Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет Комп’ютерної інженерії та управління
Кафедра Комп’ютерних інтелектуальних технологій та систем

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

Рівень вищої освіти другий (магістерський)

Інтелектуальна система для розпізнавання
номерних знаків автомобілів за допомогою технології OCR
за наявності шуму

Виконав:
студент 2 курсу, групи КІТм-21-1
Омельченко С. О.
(прізвище, ініціали)

Спеціальність 123 «Комп’ютерна інженерія»
Тип програми освітньо-професійна
Освітня програма Комп’ютерні інтелектуальні технології
Керівник ст. викладач, к.т.н. Ілюнін О.О.
(посада, прізвище, ініціали)

Допускається до захисту _____
(підпис)

Зав. кафедри _____ О. Г. Руденко
(підпис)

2022 р.

Харківський національний університет радіоелектроніки

Факультет	Комп'ютерної інженерії та управління
Кафедра	Комп'ютерних інтелектуальних технологій та систем
Рівень вищої освіти	другий (магістерський)
Спеціальність	123 Комп'ютерна інженерія
Тип програми	освітньо-професійна
Освітня програма	Комп'ютерні інтелектуальні технології

ЗАТВЕРДЖУЮ:

Зав. кафедри _____
(підпис)

«_____»_____2022р.

ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ

студентові Омельченку Сергію Олександровичу
(прізвище, ім'я, по батькові)

1. Тема роботи (проекту) Інтелектуальна система для розпізнавання номерних знаків автомобілів за допомогою технології OCR за наявності шуму.

затверджена наказом по університету від « 7 » листопада 2022р. № 1455 Ст

2. Термін подання студентом роботи до екзаменаційної комісії 16 грудня 2022 р.

3. Вихідні дані до роботи (проекту)

1) Мова програмування Python
2) Бібліотеки OpenCV, NumPy, Imutils та easyocr
3) Середовище розробки – PyCharm
4) Документація по Angular та Flask
5) Зображення номерних знаків автомобілів
6) Література

4. Перелік питань, що потрібно опрацювати в роботі

1) Аналіз предметної області
2) Використання комп'ютерного зору для розпізнавання образів
3) Цифрове зображення. Шум зображення. Причини виникнення шуму
4) Технологій розробки та інструментальні засоби. Розробка додатку. Аналіз результатів експериментальних досліджень щодо роботи програми

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (п.5 включається до завдання за рішенням кафедри)

Слайд-презентація – 15 слайдів

6. Консультанти розділів роботи (п.6 включається до завдання за наявністю консультантів згідно до наказу, зазначеному у п.1)

Найменування розділу	Консультант (посада, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1	Ознайомлення з літературою	07.11.22-08.11.22	виконано
2	Аналіз публікацій за напрямком кваліфікаційної	09.11.22-10.11.22	виконано
	роботи		
3	Огляд існуючих рішень	11.11.22-13.11.22	виконано
4	Дослідження технологій комп'ютерного зору для	14.11.22-18.11.22	виконано
	розпізнавання образів		
5	Дослідження шуму зображення	21.11.22-23.11.22	виконано
6	Вибір технологій розробки та інструментальних	24.11.22-25.11.22	виконано
	засобів		
7	Розробка програми	28.11.22-30.11.22	виконано
8	Проведення експериментальних досліджень	01.12.22-02.12.22	виконано
9	Оформлення матеріалів кваліфікаційної роботи	03.12.22-05.12.22	виконано

Дата видачі завдання 7 листопада 2022 р.

Студент _____

Керівник роботи _____ ст. викладач, к.т.н. Ілюнін О.О.
(посада, прізвище, ініціали)

РЕФЕРАТ

Пояснювальна записка кваліфікаційної роботи: 115 сторінок, 27 рисунків, 1 таблиця, 2 додатки, 44 джерел.

Метою кваліфікаційної роботи є створення ПЗ для автоматичного розпізнавання зображення номерного знаку автомобіля за допомогою технології оптичного розпізнавання символів за наявності шуму.

В ході виконання кваліфікаційної роботи було виконано наступне:

- обґрунтовано актуальність обраної теми, оглянуто наукові публікації, які стосуються теми кваліфікаційної роботи та існуючі рішення на ринку;

- розглянуто технологію комп'ютерного зору, його історію виникнення, напрямки застосування, використання технології для розпізнавання образів, алгоритми розпізнавання образів, приклади використання. Зокрема було розглянуто технологію оптичного розпізнавання символів: коротку історію, принцип роботи, варіанти використання, переваги та недоліки;

- розглянуто існуючі типи цифрових зображень, типи шуму та причини його виникнення, засоби зменшення шуму на зображенні;

- описано технології розробки та інструментальні засоби, що використовувалися для створення веб-додатку;

- створено веб-додаток, що дозволяє накладати шум на вибране зображення або автоматично розпізнавати зображення номерного знаку автомобіля за наявності шуму;

- проведено експериментальні дослідження щодо роботи програми.

КОМП'ЮТЕРНИЙ ЗІР, РОЗПІЗНАВАННЯ ОБРАЗІВ, ОПТИЧНЕ РОЗПІЗНАВАННЯ СИМВОЛІВ, ЦИФРОВЕ ЗОБРАЖЕННЯ, ЦИФРОВИЙ ШУМ ЗОБРАЖЕННЯ, ФІЛЬТРИ ЗОБРАЖЕНЬ, ANPR, ANGULAR, PYTHON, FLASK, OPENCV, NUMPY, IMUTILS EASYOCR.

ABSTRACT

Master’s thesis: 15 presentation sheets, 115 pages, 27 figures, 1 table, 2 appendices, 44 reference sources.

The purpose of the qualification work is to develop software for automatic recognition of the car license plate image using optical character recognition technology in the presence of noise.

In the course of the qualification work, the following was performed:

- the relevance of the chosen topic was described, scientific publications related to the topic of the qualification work and existing solutions on the market were reviewed;

- considered what computer vision is, its history, directions of application, use of computer vision for pattern recognition, how pattern recognition works, which algorithms are used for this, examples of use. Optical character recognition technology, its brief history, principle of operation, options for use, advantages and disadvantages were also considered;

- considered what types of digital image exist, the causes of image noise and its types, as well as ways to reduce image noise;

- development technologies and tools used to create a web application are described;

- a web application was created that allows to add noise on the selected image or automatically recognize the image of a car license plate in the presence of noise;

- experimental studies of the program work were carried out.

COMPUTER VISION, IMAGE RECOGNITION, OPTICAL CHARACTER RECOGNITION, AUTOMATIC NUMBER LICENCE PLATE RECOGNITION, DIGITAL IMAGING, IMAGE NOISE, IMAGE FILTERS, ANGULAR, PYTHON, FLASK, OPENCV, NUMPY, IMUTILS EASYOCR.

Міністерство освіти і науки України

Харківський національний університет радіоелектроніки

Факультет Комп’ютерної інженерії та управління

Кафедра Комп’ютерних інтелектуальних технологій та систем

АНОТАЦІЯ

КВАЛІФІКАЦІЙНОЇ РОБОТИ

Інтелектуальна система для розпізнавання
номерних знаків автомобілів за допомогою технології OCR
за наявності шуму

Виконав:

студент 2 курсу, групи КІТм-21-1

Омельченко С. О.

(прізвище, ініціали)

Спеціальність 123 «Комп’ютерна інженерія»

Тип програми освітньо-професійна

Освітня програма Комп’ютерні інтелектуальні технології

Керівник ст. викладач, к.т.н. Ілюнін О.О.

(посада, прізвище, ініціали)

Харків –2022

АНОТАЦІЯ

Омельченко С. О. Інтелектуальна система для розпізнавання номерних знаків автомобілів за допомогою технології OCR за наявності шуму. – Магістерська кваліфікаційна робота.

Кожного року в Україні трапляється все більше ДТП. Так за даними статистики у 2021 році було зафіксовано приблизно 154 тисячі ДТП, що на 14% більше ніж за той же період у 2020 році. Для боротьби з правопорушниками кожного року на дорогах з'являється все більше камер автофіксації порушень правил дорожнього руху. Деякі дозволяють фіксувати перевищення швидкості, інші – проїзд на червоне світло, порушення правил зупинки чи стоянки, перетин подвійної суцільної або проїзд смугою для громадського транспорту, тротуарами, смугами зустрічного руху тощо. Вся ця інформація передається до Центру обробки даних Національної поліції, де вже професійний оператор обробляє пакет інформації. Така обробка інформації є ручною і може займати великий проміжок часу.

Інша проблема, яка з'явилися вже достатньо давно – ввезення автомобілів на європейських номерах до Україні, що фактично є незареєстрованою власністю. Відстеження таких автомобілів, а саме їх номерних знаків, може значно спростити виявлення правопорушників, які їздять на автомобілях з незареєстрованими номерними знаками.

Частково ці проблеми дозволяє вирішити одна з областей використання комп'ютерного зору для розпізнавання образів – оптичне розпізнавання символів. Але через недостатню кількість світла, погодні умови та інші ситуації, зображення набуває шумів і це ускладнює процес розпізнавання номерного знаку та впливає на точність.

Таким чином, метою кваліфікаційної роботи є створення ПЗ для автоматичного розпізнавання зображень номерних знаків автомобілів за допомогою технології OCR за наявності шуму.

Об’єктом дослідження є процеси розпізнавання зображень номерних знаків автомобілів за допомогою технології OCR за наявності шуму.

Предметом дослідження є нейро-нечіткі методи та моделі розпізнавання зображень, образів за наявності шуму, а саме – номерних знаків автомобілів за допомогою технології OCR за наявності шуму.

Методи та технології КЗ – це область ШІ, яка дозволяє комп'ютерам і системам вилучати значиму інформацію з цифрових зображень, відео та інших візуальних входів і вживати заходи або давати рекомендації на основі цієї інформації. Задачі КЗ включають методи отримання, обробки, аналізу та розуміння цифрових зображень, а також вилучення даних великої розмірності з реального світу для отримання числової або символьної інформації.

До основних напрямків використання КЗ відносяться реконструкція сцени, виявлення об’єктів та подій, 3D оцінка пози, відстеження руху, охорона здоров’я та інші.

РО – один з основних напрямків використання КЗ, що використовує комп’ютерні алгоритми для виявлення закономірностей даних. Напрям класифікує дані на основі статистичної інформації або знань, отриманих із шаблонів, та їх представлення. РО досягається з використанням концепції навчання. Навчання дозволяє системі РО навчатися та адаптуватися, щоб забезпечити більш точні результати. Частина набору даних використовується для навчання системи, а решта – для її тестування.

Основними прикладами використання КЗ для РО є прогноз фондового ринку, розпізнавання текстових образів, розпізнавання почерку, розпізнавання обличчя та візуальний пошук, розпізнавання голосу та емоцій.

Технологія OCR – це ще одна область використання КЗ для РО. Це рішення для автоматизації вилучення даних з друкованого або письмового тексту зі сканованого документа або файлу зображення, а потім перетворення тексту у машиночитану форму для використання в обробці даних, таких як редагування або пошуку. OCR використовують для перетворення друкованих паперових документів у машинозчитувані текстові документи, розпізнавання паспортів для аеропортів,

вилучення контактної інформації з документів або візиток, введення даних для ділових документів, розпізнавання дорожніх знаків, а також для автоматичного розпізнавання номерних знаків.

Для того, щоб розпізнати цифрове зображення, його спочатку необхідно отримати, а вже потім передати системі розпізнавання. Цифрове зображення зберігається у двійковій формі та розділене на матрицю пікселів. Кожен піксель складається з цифрового значення одного або кількох бітів, що визначається бітовою глибиною. Цифрове значення може представляти енергію, яскравість, колір, інтенсивність, звук, висоту або класифіковане значення, отримане за допомогою обробки зображення. Таким чином, група пікселів, регулярно розташованих у значущому порядку, передає зображення.

Цифрове зображення може бути представлено у вигляді бінарного, сірого або кольорового зображень. Бінарне зображення (дворівневе, двійкове) – різновид цифрових растрових зображень, коли кожен піксель може представляти лише один із двох кольорів. Значення кожного пікселя умовно кодуються як «0» та «1». Бінарне зображення також іноді називають однобітним, монохромним або чорно-білим. Часто для створення бінарного зображення застосовують такий метод сегментації зображення як порогово значення. Найпростіші методи порогового значення замінюють кожен піксель на зображенні чорним пікселем, якщо інтенсивність зображення $I_{i,j}$ менша за фіксоване значення, яке називається пороговим значенням T або білий піксель, якщо інтенсивність пікселів перевищує цей поріг.

Зображення в градаціях сірого (або рівень сірого) – це зображення, на якому єдиними кольорами є відтінки сірого. Причина відмінності таких зображень від будь-якого іншого різновиду кольорових зображень полягає в тому, що для кожного пікселя можна надати менше інформації. У сірому зображенні кожен піксель має значення від 0 до 255, де нуль відповідає «чорному», а 255 – «білому». Значення від 0 до 255 є різними відтінками сірого, де значення, ближчі до 0, темніші, а значення, ближчі до 255, світліші.

Різниця між бінарним зображення та зображенням в градаціях сірого полягає в тому, що бінарне зображення – це зображення, яке складається з пікселів, які

можуть мати один з двох кольорів, зазвичай чорний і білий, а зображення в градаціях сірого – це різновид чорно-білого або сірого монохрому, що складається виключно з відтінків сірого.

Можна побудувати майже всі видимі кольори, комбінуючи три основні кольори червоний, зелений і синій, тому що людське око має лише три різних колірних рецептора, кожен з яких чутливий до одного з трьох кольорів. Різні комбінації в стимуляції рецепторів дозволяють людському оку розрізняти приблизно 350000 кольорів. Кольорове RGB-зображення – це багатоспектральне зображення з однією смугою для кожного кольору червоного, зеленого та синього, що створює зважене поєднання трьох основних кольорів для кожного пікселя. Повне 24-бітне кольорове зображення містить одне 8-бітне значення для кожного кольору, таким чином дає змогу відображати $2^{24} = 16777216$ різні кольори. Однак використання повного 24-розрядного зображення для збереження кольору для кожного пікселя є дорогим з точки зору обчислень і часто не обов’язкове.

Дуже часто на цифровому зображенні виникає різноманітний шум. Шум зображення складається з випадкових варіацій інформації про яскравість або колір на знімку. До основних джерел шуму в цифрових зображеннях відносяться слабке освітлення, режими високої чутливості, низька роздільність датчиків, фактори навколишнього середовища, частинки пилу в сканері, перешкоди каналу передачі. Так було виявлено, що до основних типів шуму відносяться: шум Гауса, шум «солі та перцю», спекл-шум, шум Пуассона, шум квантування, зернистість фотоплівки, анізотропний та періодичний шуми.

Для зменшення шуму зображення застосовують методи, кожен з яких використовує різні математичні та статистичні моделі. Вони широко класифікуються як класичні методи зменшення шуму (методи просторової області), методи перетворення та методи, засновані на МН та CNN. Також для зменшення шуму застосовуються різноманітні фільтри зображення. В ході виконання кваліфікаційної роботи було виявлено, що для зменшення шуму Гауса або спекл-шуму доцільно використовувати фільтр Гауса. Медіанний та консервативний

фільтри можна використовувати для зменшення шуму «солі та перцю», а двосторонній фільтр доцільно використовувати при наявних шумах Пуассона.

Результатом виконання кваліфікаційної роботи є розроблений веб-додаток, який дозволяє накладати шум на вибране зображення або автоматично розпізнавати зображення номерного знаку автомобіля за допомогою технології оптичного розпізнавання символів за наявності шуму.

Для реалізації клієнтської частини програми використано фреймворк Angular, основною метою якого є розробка односторінкових додатків. Angular надає таку функціональність як двостороннє зв'язування, що дозволяє динамічно змінювати дані в одному місці інтерфейсу при зміні даних моделі в іншому. Однією з ключових особливостей Angular є те, що він використовує мову програмування TypeScript.

Для реалізації серверної частини програми було використано високорівневу мову програмування Python та її мікрофреймворк Flask для створення легких, гнучких та масштабованих веб-додатків. Його простота у використанні та невелика кількість залежностей дозволяють працювати безперебійно, навіть якщо додаток масштабується все більше і більше.

За результатами роботи можна зробити висновок, що програма з високою точністю та швидкістю розпізнає зображення номерних знаків автомобілів за наявності шуму, за умов чіткого загального зображення та розташування номерного знаку на передньому плані.

Поставлені задачі кваліфікаційної роботи виконані в повному обсязі. Можливим варіантом подальшого розвитку проекту є створення згорткових нейронних мереж для виявлення номерних знаків автомобілів на зображенні та зменшення шуму на цьому зображенні до рівня зумовленого завданням.

КОМП’ЮТЕРНИЙ ЗІР, РОЗПІЗНАВАННЯ ОБРАЗІВ, ОПТИЧНЕ РОЗПІЗНАВАННЯ СИМВОЛІВ, АВТОМАТИЧНЕ РОЗПІЗНАВАННЯ НОМЕРНИХ ЗНАКІВ, ЦИФРОВЕ ЗОБРАЖЕННЯ, ЦИФРОВИЙ ШУМ ЗОБРАЖЕННЯ, ФІЛЬТРИ ЗОБРАЖЕНЬ, ANGULAR, PYTHON, FLASK.

Використані в роботі публікації керівника та співробітників кафедри:

1. Bezsonov O., Ilyunin O., Khusanov A., Rudenko O., Oleg Sotnikov O. Intelligent Identification System of the Process Liquid Solutions Composition. Proceedings of the 6th International Conference on Computational Linguistics and Intelligent Systems (COLINS 2022). Volume I: Main Conference, Gliwice, Poland, May 12-13, 2022. P.960–973.

Публікації здобувача за темою роботи:

1. Омельченко С. О., Сердюк Н. М. Використання комп'ютерного зору для розпізнавання образів. Глобалізація наукових знань: міжнародна співпраця та інтеграція галузей наук: матеріали II Міжнародної студентської наукової конференції (Т. 1), м. Біла Церква, 22 жовтня, 2021 рік / ГО «Молодіжна наукова ліга». – Вінниця: ГО «Європейська наукова платформа», 2021. – 130 с. URL: https://ojs.ukrlogos.in.ua/index.php/liga/issue/view/23.10.2021/618.

2. Омельченко С.О., Сердюк Н.М. Використання технології OCR для автоматичного розпізнавання європейських номерних знаків. Цифровізація науки та сучасні тренди її розвитку: матеріали II Міжнародної студентської наукової конференції (Т. 1), м. Миргород, 5 листопада, 2021 рік / ГО «Молодіжна наукова ліга». – Вінниця: ГО «Європейська наукова платформа», 2021. – 116 с. URL: https://ojs.ukrlogos.in.ua/index.php/liga/issue/view/inter-05.11.2021/629.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

ГН – глибоке навчання

КЗ – комп’ютений зір

МН – машинне навчання

НМ – нейронна мережа

ПЗ – програмне забезпечення

РО – розпізнавання образів

ШІ – штучний інтелект

OCR – оптичне розпізнавання символів (англ., Optical Character Recognition)

CNN – згоркова нейронна мережа (англ., Convolutional Neural Network)

ANPR – автоматичне розпізнавання номерних знаків (англ., Automatic Number Plate Recognition)

ВСТУП

В наш час вже неможливо уявити життя людини без комп’ютера. За останні роки цей пристрій проник майже у всі сфери людської діяльності. З розвитком елементної бази, інформаційних технологій, інтелектуальних систем, комп’ютери стали використовуватися для розв’язання все більш складних науково-технічних задач, які раніше були притаманні лише людині, а зараз можуть вирішуватися і без її участі. Однією з таких задач є використання КЗ для РО.

Основну частину інформації людина і більшість тварин отримують за допомогою зору. Зоровий канал в декілька разів надає більше корисної інформації про навколишній світ, ніж інші органи чуття. За долю секунди людина без роздуму і безпомилково може визначити різноманітні об’єкти, що знаходяться в полі зору. При цьому людина здатна не тільки назвати об’єкти, що вона бачить, але й визначити їх розмір, колір, об’єм, форму, розпізнати контури і визначити об’єкти, що розташовані на задньому плані. Це все ті навички, які необхідні системі комп’ютерного зору.

Таким чином, основну проблему, що вирішується КЗ, можна сформувати наступним чином: з огляду на двовимірне зображення, система КЗ повинна розпізнавати існуючі об'єкти і їх характеристики, такі як форми, текстури, кольори, розміри, просторове розташування серед інших об’єктів, щоб забезпечити якомога більш повний опис зображення.

На даний момент комп’ютерні методи, що імітують людський зір, використовуються не тільки для виявлення, відстежування і класифікації об’єктів, але й для таких задач як розпізнавання тексту, ідентифікації предметів і людей, відновлювання зображення, оцінки руху, автопілотування, а також діагностики захворювань. Однак імітація роботи людських органів зору виявляється завданням не з легких – те що людина може розпізнати за долю секунди для сучасного комп’ютера може складати декілька секунд, хвилин або навіть годин, так як там де людина автоматично розпізнає контури, лінії і об’єкти, комп’ютер бачить лише

великі масиви даних, які необхідно постійно обчислювати. Інша проблема, яка ускладнює завдання розпізнавання – це наявність різноманітного шуму на об’єкті. Так, людина може навіть не помітити наявність шуму і чітко визначити, що зображено, а для комп’ютера це буде лише додаткова некоректна інформація, що впливатиме на процес розпізнавання і псувати точність роботи програми.

Таким чином, метою кваліфікаційної роботи є створення ПЗ для автоматичного розпізнавання зображень номерних знаків автомобілів за допомогою технології OCR за наявності шуму.

Об’єктом дослідження є процеси розпізнавання зображень номерних знаків автомобілів за допомогою технології OCR за наявності шуму.

Предметом дослідження є нейро-нечіткі методи та моделі розпізнавання зображень, образів за наявності шуму, а саме – номерних знаків автомобілів за допомогою технології OCR за наявності шуму.

В результаті досліджень буде розроблено клієнт-серверну програму за допомогою фреймворку Angular для клієнтської частини та мови програмування Python і мікрофреймворку Flask для серверної, яка на вхід методу-контролера прийматиме зашумлене зображення, що містить номерний знак автомобіля, послідовно виконуватиме операції для оптимізації зображення та зменшення шуму на ньому, знаходитиме контур номерного знаку та розпізнаватиме текст, що зображений на ньому, а також повертатиме оброблене зображення назад користувачу.

1 АКТУАЛЬНІСТЬ ЗАДАЧІ ТА АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Актуальність обраної теми

Кожного року в Україні трапляється все більше ДТП. Так за даними статистики у 2021 році було зафіксовано приблизно 154 тисячі ДТП, що на 14% більше ніж за той же період у 2020 році. Для боротьби з правопорушниками кожного року на дорогах з’являється все більше камер автофіксації порушень правил дорожнього руху. Деякі дозволяють фіксувати перевищення швидкості, інші – проїзд на червоне світло, порушення правил зупинки чи стоянки, перетин подвійної суцільної або проїзд смугою для громадського транспорту, тротуарами, смугами зустрічного руху тощо. Вся ця інформація передається до Центру обробки даних Національної поліції, де вже професійний оператор обробляє пакет інформації. Така обробка інформації є ручною і може займати великий проміжок часу.

Достатньо давня проблема – ввезення автомобілів на європейських номерах до Україні, що фактично є незареєстрованою власністю і відстеження таких автомобілів, а саме їх номерних знаків, може значно спростити відстеження правопорушників, які їздять на автомобілях з незареєстрованими номерними знаками.

Технологія OCR – одна з областей використання комп’ютерного зору для РО, яка дозволяє частково вирішити ці проблеми. Але існують ситуації, із-за яких зображення номерного знаку, яке необхідно розпізнати, набуває шумів і це ускладнює процес розпізнавання та впливає на його точність. Тому існує необхідність визначення причин виникнення шуму на зображенні та шляхи його усунення для якіснішого розпізнавання.

1.2 Огляд наукових публікацій та існуючих рішень

Використання КЗ та технологій OCR для ANPR є сучасною, актуальною темою. Для аналізу предметної області обрано та розглянуто деякі публікації та готові популярні рішення, інформація про які буде взята за основу для виконання кваліфікаційної роботи.

У «Automatic License Plate Recognition (ALPR): A State-of-the-Art Review» [1] розкрито загальні положення про ANPR автомобілів, етапи, з яких складається розпізнавання та методи, що можуть використовуватися на кожному етапі. Виділено чотири етапи з яких складається алгоритм розпізнавання номерних знаків:

- отримання зображення автомобіля за допомогою камери;
- вилучення номерного знаку із зображення на основі деяких ознак, таких як межі, колір або наявність символів;
- сегментація номерного знаку і виділення символів, спроектувавши інформацію про їхній колір, позначивши їх або зіставивши їхні позиції з шаблонами;
- розпізнавання вилучених символів шляхом зіставлення шаблонів або використання класифікаторів, таких як НМ та нечіткі класифікатори.

Для вилучення номерного знаку із зображення було запропоновано декілька методів, кожен із яких має свої переваги та недоліки:

- метод виявлення номерного знаку за допомогою інформації про межі/краї – оскільки номерний знак зазвичай має прямокутну форму з відомим співвідношенням сторін, його можна отримати, знайшовши всі можливі прямокутники на зображенні. Цей метод є найпростіший, найшвидший та зрозумілий, але його не застосовують до складних зображень, оскільки вони надто чутливі до небажаних країв;
- метод заснований на визначенні символів номерного знаку – перевіряє зображення на наявність символів і якщо символи знайдено, їх регіон витягується як регіон номерного знаку. Недоліками є те, що цей метод займає багато часу (обробка всіх бінарних об'єктів) та продукує помилки виявлення іншого тексту на зображенні.

Для сегментації номерного знаку були запропоновані наступні методи:

- сегментація номерних знаків за допомогою підключення пікселів – сегментація виконується шляхом позначення з'єднаних пікселів у бінарному зображенні номерного знаку. Мічені пікселі аналізуються, і ті, які мають однаковий розмір і співвідношення сторін символів, вважаються символами номерного знаку. Цей метод не ідентифікує символи, якщо вони об'єднані або розбиті;

- сегментація номерних знаків за допомогою проекційних профілів – оскільки символи та фон номерного знаку мають різні кольори, вони мають протилежні двійкові значення у двійковому зображенні. Такі методи проектують двійковий вилучений номерний знак вертикально, щоб визначити початкову та кінцеву позиції символів, а потім проектують вилучені символи горизонтально, щоб виділити кожен символ окремо.

Практичну частину реалізації розпізнавання номерних знаків автомобілів у реальних умовах за допомогою CNN описано «A CNN-Based Approach for Automatic License Plate Recognition in the Wild» [2]. Авторами запропоновано ALPR систему розпізнавання зображень, що зроблені в дикій природі, мають складний фон з різними типами текстових блоків, а номерні знаки займають лише дуже невелику їх частину. Система ALPR приймає на вхід довільне зображення та виводить розпізнані номерні знаки. Система навчалася та оцінювалася на наборі даних номерних знаків із більш ніж 18000 зображень.

На етапі виявлення використовується каскадна структура, що складається з RPN (мережа пропозицій регіонів) та класифікатора R-CNN (CNN на основі регіонів). RPN – повністю згорткова мережа, яка одночасно передбачає межі об'єктів і показники об'єктності в кожній позиції. RPN навчена наскрізною для створення високоякісних регіональних пропозицій.

Полегшена мережа RPN працює швидко та ефективно. Вона приймає зображення зі зниженою частотою дискретизації як вхідні дані та генерує кандидатів на номерні знаки. Як тільки область інтересу запропонована, відповідний фрагмент зображення вирізається з вихідного зображення з високою роздільною здатністю. R- CNN виконує не тільки класифікацію, а й регресує чотири кутові

положення для кожного номерного знаку. Положення кутів використовуються для розрахунку матриці афінного перетворення для виправлення.

Розпізнавання номерних знаків досягається за допомогою паралельної мережі просторового перетворення (STN) із загальними ваговими класифікаторами. STN неявно виконує сегментацію символів та фокусується на пікселях зображення кожного символу. Зрештою, для розпізнавання кожного символу використовується класифікатор загальної ваги. Інноваційний розпізнавач загальної ваги значно зменшує розмір моделі і, що важливіше, краще використовує обмежені навчальні дані.

Результати тестування показують, що детектор працює краще, ніж швидший R- CNN (VGG), який в 1,5 рази повільніший при тестуванні і в 57 разів більший за розміром моделі. Детектор також є значно кращим, ніж існуючі рішення, зменшуючи 57,5% помилок сучасної схеми кодування послідовності символів.

Одним з існуючих рішень на ринку для розпізнавання номерних знаків автомобілів у реальному часі є система камер Axis для захоплення номерного знаку і подальшого його розпізнавання [3]. Захоплення номерних знаків (LPC) – це здатність камери знімати зображення номерних знаків, які можна прочитати. Це необхідна умова для розпізнавання номерних знаків (LPR), коли номерні знаки автоматично знаходяться і зчитуються за допомогою аналітичного ПЗ.

Спеціалізовані камери Axis LPC/LPR розроблено з урахуванням жорстких умов дорожнього руху. Компоненти підібрані таким чином, щоб витримувати погану погоду, сильний вітер і перепади температур.

Для вдалого захоплення номерного знаку використовується таке поняття як дальність виявлення – це діапазон відстаней уздовж дороги, де номерний знак видно та його можна прочитати на зображенні. В ідеалі діапазон виявлення – це повне поле зору камери, але це не завжди так. Діапазон виявлення може бути обмежений глибиною різкості камери, а транспортні засоби, які знаходяться далеко, іноді занадто малі, щоб датчик зображення добре розрізняв їх. Рекомендована мінімальна відстань захоплення залежить від швидкості транспортних засобів. Так для камер

Axis мінімальна рекомендована відстань захоплення становить 4 метри, при швидкості автомобіля 10 км/г та 30 метрів, при швидкості 130 км/г.

Після захоплення відеопотоку номерних зображень знаків необхідне спеціальне аналітичне ПЗ, щоб відокремити номерні знаки з залишкового фону зображень. Камери Axis за замовченням не мають такого забезпечення, але мають можливість додатково встановити його.

ПЗ LPR може працювати безпосередньо на камері або на віддалених серверах. Запуск ПЗ LPR на віддаленому сервері може забезпечити велику обчислювальну потужність, але для цього потрібна потокова передача відео у віддалену локацію, що вимагає достатньої пропускної здатності мережі. Масштабувати серверну систему для багатьох камер досить проблемно, оскільки обробка великої кількісті відеопотоків швидко засмічує мережу.

Запуск ПЗ LPR безпосередньо на камері означає, що лише літери та цифри номерних знаків потрібно надіслати з камери на центральний сервер. Це мінімізує вимоги до пропускної здатності мережі. Подібну розподілену систему легко масштабувати, оскільки додавання нової камери не вимагає додавання інших ресурсів до системи.

Недоліком використання алгоритмів LPR на камері є обмежена потужність обробки, через що аналіз кожного зображення займає більше часу. Він обмежує максимальну роздільну здатність, яку можна використовувати, що обмежує кількість смуг, які може охоплювати кожна камера.

Камери Axis оснащені помічником із захоплення номерних знаків – це функція, розроблена компанією, щоб допомогти правильно вирівняти та налаштувати камеру. Асистент захоплення номерного знаку автоматично надає відгук, поки користувач вирівнює камеру. Це можливо завдяки тому, що камера може вимірювати свою орієнтацію в полі тяжіння. Помічник постійно відюбражує вертикальний, горизонтальний та кут повороту, і відображує їх із попередженням, якщо вони завеликі.

Компанія FF Group – це ще одна компанія, яка існує на ринку і надає спеціальне ПЗ «NumberOk» для автоматичного розпізнавання номерних знаків

автомобілів, а також марки, моделі, кольору та типу автомобіля для контролю доступу та трафік менеджменту [4]. ПЗ працює з будь-якими IP камерами та аналоговими відеореєстраторами за протоколом RTSP.

ПЗ встановлене у системі відеоспостереження виконує розпізнавання, облік та аналіз транспортних засобів, забезпечених державним реєстраційним номером. Вхідними даними є кадри з відеопотоку. Може бути особливо корисним на таких об'єктах, як автостоянки, парковки, автомийки, резиденції та готелі, а також на контрольно-пропускних пунктах, де існує підвищена ймовірність скоєння злочину. Також ПЗ можна встановити і на робочий комп’ютер.

Розпізнаний номер, модель, марка, тип, колір та саме зображення автомобіля, час розпізнавання, напрям руху зберігаються в базі даних, щоб в подальшому була можливість передивитись детальну інформацію. Система захоплює декілька кадрів з одним номером і потім вибирається кращий результат, що зберігається в базі даних.

При налаштуванні системи можна вказати номери, виявлення яких викликає подачу сигналу тривоги. Користувач може створювати «чорний» та «білий» списки номерів.

Програма забезпечує рекордний час і точність розпізнавання. Обробка номера на комп’ютері займає всього 10-100 мс, у вбудованому процесорі камери воно вимагатиме від 100 мс. Імовірність правильного розпізнавання становить 95%.

1.3 Мета та задачі кваліфікаційної роботи

Метою кваліфікаційної роботи є створення ПЗ для автоматичного розпізнавання зображення номерного знаку автомобіля за допомогою технології OCR за наявності шуму.

Для досягнення поставленої мети необхідно вирішити наступні завдання:

- розглянути технологію КЗ, принцип роботи та напрямки його застосування;
- дослідити технологію OCR, її переваги та недоліки, основні сфери використання;

- дослідити, що таке шум зображення, причини виникнення, його види та засоби усунення;

- проаналізувати технології розробки та інструментальні засоби для реалізації поставленого завдання.

Розв’язання цих задач дозволить створити ПЗ, яке буде автоматично розпізнавати зображення номерного знаку автомобіля за наявності шуму.

Алгоритм роботи програми повинен бути наступним.

Користувач обирає зображення та операцію, яку необхідно здійснити над зображенням: накласти шум на вибране зображення або розпізнати текст номерного знаку автомобіля на зображенні. Існує можливість обрати різний тип шуму для накладання на зображення: Гаусовий шум, шум «солі та перцю», шум Пуассона або спекл-шум.

На вхід головної функції програми надходить зображення і тип операції виконання.

Якщо було обрано операцію накладання шуму на зображення, то викликається метод для накладання шуму в залежності від обраного типу. Користувачу повертається початкове зображення з накладеним шумом.

Якщо було обрано операцію розпізнавання номерного знаку автомобіля на зображенні, то відбувається виконання послідовності методів для маніпуляцій з переданим зображенням: виконання оптимізації зображення та зменшення шуму на ньому, виділення країв та контурів на зображенні, пошук контурів номерного знаку автомобіля, використання побітових операцій та масок для виділення з загального зображення форми знайденого контуру номерного знаку, вирізання контуру номерного знаку, збільшення зображення номерного знаку та розпізнавання самого тексту на номері. В результаті користувачу повертається початкове зображення з виділеним контуром номерного знаку автомобіля та текст, що було розпізнано на ньому.

2 КОМП'ЮТЕРНИЙ ЗІР ТА РОЗПІЗНАВАННЯ ОБРАЗІВ

2.1 Коротка історія комп'ютерного зору

Вчені та інженери близько 60 років намагалися розробити засоби, щоб комп'ютери могли бачити та розуміти візуальні дані. Експеримент розпочався в 1959 році, коли нейрофізіологи показали кішці масив зображень, намагаючись скорегувати відповідь у її мозку. Вони виявили, що мозок спочатку реагує на різкі краї або лінії, і з наукової точки зору це означало, що обробка зображення починається з простих форм, таких як прямі краї.

Приблизно в той же час була розроблена перша технологія сканування комп'ютерних зображень, що дозволяє комп'ютерам оцифровувати та отримувати зображення. Інший етап був досягнутий у 1963 році, коли комп'ютери змогли перетворити двовимірні зображення у тривимірні форми. У 1960 роках ШІ став академічним напрямком дослідження, а також поклав початок пошукам штучного інтелекту щодо вирішення проблеми зору людини.

У 1974 році було введено технологію OCR, яка могла розпізнавати текст, надрукований будь-яким шрифтом або гарнітурою. Аналогічно, інтелектуальне розпізнавання символів (ICR) могло розшифрувати рукописний текст за допомогою НМ. З тих пір, OCR та ICR знайшли свій шлях в обробці документів та рахунків, розпізнаванні номерних знаків автомобілів, мобільних платіжках, машинного перекладу та інших поширених додатків.

У 1982 році невролог Девід Марр встановив, що зір працює ієрархічно, і ввів алгоритми машин для виявлення країв, кутів, кривих та подібних основних форм. Одночасно комп'ютерний учений К. Фукусіма розробив мережу «клітин», які могли б розпізнавати закономірності. Мережа, яка називається Neocognitron, включала згорткові шари в НМ.

До 2000 року у центрі уваги було розпізнавання об'єктів, і у 2001 року з'явилися перші програми розпізнавання облич у реальному часі. У 2010 році набір

даних ImageNet став доступним. Він містив мільйони позначених зображень у тисячі класів об’єктів і забезпечує основу для CNN та моделей ГН, які використовуються сьогодні [5].

2.2 Комп’ютерний зір та напрямки його застосування

КЗ – це напрям досліджень ШІ, який дозволяє комп'ютерам і системам вилучати значиму інформацію з цифрових зображень, відео та інших візуальних входів і вживати заходи або давати рекомендації на основі цієї інформації [6,7]. Задачі КЗ включають методи отримання, обробки, аналізу та розуміння цифрових зображень та вилучення даних великої розмірності з реального світу для отримання числової або символьної інформації.

До основних напрямків використання КЗ можна віднести:

- реконструкція сцени – це процес реконструкції цифрової версії об’єкта реального світу із зображень або сканів об’єкта. Маючи одне або (зазвичай) кілька зображень сцени або відео, реконструкція сцени спрямована на обчислення 3D-моделі сцени. У найпростішому випадку моделлю може бути набір 3D точок. Більш складні методи створюють повну 3D модель поверхні. Поява 3D-зображень, які не потребують руху чи сканування, і пов’язаних з ними алгоритмів обробки дає змогу швидко розвиватися в цій галузі;

- виявлення об’єктів – це технологія, яка дозволяє ідентифікувати та знаходити об’єкти на зображенні чи відео. За допомогою такого типу ідентифікації та локалізації виявлення об’єктів можна використовувати для підрахунку об’єктів у сцені та визначення та відстеження їхнього точного розташування, при цьому точно позначаючи їх [8];

- виявлення подій – це сфера використання КЗ, яка аналізує вхідне відео з метою визначення того, коли сталася певна аномальна подія. Моделі ШІ для виявлення подій можна навчити на широкому діапазоні потенційних подій, від розпізнавання бійок і дорожньо-транспортних пригод до ідентифікації падінь, диму та вогню [9];

- 3D оцінка пози – застосування алгоритми ГН, щоб відстежувати відомі пози людини від сидячих до стоячих у відеокадрах;

- відстеження руху – це сфера КЗ, яке намагається стежити за рухом людини чи об’єкта в кількох кадрах у відео. Це розширене розпізнавання об’єктів. У розпізнаванні об’єктів завдання моделі ШІ – ідентифікувати різні об’єкти на зображенні. Відстеження руху залежить від розпізнавання об’єктів, щоб знайти об’єкти та їхнє початкове положення, а потім стеження за їхнім рухом протягом усього відео, розпізнаючи їх як ті самі об’єкти [10];

- охорона здоров’я – сфера охорони здоров'я використовує КЗ для широкого спектру застосувань [11]:

1) вивчення радіологічних зображень для діагностики медичних проблем від раку до COVID-19;

2) перевірка дотримання стандартів охорони здоров’я медичним персоналом: (чи миє руки, чи носить маски);

3) автоматичне створення журналів операцій шляхом виявлення етапів хірургічної процедури, від анестезії до використання хірургічних інструментів;

4) ідентифікація пацієнтів за допомогою високоточних моделей автентифікації обличчя, зменшення медичних помилок.

Також КЗ використовується для відеоспостереження, навчання, індексування, візуального обслуговування, моделювання 3D-сцен та відновлення зображення та іншого.

2.3 Розпізнавання образів за допомогою комп’ютерного зору

РО – один з основних напрямків використання КЗ, що використовує комп’ютерні алгоритми для виявлення закономірностей даних. РО класифікує дані на основі статистичної інформації або знань, отриманих із шаблонів, та їх представлення [12].

Алгоритми РО мають такі особливості:

- висока точність розпізнавання;

- можливість розпізнавання незнайомих об’єктів;
- розпізнавання предметів з різних ракурсів;
- відновлення шаблонів в разі відсутності даних;
- може виявляти шаблони, які частково приховані.

2.3.1 Як працює розпізнавання образів

РО досягається з використанням концепції навчання. Навчання дозволяє системі РО навчатися та адаптуватися, щоб забезпечити більш точні результати: так частка набору даних використовується для навчання системи, а решта – для її тестування [13]. На рис. 2.1 показано використання даних для навчання та тестування.

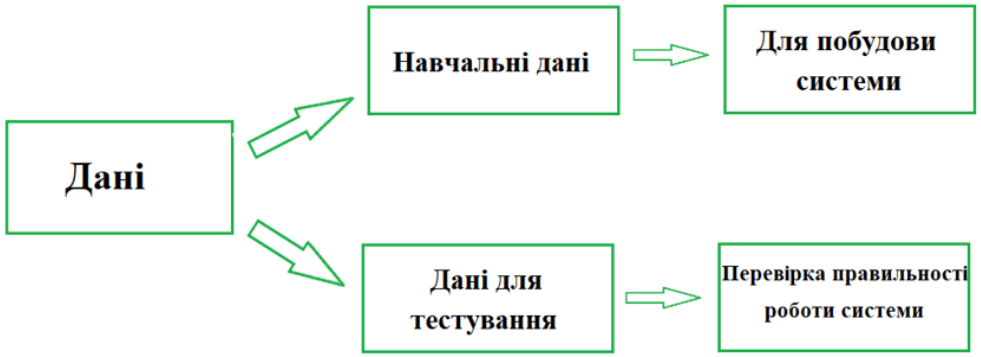


Рисунок 2.1 – Використання даних для навчання та тестування

Навчальний набір містить зображення або дані, які використовуються для навчання або побудови моделі. Правила навчання використовуються для забезпечення критеріїв вихідних рішень.

Навчальні алгоритми використовуються для узгодження заданих вхідних даних з відповідним вихідним рішенням. Потім алгоритми та правила застосовуються для полегшення навчання. Система використовує інформацію, зібрану з даних, для отримання результатів. Тестовий набір даних використовується

для перевірки точності системи. Дані тестування використовуються для перевірки того, чи досягається точний результат після навчання системи. Ці дані становлять приблизно 20% усіх даних системи РО.

Процес РО працює в п'яти основних фазах:

а) виявлення – на цьому етапі система РО перетворює вхідні дані в аналогічні дані, які систематизуються;

б) сегментація – ця фаза забезпечує ізоляцію виявлених об'єктів;

в) вилучення ознак – на цьому етапі обчислюються характеристики чи властивості об’єктів та відбувається їх відсилання для подальшої класифікації;

г) класифікація – на цьому етапі виявлені об'єкти класифікуються або розміщуються у кластерні групи;

д) післяобробка – до прийняття рішення приймаються додаткові міркування.

Нижче наведено деякі алгоритми, що використовуються для РО [14].

2.3.2 Алгоритми розпізнавання образів

2.3.2.1 Розпізнавання статистичних образів

Серед традиційних підходів РО статистичний підхід найбільш інтенсивно вивчався і застосовувався на практиці задовго до того, як методи НМ стали популярними. При статистичному розпізнаванні образ групується відповідно до його ознак, а кількість ознак d визначає, як образ розглядатиметься точкою у d-вимірному просторі. Ці функції вибрано таким чином, щоб різні образи займали місце без накладання. Метод працює так, що обрані атрибути допомагають створювати кластери. Машина навчається та адаптується, як очікується, а потім використовує образи для подальшої обробки та навчання.

2.3.2.2 Розпізнавання синтаксичних шаблонів

Синтаксичне розпізнавання шаблонів використовується для розпізнавання проблем зі складними образами, які можна вирішити, прийнявши ієрархічну перспективу. Підхід з використанням синтаксичного образу спирається на примітивних підобразах (наприклад, літери алфавіту). Образ описується залежно від того, як примітиви взаємодіють між собою. Прикладом цієї взаємодії є те, як вони зібрані в слова та речення. Наведені зразки навчання показують, як розробляються граматичні правила і як пізніше речення будуть «прочитані». Синтаксичний підхід може призвести до комбінаторного вибуху ймовірностей, які потрібно дослідити, що вимагає великих навчальних наборів та дуже великих обчислювальних зусиль.

2.3.2.3 Відповідність образу

Метод відповідності образу один з найпростіших і ранніх підходів до РО. Моделі відповідності образів намагаються виявити подібність між двома сутностями одного типу: у зразку на основі еталонного образу. Таким чином, техніка відповідності образу зазвичай використовується в цифровій обробці зображень для виявлення невеликих ділянок зображення, які відповідають зображенню шаблону. Типовими реальними прикладами є обробка медичних зображень, контроль якості у виробництві, навігація роботів або розпізнавання обличчя.

2.3.2.4 Розпізнавання образів нейронною мережею

РО за допомогою НМ, що представлено на рис. 2.2, на даний момент є найпопулярнішим методом виявлення образів. НМ базуються на паралельних субодиницях, що називаються нейронами, які імітують процес прийняття рішень людиною. НМ можна розглядати як масово паралельні обчислювальні системи, що складаються з величезної кількості простих процесорів з безліччю взаємозв’язків (нейронів). Найбільш популярною та успішною формою МН з використанням НМ є глибоке навчання, яке застосовує глибокі CNN для вирішення завдань класифікації.

Сьогодні РО з використанням НМ має перевагу над іншими методами, оскільки може неодноразово змінювати ваги на ітераційних образах.

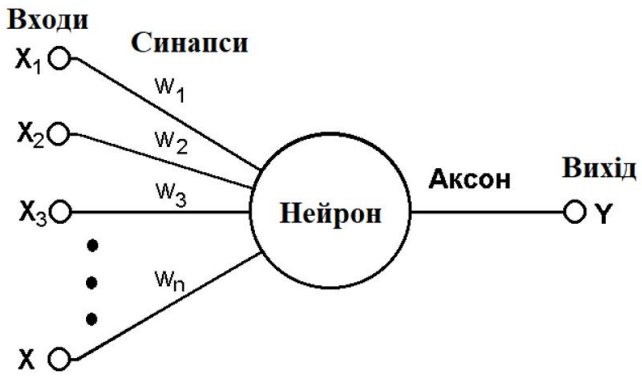


Рисунок 2.2 – Концепція нейронної мережі для виявлення закономірностей у даних

2.3.2.5 Нечіткі алгоритми

В основі нечітких алгоритмів застосовується концепція нечіткої логіки, яка використовує значення істинності від 0 до 1. У нечіткій моделі застосовується набір правил для відповідності певного набору входу відповідним результатам виходу. Ця модель дає високі результати, оскільки підходить для невизначених областей [15].

2.3.2.6 Гібридні методи розпізнавання образів

Багато проектів РО базуються на гібридних моделях для підвищення продуктивності розпізнавання для конкретних випадків використання, залежно від типу та наявності даних. Наприклад, методи ГН досягають значних результатів, але є досить затратними, тоді як «легші» математичні методи зазвичай є більш ефективними. Використання гібридної моделі підвищить продуктивність усієї програми або системи виявлення.

2.3.3 Приклади використання комп’ютерного зору для розпізнавання образів

До основних прикладів використання КЗ для РО можна віднести:

- прогноз фондового ринку – класичне складне завдання з метою оцінки майбутньої вартості акцій компанії чи інших активів, що торгуються;

- розпізнавання текстових образів – РО на основі МН використовується для генерації, аналізу та перекладу тексту. Образи використовуються для розуміння людської мови та створення текстових повідомлень. Розпізнавання тексту в словах використовується для класифікації документів та автоматичного виявлення конфіденційних текстових уривків. Тому розпізнавання текстових шаблонів використовується у галузях фінансів та страхування для виявлення шахрайства;

- розпізнавання почерку – розпізнавання рукописного тексту використовується для порівняння шаблонів у рукописному тексті або підписах для ідентифікації шаблонів. Для комп'ютерного розпізнавання рукописних слів задіяні різні програми. Однак розпізнавання та виділення рукописних слів є складним полем, оскільки рукописний текст містить неправильні та складні форми;

- розпізнавання обличчя та візуальний пошук – алгоритми розпізнавання зображень спрямовані на виявлення закономірностей у візуальних образах для розпізнавання конкретних об'єктів. Типовим завданням розпізнавання зображень є класифікація зображень, яка використовує НМ для позначення зображення або сегмента зображення на основі зображеного. Це основа візуального пошуку, де користувачі можуть легко шукати та порівнювати позначені зображення;

- розпізнавання голосу – системи розпізнавання голосу дозволяють машинам приймати та інтерпретувати диктування або виконувати голосові команди та відповідно взаємодіяти. Розпізнавання мовлення базується на МН, що дозволяє розпізнавати та перекладати розмовну мову;

- системи розпізнавання емоцій – МН застосовується до зображень або відеоматеріалів для аналізу та виявлення людських емоцій аудиторії. Мета – вказати настрій, думку та наміри аудиторії чи клієнтів. Ці знання використовуються для покращення маркетингових компаній та досвіду клієнтів.

2.3.4 Переваги розпізнавання образів

Методи РО надають різні переваги залежно від застосування. Загалом, пошук закономірностей у даних допомагає аналізувати та прогнозувати майбутні тенденції або розробляти системи раннього попередження на основі конкретних індикаторів шаблону. До додаткових переваг РО можна віднести:

- ідентифікація – виявлені закономірності допомагають ідентифікувати об’єкти під різними кутами та відстанями (наприклад, у ГН на основі відео) або ідентифікувати небезпечні події. РО використовується для ідентифікації людей із глибоким відеонавчанням за допомогою розпізнавання обличчя або аналізу рухів. Нові системи ШІ можуть ідентифікувати людей вимірюючи їхню ходу або за моделлю ходьби;

- виявлення корисної інформації – алгоритми РО дозволяють «мислити нестандартно» та виявляти випадки, які люди не бачать чи не помічають. Шаблони алгоритму можуть виявляти дуже тонкі зрушення в даних або кореляції між факторами у величезній кількості даних. Це дуже важливо для медичних випадків – наприклад, моделі ГН використовуються для діагностики пухлин головного мозку шляхом отримання зображень магнітно-резонансної томографії. У інформаційній безпеці та ІТ популярним прикладом розпізнавання шаблонів є їх зіставлення із шаблонами з бази системи виявлення вторгнень (IDS) для моніторингу комп’ютерних мереж або систем на предмет зловмисної активності або порушень;

- прогнозування – дані оперативного прогнозування та стратегічного прогнозування майбутніх подій відіграють важливу роль у багатьох проектах РО, наприклад, на торгових ринках для прогнозування цін на акції та інших інвестиційних можливостей або виявлення тенденцій для маркетингових цілей;

- прийняття рішень – сучасні методи МН надають високоякісну інформацію на основі шаблонів, виявлених майже в реальному часі. Це дає змогу приймати рішення на основі надійних оперативних даних. Вирішальним фактором є швидкість сучасних ШІ-систем РО, які перевершують звичайні методи та створюють нові програми. Наприклад, медичне РО для виявлення параметрів ризику в даних, швидкого надання лікарям важливої інформації;

- аналітика великих даних – за допомогою НМ стало можливим виявляти закономірності у величезних обсягах даних. Це дозволило використовувати випадки, які були б неможливими за допомогою традиційних статистичних методів. Розпізнавання шаблонів є життєво важливим у галузі медицини, для судово-медичного аналізу та секвенування ДНК. Наприклад, його використовували для розробки вакцин для боротьби з коронавірусом COVID-19.

Алгоритми розпізнавання шаблонів можна застосовувати до різних типів цифрових даних, включаючи зображення, тексти або відео. Пошук закономірностей дозволяє класифікувати результати для прийняття обґрунтованих рішень. РО можна використовувати для повної автоматизації та вирішення складних аналітичних задач.

2.4 Оптичне розпізнавання символів

Ще однією з областей використання КЗ для РО є оптичне розпізнавання символів. Технологія OCR – це рішення для автоматизації вилучення даних з друкованого або письмового тексту, зі сканованого документа або файлу зображення, з наступним перетворенням тексту у машиночитану форму для використання в обробці даних, таких як редагування або пошуку [16].

2.4.1 Історія оптичного розпізнавання символів

Найдавніше використання OCR можна простежити до телеграфної технології та пристроїв для читання для сліпих.

Емануель Голдберг винайшов OCR-подібну машину. Він читав символи та перетворював їх у стандартний телеграфний код. Приблизно в той же час Едмунд Фурньє д’Альбе винайшов оптофон. Подібно до винаходу Голдберга, це був портативний сканер, який виробляв тони, що відповідали певним літерам або символам під час руху по сторінці.

В кінці 1920-х, на початку 1930-х років Голдберг розробив машину для пошуку в архівах мікрофільмів за допомогою оптичного розпізнавання коду. Він назвав це своєю «Статистичною машиною». У 1931 році він запатентував цей винахід, який пізніше придбала IBM.

У 1974 році Рей Курцвейл заснував компанію Kurzweil Computer Products, Inc., чий OCR-продукт міг розпізнавати текст, надрукований практично будь-яким шрифтом. Він вирішив, що найкращим застосуванням цієї технології буде пристрій для МН для сліпих, тому він створив машину для читання, яка може читати текст вголос у форматі перетворення тексту в голосові повідомлення. У 1980 році Курцвейл продав свою компанію Xerox, яка була зацікавлена в подальшій комерціалізації перетворення тексту з паперу в цифрову форму.

Технологія OCR стала популярною на початку 1990-х років під час оцифрування історичних газет. Сучасні рішення після ряду вдосконалень мають можливість забезпечити майже ідеальну точність OCR. Передові методи використовуються для автоматизації складних процесів обробки документів. До появи технології OCR єдиним засобом цифрового форматування документів було повторне введення тексту вручну. Це не тільки забирало багато часу, але й супроводжувалося неминучими неточностями та помилками. Сьогодні послуги OCR широко доступні. Наприклад, Google Cloud Vision OCR використовується для сканування та зберігання документів на смартфоні [17].

2.4.2 Принцип роботи оптичного розпізнавання символів

OCR використовує сканер для обробки фізичної форми документа. Після копіювання всіх сторінок ПЗ OCR перетворює документ у двоколірну або чорно-білу версію. Відскановане зображення або растрове зображення аналізується на наявність світлих і темних ділянок, і темні ділянки ідентифікуються як символи, які потрібно розпізнати, а світлі ділянки ідентифікуються як фон. Потім темні області обробляються для пошуку літер алфавіту або цифр. Цей етап, як правило, передбачає націлювання на один символ, слово або блок тексту за раз. Потім

символи ідентифікуються за допомогою одного з двох алгоритмів – РО або розпізнавання ознак.

РО використовується, коли програма OCR передає приклади тексту в різних шрифтах і форматах для порівняння та розпізнавання символів у відсканованому документі чи файлі зображення.

Розпізнавання ознак відбувається, коли OCR застосовує базу правил щодо особливостей певної літери чи цифри для розпізнавання символів у відсканованому документі. Ознаки включають кількість ліній під кутом, перехресних ліній або кривих у символі. Коли символ ідентифікується, він перетворюється на код ASCII, який комп'ютерні системи використовують для подальших маніпуляцій.

Програма OCR також аналізує структуру зображення документа. Він ділить сторінку на такі елементи, як блоки текстів, таблиць або зображень. Рядки поділені на слова, а потім на символи. Після виділення символів програма порівнює їх із набором шаблонних зображень. Після обробки всіх ймовірних збігів програма представляє результат – розпізнаний текст [18].

2.4.3 Варіанти використання оптичного розпізнавання символів

Найвідомішим варіантом використання технології OCR є перетворення друкованих паперових документів у машинозчитувані текстові документи. Коли відсканований паперовий документ проходить обробку OCR, текст документа можна редагувати за допомогою текстового процесора, наприклад Microsoft Word або Google Docs.

Розпізнавання символів часто використовується як прихована технологія, що забезпечує роботу багатьох добре відомих систем і служб у повсякденному житті. Важливі випадки використання технології OCR включають автоматизацію введення даних, допомогу сліпим і людям із вадами зору та індексування документів для пошукових систем, таких як паспорти, номерні знаки, рахунки-фактури, банківські виписки, візитні картки та автоматичне розпізнавання номерних знаків.

OCR дозволяє оптимізувати моделювання великих даних, перетворюючи паперові документи та документи зі сканованими зображеннями в машиночитані PDF-файли з можливістю пошуку. Обробку та отримання цінної інформації неможливо автоматизувати без попереднього застосування оптичного розпізнавання тексту в документах, де ще немає текстових шарів.

Завдяки розпізнаванню тексту OCR скановані документи можна інтегрувати в систему великих даних, яка тепер може зчитувати дані клієнтів із банківських виписок, контрактів та інших важливих друкованих документів. Замість того, щоб співробітники перевіряли незліченну кількість документів із зображеннями та вручну вводили вхідні дані, організації можуть використовувати OCR для автоматизації на етапі введення інтелектуального аналізу даних. ПЗ OCR може ідентифікувати текст на зображенні, витягувати текст із зображень, зберігати текстовий файл і підтримувати формати jpg, jpeg, png, bmp, tiff, pdf та інші.

Менш відомі, але такі ж важливі варіанти використання технології OCR включають:

- розпізнавання паспортів для аеропортів;
- розпізнавання дорожніх знаків;
- вилучення контактної інформації з документів або візиток;
- подолання систем захисту від ботів CAPTCHA;
- введення даних для ділових документів (банківські виписки, рахунки-фактури, квитанції) [19].

2.4.4 Переваги та недоліки OCR

Основна перевага технології OCR – спрощення процесу введення даних, легкий пошук, редагування та зберігання тексту. OCR дозволяє компаніям і окремим особам зберігати файли на своїх комп’ютерах, ноутбуках та інших пристроях, забезпечуючи постійний доступ до всієї документації.

Також до переваг OCR можна віднести наступні:

- покращена точність – програмне розпізнавання символів усуває людські помилки, що забезпечує підвищену точність;

- прискорення процесів – технологія перетворює неструктуровані дані в інформацію, доступну для пошуку, забезпечуючи доступ до необхідних даних з більшою швидкістю, а отже прискорюючи бізнес-процеси;

- економічно – технологія OCR не потребує багато ресурсів, що зменшує витрати на обробку та відповідно зменшує загальні витрати бізнесу;

- підвищення продуктивності – легкий доступ до даних з можливістю пошуку створює вільне від стресу середовище для співробітників, дозволяючи їм зосередитися на головних цілях, підвищуючи продуктивність бізнесу.

Технології OCR притаманні і свої недоліки, до яких можна віднести наступні:

- якість розпізнавання OCR не завжди висока – залежить від якості вхідного зображення, яке надається до нього. Це означає, що якщо на зображенні є якісь недоліки, OCR буде важче вилучити з нього текст. Помилки OCR можуть бути навіть важчим для виправлення, оскільки вони часто вимагають від користувача виправити помилки OCR перед повторною обробкою;

- час – OCR може повільно працювати. Це тому, що технологія OCR має аналізувати кожне зображення та перетворювати його на текст, що потребує певного часу. Це може стати проблемою, якщо потрібно перетворити великий документ на текст;

- неточність – пояснюється тим, що технологія OCR не є 100% точною, і вона іноді може призвести до помилок під час перетворення зображень у текст. Наприклад, OCR може помилково ввести малу літеру «l» за «1» або «b» за «8». Це може спричинити проблеми, якщо текст використовується для критичних цілей, наприклад, у юридичному документі;

- втрата форматування документів – ще одним недоліком OCR є те, що іноді під час процесу втрачається форматування вихідних документів. Це може призвести до того, що текст буде важко прочитати або важко зрозуміти;

- відсутність інформації про деякі символи, наприклад про знаки пунктуації – є багато знаків пунктуації, які не можуть бути прочитані ПЗ OCR, тому що вони

замалі чи несуміжні, або тому, що вони перевернуті чи задом наперед. Пунктуаційні помилки також можуть виникати, якщо користувач вводить неправильний розділовий знак.

- неможливість розпізнати деякі мови належним чином – якщо текст написано мовою, для якої немає мовного пакету OCR. Мовні пакети OCR – це необов'язковий компонент, який можна додати до інсталяції OCR [20].

2.5 Технологія OCR та автоматичне розпізнавання номерних знаків

Ще однією сферою використання технології OCR є автоматичне розпізнавання номерних знаків транспортних засобів без будь-якої людської взаємодії. Система ANPR застосовує різні методи обробки зображень, щоб швидко й автоматично ідентифікувати транспортні засоби на відео чи фотознімках з камер (рис. 2.3).

Рисунок 2.3 – Розпізнавання тексту на номерному знаку

ANPR можна використовувати для різних цілей, наприклад, для відстеження руху транспортних засобів, ідентифікації конкретних автомобілів, автоматизованого контролю за паркуванням і так далі. Використання систем ANPR стає все більш популярним, оскільки технологія швидко розвивається з появою машинного та глибокого навчання, вартість обчислень зменшується, а точність застосованих методів обробки зображень зростає.

Системи ANPR застосовують OCR у поєднанні з іншими методами обробки зображень для зчитування номерних знаків транспортних засобів. ANPR є однією з

найточніших і широко застосовуваних систем КЗ. Методи, які використовує ANPR, постійно вдосконалюються, щоб підвищити продуктивність, точність, економічну ефективність, надійність і масштабованість ПЗ ANPR.

Конвеєр візуалізації, що широко використовуються в ANPR, містить необхідні кроки перетворення вхідного зображення або відео в значущу повну інформацію. До таких методів відносяться:

- виявлення об’єктів у реальному часі – використовує ГН для розпізнавання транспортних засобів і різних класів транспортних засобів (автобус, вантажівка, легковий автомобіль, мікроавтобус, мотоцикл тощо) у зображеннях відеопотоків. Сучасні алгоритми виявлення об’єктів, такі як YOLOv3 або YOLOv7, використовують НМ, навчені на наборі даних зображень;

- обробка зображень, що включає традиційні методи КЗ, які використовуються для нормалізації та підготовки зображень до обробки за допомогою алгоритму OCR. Оскільки додатки ANPR зазвичай використовуються в складних реальних умовах з різним освітленням, погодою та непослідовними налаштуваннями, функції обробки зображень використовуються для підвищення різкості, корекції кольорів або обрізання зображень, щоб значно покращити вихідні результати [21].

3 ШУМ ЗОБРАЖЕННЯ

3.1 Цифрове зображення

Цифрове зображення – зображення, що зберігається у двійковій формі та розділене на матрицю пікселів. Кожен піксель складається з цифрового значення одного або кількох бітів, що визначається бітовою глибиною. Цифрове значення може представляти енергію, яскравість, колір, інтенсивність, звук, висоту або класифіковане значення, отримане за допомогою обробки зображення. Цифрове зображення зберігається як растр і може містити одну або кілька смуг.

Цифрове зображення – це дискретизоване та квантоване представлення аналогового зображення [22]. Таким чином, воно представляє аналогове зображення з використанням кінцевої колекції зразків. Зображення складається з основної одиниці під назвою «піксель». Цифрове зображення – це двовимірний масив або двовимірна послідовність пікселів. Таким чином, група пікселів, регулярно розташованих у значущому порядку, передає зображення. Цифрове зображення може бути представлено у вигляді бінарного, сірого або кольорового зображень [23].

3.1.1 Бінарне зображення

Бінарне зображення (дворівневе, двійкове) – різновид цифрових растрових зображень, коли кожен піксель може представляти лише один із двох кольорів. Значення кожного пікселя умовно кодуються як «0» та «1» [24]. Значення «0» умовно називають заднім планом чи тлом, а «1» – переднім планом. Часто при зберіганні цифрових бінарних зображень застосовується бітова карта, де використовують один біт інформації для представлення одного пікселя. Бінарне зображення також іноді називають однобітним, монохромним або чорно-білим. Бінарні зображення можна розглядати, як окремий випадок індексованого кольорового зображення з палітрою з двох кольорів різних відтінків або як окремий

випадок напівтонового зображення, при використанні кольорів одного відтінку з різною яскравістю.

Часто для створення бінарного зображення застосовують такий метод сегментації зображення як порогове значення. Найпростіші методи порогового значення замінюють кожен піксель на зображенні чорним пікселем, якщо інтенсивність зображення $I_{i,j}$ менша за фіксоване значення, яке зветься пороговим (T) або білий піксель, якщо інтенсивність пікселів перевищує цей поріг.

Програмна реалізація створення бінарного зображення за допомогою методу порогово значення на мові програмування Python з використання бібліотеки OpenCV наведено нижче:

```
# зчитування зображення
image = cv2.imread('input.jpg')
# застосування методу порогової обробки
# всі значення пікселів вище 120 будуть встановлені на 255
ret, thresh = cv2.threshold(image, 120, 255, cv2.THRESH_BINARY)
# виведення результату
cv2.imshow('Binary Threshold', thresh)
```

Результат створення бінарного зображення представлено на рис. 3.1.

а) б)

Рисунок 3.1 – Результат створення бінарного зображення: а) оригінальне зображення; б) бінарне зображення, отримане в результаті використання порогового значення для ідентифікуємих пікселів

3.1.2 Зображення в градаціях сірого

Зображення в градаціях сірого (або рівень сірого) – це просто зображення, на якому єдиними кольорами є відтінки сірого [25]. Причина відмінності таких зображень від будь-якого різновиду полягає в тому, що для кожного пікселя потрібно надати менше інформації ніж для кольорових зображень. У сірому зображенні кожен піксель має значення від 0 до 255, де нуль відповідає «чорному», а 255 – «білому». Значення від 0 до 255 є різними відтінками сірого, де значення, ближчі до 0, темніші, а значення, ближчі до 255, світліші.

Насправді «сірий» колір – це колір, у якому червоний, зелений і синій компоненти мають однакову інтенсивність у просторі RGB, тому необхідно вказати лише одне значення інтенсивності для кожного пікселя, на відміну від трьох інтенсивностей, необхідних для представлення кожного пікселя у повнокольоровому RGB-зображенні [26].

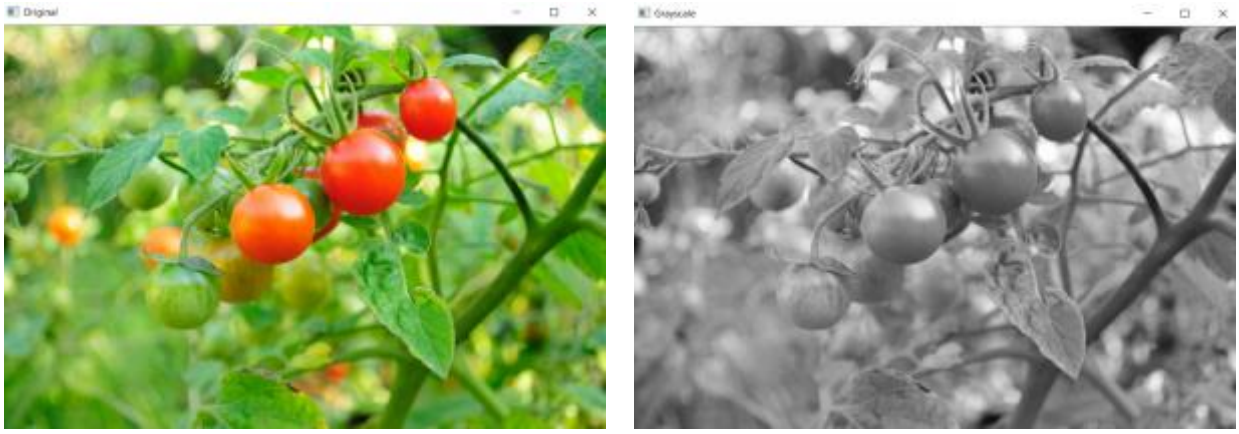
Часто інтенсивність градацій сірого зберігається як 8-бітове ціле число, що дає 256 можливих різних відтінків сірого від чорного до білого. Якщо рівні розподілені рівномірно, різниця між послідовними рівнями сірого буде значно кращою, ніж здатність людського ока розділяти рівні сірого.

Зображення у відтінках сірого дуже поширені, частково через те, що більшість сучасних дисплеїв і обладнання для захоплення зображень підтримує лише 8-бітні зображення. Крім того, зображень у відтінках сірого цілком достатньо для багатьох завдань, тому немає потреби використовувати складніші кольорові зображення, які важче обробляти.

Програмна реалізація створення зображення в градаціях сірого на мові програмування Python з використанням бібліотеки OpenCV наведено нижче:

```
# зчитування зображення
image = cv2.imread('input.jpg')
# створення градації сірого шляхом перетворення
# кольорового зображення
gray_image = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
# виведення результату
cv2.imshow('Gray image', gray_image)
```

Результат створення зображення в градаціях сірого представлено на рис. 3.2.



а) б)

Рисунок 3.2 – Результат створення зображення в градаціях сірого:
а) оригінальне зображення; б) зображення в градаціях сірого

3.1.3 Кольорове зображення

Можна побудувати майже всі видимі кольори, комбінуючи три основні кольори червоний, зелений і синій, тому що людське око має лише три різних колірних рецептора, кожен з яких чутливий до одного з трьох кольорів. Різні комбінації в стимуляції рецепторів дозволяють людському оку розрізняти приблизно 350000 кольорів. Кольорове RGB-зображення – це багатоспектральне зображення з однією смугою для кожного кольору червоного, зеленого та синього, що створює зважене поєднання трьох основних кольорів для кожного пікселя [26].

Повне 24-бітне кольорове зображення містить одне 8-бітне значення для кожного кольору, таким чином дає змогу відображати $2^{24} = 16777216$ різні кольори.

Однак використання повного 24-розрядного зображення для збереження кольору для кожного пікселя має високу вартість з точки зору обчислень і часто необов’язкове. Таким чином, колір для кожного пікселя кодується в одному байті, що призводить до 8-бітного кольорового зображення. Процес скорочення подання кольору з 24-бітів до 8-бітів, відомий як квантування кольорів, обмежує кількість

можливих кольорів до 256. Однак зазвичай немає видимої різниці між 24-кольоровим зображенням і тим самим зображенням, що відображається за допомогою 8 біт. 8- бітні кольорові зображення базуються на кольорових картах, які є пошуковими таблицями, що беруть значення 8-бітного пікселя як індекс і забезпечують вихідне значення для кожного кольору.

Програмна реалізація створення кольорового зображення на мові програмування Python з використання бібліотеки Numpy наведено нижче:

```
# висота та ширина майбутнього зображення
height=512
width=512
#створення 3-канального зображення з повністю чорним фоном
image = np.zeros((height,width,3), np.uint8)
# встановлення нового кольору
image[:]=(0,124,255)
cv2.imshow('3 Channel Window', image)
```

Результат створення кольорового зображення представлено на рис. 3.3.

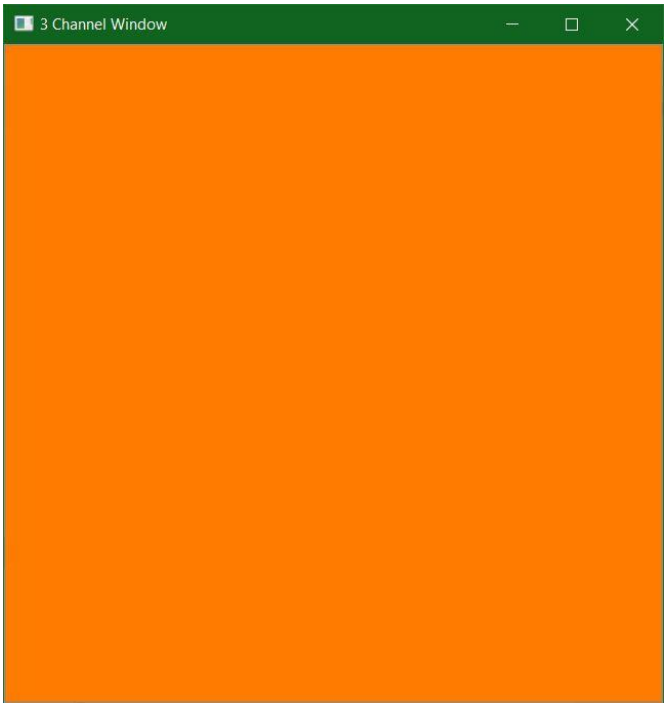


Рисунок 3.3 – Результат створення кольорового зображення

3.2 Причини виникнення шуму на зображенні

Шум зображення складається з випадкових варіацій інформації про яскравість або колір на знімку, який ви намагаєтеся зробити. Більш за все це проявляється в умовах слабкого освітлення та/або під час зйомки з низькою роздільністю датчиків. Чим більша компенсація налаштування слабкого освітлення за допомогою посилення та чутливості до світла, тим більше шуму створюється в результаті [27].

Наявність шуму в зображенні може бути адитивною або мультиплікативною. У моделі адитивного шуму до оригінального сигналу додається додатковий шумовий сигнал, щоб отримати пошкоджений шумовий сигнал, який відповідає такому правилу:

$$w(x, y) = s(x, y) + n(x, y), \tag{3.1}$$

де $s(x, y)$ – представляє початкову інтенсивність зображення, а $n(x, y)$ представляє шум, який подається для створення спотвореного сигналу $w(x, y)$ у позиції пікселя (x, y).

Подібним чином модель мультиплікативного шуму множить вихідний сигнал на сигнал шуму.

Під час отримання та передачі зображення на зображенні може з’явитися шум. Введення шуму в зображення може бути викликано кількома факторами. Кількісна оцінка шуму визначається кількістю пошкоджених пікселів у зображенні.

Шум зображення може варіюватися від майже невидимих цяток на цифровому знімку, зробленому при хорошому освітленні, до оптичних і радіоастрономічних зображень, які є майже повністю шумовими, з яких шляхом складної обробки можна отримати невелику кількість інформації. Такий рівень шуму був би недоречним для фотографії, оскільки неможливо було б ідентифікувати об’єкт.

Нижче наведено основні джерела шуму в цифрових зображеннях [28]:

- налаштування слабкого освітлення – у добре освітленому середовищі світла достатньо, щоб подолати небажані дані, створені шумом датчика. Умови слабкого

освітлення, з іншого боку, можуть призвести до отримання більшої кількості шумових даних, ніж світлових, що призводить до дуже помітних рівнів шуму;

- режими високої чутливості – у цифровій фотографії висока чутливість є результатом того, що пристрої із зарядовим зв'язком посилюють дані, які вона вимірює. Але це підсилення не розрізняє дані про світло та дані про шум – тобто обидва посилюються і тому ці режими часто призводять до більш помітного шуму;

- датчики слабкої продуктивності – чим менші фотоночутливі елементи, тим менше фотонів накопичуватиметься на піксель під час експозиції. Менше фотонів на піксель означає більше шуму на піксель, оскільки сигнал потрібно посилити більше, щоб зчитати заряд;

- фактори навколишнього середовища – іноді сигнали, що надходять за межі камери (наприклад, від космічного випромінювання чи потужних радіосигналів), можуть заважати та впливати на рівень шуму в фінальному знімку;

- низька освітленість та температура датчика можуть спричинити шум зображення;

- частинки пилу в сканері можуть викликати шум у цифровому зображенні;

- перешкоди каналу передачі;

- щільність розміщення пікселів (розмір осередків матриці) – оптимальний розмір лежить у межах від 6 до 11 мкм. У компактних камерах розмір осередків становить 3-5 мкм, що й зумовлює високий рівень шумів.

3.3 Типи шумів зображення

3.3.1 Шум Гауса

Шум Гауса – це статистичний шум із функцією щільності ймовірності, що дорівнює нормальному розподілу. Гаусовий шум має рівномірний розподіл по всьому сигналу.

Зображення з шумами містить пікселі, які складаються із суми вихідних значень пікселів плюс випадкове значення шуму Гауса. Функція розподілу

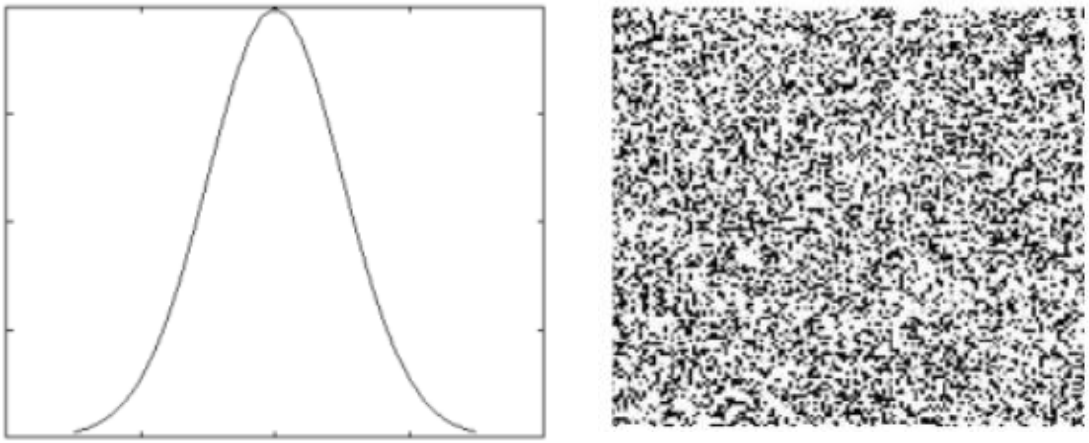
ймовірностей для розподілу Гауса має форму дзвона. Адитивний білий шум Гауса є найпоширенішим застосуванням Гаусового шуму в програмах.

Функція щільності ймовірності випадкової величини Гауса $p_G(z)$, визначається як [29]:

$$p_G(z) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(z-\mu)^2}{2\sigma^2}}, \tag{3.2}$$

де z – представляє рівень сірого, μ – середнє значення сірого, а σ – його стандартне відхилення.

На рис. 3.4 показано функцію розподілу ймовірностей Гаусового шуму та піксельне подання Гаусового шуму.

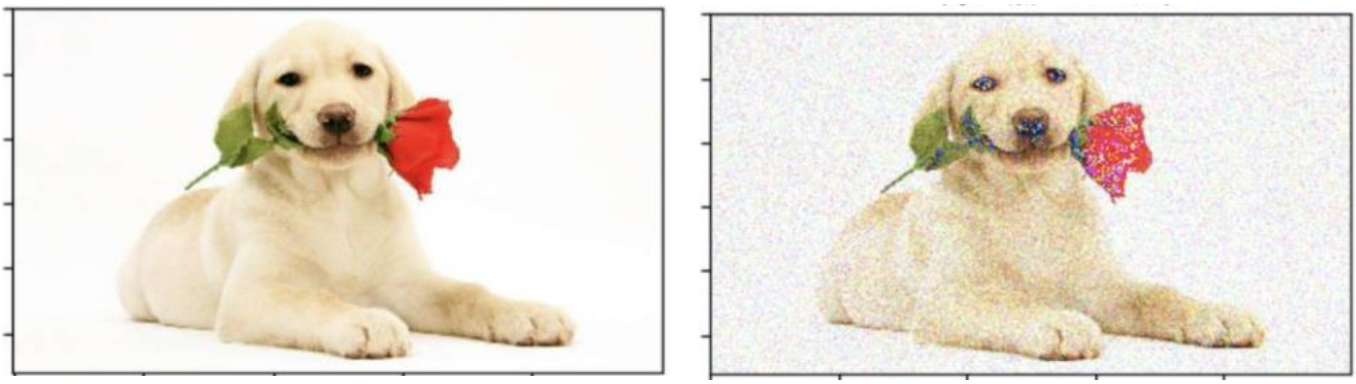


а) б)

Рисунок 3.4 – Функція розподілу Гауса: а) розподіл Гаусового шуму; б) піксельне подання Гаусового шуму

Основні джерела Гаусового шуму в цифрових зображеннях виникають під час отримання зображення. Наприклад, шум датчика, спричинений поганим освітленням, високою температурою або передачею даних.

Приклад зображення з шумом Гауса представлено на рис. 3.5.



а) б)

Рисунок 3.5 – Приклад зображення: а) оригінальне зображення; б) результат накладання шуму Гауса на зображення

3.3.2 Шум «солі та перцю»

Тип шуму, який зазвичай можна побачити на фотографіях – це шум «солі та перцю». Він проявляється у вигляді білих і чорних пікселів, які з’являються через випадкові проміжки часу. Помилки під час передачі даних викликають появу такого виду шуму. Пошкоджені пікселі по черзі встановлюються на мінімальне та найвище значення, надаючи зображенню вигляд «солі» і «перцю».

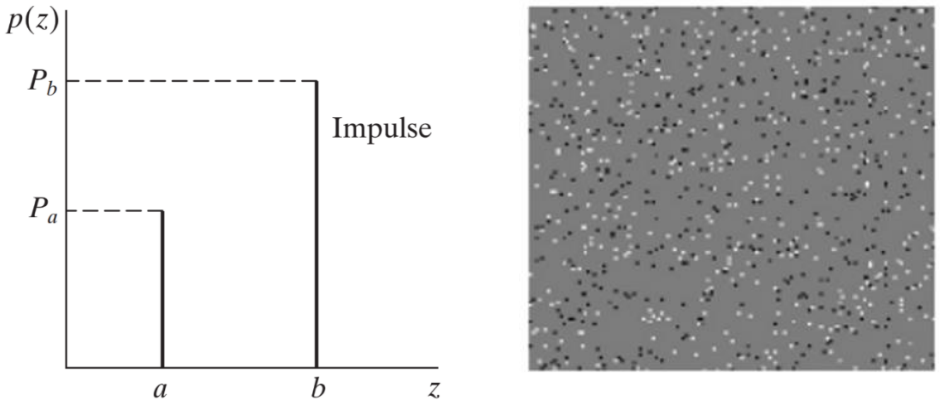
Функцію розподілу ймовірності шуму «солі та перцю» наведено нижче:

$$p(z) = \begin{cases} P_a, z = a \\ P_b, z = b \\ 0 \end{cases}, \tag{3.3}$$

де P_a – ймовірність значення шуму «солі», P_b – ймовірність значення шуму «перцю».

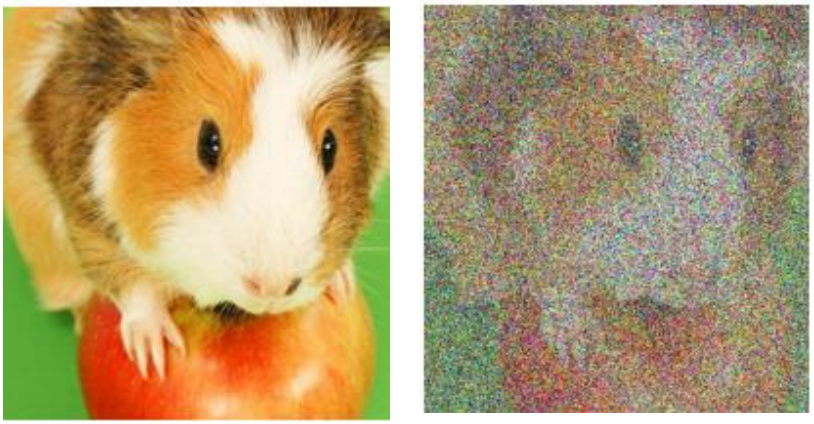
Як можна побачити з формули, «сіль» і «перець» не обов’язково повинні бути чорними і білими. Значення P_a і P_b у шумі «солі та перцю» різні. Імовірність кожного в середньому становить менше 0,1. Також можливо встановити ці імпульсні значення на власні значення пікселів. На рис. 3.6 представлено графік

розподілу ймовірності шуму «солі та перцю» та його піксельне представлення, а на рис. 3.7 наведено приклад зображення з цим шумом.



а) б)

Рисунок 3.6 – Шум «солі та перцю»: а) графік розподілу ймовірності шуму; б) піксельне представлення шуму



а) б)

Рисунок 3.7 – Приклад зображення: а) оригінальне зображення; б) пошкоджене зображення з 70% шуму «солі та перцю»

3.3.3 Спекл-шум

На відміну від Гаусового шуму або шуму «солі та перцю», спекл-шум є мультиплікативним шумом (випадковий шум сигналу, помножений на відносний

сигнал протягом захоплення). У діагностичних дослідженнях це знижує якість зображення, надаючи зображенням вигляд хвилі зворотного розсіювання, спричиненої багатьма мікроскопічними розсіяними відображеннями, що протікають через внутрішні органи. Це ускладнює для спостерігача розрізнення дрібних деталей на зображеннях.

Зі спекл-шумом боротися складніше і важче через такі причини:

- компоненти шуму множаться на кожен піксель вихідного зображення;

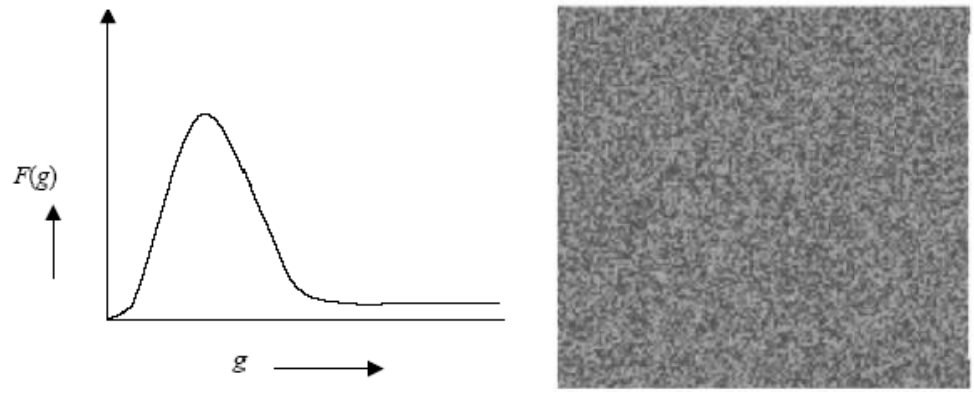
- не має нормального розподілу і досить близький до розподілу Релея та гамма-розподілу.

Функцію спекл-шуму з гамма розподілом можна представити у вигляді [30]:

$$F(g) = \frac{g^{a-1}}{(a-1)!a^a} e^{\frac{-g}{a}}, \tag{3.4}$$

де a^a – дисперсія, g – рівень сірого.

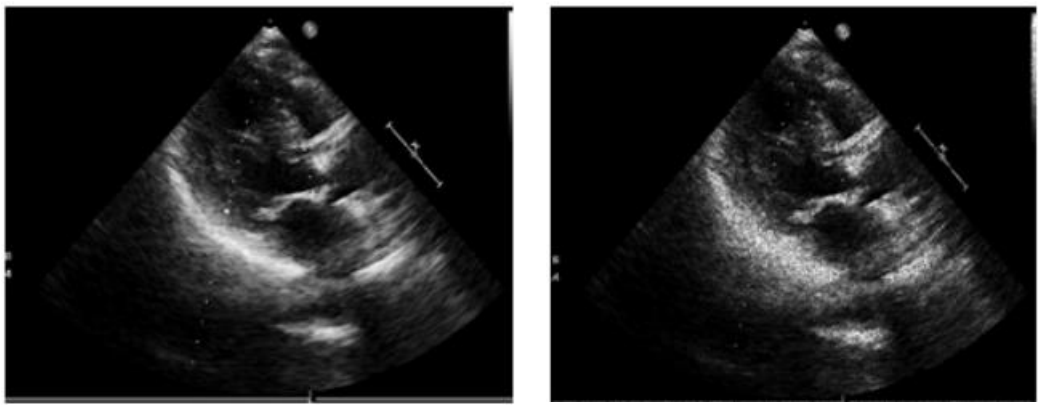
Графік розподілу і піксельне представлення цього шуму показано на рис. 3.8.



а) б)

Рисунок 3.8 – Спекл-шум: а) гамма-розподіл; б) піксельне представлення шуму

Цей тип шуму можна знайти в широкому діапазоні систем, включаючи зображення радарів із синтезованою апертурою (SAR), ультразвукове зображення та багато іншого. Приклад зображення зі спекл-шумом представлено на рис. 3.9.



а) б)

Рисунок 3.9 – Приклад зображення: а) оригінальне зображення; б) зображення зі спекл-шумом

3.3.4 Шум Пуассона

Шум Пуассона створюється нелінійними відгуками детекторів зображення і регістраторами. Цей тип шуму визначається за даними зображення. Оскільки процедури виявлення та реєстрації включають довільне випромінювання електронів із розподілом Пуассона та середнім значенням відгуку, використовується формула у вигляді [31]:

$$P(k) = \frac{\lambda^k e^{-\lambda}}{k!}, \tag{3.5}$$

де, k – значення 0,1,2 і так далі, а λ – середнє число входжень в інтервалі.

Появу цього шуму можна побачити через статистичну природу електромагнітних хвиль, таких як рентгенівське випромінювання, видиме світло та гамма-промені.

На рис. 3.10 наведено криву Пуассона з різними значеннями k та λ, а на рис. 3.11 наведено приклад зображення із шумом Пуассона.

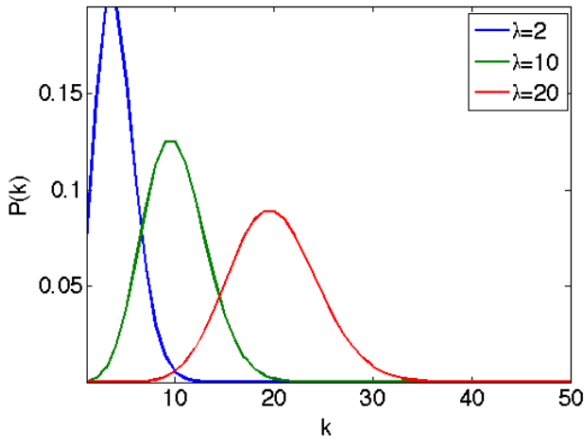
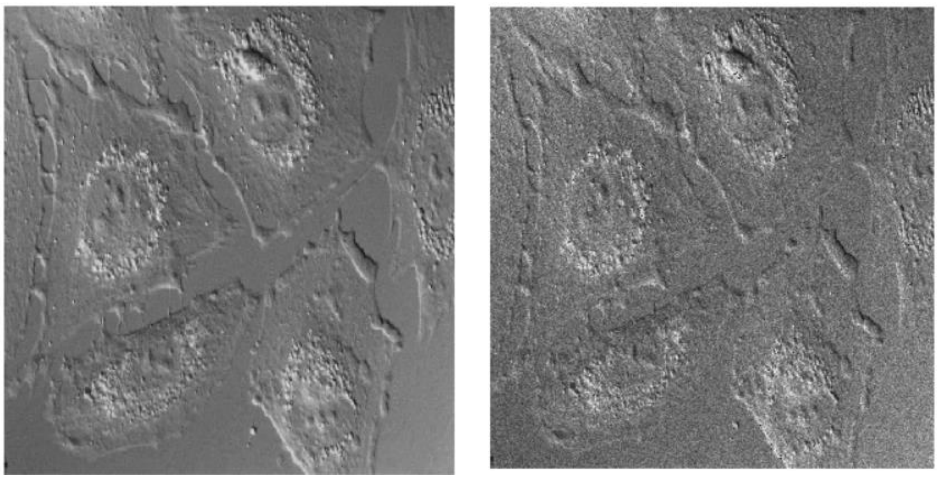


Рисунок 3.10 – Крива Пуассона з різними значеннями *k* та λ.



а) б)

Рисунок 3.11 – Приклад зображення: а) оригінальне зображення; б) зображення з доданим 15% шумом

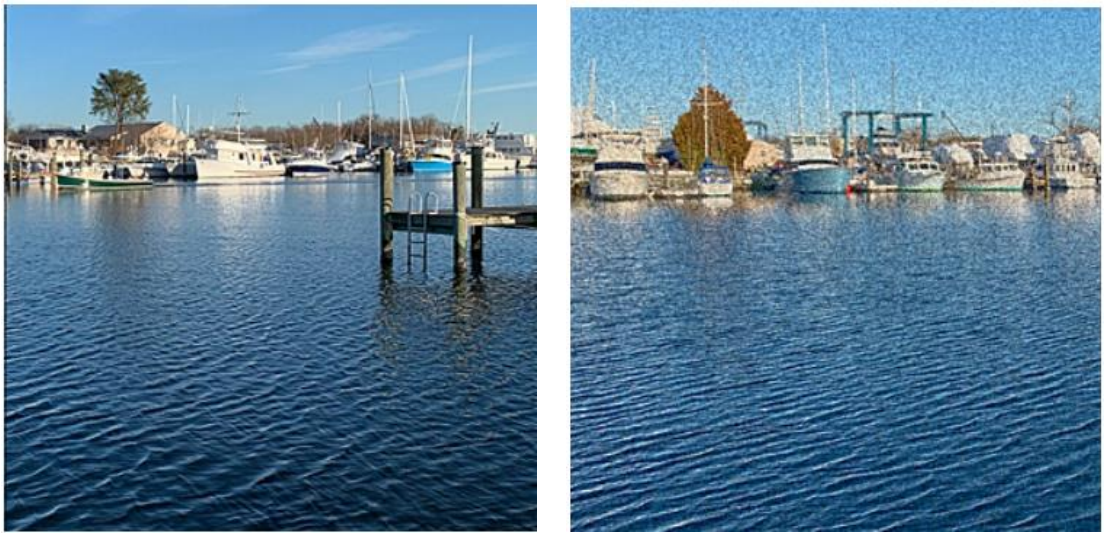
3.3.5 Шум квантування

Шум, спричинений квантуванням пікселів розпізнаного зображення до кількох дискретних рівнів, відомий як шум квантування. Має приблизно рівномірний розподіл. У контексті обробки зображень, квантування досягається стисненням діапазону значень в одне ціле значення або менший репрезентативний набір вибірок. Шум квантування зазвичай спричинений помилками округлення під час цього процесу.

3.3.6 Зернистість фотоплівки

Зернистість фотоплівки є залежним від сигналу шумом зі статистичним розподілом, подібним до дробового шуму. Якщо зерна фотоплівки розподілені рівномірно (однакова кількість на площу), і якщо кожне зерно має однакову й незалежну ймовірність перетворитися на темно-сріблясте зерно після поглинання фотонів, то кількість таких темних зерен у ділянці буде випадковою з біноміальною величиною розподілу. У областях, де ймовірність низька, цей розподіл буде близьким до класичного розподілу Пуассона дробового шуму [32].

Приклад зображення з додаванням шуму зернистості фотоплівки представлено на рис. 3.12.



а) б)

Рисунок 3.12 – Приклад зображення: а) оригінальне зображення; б) зображення з додаванням шуму зернистості плівки

3.3.7 Анізотропний шум

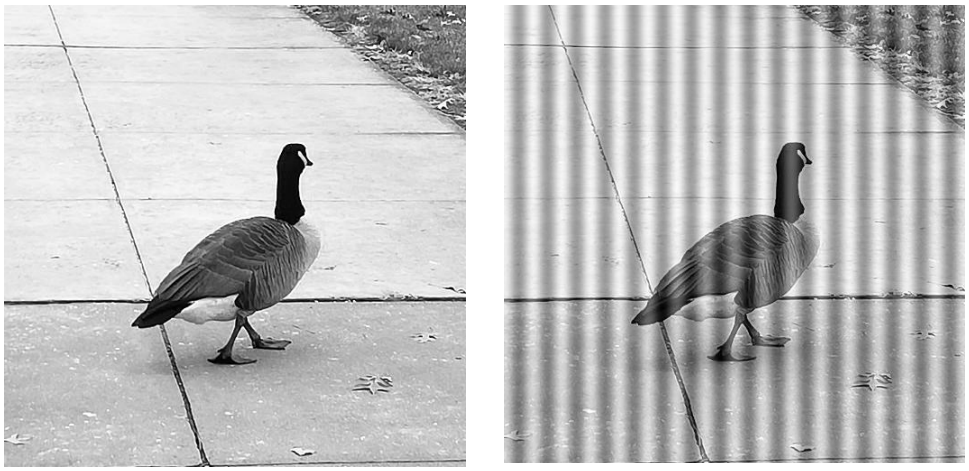
Анізотропний (неізотропний) шум виникає, коли показання датчика дискретизуються або квантуються. Цей тип шуму знижує сприйняту роздільної здатності зображення на знімках, які піддаються впливу, змішуючи дрібні деталі, створюючи візерунки, яких насправді немає, або сприймаючи прямі лінії як нерівні.

Цей тип шуму у відеокамерах зазвичай проявляється, коли власна роздільна здатність датчика набагато вища, ніж сенсора, що записує.

3.3.8 Періодичний шум

Загальним джерелом періодичного шуму в зображенні є електричні перешкоди під час процесу захоплення зображення. Зображення, на яке впливає періодичний шум, виглядатиме так, ніби поверх вихідного зображення додано повторюваний візерунок. У частотній області цей тип шуму можна розглядати як дискретні спайки. Значного зменшення цього шуму можна досягти шляхом застосування режекторних фільтрів у частотній області.

Приклад зображення, що містить періодичний шум представлено на рис. 3.13.



а) б)

Рисунок 3.13 – Приклад зображення: а) оригінальне зображення; б) зображення , що містить періодичний шум

3.4 Засоби зменшення шуму зображення

Методи зменшення шуму використовують різні математичні та статистичні моделі та класифікуються як класичні методи просторової області, методи перетворення та методи, засновані на МН та CNN.

3.4.1 Класичне усунення шумів

Методи просторової області спрямовані на видалення шуму шляхом обчислення рівня сірого кожного пікселя на основі кореляції між пікселями та ділянками зображення на вихідному зображенні. Значення сірого пікселя визначається як його яскравість, при цьому мінімальне значення дорівнює 0, а максимальне значення залежить від глибини оцифрування зображення. Для 8-бітної глибини це максимальне значення становить 255. Ці методи можна додатково розділити на методи фільтрації та методи варіаційного усунення шумів.

Фільтрація просторової області базується на принципі, згідно з яким шум займає вищу область частотного спектру. Далі цю частоту можна виділити та відняти за допомогою таких фільтрів, як: середнього значення, Вінера або зваженої медіани. Недоліком цього методу є можливе розмивання та пом’якшення інших високочастотних елементів, таких як краї та текстури, що є небажаним результатом.

Варіаційні методи зменшення шуму базуються на принципі, що сигнали з надмірною та, можливо, помилковою деталізацією (тобто шумом) мають високу загальну варіацію, яка пов’язана з частотою, але відрізняється від неї. Тому зменшення загальної варіації за умови, що вона близька до вихідного сигналу, зменшить надмірний шум, зберігши такі деталі, як краї.

3.4.2 Методи перетворення

На відміну від просторових методів, де усунення шуму виконується на фактичному зображенні, методи перетворення спочатку «транспонують» дане зображення в іншу область шляхом виконання набору математичних операцій, відомих як перетворення. Далі вони виконують усунення шумів на трансформованому зображенні, враховуючи відмінності фактичного зображення від небажаного шуму. Методи перетворення класифікуються на адаптивні та не адаптивні до даних залежно від типу трансформації, що виконується на зображенні.

Недоліком цих методів є те, що вони більш вимогливі до обчислювальної потужності, тоді як методи просторової області є швидшими та простішими. З іншого боку, методи трансформації можуть бути надзвичайно ефективними в деяких випадках, тому це компроміс між класичною швидкістю та візуальною якістю.

3.4.3 Методи машинного навчання

Хоча представлені вище методи усунення шуму добре усувають шум із зображень і відео, проте вони вимагають багато часу та ресурсів для обчислень. Для необізнаних на цих методах користувачів це означає значне ручне налаштування ПЗ з крутими кривими навчання. Методології ГН (часто із залученням CNN) були створені для вирішення цієї проблеми.

У МН для усунення шумів використовуються складні алгоритми, які навчаються на наборі з тисяч зображень, що містять пари зображень зі зниженим рівнем якості. Також алгоритми налаштовані на роботу для ідентифікації функції відображення, яка є невідомою базовою функцією, яка є узгодженою при зіставленні вхідних даних з вихідними у цільовій області та в результаті створює набір даних. Алгоритми намагаються вивчити зв’язок між двома наборами зображень. Отримавши знання, система буде в змозі автоматично застосовувати їх до зображень та відео з аналогічним шумом для створення версії з усуненим шумом.

3.4.4 Автоматичні кодери та декодери

Ще один з широко використовуваних підходів для усунення шумів – це автоматичні кодери, які є штучною НМ, що в основному використовується для стиснення та розпакування даних шляхом використання кодерів і декодерів. Щоб використовувати автоматичні кодери для усунення шумів, їм необхідно навчання із зашумленими зображеннями в якості функцій входу і очищеними зображеннями в якості цілей. Цей підхід дає швидкі та задовільні результати.

Приклад усунення шумів за допомогою автоматичних кодерів представлено на рис. 3.14.

Рисунок 3.14 – Приклад усунення шумів за допомогою автоматичних кодерів

4 ПРОГРАМНА РЕАЛІЗАЦІЯ

4.1 Технології розробки та інструментальні засоби

Для реалізації програми, яка буде автоматичного розпізнавати зображення номерного знаку автомобіля в умовах шуму за допомогою технології OCR було обрано клієнт-серверну архітектуру.

Для реалізації клієнтської частини програми було обрано фреймворк JavaScript, написаний на мові програмування TypeScript – Angular.

Для реалізації серверної частини програми було обрано високорівневу мову програмування загального призначення Python та її веб-фреймворк Flask для побудови веб-застосунків, а також наступні бібліотеки: OpenCV, NumPy, Imutils та easyocr.

Для реалізації програми буде використано кросплатформове середовище розробки PyCharm для мови програмування Python.

4.1.1 Фреймворк Angular

Angular – це платформа JavaScript із відкритим кодом, написана на TypeScript. Google підтримує цю платформу, і її основна мета – розробка односторінкових програм. Як фреймворк Angular має явні переваги, а також забезпечує стандартну структуру для роботи розробників. Це дозволяє користувачам створювати великі додатки зручним для обслуговування засобом [33]. Фреймворки загалом підвищують ефективність і продуктивність веб-розробки, забезпечуючи узгоджену структуру додатку та безліч бібліотечних функцій фрейму.

До основних особливостей Angular можна віднести:

- об'єктна модель документа (DOM) – розглядає документ XML або HTML як структуру дерева, в якій кожен вузол представляє частину документа. Angular використовує звичайний DOM. Нехай на одній сторінці HTML зроблено десять

оновлень. Замість того, щоб оновлювати ті, які вже були оновлені, Angular оновить всю деревовидну структуру тегів HTML;

- TypeScript – визначає набір типів для JavaScript, що допомагає користувачам писати код JavaScript, який легше зрозуміти. Увесь код TypeScript компілюється за допомогою JavaScript і може безперебійно працювати на будь-якій платформі. TypeScript не є обов'язковим для розробки програми в Angular;

- зв'язування даних – це процес, який дозволяє користувачам керувати елементами веб-сторінки через веб-браузер. Він використовує динамічний HTML і не вимагає складних сценаріїв або програмування. Зв'язування даних використовується на веб-сторінках, які містять інтерактивні компоненти, такі як калькулятори, навчальні посібники, форуми та ігри. Це також забезпечує краще поступове відображення веб-сторінки, коли сторінки містять велику кількість даних. Angular використовує двосторонню прив'язку. Стан моделі відображає будь-які зміни, зроблені у відповідних елементах інтерфейсу користувача. І навпаки, стан інтерфейсу користувача відображає будь-які зміни в стані моделі. Ця функція дозволяє інфраструктурі підключати DOM до даних моделі через контролер;

- тестування – Angular використовує структуру тестування Jasmine. Фреймворк Jasmine надає численні функції для написання різних видів тестів. Karma – це засіб виконання завдань для тестів, який використовує файл конфігурації для встановлення початкової, звітної та тестової структури.

Angular – це повноцінний фреймворк, який використовує схему розділення даних застосунку модель-представлення-контролер (MVC). Він містить чіткі вказівки щодо того, як має бути структурована програма, і пропонує двонаправлений потік даних, забезпечуючи реальний DOM.

4.1.2 Мікрофреймворк Flask

Flask – це мікрофреймворк Python для розробки веб-додатків. Його розробив Армін Роначер, який очолював міжнародну команду ентузіастів Python під назвою Россо. Був випущений у квітні 2010 року.

До основник компонентів Flask входять [34]:

- Werkzeug – це бібліотека службових програм для мови програмування Python для додатків інтерфейсу шлюзу веб-сервера (WSGI). WSGI – це специфікація універсального інтерфейсу між веб-сервером і веб-додатками. Werkzeug може створювати екземпляри об’єктів для запитів, відповідей і службових функцій. Він може бути використаний як основа для спеціального ПЗ;

- Jinja – є рушієм шаблонів для мови програмування Python. Подібно до веб-фреймворку Django, він обробляє шаблони в пісочниці;

- MarkupSafe – це бібліотека обробки рядків для мови програмування Python. Однойменний тип MarkupSafe розширює рядковий тип Python і позначає його вміст як «безпечний». Поєднання MarkupSafe зі звичайними рядками автоматично екранує непомічені рядки, уникаючи подвійного екранування вже позначених рядків;

- ItsDangerous – безпечна бібліотека серіалізації даних для мови програмування Python. Вона використовується для збереження сеансу програми Flask у файлі cookie, не дозволяючи користувачам змінювати вміст сеансу.

До основних переваг мікрофреймворку Flask відносяться:

- масштабованість – можна використовувати його для неймовірно швидкого розвитку технологічного проекту, наприклад веб-додатка. Якщо необхідно створити програму, яка починається з малого, але має потенціал для швидкого розвитку та в напрямках, які ви ще не повністю опрацьовані – це ідеальний вибір. Простота у використанні та невелика кількість залежностей дозволяють йому працювати безперебійно, навіть якщо він масштабується все більше і більше;

- гнучкість – дозволяє проекту легко рухатися в іншому напрямку, при цьому не руйнуючи головну конструкцію програми, коли інша частина буде змінена;

- легкість – мікрофреймворк не покладається на велику кількість розширень, щоб функціонувати. Flask також підтримує модульне програмування, де його функціональні можливості можна розділити на кілька взаємозамінних модулів. Кожен модуль діє як незалежний будівельний блок, який може виконувати одну частину функціональності. Разом це означає, що всі складові частини конструкції є гнучкими, рухливими та перевіряються самостійно;

4.1.3 Мова програмування Python

Python – високорівнева мова програмування загального призначення з динамічною строгою типізацією, що підтримує автоматичне управління пам’яттю, орієнтована на максимальне покращення продуктивності розробника, читаності коду та його якості. Також існує можливість перенесення програм написаних цією мовою.

Мова програмування Python – це повністю об’єктно-орієнтована мова. Однією з головних особливостей мови є виділення блоків коду за допомогою пробільних відступів. Мінімалістичний синтаксис мови, який не зобов’язує розробника часто звертатися до документації – це ще одна перевага мови програмування Python. Мова програмування також відома як інтерпретована і може використовуватися для написання скриптів. До недоліків мови програмування можна віднести нижчу швидкість роботи і більш високе споживання пам'яті написаних нею програм порівняно з аналогічним кодом, який написаний компільованими мовами, такими як C чи C++.

Python є мультипарадигмальною мовою програмування, що підтримує імперативне, процедурне, структурне, об'єктно-орієнтоване програмування, метапрограмування та функціональне програмування. До основних архітектурних рис мови відносяться: динамічна типізація, автоматичне управління пам'яттю, повна інтроспекція, механізм обробки винятків, підтримка багатопоточних обчислень із глобальним блокуванням інтерпретатора (GIL), високорівневі структури даних. Підтримується розбиття програм на модулі, які, в свою чергу, можуть поєднуватися в пакети.

Стандартна бібліотека включає великий набір корисних функцій, що переносяться, починаючи з можливостей для роботи з текстом і закінчуючи засобами для написання мережевих додатків. Додаткові можливості, такі як математичне моделювання, робота з обладнанням, написання веб-додатків або розробка ігор можуть реалізовуватися за допомогою великої кількості сторонніх

бібліотек, а також інтеграцією бібліотек, написаних на C або C++, при цьому і сам інтерпретатор Python може інтегруватися в проекти, написані цими мовами.

Python являє собою однією з найпопулярніших мов, яка використовується в аналізі даних, МН, DevOps та веб-розробці, а також в інших сферах, включаючи розробку ігор. За рахунок читабельності, простого синтаксису та відсутності необхідності в компіляції мова добре підходить для навчання програмування, дозволяючи концентруватися на вивченні алгоритмів, концептів та парадигм [35].

4.1.4 Середовище розробки PyCharm

PyCharm – інтегроване середовище розробки мови програмування Python. PyCharm розроблена компанією JetBrains на основі IntelliJ IDEA [36].

До основні можливостей програми можна віднести:

- допомога при написанні коду – PyCharm робить розробку максимально продуктивною завдяки функціям автодоповнення та аналізу коду, миттєвому підсвічуванню помилок та швидким виправленням;

- вбудовані інструменти для розробників – PyCharm пропонує великий набір вбудованих інструментів з коробки: відладчик та інструмент запуску тестів, профільник Python, повнофункціональний термінал, SSH-термінал, підтримує можливості віддаленої розробки та віддалені інтерпретатори, інструменти для роботи з базами даних. IDE інтегрована з популярними системами контролю версій, а також з Docker та Vagrant;

- веб розробка – PyCharm надає повноцінну підтримку різних веб-фреймворків та платформ для розробки на Python, підтримує темплейтні мови цих фреймворків, а також JavaScript, CoffeeScript, TypeScript, HTML/CSS, AngularJS, Node.js та багато інших;

- інструменти для наукових обчислень – з PyCharm можливо працювати з IPython Notebook, запускати команди в інтерактивній консолі Python, підключення бібліотек Anaconda, Matplotlib та NumPy;

- крос-платформна IDE – PyCharm можливо встановити на Windows, MacOS та Linux за допомогою одного ліцензійного ключа, налаштувати робоче середовище за бажанням користувача: колірна схема, «гарячі» клавіші, емуляція VIM.

4.1.5 Бібліотека OpenCV

OpenCV – бібліотека ПЗ для комп'ютерного зору та машинного навчання з відкритим вихідним кодом. OpenCV як і основний інтерфейс написані на C++, але з частковим використанням інтерфейсу C. Усі нові розробки та алгоритми з'являються в інтерфейсі C++. Підтримується Python, Java та MATLAB/OCTAVE [37].

Підтримує операційні системи: Windows, Linux, Mac OS, iOS і Android. Усі додатки OpenCV написано для використання в реальному часі для підвищення ефективності обчислень на оптимізованому C/C++, щоб скористатися перевагами багатоядерної обробки [38].

Нижче наведено основні модулі бібліотеки OpenCV:

- основна функціональність (Core Functionality) – цей модуль охоплює основні структури даних Scalar, Point, Range, що використовуються для створення програм OpenCV; також включає багатовимірний масив Mat для зберігання зображень;

- обробка зображень (Image Processing) – охоплює різні операції обробки зображень: фільтрація, геометричні перетворення зображень, перетворення простору кольорів, гістограми тощо;

- відео (Video) – аналіз відео: оцінка руху, віднімання фону та відстеження об'єктів;

- calib3d – модуль містить базові алгоритми геометрії кількох ракурсів, калібрування одиночної та стереокамери, оцінки пози об'єкта, стереовідповідності та елементи 3D-реконструкції;

- features2d – містить поняття виявлення та опису ознак;

- Objdetect – включає виявлення об'єктів і екземплярів попередньо визначених класів, таких як обличчя, очі, чашки, люди, автомобілі тощо;

- Highgui – інтерфейс із простими можливостями користувача.

4.1.6 Бібліотека NumPy

NumPy – це фундаментальний пакет для наукових обчислень на Python. Це бібліотека, яка пов’язує об’єкт з багатовимірним масивом, різні похідні об’єкти (такі як маскові масиви та матриці) та асортимент процедур для швидких операцій з масивами, включаючи математичні, логічні, маніпулювання фігурами, сортування, вибір, введення -виведення та багато іншого.

Бібліотека NumPy надає реалізації обчислювальних алгоритмів, оптимізованих для роботи з багатовимірними масивами. Будь-який алгоритм, виражений у вигляді послідовності операцій над масивами (матрицями) і реалізований з використанням NumPy, за швидкодією еквівалентний коду, що виконується MATLAB [39].

4.1.7 Бібліотека Imutils

Imutils – бібліотека, яка надає серію зручних функцій для полегшення основних функцій обробки зображень, таких як переклад, обертання, зміна розміру, відображення зображень, сортування контурів, виявлення країв та багато іншого з OpenCV та Python [40];

4.1.8 Бібліотека EasyOCR

EasyOCR – пакет Python, що конвертуватує зображення в текст. Це найпростіший спосіб реалізувати OCR і має доступ до понад 70 мов, включаючи англійську, китайську, японську, корейську, гінді та інші. EasyOCR створено на основі бібліотеки ГН Python і Pytorch, наявність графічного процесора може прискорити весь процес виявлення. Частина виявлення використовує алгоритм CRAFT, а модель розпізнавання – CRNN. Він складається з 3 основних компонентів:

вилучення функцій (Resnet), маркування послідовності (LSTM) та декодування (CTC). EasyOCR не має багато програмних залежностей, його можна безпосередньо використовувати з його API [41].

4.2 Програмна реалізація

4.2.1 Клієнтська частина

Структура проекту клієнтської частини, яка представлена на рис. 4.1 складається з наступних елементів: компонентів, директив, моделі даних та сервісів.

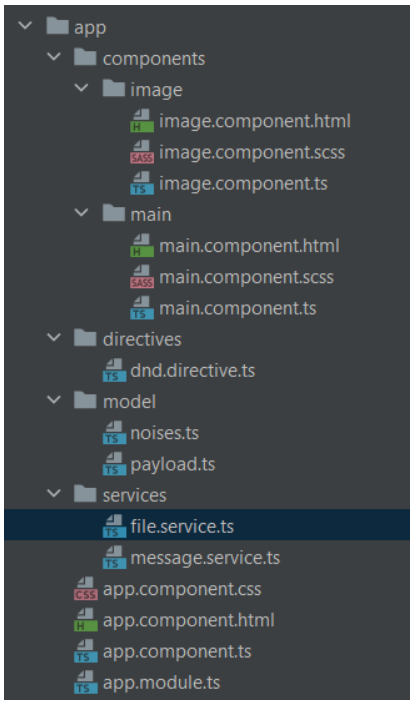


Рисунок 4.1 – Структура проекту клієнтської частини

Компонент ImageComponent призначений для відображення частини інтерфейсу користувача по вибору необхідного зображення. Компонент містить клас, в якому реалізовані функції по вибору необхідного зображення, можливості його видалення та вибору іншого, а також передачі вибраного зображення до компоненту MainComponent для подальшої обробки вибраного зображення.

Компонент MainComponent призначений для відображення всього контенту, який користувач бачить після запуску програми. Компонент містить функції для відображення вибраного зображення та відправку його до серверної частити програми для виконання однієї з функцій: накладання шуму на зображення або розпізнавання номерного знаку, який зображений на ньому. Також клас компоненту надає можливість завантажити зображення, яке було отримано в результаті роботи серверної частини програми.

У директиві DndDirective реалізовані функції для вибору зображення за допомогою методу «Drag-and-Drop».

Модель даних містить класи, які використовуються при виборі типу шуму, яке необхідно накласти на зображення та при відправці даних до серверної частини.

Клас FileService містить метод, який виконує відправку вибраного зображення та типу шуму до серверної частини за допомогою методу запиту POST із класу HttpClient. Також цей клас оброблює помилки, які можуть виникнути при передачі або отриманні даних з серверної частини.

Клас MessageService призначений для виведення повідомлень користувачу про помилки, які сталися при роботі програми.

MainComponent – основний будівельний блок для Angular додатку. Він включає в себе всі користувальницькі компоненти, які використовуються в програмі.

Для запуску клієнтської частини програми необхідно у командному рядку (терміналі) перейти до папки проекту і потім виконати команду «ng serve». Результатом успішного запуску програми є повідомлення «Compiled successfully» в терміналі, що свідчить про успішну побудову проекту. Після цього необхідно перейти до веб-браузеру та звернутися до клієнтської частини програми за посиланням http://localhost:4200/.

Приклад результату роботи клієнтської частини програми з розпізнаним номерним знаком представлено на рис. 4.2.

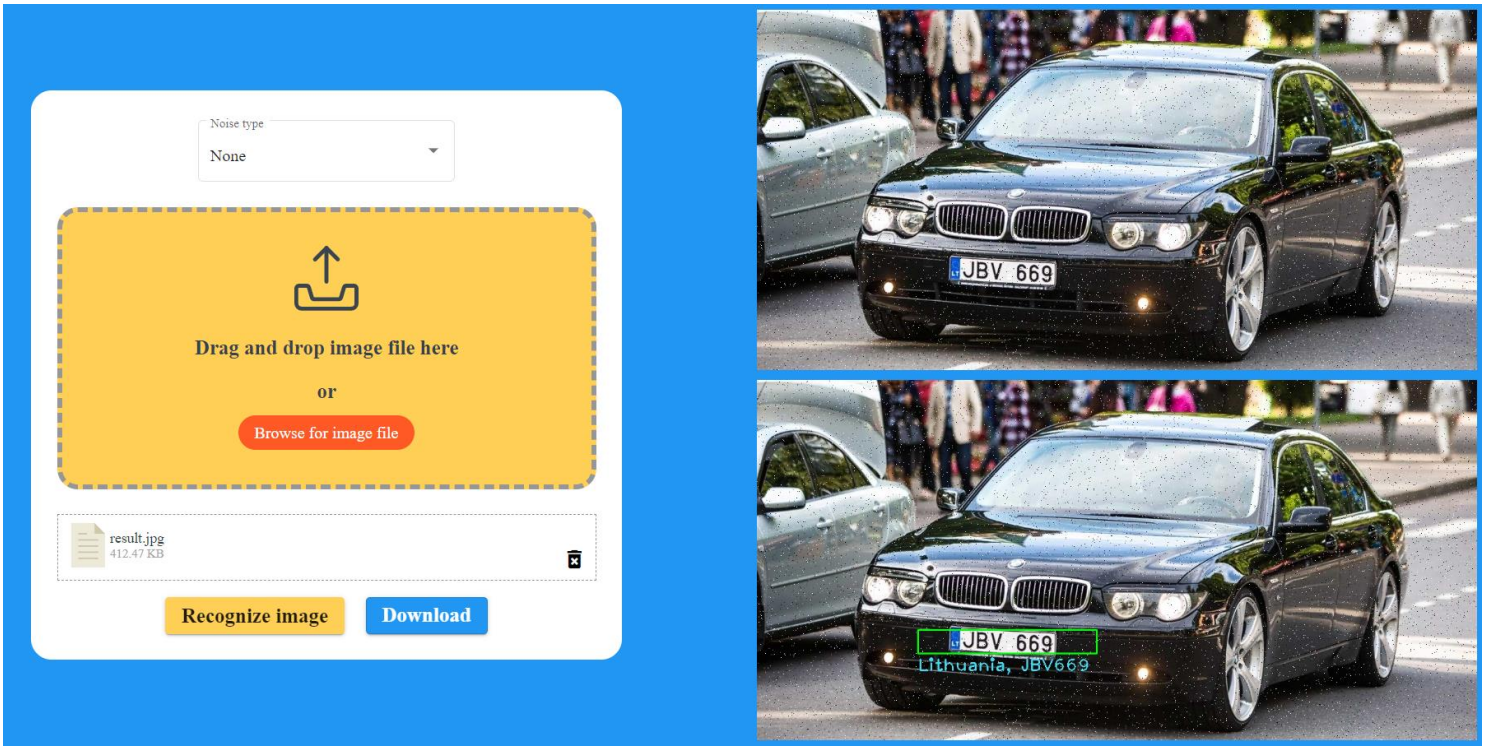


Рисунок 4.2 – Приклад результату роботи клієнтської частини програми з розпізнаним номерним знаком

4.2.2 Серверна частина

Серверна частина програми складається з наступних методів:

- main() – основний метод програми, який запускає роботу серверної частини програми;

- index() – метод-контролер, який починає роботу при отриманні запиту POST з клієнтської частини програми за адресою «http://localhost:5000/». В цьому методі відбувається зчитування даних, які були передані з клієнтської частини: тип шуму зображення та саме зображення, виклик методів для роботи з зображенням та повернення обробленого зображення назад до клієнтської частини. Тип шуму зображення може мати одне з наступних значень:

1) GAUSSIAN – на вхідне зображення буде накладено шум Гауса;

2) SALT_PAPER – на вхідне зображення буде накладено шум «солі та перцю»;

3) POISSON – на вхідне зображення буде накладено шум Пуассона;

4) SPECKLE – на вхідне зображення буде накладено спекл-шум;

5) NONE – вхідне зображення вже має шум та на ньому необхідно розпізнати номерний знак автомобіля.

- gaussian(img) – метод, який приймає в якості параметра зображення та виконує функції для накладання Гаусового шуму на вхідне зображення. В результаті повертає вхідне зображення з накладеним шумом;

- salt_paper(img) – метод, який приймає в якості параметра зображення та виконує функції для накладання шуму «солі та перцю» на вхідне зображення. В результаті повертає вхідне зображення з накладеним шумом;

- poisson(img) – метод, який приймає в якості параметра зображення та виконує функції для накладання шуму Пуассона на вхідне зображення. В результаті повертає вхідне зображення з накладеним шумом;

- specle(img) – метод, який приймає в якості параметра зображення та виконує функції для накладання спекл-шуму на вхідне зображення. В результаті повертає вхідне зображення з накладеним шумом;

- recognize(img) – метод, який приймає в якості параметра зображення та виконує послідовність необхідних дії для того, щоб виділити номерний знак автомобіля на вхідному зашумленому зображенні та розпізнати текст на ньому. В результаті повертає вхідне зображення з виділеним контуром номерного знаку автомобіля та текстом, який було розпізнано на ньому.

Далі буде розглянуто процес вилучення номерного знаку автомобіля з вхідного зображення та його розпізнавання [42].

Приклад зображення автомобіля з накладеним Гаусовим шумом, яке буде використовуватися для автоматичного розпізнавання номерного знаку представлено на рис. 4.3.

Рисунок 4.3 – Зашумлене зображення автомобіля, для якого буде виконуватись розпізнавання номерного знаку

Для збільшення контрастності зображення, що допомагає краще витягувати інформацію з зображення, необхідно виконати нормалізацію зображення за допомогою функції normalize() з бібліотеки cv2. Функція приймає наступні параметри:

- source_array – це масив, що відповідає вхідному зображенню, яке необхідно нормалізувати;

- target_array – це масив, що відповідає нормалізованому вихідному зображенню;

- alpha – являє собою нижнє граничне значення діапазону;

- beta – являє собою верхнє граничне значення діапазону;

- normalization_type – представляє тип нормалізації.

Для оптимізації пошуку номерного знаку, щоб не використовувати три прошарки RGB, а лише один сірий – зображення необхідно перевести до сірого формату. Для цього викликати метод cvtColor() з бібліотеки cv2 з наступними параметрами (img, cv2.COLOR_BGR2GRAY), де img – вхідне зображення,

cv2.COLOR_BGR2GRAY – перетворення до сірого формату. Результат нормалізації зображення та переведення його до сірого формату представлено на рис. 4.4.

Рисунок 4.4 – Результат нормалізації зображення та переведення його до сірого формату

Код програми для нормалізації вхідного зображення та переведення його до сірого формату наведено нижче:

```
img = cv2.normalize(src=img, dst=None, alpha=0, beta=255,
norm_type=cv2.NORM_MINMAX, dtype=cv2.CV_8U)
gray = cv2.cvtColor(img, cv2.COLOR_RGB2GRAY)
```

Далі, щоб зменшити шум зображення викликається метод cv2.medianBlur(src, dst, ksize), де src – зображення перетворене до формату сірого, dst – None та ksize – лінійний розмір апертури (має бути непарним і більше 1).

Після того, як зменшили шум на зображенні, потрібно знайти всі краї зображення. Для цього використовується метод cv2.Canny(image, threshold1, threshold2), де image – зображення зі зменшеним шумом, threshold1 та threshold2 –

перший та другий поріг гістерезису, чим більша різниця між ними, тим менше країв буде знайдено.

На рис. 4.5 представлено результат накладання фільтру та знаходження країв зображення.

Рисунок 4.5 – Результат накладання фільтру та знаходження країв зображення

Код програми для накладання фільтру та знаходження країв зображення наведено нижче:

```
imgFilter = cv2.medianBlur(gray, 3)
edges = cv2.Canny(imgFilter, 50, 200)
```

Коли знайшлися всі краї зображення, необхідно визначити всі контури. Контури є корисним інструментом для аналізу форми та виявлення / розпізнавання об’єктів. Для їх знаходження використовується функція cv2.findContours(image, mode, method), де image – зображення, mode – режим алгоритму пошуку контурів, method – алгоритм наближення контуру. Далі потрібно зчитати знайдені контури за

допомогою методу grab_contours() з бібліотеки imutils та відсортувати контури за допомогою вбудованого методу в Python sorted().

Знайдені контури представляються собою масив. Його потрібно перебрати у циклі і в ньому знайти форму контуру, яка максимально наближена на форму квадрата – це і буде контур номерного знаку. Для цього використовується метод cv2.approxPolyDP(curve, epsilon, closed=true), де curve – контур на і-тій ітерації, epsilon – параметр, що визначає точність наближення до квадрату, closed=true – апроксимована крива закрита (її перша і остання вершини з'єднані). Далі потрібно перевірити, якщо довжина отриманого контуру на і-тій ітерації складається з 4 елементів (координати 4 сторін) – це і є знайдений контур номерного знаку, який схожий на квадрат і відбувається вихід з циклу.

Програмна реалізація знаходження контурів номерного знаку представлена наступним кодом:

```
# знаходження контурів зображення
cont = cv2.findContours(edges.copy(), cv2.RETR_TREE,
cv2.CHAIN_APPROX_SIMPLE)
cont = imutils.grab_contours(cont) # зчитування контурів
# сортування контурів
cont = sorted(cont, key=cv2.contourArea, reverse=True)
pos = None
for c in cont:
     # Знаходження закритих контурів
     approx = cv2.approxPolyDP(c, 8, True)
     # Якщо довжина складається з 4 елементів, то
     # це координати 4 кутів квадрата
     if(len(approx)) == 4:
               pos = approx
               break
```

Після цього, потрібно виділити з загального зображення форму знайденого контуру номерного знаку. Для цього використовуються побітові операції та маски. Метод zeros(img, type) з бібліотеки NumPy використовується для створення маски, який приймає параметри img – це зображення сірого формату, type – тип даних. Цей метод повертає новий масив заданої форми та типу, заповнений нулями. За допомогою методу cv2. drawContours(image, contours, contourIdx, color thickness) відбувається рисування контурів. Де image – маска, contours – знайдений контур

номерного знаку, contourIdx=0, color – колір контурів, thickness – товщина ліній, якими промальовані контури. Потім потрібно виконати побітову операцію за допомогою методу cv2.bitwise_and(img, img, mask), щоб виділити контур знайденого номерного знаку з початкового зображення. Img – початкове завантажене зображення, mask – створена маска.

Код програми для виконання побітових операцій і накладання маски наведено нижче:

```
mask = np.zeros(gray.shape, np.uint8)
cv2.drawContours(mask, [pos], 0, 255, -1)
bitwise_img = cv2.bitwise_and(img, img, mask=mask)
```

Результат виконання побітової операції і накладання маски представлено на рис. 4.6.

Рисунок 4.6 – Результат виконання побітової операції і накладання маски

Тепер потрібно вирізати з зображення лише знайдений номерний знак. Для цього потрібно знайти координати x та y за допомогою методу where(mask == 255) з бібліотеки NumPy. Цей метод повертає 2 масиви з усіма точками для x та y

відповідно. З цих масивів потрібно знайти мінімальні та максимальні координати для точок (x1, y1) та (x2, y2) за допомогою методу min() та max() для координат x та y. Після цього можна обрізати зображення вказавши нові координати як gray_img[x1:x2, y1:y2], де gray_img – зображення сірого формату.

Так як номерний знак вирізається з початкового зображення, до нього знову необхідно застосувати фільтр для зменшення шуму на зображенні. Для цього використовується метод cv2.bilateralFilter(src, d, sigmaColor, sigmaSpace), який зменшує кількість точок на зображенні і оптимізує його. Де src – зображення, до якого застосовуються метод, d – діаметр кожного району пікселів, який використовується під час фільтрації, sigmaColor – колірний простір, чим більше значення, тим більше пікселів з однаковим кольором будуть змішуватися по кольору, sigmaSpace – координатний простір, чим більше значення, тим більше пікселів з однаковим кольором будуть змішуватися по координатам.

Для того, щоб збільшити точність розпізнавання символів на номерному знаку, необхідно збільшити вирізаний контур номерного знаку за допомогою методу cv2.resize(src, dsize, fx, fy, interpolation), де src – вирізане зображення номерного знаку, dsize = None, fx,fy – коефіцієнти масштабування по горизонтальній та вертикальній осях, interpolation – алгоритм інтерполяції.

Результат виконання операцій вирізання номерного знаку, зменшення шуму на ньому та його збільшення представлено на рис. 4.7.

Програмна реалізація вирізання номерного знаку, зменшення шуму на ньому та його збільшення наведено нижче:

```
(x, y) = np.where(mask == 255)
(x1, y1) = (np.min(x), np.min(y))
(x2, y2) = (np.max(x), np.max(y))
# Перетворення зображення до формату сірого
gr = cv2.cvtColor(img, cv2.COLOR_RGB2GRAY)
cropp = gr[x1:x2, y1 - 30:y2 + 70]
imgFilter = cv2.bilateralFilter(cropp, 11, 30, 50)
cropp = cv2.resize(imgFilter, None, fx=3, fy=3,
          interpolation=cv2.INTER_AREA)
```

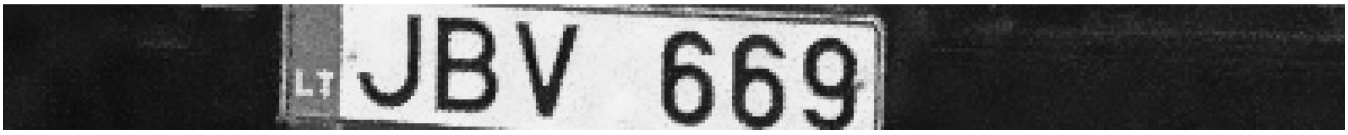


Рисунок 4.7 – Результат виконання операцій вирізання номерного знаку, зменшення шуму на ньому та його збільшення

Далі потрібно згладити отримане зображення за допомогою функції cv2.GaussianBlur(src, dst, ksize, sigmaX), де src – вирізане зображення номерного знаку, dst – вихідне зображення того самого розміру та типу, що й src, ksize – розмір ядра Гауса, sigmaX – стандартне відхилення ядра Гауса в напрямку X. Після цього відбувається збільшення чіткості отриманого зображення. Результат виконання операцій згладжування та збільшення чіткості зображення представлено на рис. 4.8.

Код програми для згладжування за збільшення чіткості зображення наведено нижче:

```
blur = cv2.GaussianBlur(cropp, (7, 7), 0)
kernel = np.array([[-1, -1, -1], [-1, 9, -1], [-1, -1, -1]])
im = cv2.filter2D(blur, -1, kernel)
```

Рисунок 4.8 – Результат згладжування та збільшення чіткості зображення

Після цього можна застосовувати функцію Reader(language).readtext(img) з бібліотеки easyocr для розпізнавання символів. Де language – мова розпізнавання, img – збільшене та інвертоване зображення. Команди розпізнавання символів номерного знаку наведено нижче:

```
text = easyocr.Reader(['en'])
text = text.readtext(im)
```

Результат розпізнавання тексу номерного знаку представлено на рис. 4.9.

```
[([[255, 53], [301, 53], [301, 89], [255, 89]], 'LT', 0.3824734330900871),
([[301, 0], [520, 0], [520, 112], [301, 112]], 'JBV', 0.999844111835554),
([[572, 9], [775, 9], [775, 117], [572, 117]], '669', 0.9974711599300832)]
```

Рисунок 4.9 – Результат розпізнавання тексу номерного знаку

Далі відбувається розбиття отриманих результатів і знаходження корисної інформації в цих даних. Також тут знаходиться код країни, який звіряється з європейськими реєстраційними знаками транспортних засобів, і на основному зображенні виділяється область номерного знаку, а під нею – розпізнаний текст: країна та сам номер.

Результат розпізнавання країни та номеру на номерному знаку представлено на рис. 4.10.

Рисунок 4.10 – Результат розпізнавання країни та номеру на номерному знаку

4.3 Результати експериментальних досліджень

Система для автоматичного розпізнавання зображень номерних знаків автомобілів в умовах шуму було протестована для різних вимірювань продуктивності та точності. Для розрахування коефіцієнту успішності роботи алгоритму програми застосовується наступна формула:

$$SR = \frac{NS_s}{TN_s} * 100, \tag{4.1}$$

де SR – коефіцієнт успішності роботи алгоритму, NS_s – кількість успішних зразків та TN_s – загальна кількість зразків зображень.

Результати аналізу точності роботи алгоритму розпізнавання представлені в таблиці 4.1.

Таблиця 4.1 – Результати аналізу точності роботи алгоритму розпізнавання

Назва операції	Загальна кількість зразків	Кількість успішних зразків	Кількість невдалих спроб	Коефіцієнт успішності (%)
Вилучення та виявлення номерного знаку	50	46	4	92
Розпізнавання тексту номерного знаку	46	44	2	95,65

Показник успішності вилучення та виявлення номерного знаку на зображенні становить 92%, що свідчить про досить високий показник роботи алгоритму. На етапі розпізнавання символів із 46 зображень номерних знаків було розпізнано 44 з них і точність розпізнавання склала 95,65%.

Для вилучення номерного знаку та його розпізнавання системі знадобилося в середньому 4 секунди. Ключовими факторами, які визначали продуктивність роботи алгоритму програми були розмір вхідного зображення, його чіткість та вид шуму, який було накладено на вхідне зображення. Графік залежності часу виконання алгоритму від розміру вхідного зображення наведено на рис. 4.11.

Рисунок 4.11 – Графік залежності часу виконання алгоритму від розміру вхідного зображення

Для порівняння результатів роботи алгоритмів вилучення та розпізнавання номерного знаку було обрано результати алгоритмів, представлені у публікаціях «Automatic Number Plate Recognition System Using OpenCV And Machine Learning» [43] та «Automatic License Plate Recognition using Python and OpenCV» [44].

Показники успішності вилучення та виявлення номерних знаків в обох випадках становлять 92%, а точність розпізнавання символів складає 94.3%, що на 1.35% менше ніж в запропонованому алгоритмі кваліфікаційної роботи.

Для вилучення номерного знаку та його подальшого розпізнавання системі, запропонованій у першій публікації знадобилося більше двадцяти секунд, а системі

з другої – більше 18 секунд. Але подальші дослідження з оптимізації алгоритму та коду знизили час роботи алгоритму, запропонованому у другій публікації, до двох секунд. Ключовими факторами, які визначали продуктивність роботи алгоритмів, були розмір вхідного зображення та його чіткість.

Порівнюючи розглянуті алгоритми та алгоритм, запропонований у кваліфікаційній роботі, можна зробити висновок, що всі системи з високою точністю здатні вилучати та розпізнавати номерні знаки автомобілів. Тривалість роботи алгоритму, реалізованому у кваліфікаційній роботі складає в середньому від 2 до 6 секунд і може бути зменшено у подальшій оптимізації роботи алгоритму. Крім того, запропонований алгоритм дозволяє розпізнавати зображення номерних знаків автомобілів за наявності таких шумів зображення як шум Гауса, шум «солі та перцю», спекл-шум або шум Пуассона.

Тестування запропонованого алгоритму для розпізнавання зображень номерних знаків автомобілів в умовах шуму виконувалося з використанням мобільного процесору Intel Core i5-8300h без використання графічних ядер CUDA.

ВИСНОВКИ

В результаті виконання кваліфікаційної роботи розроблено клієнт-серверний додаток за допомогою фреймворку Angular для клієнтської частини та мови програмування Python і мікрофреймворку Flask для серверної, який дозволяє накласти шум на вибране зображення або автоматично розпізнати зображення номерного знаку автомобіля за допомогою технології оптичного розпізнавання символів за наявності шуму.

У першому розділі «Актуальність задачі та аналіз предметної області» розглянуто актуальність обраної теми кваліфікаційної роботи, оглянуто наукові публікації, які стосуються теми кваліфікаційної роботи та існуючі рішення на ринку. Також в цьому розділі було сформовано мету та задачі кваліфікаційної роботи.

У розділі «Комп’ютерний зір та розпізнавання образів» було розглянуто коротку історію комп’ютерного зору, поняття технологій комп’ютерного зору та напрямки застосування. Також тут були розглянуті алгоритми та приклади використання. Окремо було розглянуто технологію оптичного розпізнавання символів, принцип роботи, варіанти використання, переваги та недоліки.

У третьому розділі «Шум зображення» було розглянуто типи цифрового зображення, причини виникнення шуму зображення та його типи, а також засоби зменшення шуму на зображенні. Так було наведено типи фільтрів для зменшення різноманітних шумів. Фільтр Гауса доцільно використовувати, коли на зображенні наявний шум Гауса або спекл-шум. Медіанний та консервативний фільтри можна використовувати для зменшення шуму «солі та перцю», а двосторонній фільтр (bilateral filter) варто використовувати, коли наявний шум Пуассона.

У розділі «Програмна реалізація» було описано технології розробки та інструментальні засоби, які використовувалися для створення веб-додатку. Так для реалізації клієнтської частини додатку було використано фреймворк Angular, а для реалізації серверної частини – мікрофреймворк Flask та мову програмування Python. Також були використані бібліотеки OpenCV, NumPy, Imutils та EasyOCR. В цьому

розділі було також описано структуру проекту та які модулі для чого використовуються, описано процес автоматичного розпізнавання зображення номерного знаку автомобіля в умовах шуму. Наприкінці розділу було наведено результати експериментальних досліджень щодо роботи програми та порівняння їх з результатами аналогічних рішень.

За результатами роботи можна зробити висновок, що програма з високою точністю та швидкістю розпізнає зображення номерних знаків автомобілів за наявності шуму, за умов чіткого загального зображення та розташування номерного знаку на передньому плані.

В подальшому можливе вдосконалення системи розпізнавання зображень номерних знаків автомобілів та зменшення шуму на ньому, використовуючи згорткові нейронні мережі, що має збільшити точність та якість розпізнавання.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Shan Du, Mahmoud Ibrahim, Mohamed Shehata and Wael Badawy. Automatic License Plate Recognition (ALPR): A State-of-the-Art Review. IEEE Transactions on circuits and systems for video technology, vol. 23, no. 2, February 2013.

2. M. Dong, D. He, C. Luo, D. Liu, W. Zeng. A CNN-Based Approach for Automatic License Plate Recognition in the Wild. British Machine Vision Conference, London, January 2017.

3. Axis. Automated license plate recognition. URL: https://www.axis.com/solutions/license-plate-recognition (дата звернення: 20.09.2022).

4. NumberOk. License plates recognizer. URL: https://number-ok.com (дата звернення: 23.09.2022).

5. История компьютерного зрения. URL: https://shalaginov.com/2020/05/16/computer-vision-history/ (дата звернення: 25.09.2022).

6. What is computer vision? URL: https://www.ibm.com/topics/computer-vision (дата звернення: 26.09.2022).

7. Омельченко С. О., Сердюк Н. М. Використання комп'ютерного зору для розпізнавання образів. Глобалізація наукових знань: міжнародна співпраця та інтеграція галузей наук: матеріали II Міжнародної студентської наукової конференції (Т. 1), м. Біла Церква, 22 жовтня, 2021 рік / ГО «Молодіжна наукова ліга». – Вінниця: ГО «Європейська наукова платформа», 2021. – 130 с. URL: https://ojs.ukrlogos.in.ua/index.php/liga/issue/view/23.10.2021/618.

8. Object Detection Guide. URL: https://www.fritz.ai/object-detection/#:~:text=Object%20detection%20is%20a%20computer,all%20while%20accurately%20labeling%20them (дата звернення: 27.09.2022).

9. What is event detection? URL: https://chooch.ai/computer-vision/what-is-event-detection/ (дата звернення: 28.09.2022).

10. What is motion tracking? URL: https://chooch.ai/computer-vision/what-is-motion-tracking/ (дата звернення: 29.09.2022).

11. Computer Vision For Healthcare. URL: https://chooch.ai/computer-vision/ (дата звернення: 29.09.2022).

12. A Gentle Introduction to Object Recognition With Deep Learning. URL: https://machinelearningmastery.com/object-recognition-with-deep-learning/ (дата звернення: 30.09.2022).

13. Understanding Pattern Recognition in Machine Learning. URL: https://www.section.io/engineering-education/understanding-pattern-recognition-in-machine-learning/ (дата звернення: 30.09.2022).

14. What is Pattern Recognition? A Gentle Introduction (2021). URL: https://viso.ai/deep-learning/pattern-recognition/ (дата звернення: 2.10.2022).

15. Bezsonov O., Ilyunin O., Khusanov A., Rudenko O., Oleg Sotnikov O. Intelligent Identification System of the Process Liquid Solutions Composition. Proceedings of the 6th International Conference on Computational Linguistics and Intelligent Systems (COLINS 2022). Volume I: Main Conference, Gliwice, Poland, May 12-13, 2022. P.960–973.

16. What is Optical Character Recognition (OCR) Technology? URL: https://www.hyland.com/en/resources/terminology/data-capture/what-is-optical-character-recognition-ocr (дата звернення: 05.10.2022).

17. What is Optical Character Recognition (OCR) and What Does It Do? URL: https://docparser.com/blog/what-is-ocr/ (дата звернення: 07.10.2022).

18. What Is Optical Character Recognition (OCR)? URL: https://www.ibm.com/cloud/blog/optical-character-recognition (дата звернення: 13.10.2022).

19. Optical character recognition (OCR). URL: https://searchcontentmanagement.techtarget.com/definition/OCR-optical-character-recognition (дата звернення: 20.10.2022).

20. Discreet Disadvantages of Optical Character Recognition. URL: https://theecmconsultant.com/disadvantages-of-optical-character-recognition/ (дата звернення: 06.10.2022).

21. Automatic Number Plate Recognition (ANPR). URL: https://viso.ai/computer-vision/automatic-number-plate-recognition-anpr/ (дата звернення: 25.10.2022).

22. Digital image. URL: https://support.esri.com/en/other-resources/gis-dictionary/term/53fc751d-a51b-4b40-a081-5d57ecc9904b (дата звернення: 27.10.2022).

23. V. M. Mohan, R. K. Durga, S. Devathi and K. Srujan Raju. Image Processing Representation UsingBinary Image; Grayscale, Color Image and Histogram. Proceedings of the Second International Conference on Computer and Communication Technologies (pp.353-361).

24. Бинарное изображение. URL: https://www.wiki.ru-ru.nina.az/Бинарное_изображение.html (дата звернення: 27.10.2022).

25. Grayscale Images. URL:https://homepages.inf.ed.ac.uk/rbf/HIPR2/gryimage.htm (дата звернення: 27.10.2022).

26. Color Images. URL: https://homepages.inf.ed.ac.uk/rbf/HIPR2/colimage.htm (дата звернення: 27.10.2022).

27. A beginner's guide to image noise. URL: https://www.pixop.com/blog/image-noise-causes (дата звернення: 29.10.2022).

28. A Guide to Different Types of Noises and Image Denoising Methods. URL: https://analyticsindiamag.com/a-guide-to-different-types-of-noises-and-image-denoising-methods/ (дата звернення: 29.10.2022).

29. Gaussian noise. URL: https://en.wikipedia.org/wiki/Gaussian_noise (дата звернення: 31.10.2022).

30. J. Jaybhay, R. Shastri. A study of speckle noise reduction filters. Signal & Image Processing: An International Journal (SIPIJ) Vol.6, No.3, June 2015.

31. Poisson. URL: https://docs.juliahub.com/Noise/qByhT/0.2.0/man/poisson/ (дата звернення: 01.11.2022).

32. Film grain. URL: https://en.wikipedia.org/wiki/Image_noise#Film_grain (дата звернення: 01.11.2022).

33. What is Angular? Architecture, Features, and Advantages. URL: https://www.simplilearn.com/tutorials/angular-tutorial/what-is-angular (дата звернення: 05.11.2022).

34. Flask. URL: https://en.wikipedia.org/wiki/Flask_(web_framework) (дата звернення: 07.11.2022).

35. Python – что это такое. URL: https://mchost.ru/articles/chto-takoe-python/ (дата звернення: 08.11.2022).

36. Возможности PyCharm. URL: https://www.jetbrains.com/ru-ru/pycharm/features/ (дата звернення: 09.11.2022).

37. About OpenCV. URL: https://opencv.org/about (дата звернення: 10.11.2022).

38. OpenCV – Overview. URL: https://www.geeksforgeeks.org/opencv-overview/ (дата звернення: 11.11.2022).

39. What is NumPy? URL: https://numpy.org/doc/stable/user/whatisnumpy.html (дата звернення: 12.11.2022).

40. Imutils. URL: https://pypi.org/project/imutils/ (дата звернення: 13.11.2022).

41. Easyocr. URL: https://pypi.org/project/easyocr (дата звернення: 14.11.2022).

42. Омельченко С.О., Сердюк Н.М. Використання технології OCR для автоматичного розпізнавання європейських номерних знаків. Цифровізація науки та сучасні тренди її розвитку: матеріали II Міжнародної студентської наукової конференції (Т. 1), м. Миргород, 5 листопада, 2021 рік / ГО «Молодіжна наукова ліга». – Вінниця: ГО «Європейська наукова платформа», 2021. – 116 с. URL: https://ojs.ukrlogos.in.ua/index.php/liga/issue/view/inter-05.11.2021/629.

43. Basil B., Jikku J., Lithin T., ShellyShiju G. Automatic Number Plate Recognition System Using OpenCV And Machine Learning. National Conference in Emerging Computer Applications (NCECA2019).

44. K.M. Sajjad. Automatic License Plate Recognition using Python and OpenCV. Department of Computer Science and Engineering M.E.S. College of Engineering, Kuttippuram, Kerala.