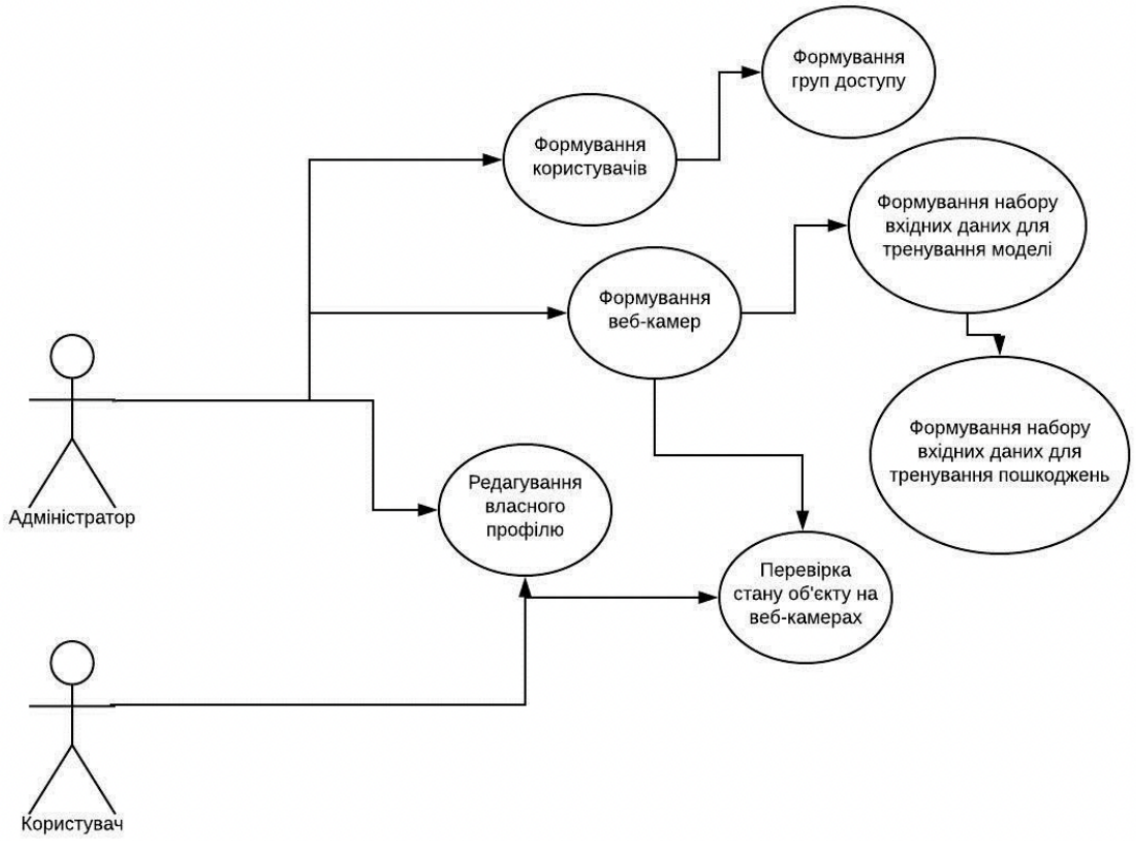

Рисунок 3.4 – Діаграма прецедентів програми

3.3 Інструкція користувача та тестування програми

Щоб використовувати функцію системи відстеження, користувачу необхідно увійти (рис. 3.5) [24].

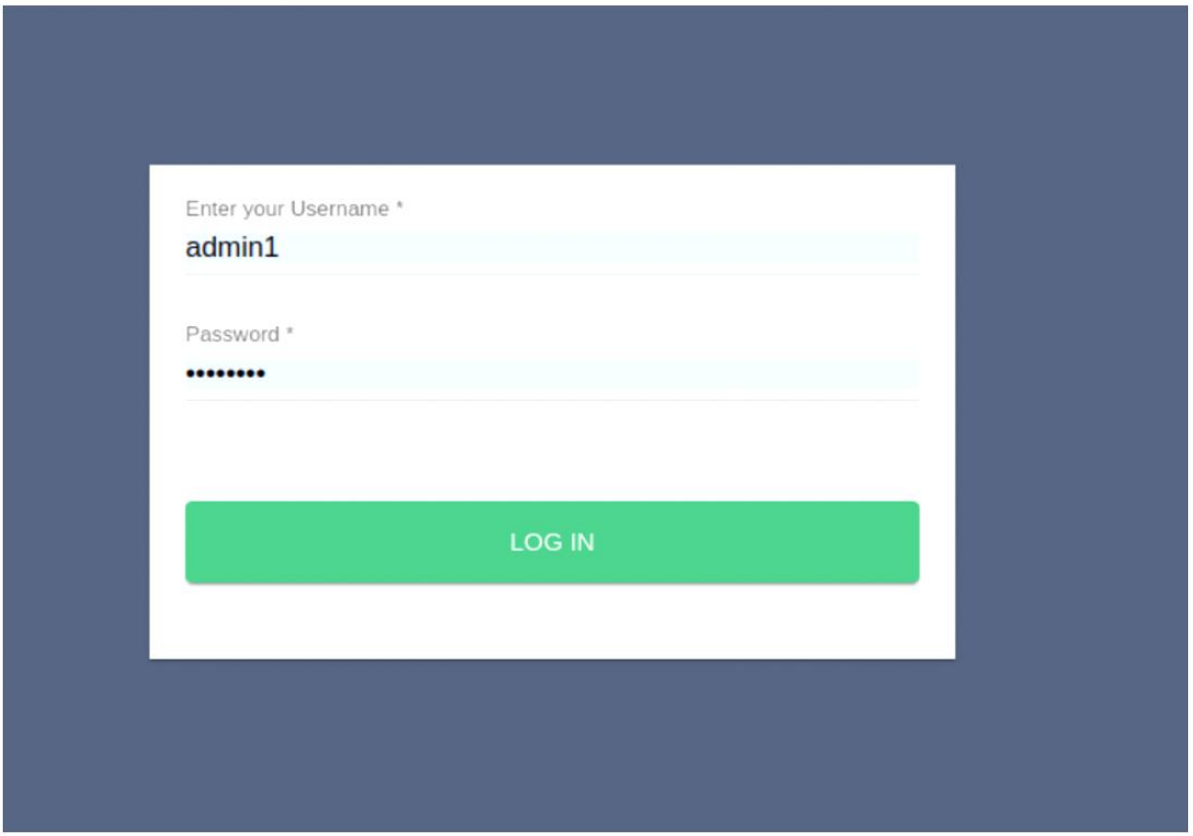

Рисунок 3.5 – Форма логіну до системи

Після успішного логіну в систему користувач потрапляє на контрольну панель. На цьому кроку можна додати камери спостереження та набори тренувальних даних (рис 3.6).

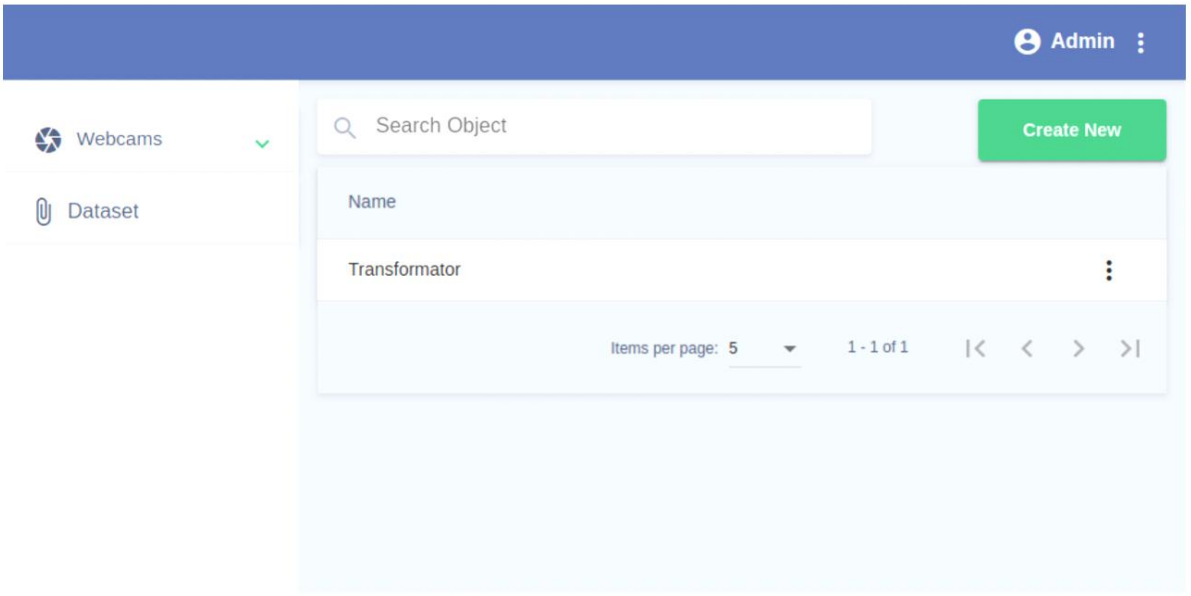

Рисунок 3.6 – Контрольна панель

Користувач, якщо є така потреба, може відредагувати свій профиль. Для цього йому потрібно відкрити випадаюче меню зверху справа (рис. 3.7).

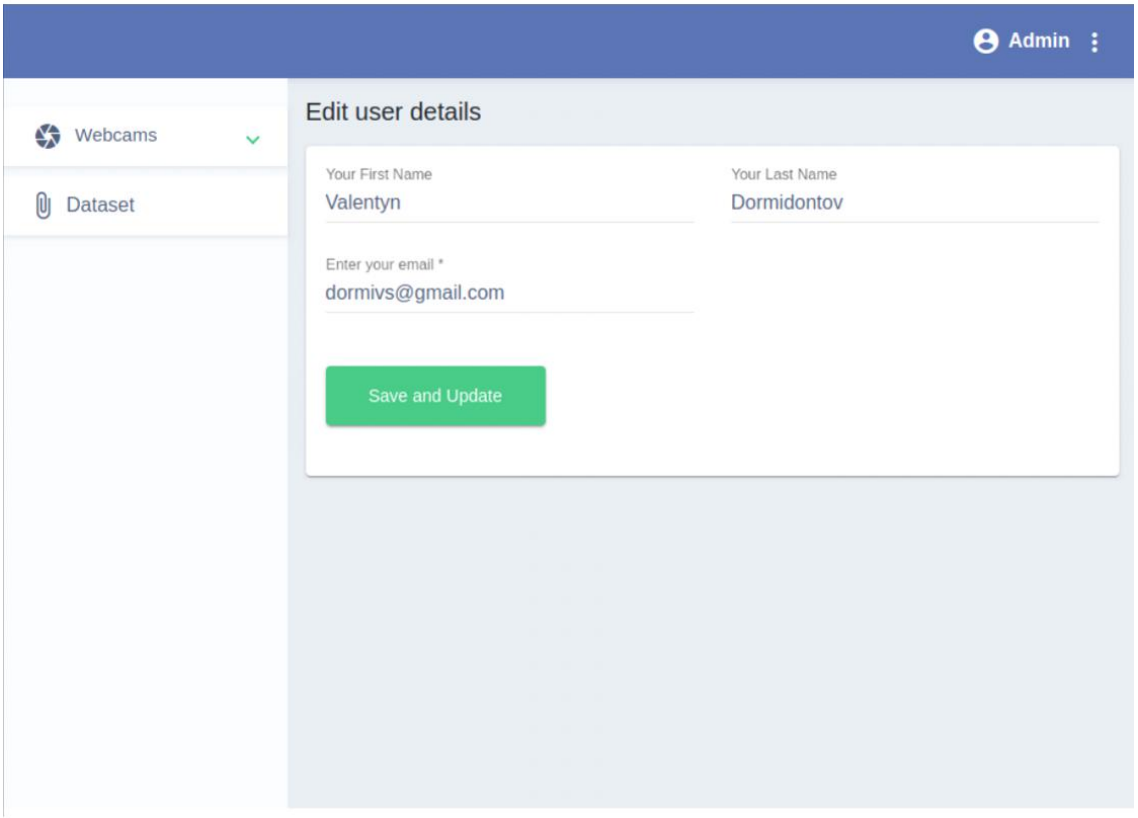

Рисунок 3.7 – Сторінка для редагування власного профілю

Адміністратор має доступ до сторінок управління групами та користувачами (рис. 3.8).

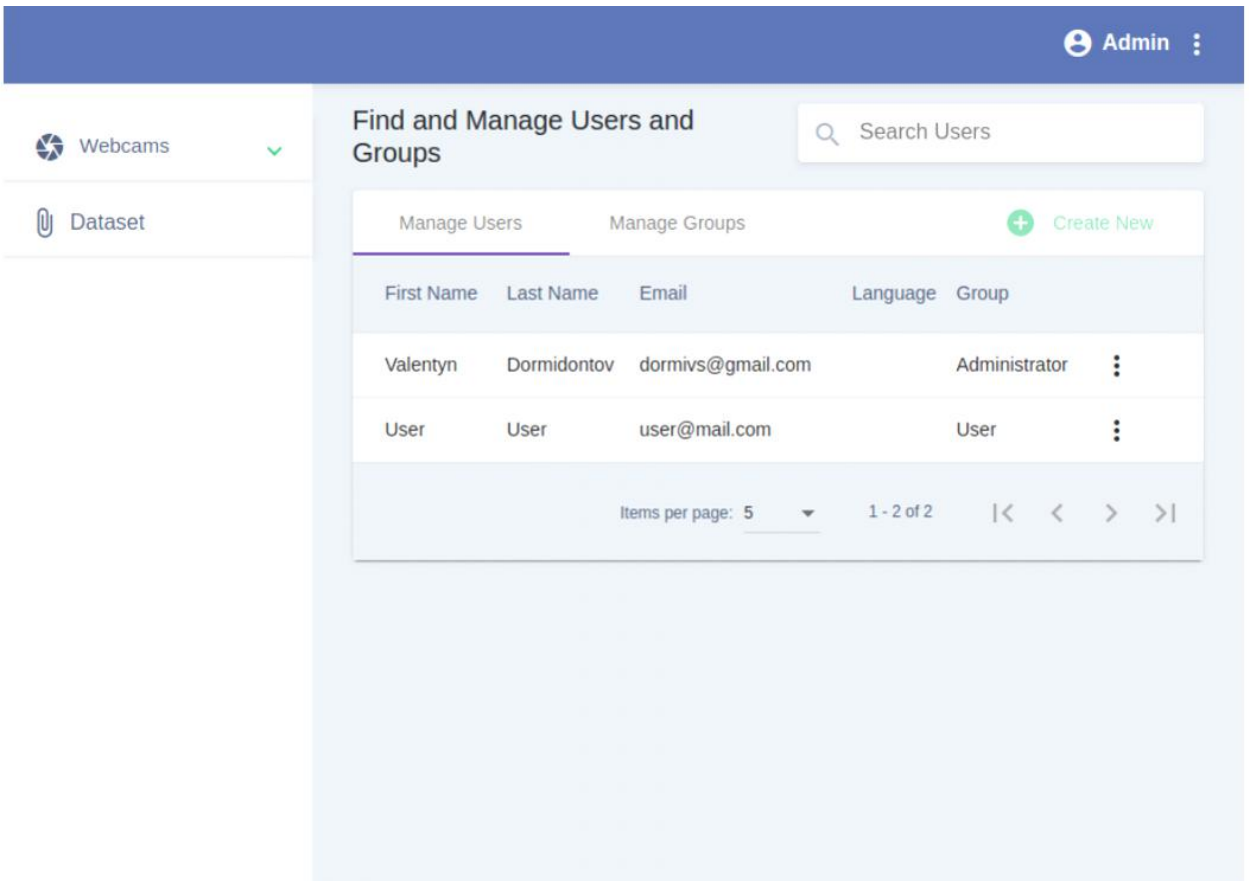

Рисунок 3.8 – Сторінка для управління користувачами

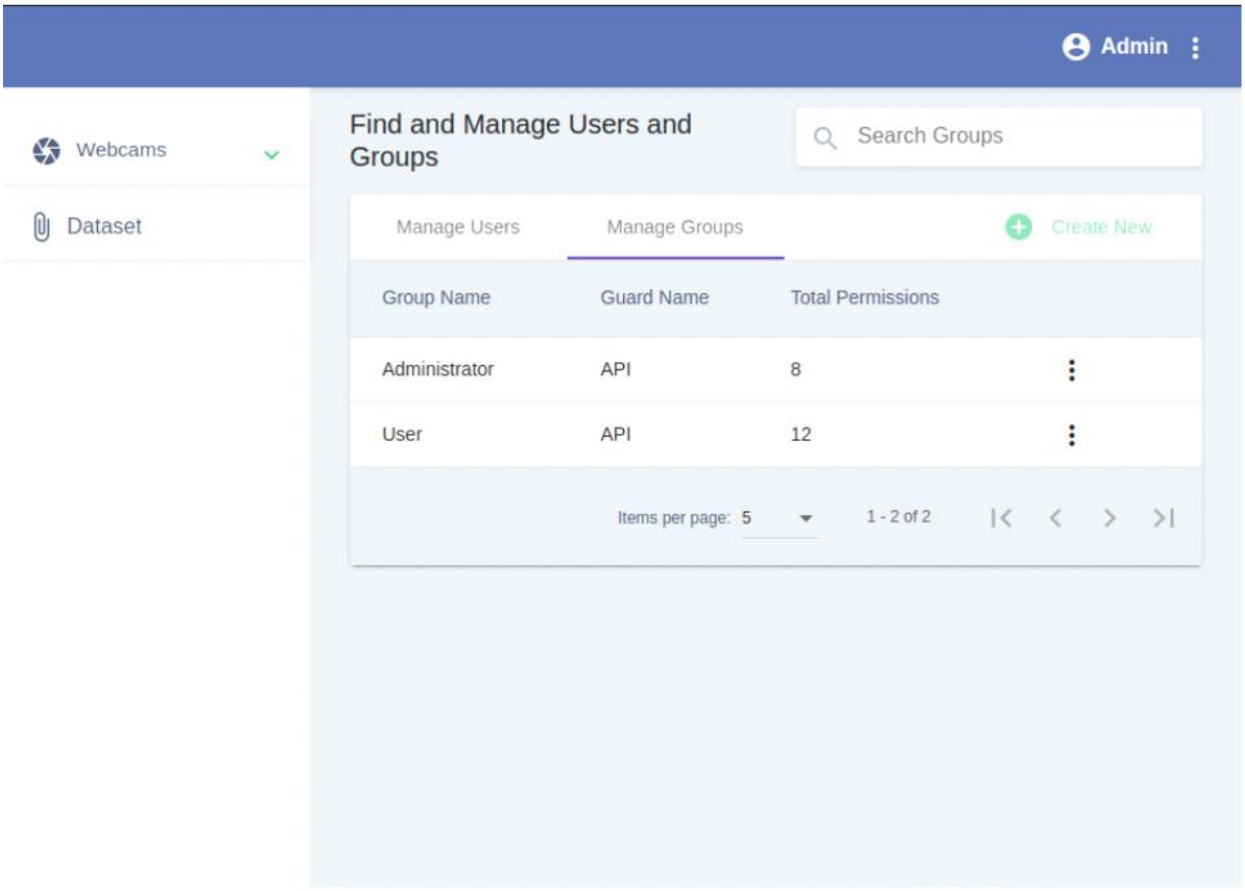

Рисунок 3.9 – Сторінка для управління групами

Адміністратор може створювати нових користувачів, групи та отримувати доступ до певних областей програми, а саме до веб-камер, наборів даних, користувачів тощо. Щоб перевірити об’єкт, необхідно заповнити набір даних для ідентифікації об’єкта, тобто завантажити кілька зображень об’єкту обведених в рамку як показано на рисунку 3.10.

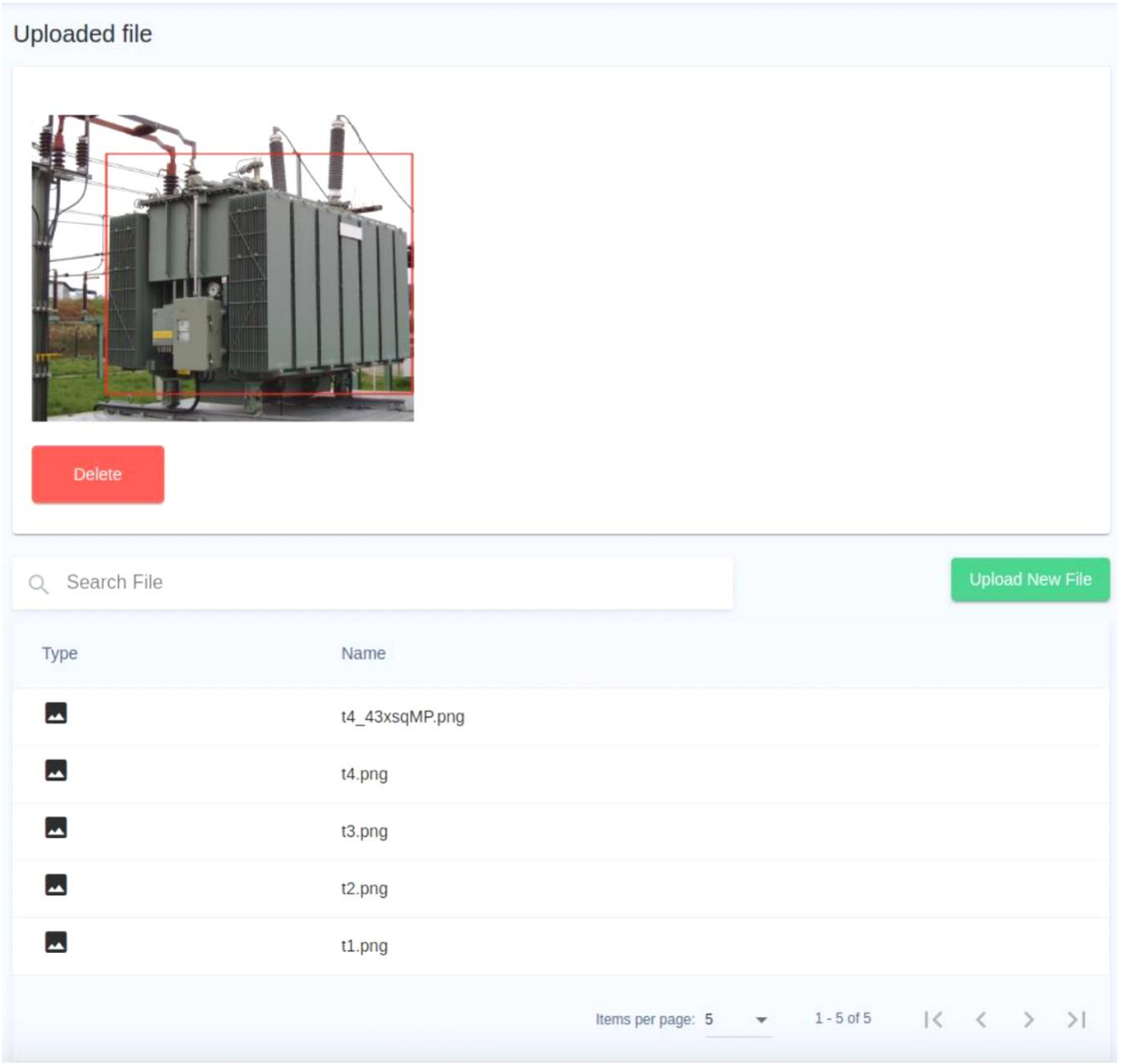

Рисунок 3.10 – Додавання тренувального набору даних

Крім того, система надає джерело даних з камери (рис. 3.11). Система розроблена з програмним забезпеченням для підключення камери до сервера та збереження фотографій камери у вибраній папці в структурі файлового медіа системи, а також для вказівки назви, опису та вибраного набору даних.

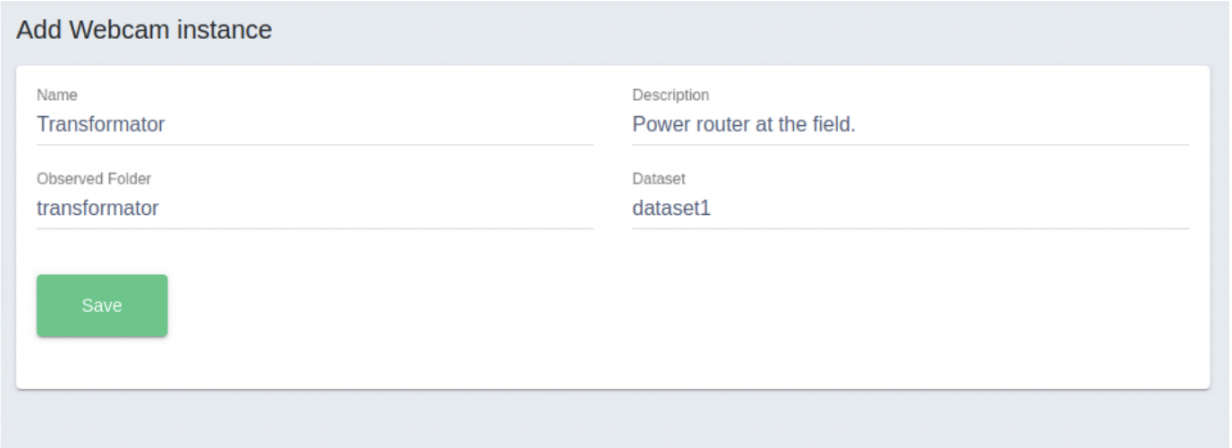

Рисунок 3.11 – Вікно для додавання камери

Після додавання одразу розпочнеться тренування моделі. Результат доданої камери показаний на рисунку 3.12.

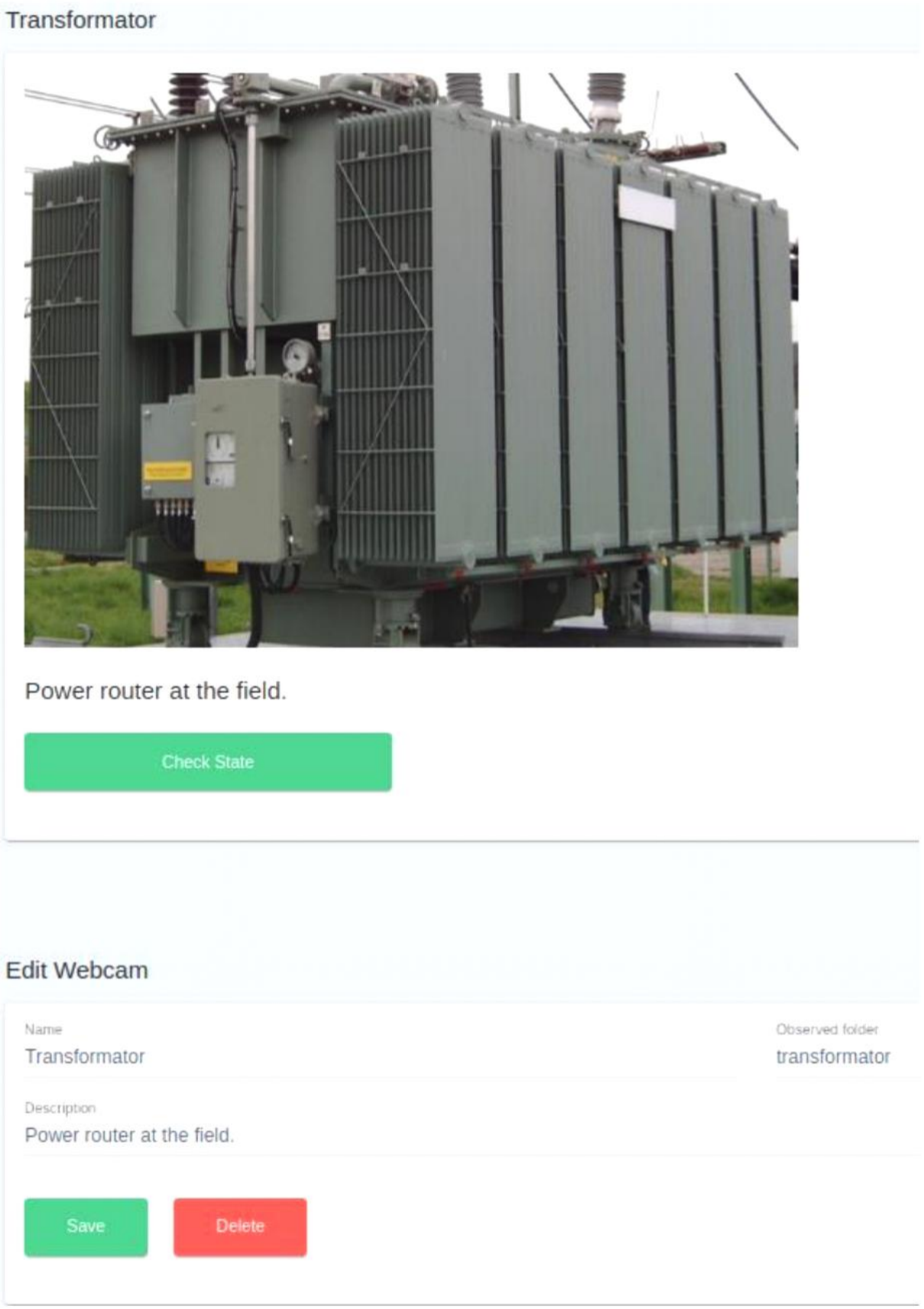

Рисунок 3.12 – Сторінка з доданою камерою спостереження

Щоб перевірити стан об’єкта спостереження, користувачу треба натиснути кнопку “Check State”. Система запросить з вказаної папки останнє за датою записане зображення (потенційно з камери спостереження) відбудеться сканування об’єкту. Як результат: користувачу буде виведений отримає зображення з маркованим дефектом на об’єкті (за наявності) (рис. 3.13).

Рисунок 3.13 – 5.12–Результат роботи виявлення дефекту.

ВИСНОВКИ

У рамках кваліфікаційної роботи було виконано аналіз питання сегментації та обробки зображень із застосуванням сучасних алгоритмів і технологій. Під час роботи були використані такі бібліотеки, як Keras та Tensorflow. Програмне забезпечення реалізоване у вигляді веб-системи для обробки зображень на прикладі фото та розпізнавання на ньому об`єктів. Далі було проведено аналіз об’єкта на предмет його пошкодження. За наявності пошкоджень, вони позначаються червоною рамкою для аналізу користувачем.

Виконується функція обробки результатів спільного використання динамічного набору зображень на прикладі промислових об’єктів. Реалізовано аналіз об’єкта зображення. Результат обробки зображення об’єкта з позначкою пошкодження виводиться для подальшого аналізу користувачем.

Система розроблена з можливістю розширення функціональності в майбутньому. Виконано всі завдання, необхідні для досягнення мети роботи, а саме сегментація та обробка зображення для моніторингу навколишнього середовища.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Yamashita, R., Nishio, M., Do, R. K. G., & Togashi, K. (2018). Convolutional neural networks: an overview and application in radiology. Insights into imaging, 9(4), 611-629.
2. Modarres, C., Astorga, N., Droguett, E. L., & Meruane, V. (2018). Convolutional neural networks for automated damage recognition and damage type identification. Structural Control and Health Monitoring, 25(10), e2230.
3. Kirichenko, L., Radivilova, T., & Bulakh, V. (2018). Machine learning in classification time series with fractal properties. Data, 4(1), 5.
4. Protopapadakis, E., & Doulamis, N. (2015, December). Image based approaches for tunnels' defects recognition via robotic inspectors. In International Symposium on Visual Computing (pp. 706-716). Springer, Cham.
5. Kirichenko, L., Radivilova, T., & Bulakh, V. (2019, May). Binary classification of fractal time series by machine learning methods. In International Scientific Conference “Intellectual Systems of Decision Making and Problem of Computational Intelligence” (pp. 701-711). Springer, Cham.
6. Shin, H. K., Ahn, Y. H., Lee, S. H., & Kim, H. Y. (2020). Automatic concrete damage recognition using multi-level attention convolutional neural network. Materials, 13(23), 5549.
7. Yi, L., Li, G., & Jiang, M. (2017). An end-to-end steel strip surface defects recognition system based on convolutional neural networks. steel research international, 88(2), 1600068.
8. Kirichenko, L., Zinchenko, P., & Radivilova, T. (2020, May). Classification of time realizations using machine learning recognition of recurrence plots. In International Scientific Conference “Intellectual Systems of Decision

Making and Problem of Computational Intelligence" (pp. 687-696). Springer, Cham.

9. Li, Z., Liu, F., Yang, W., Peng, S., & Zhou, J. (2021). A survey of convolutional neural networks: analysis, applications, and prospects. IEEE Transactions on Neural Networks and Learning Systems.
10. Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., ... & Fei-Fei, L. (2015). Imagenet large scale visual recognition challenge. International journal of computer vision, 115(3), 211-252.
11. Kirichenko, L., Radivilova, T., & Bulakh, V. (2020). Machine learning classification of multifractional Brownian motion realizations.
12. Kirichenko, L., Radivilova, T., Bulakh, V., Zinchenko, P., & Alghawli, A. S. (2020, August). Two approaches to machine learning classification of time series based on recurrence plots. In 2020 IEEE Third International Conference on Data Stream Mining & Processing (DSMP) (pp. 84-89). IEEE.
13. LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. nature, 521(7553), 436-444.
14. Minaee, S., Boykov, Y. Y., Porikli, F., Plaza, A. J., Kehtarnavaz, N., & Terzopoulos, D. (2021). Image segmentation using deep learning: A survey. IEEE transactions on pattern analysis and machine intelligence.
15. Xuan, X., Zhang, X., Kwon, O. H., & Ma, K. L. (2022). VAC-CNN: A Visual Analytics System for Comparative Studies of Deep Convolutional Neural Networks. IEEE Transactions on Visualization and Computer Graphics, 28(6), 2326-2337.
16. Django documentation. URL: https://docs.djangoproject.com/en/4.0/ (дата звернення 25.04.2022)
17. Python 3 documentation. URL: (дата звернення 26.04.2022)

18. TensorFlow documentation. URL: https://www.tensorflow.org/api_docs/python/tf/all_symbols (дата звернення 27.04.2022)
19. Manaswi, N. K. (2018). Understanding and working with Keras. In Deep Learning with Applications Using Python (pp. 31-43). Apress, Berkeley, CA.
20. Keras documentation. URL: https://keras.io/guides/ (дата звернення 27.04.2022)
21. Sinha, D., & El-Sharkawy, M. (2019, October). Thin mobilenet: An enhanced mobilenet architecture. In 2019 IEEE 10th annual ubiquitous computing, electronics & mobile communication conference (UEMCON) (pp. 0280-0285). IEEE.
22. NumPy documentation. URL: https://numpy.org/doc/stable/user/index.html#user (дата звернення 28.04.2022)
23. SciPy documentation. URL: https://docs.scipy.org/doc/scipy/tutorial/index.html#user-guide
24. Angular 6 documentation. URL: https://angular.io/docs
25. ImageAI documentation. URL: https://imageai.readthedocs.io/en/latest/
26. Степанов, Л. В. (2009). Моделювання конкуренції в умовах ринку: навч. посібник.