Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

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

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти перший (бакалаврський)

РОЗРОБКА СИСТЕМИ РОЗПІЗНАВАННЯ ЖЕСТІВ РУК
(тема)

Виконав:
студент 4 курсу, групи ІТІНФ-19-1

Слободенюк І.Ф.
(прізвище, ініціали)

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

Тип програми освітньо-професійна

Освітня програма Інформатика
(повна назва освітньої програми)

Керівник ст. викл. Путятіна О.Є.
(посада, прізвище, ініціали)

Допускається до захисту

Зав. кафедри _____ Кобилін О.А.
(підпис) (прізвище, ініціали)

2023 р.

Харківський національний університет радіоелектроніки

Факультет Інформаційно-аналітичних технологій та менеджменту
(повна назва)

Кафедра Інформатики
(повна назва)

Рівень вищої освіти перший (бакалаврський)

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

Тип програми освітньо-професійна

Освітня програма Інформатика
(повна назва освітньої програми)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
«____»_____ 2023 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

студентові Слободенюку Івану Федоровичу
(прізвище, ім'я, по батькові)

1. Тема роботи Розробка системи розпізнавання жестів рук

затверджена наказом університету від 15 травня 2023 року № 474 Ст

2. Термін подання студентом роботи до екзаменаційної комісії 29 травня 2023 р.

3. Вихідні дані до роботи методична та технічна література, матеріали конференцій, дані інтернет-мережі, бібліотека комп'ютерного зору з відкритим кодом OpenCV, бібліотека машинного навчання MediaPipe.

4. Перелік питань, що потрібно опрацювати в роботі
1. Огляд методів розпізнавання жестів рук та аналіз їх можливостей.
2. Аналіз технічних засобів розробки.
3. Проєктування процесу розпізнавання жестів.
4. Розробка застосунку для розпізнавання жестів у режимі реального часу.
5. Тестування системи розпізнавання жестів рук.

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (п.5 включається до завдання за рішенням випускової кафедри) Актуальність проблеми обробки зображень, постановка задачі, тестові зображення.

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

Найменування розділу	Консультант (посада, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата
Консультант з дотримання діючих стандартів та норм	Доцент Творошенко І.С.		

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Терміни виконання етапів роботи	Примітка
1	Отримання завдання на кваліфікаційну роботу	10.04.2023	
2	Аналіз завдання, підбір літератури	11.04.23-17.04.23	
3	Аналіз літератури з досліджуваної проблеми	18.04.23-20.04.23	
4	Аналіз технічних засобів	21.04.23-30.04.23	
5	Розробка методу	01.05.23-14.05.23	
6	Програмна реалізація	15.05.23-23.05.23	
7	Оформлення пояснювальної записки	24.05.23-26.05.23	
8	Перевірка на плагіат	27.05.23	
9	Рецензування	28.05.23	
10	Підготовка презентації та доповіді	29.05.23-30.05.23	
11	Занесення роботи в електронний архів	31.05.23	
12	Попередній захист кваліфікаційної роботи	07.06.23	

Дата видачі завдання 10 квітня 2023 р.

Студент _____
(підпис)

Керівник роботи _____ ст. викл. Путятіна О.Є.
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ/ABSTRACT

Пояснювальна записка до кваліфікаційної роботи: 61 с., 27 рис., 39 джерел.

КОМП'ЮТЕРНИЙ ЗІР, РОЗПІЗНАВАННЯ ЖЕСТІВ, НЕЙРОННА МЕРЕЖА, ДАКТИЛЬНИЙ АЛФАВІТ, OPENCV, PYTHON.

Об'єктом роботи є послідовність відеокадрів, зображуючих конфігурації жестів рук.

Метою роботи є розробка системи, що базуються на використанні комп'ютерного зору, яка дозволяє у реальному часі детектувати ознаки та класифікувати жести рук для перетворення їх у простий текст.

У даній роботі спроєктовано і розроблено застосунок з використанням нейронної мережі та комп'ютерного зору, що реалізує розпізнавання дактильного алфавіту. Комплексно розглянуто теми дактильного алфавіту, мови жестів та її структури. Проведено аналіз методів розпізнавання жестів, класифікації зображень та відео, детектування та обробка конфігурацій для потреб перекладу у текст. Для розпізнавання жестів та обробки зображень застосовано бібліотеку комп'ютерного зору OpenCV. Код програми написано мовою програмування Python.

У результаті роботи здійснена програмна реалізація системи для розпізнавання жестів рук.

COMPUTER VISION, GESTURE RECOGNITION, NEURAL NETWORK, MANUAL ALPHABET, OPENCV, PYTHON.

The object of the work is the process of gesture recognition corresponding to finger alphabet signs.

The aim is to develop a system that, using computer vision, can detect and classify hand gestures for converting them into simple text.

In this work, an application was designed and developed using a neural network and computer vision that implements the recognition of the manual alphabet. The topics of the finger alphabet, sign language, and its structure were comprehensively considered. An analysis was conducted of methods for gesture recognition, image and video classification, configuration detection, and processing for translation into text. The OpenCV computer vision library was used for gesture recognition and image processing. The program code was written in the Python programming language.

As a result of the work, a software implementation of a system for recognizing hand gestures has been carried out.

ЗМІСТ

Перелік умовних позначень, символів, одиниць, скорочень і термінів 6
Вступ 7
1 Аналіз предметної області розпізнавання жестів 9
1.1 Загальна інформація про мову жестів 9
1.2 Структура мови жестів 11
1.3 Структура системи розпізнавання жестів рук 12
1.4 Огляд існуючих методів розпізнавання жестів 14
1.4.1 Розпізнавання з використанням трекерів 15
1.4.2 Розпізнавання з використанням комп'ютерного зору 17
1.4.3 Розпізнавання з використанням нейронних мереж 19
1.5 Постановка задачі 20
2 Проєктування системи розпізнавання жестів 23
2.1 Вибір мови програмування 23
2.2 Формування ієрархічної моделі класів згідно з SOLID 25
2.3 Створення власного набору даних 28
2.4 Проєктування процесу розпізнавання жестів 30
2.5 Застосування алгоритму класифікації 36
3 Розробка та тестування застосунку 38
3.1 Розробка прототипу 38
3.2 Реалізація застосунку 41
3.2.1 Mediapipe та OpenCV 42
3.2.2 Алгоритм створення датасету 44
3.2.3 Вихідна класифікація 47
3.3 Тестування застосунку 51
3.4 Інструкція користувача 54
Висновки 57
Перелік джерел посилання 58

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

КЗ – комп’ютерний зір

ШНМ – штучна нейронна мережа

ЗНМ – згорткова нейронна мережа

РЖ – розпізнавання жестів

ДА – дактильний алфавіт

SOLID – акронім п’яти принципів дизайну програмного забезпечення

ВСТУП

Світ безперервно змінюється та розвивається завдяки науковим відкриттям і технологічному прогресу. У сучасному інформаційному суспільстві технології стали невід'ємною частиною життя людей. Одним з найбільш визначальних технологічних явищ є розвиток мобільних і комп'ютерних застосунків, що приводить до появи великої кількості можливостей для використання у повсякденному житті. Вони допомагають у всьому – від повсякденних завдань до довгострокових цілей. З цією метою і розробляються спеціальні сервіси з функціональними можливостями, котрі значно відрізняються в залежності від області їх застосування.

В останні роки значний прогрес був зроблений в галузі комп'ютерного зору та машинного навчання, які дозволяють розв'язувати складні задачі розпізнавання образів. Одним з напрямів, де такі технології можуть знайти своє застосування, є розпізнавання жестів рук. Це особливо актуально в контексті розвитку інтерактивних систем управління комп'ютером та роботами, передачі інформації, тощо. Розпізнавання жестів рук має багато викликів, пов'язаних зі змінністю зображень та різноманітністю виконання жестів різними користувачами. Для вирішення цих викликів можна використовувати різноманітні методи машинного навчання, зокрема нейронні мережі та глибоке навчання.

Буде розглянуто створення системи розпізнавання з використанням методів машинного навчання та комп'ютерного зору. Планується провести дослідження різних методів розпізнавання жестів рук та їхніх комбінацій з метою вибору найбільш оптимального підходу для нашої системи. Результатом роботи буде програмна система, яка зможе розпізнавати жести рук з високою точністю та швидкістю, що може мати практичне застосування в різних сферах.

Актуальність роботи полягає у тому, що зараз українське суспільство стикається з проблемами доступності інформації та комунікації для людей з різними видами обмежень, включаючи людей з вадами слуху та мовлення. Розробка системи розпізнавання жестів рук може допомогти забезпечити доступ до інформації та комунікації для цієї групи людей.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ РОЗПІЗНАВАННЯ ЖЕСТІВ

1.1 Загальна інформація про мову жестів

Мова жестів – це природна мова, яка допомагає людям спілкуватися без залучення голосового слова, і тому найчастіше використовується людьми з вадами слуху або мовлення [1].

Комунікація з допомогою жестів або рухів тіла існувала протягом більшості історії людства, з посиланнями на це, що датуються ще з часів Стародавньої Греції V століття до н.е. Однак розвиток повноцінної мови без вживання вербального мовлення має свої корені в мовах жестів сільських спільнот. У спільнотах з високим рівнем генетичної глухоти, мови жестів розвинулися в метод комунікації, який міг бути зрозумілим та доступним кожному члену спільноти, не тільки глухим або людям з вадами слуху. Багато ранніх шкіл для глухих намагалися придушити мову жестів, замість цього намагаючись змусити людей з вадами слуху вивчити мовлення. Як результат, перші витоки мови жестів розвинулися як природна мова, а не формалізована мова.

Опублікована в Лондоні у 1698 році, книга Digiti-Lingua є визначною подією в історії мови жестів. Вона надала найраннішу форму сучасного двохручного алфавіту та перші зображення приголосних жестів, які широко використовуються в мові жестів [2].

Анонімний автор цього 30-сторінкового брошуру твердив, що не використовував жодної іншої мови, «будучи змушений (через нещасний недуг) до таких або подібних методів спілкування протягом останніх десяти років». Книга також стверджувала, що показує «як будь-які двоє людей можуть здійснювати розмову лише пальцями, і як у темряві, так і на світлі». Не зроблено явної посилання на глухих людей і насправді не зовсім зрозуміло, чи автор сам був глухим. Він запропонував використовувати свою книгу, коли

потрібна тиша і таємниця, або просто для розваг. Розкладна таблиця надавала чіткі зображення, які показували, як здійснювати жести.

Після цього з'явилися докази розвитку повноцінних мов жестів у Франції в XVIII столітті та в колоніях Америки (пізніше Сполучених Штатах) у XIX столітті.

Сьогодні на світі використовують понад 300 мов жестів, серед яких найвідомішою є американська мова жестів, особливо поширена в США та більшості англомовних країн, французька мова жестів та британська мова жестів. Крім того, мова жестів є однією з найпопулярніших других (або третіх) мов, яку люди обирають вивчати. Українська жестова мова належить до сім'ї французької жестової мови.

Зростаюча популярність мов жестів в усьому світі свідчить про значний потенціал розробки систем розпізнавання жестів рук для полегшення комунікації людей з обмеженими можливостями слуху та мовлення.

Порушення слуху у зрілому віці виникають і приблизно у кожної третьої людини старше 65 років. Найчастіше – в Південній Азії і в Африці на південь від Сахари. В Україні, згідно даних ВООЗ, порушення слуху різного ступеня є у 2,1 мільйонів громадян; з них приблизно для 230 тисяч основним або одним з основних способів комунікації є жестова мова [3].

Українська дактильна абетка – допоміжна система української жестової мови, в якій кожному жесту однієї руки (зазвичай нерухомою правою, зігнутою в лікті) відповідає літера української мови; багато дактилем зовні схожі на відповідні літери української абетки (рис. 1.1). Промовляння ведеться за правилами української орфографії.

Дактилологія відрізняється від звичайних жестів, що позначають поняття або комплекс понять. Дактилеми використовуються для промовляння допоміжних слів, слів, у яких відсутнє жестове позначення («дефрагментація», «геном», деяких топонімів та власних назв), а також у разі, якщо потрібно прояснити значення того чи іншого слова [4].

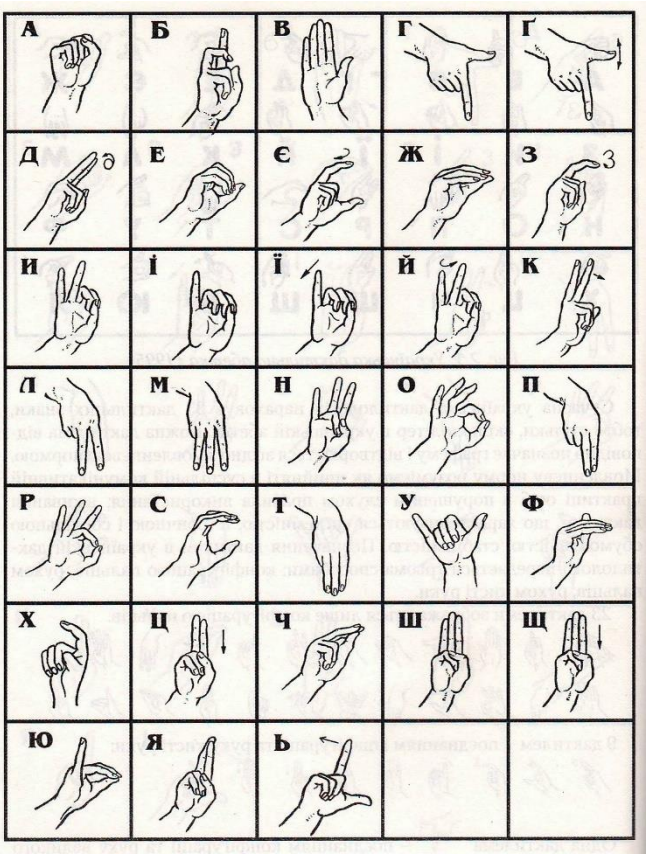


Рисунок 1.1 – Українська дактильна абетка

1.2 Структура мови жестів

Жест може поділятись на 4 компонента: рух, місце виповнення, орієнтація та конфігурація руки. Важливим моментом є врахування виразу обличчя та губ, тобто міміки [5]. Коли людина стикається з незнайомим словом, чи якщо для нього ще не придумали жесту, то тоді вибір спадає на використання дактильної абетки. Чуючі люди можуть сприймати за жестову мову пальцевий алфавіт. Однак ці речі різні: жестові конфігурації представляють ціле слово, або фразу, а пальцевим алфавітом передаються звуки, літери (у порядку як у слові).

Сучасна українська ДА нараховує 33 дактильних знаки, тобто стільки ж, скільки літер в українській абетці. Кожна дактилема відповідно позначає графему і відтворюється згідно з мовленнєвою нормою. Мовленнєву норму розуміємо як прийняті в суспільній комунікативній практиці осіб із

порушеним слухом правила використання, вживання дактилем, що характеризуються системністю, історичною і соціальною обумовленістю, стабільністю [4]. Позначення дактилем в українській дактилології передається трьома способами:

– конфігурацією пальців;
– рухом пальців;
– рухом п’ястку руки.

23 дактилеми, такі як: А, Б, В, Г, Е, Ж, И, І, Л, М, Н, О, П, Р, С, Т, У, Ф, Х, Ш, Ч, Ю, Я – зображаються лише конфігурацією пальців. 9 дактилем: Д, Є, З, Ї, Й, К, Ц, Щ, Ь – через поєднання руху п’ястку руки та конфігурації: Одна дактилема Ґ – поєднанням конфігурації та руху великого пальця.

Коли мова йде про граматику, мова жестів використовує структуру «тема-коментар». Тема пов’язана з підметом. У мові жестів цю тему можна підписати в початку або в кінці речення. Фактично, жестомовець може використовувати тему в обох місцях і все ще буде граматично правильно. Зазвичай темою є займенник. На прикладі американської мови, у мові жестів фразу «I am an employee» («Я є працівником») можна представити так: «I employee» («Я працівник»), «Employee I» («Працівник я»), «I employee I» («Я працівник я»). Іноді до речення мови жестів додається третя частина – час, або часова форма [6].

1.3 Структура системи розпізнавання жестів рук

У контексті комп’ютерного зору та розпізнаванні образів, жест – це форма невербального спілкування, в якій рухи тіла людини або його частини використовуються для передачі інформації. Він може містити дію як і однією, так і декількома частинами людського тіла, й нести емоційне навантаження, що робить його інформаційним засобом взаємодії.

Жести рук є аспектом мови тіла, який може бути переданий через центр долоні, позицію пальців та форму, створену рукою. Жести рук можуть бути класифіковані на статичні та динамічні. Як і підказує його назва, статичний жест відноситься до стабільної форми руки, тоді як динамічний жест складається з серії рухів руки, таких як махання. У межах жесту є різноманітні рухи руки; наприклад, рукостискання відрізняється від людини до людини та змінюється відповідно до часу та місця. Основна відмінність між поставою та жестом полягає в тому, що постава більше фокусується на формі руки, тоді як жест фокусується на руху руки. Основні підходи до дослідження жестів рук можуть бути класифіковані на основі датчика, який використовується: оснований на датчику носимої рукавички та оснований на датчику зображення камери [6].

Класифікація жестів також є і за способом обробки інформації. Статичні жести, для яких важлива фінальна конфігурація, відносятся до тих, що обробляються після введення інформації. Для динамічних жестів обов’язковою є постійна обробка під час виконання дії.

Спрощена система розпізнавання жестів рук складається з таких основних етапів (рис. 1.2):

Крок 1. Введення потоку інформації.

Крок 2. Перетворення жесту у цифрову форму.

Крок 3. Застосування конкретного методу розпізнавання образів.

Крок 4. Результат розпізнавання жесту.

Пристрій введення інформації у систему є точкою входу. Він перетворює жест у цифрову форму і передає його у метод розпізнавання, який порівнює отриману інформацію з шаблонами, які знаходятся у базі даних. У результаті метод повертає розпізнаний жест, який, відповідно до завдання, можна використовувати у наступних процесах. Наприклад, сенсор Kinect використовується для взаємодії з ігровою приставкою Xbox 360 через усні команди, пози тіла і жести рук [7]. Такий підхід дозволяє використовувати

жести для управління відеогрою, здійснюючи, на основі введеної інформації, відповідні рухи персонажем.

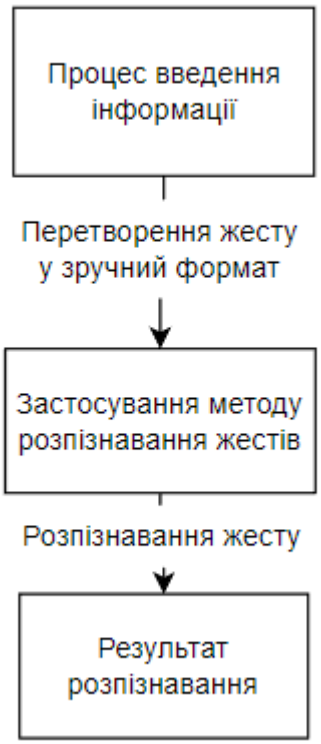


Рисунок 1.2 – Спрощена структура системи розпізнаваня жестів рук

1.4 Огляд існуючих методів розпізнавання жестів

Зараз на ринку існує багато технологій та рішень для розпізнавання жестів, які використовують різні методи та підходи.

Одним з популярних підходів є використання носимих датчиків, таких як рукавиці [8, 9], що містять акселерометри та гіроскопи. Ці датчики дозволяють отримати точні дані щодо рухів рук та жестів. Однак, використання таких датчиків може бути не зручним та незручним у використанні.

Іншим підходом є використання візуальних датчиків, зокрема камер [10, 11], для отримання зображень жестів. Найпоширеніші методи в цьому напрямку включають в себе аналіз кольору, форми та текстури руки та її рухів.

Однак, ці методи можуть бути відносно складними та залежними від умов освітлення та фону.

Третім підходом є використання комбінованих методів, які поєднують у собі як візуальні, так і носимі датчики. Це дозволяє отримувати більш точні та надійні дані щодо жестів [12].

Одним з найбільш перспективних методів отримання інформації про жести є електроміографія (ЕМГ). Отримані сигнали є вхідними даними для методів класифікації жестів. Перевагами систем розпізнавання жестів на основі ЕМГ є малі масогабаритні параметри та можливість реалізації таких систем у вигляді портативних пристроїв [13].

Методи, засновані на аналізі зовнішніх ознак жесту, аналізують загалом тільки зовнішній вигляд (позиції, форму, і т.д.) об’єкта. Для розпізнавання найчастіше не потрібне знання про фізичні властивості даного об’єкту, тому ця інформація не зберігається.

1.4.1 Розпізнавання з використанням трекерів

Одним з вдалих прикладів використання трекерів можна вважати «Language of Glove» – це проєкт студентів з каліфорнії, представлений у вигляді автономної рукавички для сурдоперекладу (рис. 1.3). Вона здатна бездротово перекладати американську жестову мову в текст, що може бути відображений на комп’ютері або смартфоні.

Основними компонентами пристрою є датчики напруження, що складаються з п’єзорезистивного композиту з частинками вуглецю, вбудованими у фтореластомер. Ці датчики інтегровані з носимим електронним модулем, що складається з дигітайзерів, мікроконтролерів та Bluetooth-радіо. Кінцевий аналіз передбачає пікову напруженість на датчиках в розмірі 5%, коли суглоби повністю зігнуті [13].

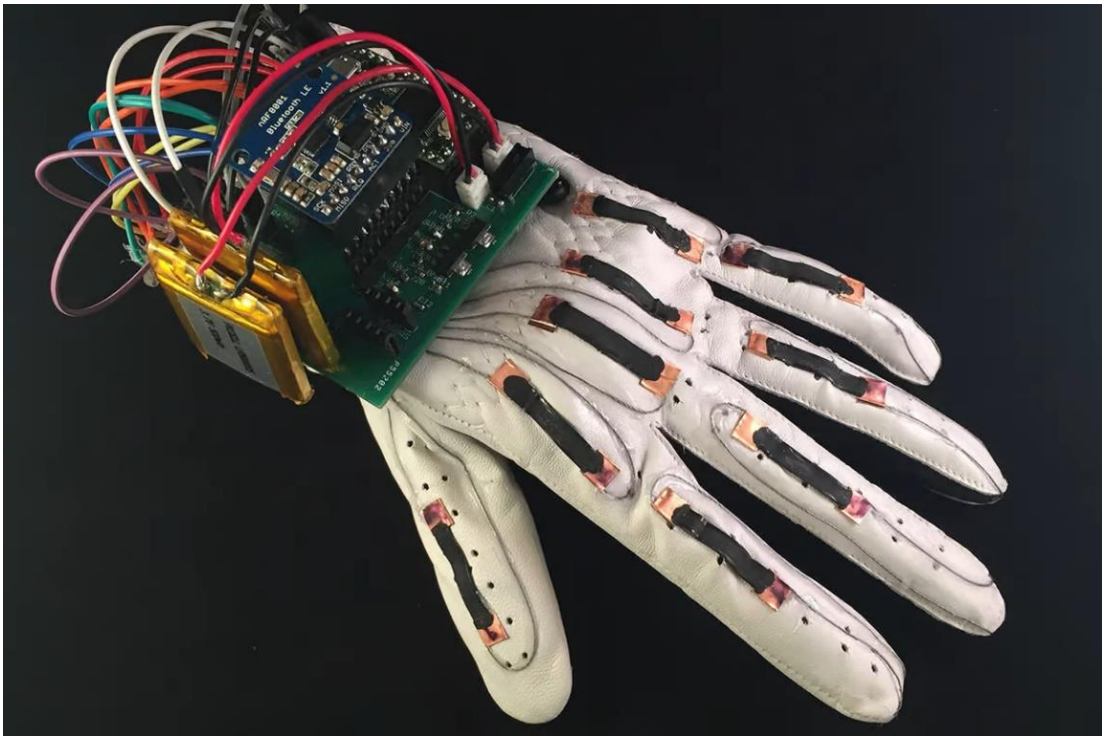


Рисунок 1.3 – Проєкт «Language of Glove»

Прикладом успішних візуальних датчиків, зокрема камер, є сенсор Kinect. Kinect (раніше Project Natal) – безконтактний сенсорний ігровий контролер, спочатку представлений для консолі Xbox 360, і значно пізніше для персональних комп'ютерів під керуванням Windows показано на рисунку 1.4. Він складається з двох сенсорів глибини, кольорової відеокамери та мікрофонної решітки. Датчик глибини складається з інфрачервоного проєктора, об'єднаного з монохромною матрицею, що дозволяє датчику Kinect отримувати тривимірне зображення при майже будь-якому природному та штучному освітленні [10].

Камера 3D, така як Kinect, надає необхідну інформацію для 3D-реконструкції. Для кожного 2D-пікселя на площині зображення вона не тільки записує колір цього пікселя, тобто колір початкової 3D-точки, але й відстань цієї 3D-точки вздовж її проекційної лінії. Знаючи проекційну лінію 2D-пікселя та відстань вздовж цієї проекційної лінії, кожен піксель проєктується назад у 3D-простір, що ефективно відтворює початково захоплені 3D-об'єкти [14].

Рисунок 1.4 – Безконтактний сенсорний ігровий контролер Kinect

1.4.2 Розпізнавання з використанням комп’ютерного зору

Комп’ютерний зір (КЗ) є галуззю штучного інтелекту, що дозволяє комп’ютерам та системам отримувати змістовну інформацію з цифрових зображень, відео та інших візуальних даних, а також вживати заходи або робити рекомендації на основі цієї інформації. Якщо штучний інтелект дозволяє комп’ютерам мислити, то комп’ютерне зорове сприймання дозволяє їм бачити, спостерігати і розуміти.

КЗ навчає машини виконувати ці функції, але з меншим часом, використовуючи камери, дані та алгоритми, замість сітківок, зорових нервів та зорової кори. Оскільки система, навчена оглядати продукти або спостерігати за виробничим активом, може аналізувати тисячі продуктів або процесів за хвилину, помічаючи непомітні дефекти чи проблеми, вона має потенціал швидко перевершити людські можливості [15].

Методи розпізнавання жестів руки часто зосереджують увагу на техніках комп’ютерного зору. У цьому контексті є сім найпоширеніших технік, таких як визначення кольору шкіри, зовнішній вигляд, рух, скелет, глибина, 3D-модуль, глибоке навчання які підтримують ці техніки [16].

Задачі КЗ полягають у аналізі зображення або відеопотоку з метою виділення фрагменту, що містить необхідну інформацію. Для цього зазвичай

використовують прямокутну область, яка обмежує вихідний фрагмент, або виділяє його пікселі та орієнтири (рис. 1.5).

Запуск програмного продукту для комп’ютерного зору вимагає великої обчислювальної потужності. Особливо це стосується застосунків у режимі реального часу, оскільки їм потрібно обробити та зберегти набагато більше даних. Це означає, що потрібно залучити значну кількість апаратного забезпечення.

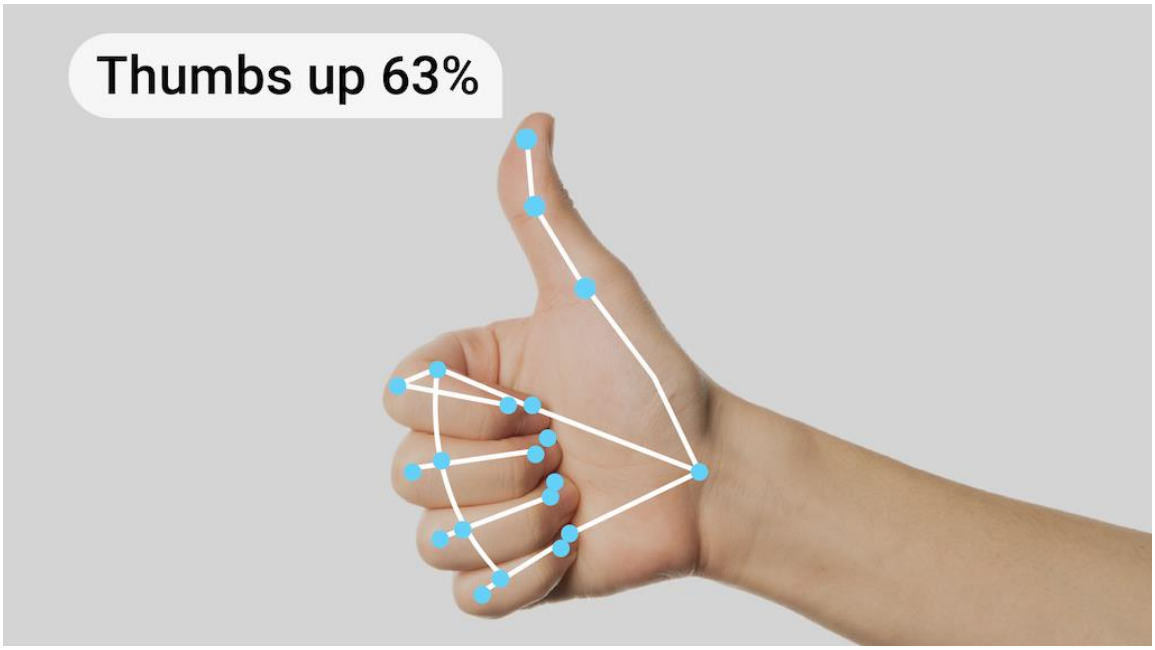


Рисунок 1.5 – Приклад роботи розпізнавання жестів руки з орієнтирами

Мета завдання ідентифікації полягає в класифікації всього зображення, для чого спочатку виділяються ключові області, за якими відбувається класифікація, наприклад, за допомогою рішаючих дерев або згорткових нейронних мереж. Однак завдання комп’ютерного зору є складним через труднощі повного аналізу зображень, що обмежують дослідників та змушують їх розділяти завдання на менш складні підзадачі, такі як нормалізація зображень, покращення їх якості, стандартизація, пошук особливих характеристик тощо.

1.4.3 Розпізнавання з використанням нейронних мереж

Звичайні моделі розпізнавання образів обмежені в своїй здатності обробляти природні дані в їх початковому вигляді. Протягом десятиліть створення системи розпізнавання образів значного досвіду в предметній області, щоб розробити засіб екстракції ознак, що перетворював би необроблені дані (наприклад, значення пікселів зображення) в підходяще внутрішнє подання або вектор ознак, з якого підсистема навчання, часто класифікатор, може виявляти або класифікувати образи вхідних даних [16].

Згорткові нейронні мережі (ЗНМ) аналогічні традиційним штучним нейронним мережам (ШНМ) тим, що вони складаються з нейронів, які самооптимізуються за допомогою навчання. Кожен нейрон отримує вхідні дані і виконує операції (такі як скалярний добуток, за яким слідує нелінійна функція), що є основою багатьох штучних нейронних мереж. Від вхідного вектору зображення до кінцевого виведення класового балу, вся мережа все ще виражає одну функцію сприйняття. Останній шар містить функції втрат, пов’язані з класами, і всі звичайні поради та прийоми, розроблені для традиційних ШНМ, все ще застосовуються. Єдиною помітною відмінністю між ЗНМ та традиційними ШНМ є те, що ЗНМ переважно використовуються в області розпізнавання образів в межах зображень. Це дозволяє закодувати властивості, специфічні для зображення або інших образів, в архітектуру мережі, зробивши мережу більш придатною для завдань, пов’язаних з зображеннями, та подальшого зменшення параметрів, необхідних для налаштування моделі [17].

ЗНМ складається з шарів входу та виходу, а також із декількох прихованих шарів. Приховані шари ЗНМ зазвичай складаються зі згорткових шарів, агрегувальних шарів, повноз’єднаних шарів та шарів нормалізації. Цей процес описують в нейронних мережах як згортку за домовленістю. З математичної точки зору він є радше взаємною кореляцією, ніж згорткою. Це

має значення лише для індексів у матриці, й відтак які ваги на якому індексі розташовуються [18].

Після декількох згорткових і агрегувальних шарів може бути один або кілька повнозв'язних шарів, які беруть всі виходи нейронів попереднього шару і з'єднують їх з кожним нейроном поточного шару для генерації глобальної семантичної інформації [19]. Повнозв'язний шар не завжди необхідний, оскільки його можна замінити згортковим шаром 1 × 1 [20].

Перед розпізнаванням жесту необхідно витягти характеристики зображення. Після цього можна використати будь-який метод класифікації. Для успішної класифікації та розпізнавання необхідна значна кількість характеристик.

Навчання ЗНМ – це задача глобальної оптимізації. Мінімізуючи функцію втрат, можна знайти найбільш підходящий набір параметрів. Стохастичний градієнтний спуск – найпоширеніше рішення для оптимізації згорткової нейронної мережі [21].

1.5 Постановка задачі

Під час написання першого розділу, було наведено інформацію про мову жестів, і саме про українську дактильну абетку. Розібрана структура мови жестів і описана загальна структура системи розпізнавання. Наведені відомості з теорії комп'ютерного зору та нейронних мереж. Виявлення об'єктів – це складна процедура, яка відбувається в кілька етапів.

Існуючі методи розпізнавання можна поділити на 3 загальні групи:

– методи на основі візуальних сенсорів, зокрема камер;
– методи на основі носимих датчиків, таких як рукавиці;
– гібридні методи на основі тривимірних сенсорів.

Перша група методів дозволяє розпізнавати статичні та динамічні жести руки з точністю розпізнавання більше 90% у деяких випадках. Проте,

головним недоліком цих методів є їх чутливість до змін освітлення, яку в деяких роботах намагаються усунути за допомогою використання кольорових рукавичок або створенням однорідного фону. Однак, це може зробити взаємодію між людиною та машиною незручною та неприродною.

Методи другої групи забезпечують високу якість розпізнавання позицій ключових точок руки людини, проте вимагають використання спеціальних рукавичок.

Методи третьої групи дозволяють розпізнавати ключові точки тіла людини в реальному часі та є незалежними від зміни освітлення. Проте, на сьогоднішній день не існує готових рішень, що забезпечують високу точність розпізнавання конфігурацій та динамічних жестів руки людини.

У завданнях розпізнавання зображень часто використовують згорткові нейронні мережі. Зазвичай такі мережі складаються з трьох шарів: перший – згортковий, відповідає за виділення ознак зображення; другий – пулінговий, призначений для зменшення розміру зображення з метою прискорення обчислень; третій – повнозв’язний, використовується для класифікації результатів. Також важливим елементом розпізнавання є комп’ютерний зір, який аналізує зображення або відео з метою виявлення фрагменту, що містить необхідну інформацію.

Таким чином, автоматизація розпізнавання жестів рук та дактильного алфавіту є актуальним завданням для обробки і розпізнавання зображень сучасного світу, оскільки, наприклад, дозволить людям з мовнослуховими вадами передавати інформацію з іншими без залучення писання чи експерта з перекладу. Іншим прикладом може слугувати управління роботами або комп’ютерними системами безконтактно.

Об’єктом роботи є послідовність відеокадрів, зображуючих конфігурації жестів рук.

Метою роботи є розробка системи, що базуються на використанні комп’ютерного зору, яка дозволяє у реальному часі детектувати ознаки та класифікувати жести рук для перетворення їх у простий текст.

Для досягнення мети необхідно вирішити такі завдання:

– провести аналіз існуючих методів розпізнавання жестів;

– зібрати набір даних, який складається з фотографій конфігурацій рук дактильної абетки або жестів;

– розробити класифікатор розпізнавання конфігурацій;

– протестувати роботу застосунку.

2 ПРОЄКТУВАННЯ СИСТЕМИ РОЗПІЗНАВАННЯ ЖЕСТІВ

2.1 Вибір мови програмування

Вибір мови програмування для розпізнавання жестів рук може бути складним завданням, оскільки існує багато мов програмування з різними особливостями та перевагами. Однак, для досягнення мети розпізнавання жестів рук, важливо вибрати мову програмування, яка забезпечує швидку та точну обробку великих обсягів даних, а також має добре розроблені інструменти для роботи з образами та машинним навчанням.

Існує декілька мов програмування, які можна використовувати для завдання розпізнавання жестів. Для розробки програм для обробки зображень і розпізнавання жестів, найчастіше використовуються такі мови програмування, як Python, C++ та Java.

Python – це інтерпретована, об’єктно-орієнтована, високорівнева мова програмування з динамічною семантикою. Її вбудовані високорівневі структури даних, в поєднанні з динамічною типізацією та зв’язуванням, роблять її дуже привабливою для швидкого розроблення програм, а також для використання як мови скриптування або з’єднувальної мови для з’єднання наявних компонентів між собою [22].

Python є дуже популярною мовою програмування у сфері комп’ютерного зору і обробки зображень. Вона має багато бібліотек для роботи з зображеннями, таких як Mediapipe та OpenCV, які дозволяють розпізнавати жести та інші дії з використанням камери. Python є досить простим у використанні, має велику кількість документації та навчальних матеріалів, що робить його популярним вибором для початківців.

C++ – це мова програмування, стандартизована, загального призначення, об’єктно-орієнтована, скомпільована мова програмування. Кожен компілятор C++ супроводжується набором корисних функцій та

контейнерів, які називаються C++ стандартною бібліотекою. Bjarne Stroustrup створив C++ як розширення мови програмування C, але C++ еволюціонував до повністю іншої мови програмування [23].

За допомогою C++ можна використовувати різні бібліотеки для обробки зображень, також зокрема OpenCV, що містить багато алгоритмів для обробки та розпізнавання образів, включаючи розпізнавання образів. Крім того, C++ дозволяє працювати зі звуком та відео, що може бути корисно при розпізнаванні жестів рук. Однак, вона є більш складною мовою програмування порівняно з Python, що може забрати більше часу на розробку програми.

Java – це об’єктно-орієнтована мова програмування, яка використовується для розробки різноманітних застосунків, від веб-застосунків до мобільних застосунків та сервісів. Java була розроблена у 1995 році компанією Sun Microsystems і відтоді стала однією з найбільш популярних мов програмування [24].

Щодо розпізнавання образів Java може використовуватись для розробки програмного забезпечення, яке взаємодіє з камерами та датчиками руху для розпізнавання рухів рук та жестів. Наприклад, JavaFX – це графічна бібліотека для Java, яка надає можливості для розробки інтерактивних графічних інтерфейсів користувача, що можуть взаємодіяти з рухами рук та жестами.

Використання мови Python для розпізнавання образів є вибором для розробки завдяки широкій підтримці бібліотек машинного навчання та комп’ютерного зору. Python дозволяє легко писати чистий, зрозумілий код, що знижує час розробки та сприяє більшій розумінню коду.

Крім того, мається доступ до великої кількості розширень та бібліотек, що полегшує розробку складних моделей розпізнавання образів. Також, для виконання складних завдань може знадобитися велика обчислювальна потужність та графічний процесор.

2.2 Формування ієрархічної моделі класів згідно з SOLID

Написання чистого коду – одна з основних засад розробки програмного забезпечення. Існують різні підходи до проєктування програмного забезпечення і забезпечення зрозумілої, гнучкої та підтримуваної кодової бази. Одним з основних принципів, що дозволяють ефективно будувати архітектуру програмного додаткку є SOLID.

Принцип SOLID – це принцип і шаблон проєктування, призначений для того, щоб зробити код програмного забезпечення зручним для підтримки. При створенні об’єкта, структура об’єкта має важливе значення в процесі розробки програмного забезпечення, особливо при визначенні меж доступності. Формування ієрархічної моделі класів згідно з принципами SOLID передбачає створення моделі класів, яка буде легко зрозумілою, масштабованою, змінною і легко тестованою [25].

Нижче наведено п’ять переваг застосування принципу SOLID у процесі розробки програмного забезпечення:

– доступність. Цілісність стабільних об’єктно-орієнтованих застосунків забезпечує легкий доступ до об’єктів, усуваючи ризики ненавмисного успадкування;

– спрощення рефакторингу. Погано структуровані програмні додатки важко піддаються рефакторингу, але рефакторинг кодової бази за принципом SOLID значно спрощується;

– розширюваність. Якщо розширювати функції програми не тактовно, це може вплинути на існуючі функціональні можливості та спричинити непередбачувані проблеми;

– відладка. Коли програмні додатки не спроєктовані належним чином, а кодова база кластеризована, їх важко налагоджувати;

– читабельність. Добре спроєктовану кодову базу легко зрозуміти і прочитати.

Нижче наведено список цих принципів та їх детальне пояснення:

– принцип єдиного обов’язку (Single Responsibility Principle – SRP): клас повинен мати лише один обов’язок. Інакше кажучи, кожен клас повинен вирішувати лише одну проблему. Якщо клас виконує більше, ніж одну функцію, то він стає складним для розуміння і підтримки. Якщо відповідальність класу змінюється, то варто перерозподілити його функції на декілька класів;

– принцип відкритості/закритості (Open/Closed Principle – OCP): клас повинен бути відкритий для розширення, але закритий для змін. Це означає, що поведінка класу повинна змінюватися лише через додавання нового коду, а не зміни існуючого. Ідея принципу відкритості-закритості полягає в тому, що існуючі, добре протестовані класи потрібно змінювати, коли потрібно щось додати. Проте, зміна класів може призвести до проблем або помилок. Замість того, щоб змінювати клас, ви просто можете розширити його;

– принцип підстановки Лісков (Liskov Substitution Principle – LSP): об’єкти підкласів повинні бути замінювані об’єктами базового класу без зміни правильності програми. Це означає, що якщо клас B є підкласом класу A, то є можливим передача об’єкту класу B будь-якому методу, який очікує об’єкт класу A, і метод не повинен видавати ніяких хибних результатів у цьому випадку;

– принцип розділення інтерфейсу (Interface Segregation Principle – ISP): клієнти не повинні залежати від методів, які вони не використовують. Це означає, що інтерфейси повинні бути розбиті на менші частини, щоб клієнти могли реалізовувати тільки ті методи, які їм потрібні. Загальна ідея принципу розділення інтерфейсів полягає в тому, що краще мати багато маленьких інтерфейсів, ніж кілька великих;

– принцип інверсії залежностей (Dependency Inversion Principle – DIP): модулі високого рівня не повинні залежати від модулів нижчого рівня. Обидва типи модулів повинні залежати від абстракцій, а не від конкретики.

Використання цих принципів при формуванні ієрархічної моделі класів допомагає створити код, який є більш ефективним, простим для розуміння та модифікації, а також більш масштабованим. Це дозволяє зменшити кількість помилок, які можуть виникнути в процесі розробки програмного забезпечення.

Грунтуючись на цих базових принципах, було створено ряд сутностей, на які буде опиратися в подальшому алгоритм. Нижче представлена їх будова у форматі UML-діаграм (рис. 2.1 – рис. 2.3).

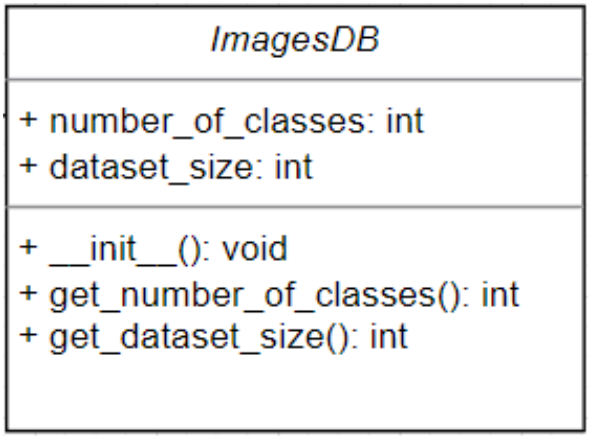


Рисунок 2.1 – Схема класу ImagesDB

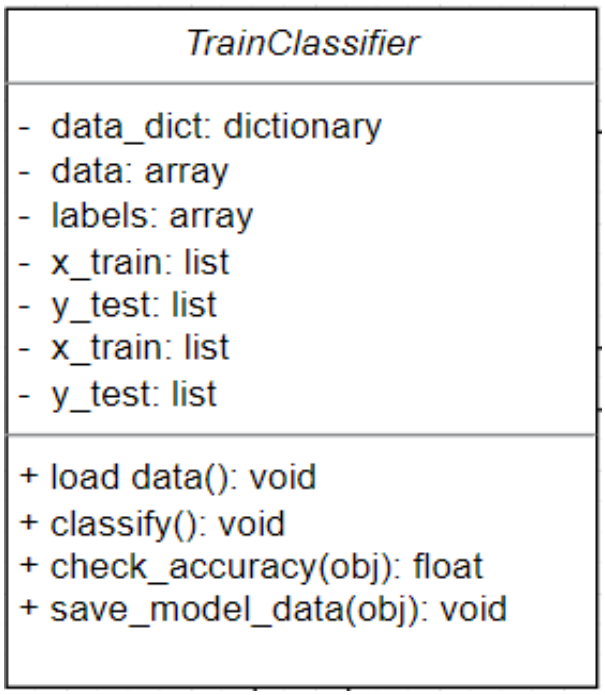


Рисунок 2.2 – Схема класу TrainClassifier

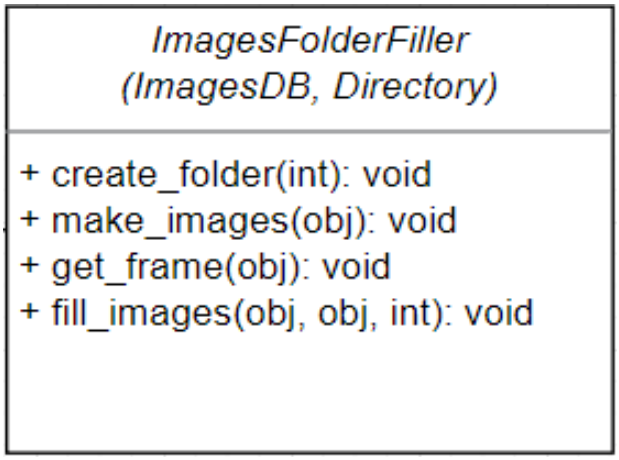


Рисунок 2.3 – Схема класу ImagesFolderFiller

2.3 Створення власного набору даних

Створення власного набору даних для розпізнавання жестів рук є складним і важливим етапом у розробці системи машинного навчання. Для досягнення максимальної ефективності алгоритмів машинного навчання, необхідно мати достатньо велику кількість різноманітних даних.

На щастя, на сьогодні існує велика кількість готових наборів даних (датасетів), але деякі завдання вимагають створення окремого набору даних. Одні з найбільш відомих датасетів цієї галузі: HaGRID, EgoGesture, SHREC, мають до 500000 варіантів фото.

Зазвичай, для створення власного датасету зображень можна використати картинки з інтернету. Деякі програми навіть дозволяють завантажувати більше 1000 зображень за один раз. Однак, у випадку українського дактильного алфавіту немає великих готових наборів даних, тому жести будуть фотографуватися власноруч. Для цього необхідно спочатку визначити кількість літер у дактильному алфавіті, що відповідає кількості літер у звичайному українському алфавіті. Різницю англійського дактильного алфавіту та українського показано на рисунку 2.4.

Наступним етапом є визначення списку жестів, які входять до складу набору. В залежності від конкретних потреб, можуть включатися як статичні,

так і динамічні жести. Проте, деякі динамічні жести у статичному вигляді відповідають іншим статичним жестам, тому вони будуть виключені з набору.

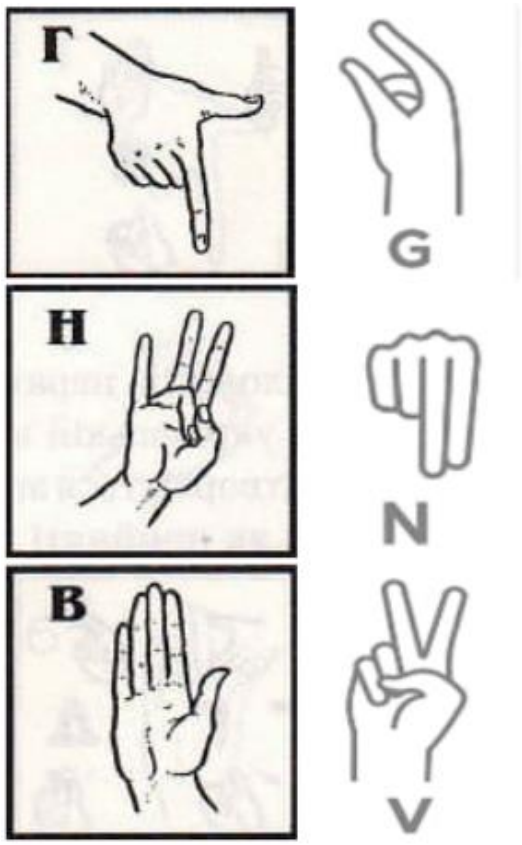


Рисунок 2.4 – Показ різниці американського та українського алфавітів

В набір даних включатимуться 28 різних жестів, відповідаючих українському дактильному алфавіту, при цьому, деякі жести, які мають динамічну природу, будуть виключені, оскільки набір даних складатиметься зі статичних зображень, та більше ніж 5 загальновживаних жестів, які відповідають словам або фразам. Деякі літери, такі як «ґ», «ї», «й», «щ», можуть бути замінені на «г», «і», «и», «ш» через їхню схожість на статичних зображеннях.

В кожному з, більш ніж 35, класів буде по 100 зображень, загалом 3500 екземплярів. Щоб забезпечити однаковий розмір зображень у наборі даних, було обрано розмір 640×480 пікселів. Щоб полегшити та прискорити обробку зображень, кольорові зображення фільтруються та перетворюються на у канал сірих, що дозволяє нейромережі легше опрацьовувати фотографії.

2.4 Проєктування процесу розпізнавання жестів

Проєктування процесу розпізнавання жестів – це складний процес, який включає в себе декілька етапів. Першим етапом є збір даних, що потрібні для навчання алгоритмів розпізнавання жестів. Наступним етапом є підготовка цих даних для навчання, яка включає в себе обробку, нормалізацію та видалення шуму.

Після підготовки даних переходимо до навчання алгоритмів розпізнавання жестів. Для цього необхідно визначити оптимальні параметри моделі, такі як архітектура нейронної мережі, параметри оптимізації, функцію витрат і так далі. Після успішного навчання моделі можна перейти до її тестування на наборі даних, який раніше не використовувався для навчання. Тестування дозволяє оцінити точність та ефективність моделі. Останнім етапом проєктування процесу розпізнавання жестів є оптимізація моделі. Це означає, що потрібно зробити модель більш точною та ефективною. Оптимізація може бути досягнута шляхом підвищення кількості навчальних даних, зміни параметрів моделі, оптимізації функції витрат і так далі [26].

Спрощений процес розпізнавання жестів з вебкамери був зображений на рисунку 2.5 та складається з трьох основних кроків, які йдуть у циклі:

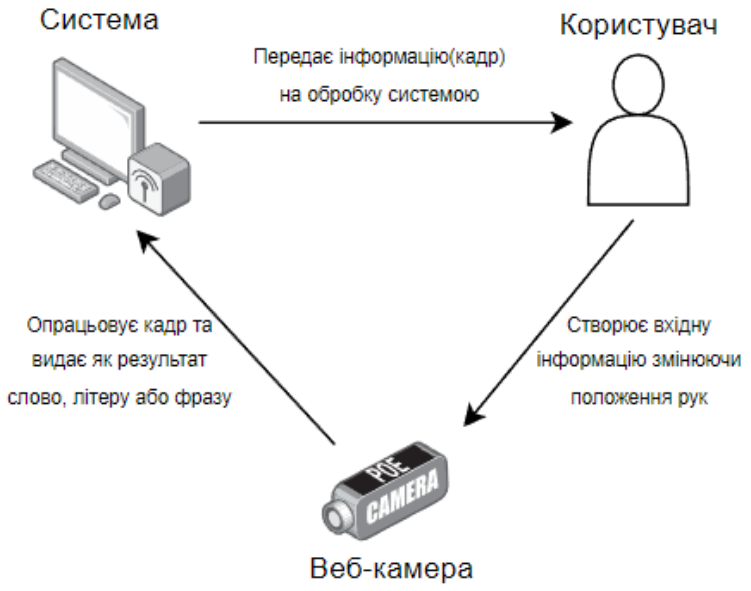


Рисунок 2.5 – Показ процесу розпізнавання жестів з вебкамери

Процес розпізнавання жестів з вебкамери може бути реалізований наступним чином:

– захоплення зображення. Користувач запускає програму та показує жести. Вебкамера захоплює кольорові зображення з частотою оновлення 30 кадрів за секунду. Кожне зображення передається на попередню обробку;

– попередня обробка. Кольорове зображення перетворюється у сіре монохромне, що зменшує обсяг даних і полегшує подальшу обробку. Також можливі інші операції, такі як зміна розміру зображення, підсилення контрастності, позбавлення від шуму тощо. Після цього значення пікселів нормалізуються до діапазону від 0 до 1;

– розпізнавання. Оброблене зображення передається на нейромережу, яка здійснює розпізнавання жестів. Нейромережа може бути навчена на великому наборі даних з різними жестами. Після розпізнавання жесту, програма повертає його переклад. Для покращення точності розпізнавання можуть бути використані різні техніки, такі як збільшення розміру навчального набору даних, зменшення кількості параметрів нейромережі, додавання додаткових шарів тощо.

Для здійснення розпізнавання жестів було використано модуль комп’ютерного зору Python OpenCV, який забезпечує можливість роботи з зображеннями, відео та трансляціями з вебкамери, а також надає функції для захоплення кадрів та обробки зображень.

Архітектура нейронної мережі складається з двох основних частин: спочатку вона використовує модель детектування ключових точок, щоб знайти руки на зображенні, а потім вона використовує модель оцінки, щоб визначити позиції цих ключових точок на руках. Для детекції рук використовується модель BlazePalm, яка є попередньо навченою на даних з понад 30 тисяч різних рук та використовує архітектуру EfficientDet для швидкого та точного виявлення рук на зображенні.

BlazePalm – це глибока нейронну мережу, створену для виявлення долонь та їх орієнтації на зображенні в режимі реального часу [27]. BlazePalm

використовує архітектуру відкритого коду EfficientDet, що забезпечує швидку та ефективну роботу мережі на різних платформах. Вона приймає на вхід кольорове зображення розміром 256×256 пікселів та використовує внутрішню згорткову мережу для отримання векторів ознак на кожній ітерації. Потім, за допомогою non-maximum suppression (NMS), мережа вибирає найбільш імовірні прямокутники, що містять долоні.

Non-maximum suppression (NMS) – це метод, що використовується для підтримки локалізації об’єктів на зображеннях та відео. Він є важливою складовою для обробки об’єктів у комп’ютерному зорі та інших завданнях комп'ютерного зору, таких як детекція обличчя та об’єктів, виявлення ключових точок, відстеження об’єктів тощо.

Алгоритм працює на вхідному наборі об’єктних рамок та їх відповідних оцінок, щоб прибрати рамки з невідповідними об’єктами та об’єднати перекриваючіся рамки. Припустимо, що набір містить N рамок та їх оцінки у вигляді вектора s з N елементами. Тоді для кожного класу об’єктів процедура діє наступним чином:

Крок 1. Об’єктні рамки сортуються за оцінкою s у зворотньому порядку (від найбільшої до найменшої).

Крок 2. Обирається об’єктна рамка b з найвищою оцінкою $s[0]$ та додається в список вибраних об’єктів. Це гарантує, що алгоритм завжди вибирає рамку з найвищою оцінкою.

Крок 3. Об’єктні рамки, які перетинаються з вибраною рамкою b на більше, ніж задане порогове значення IoU (Intersection over Union), видаляються зі списку.

Крок 4. Повторюються Кроки 2 та 3 для вибору наступної найвищої об’єктної рамки та її додавання до списку вибраних об’єктів.

Крок 5. Повертається список вибраних об’єктних рамок.

Формула для розрахунку *IoU* між двома об’єктними рамками *A* та *B* має наступний вигляд:

$$IoU(A, B) = \frac{Area\ of\ Intersection\ (A,B)}{Area\ of\ Union\ (A, B)}, \quad (2.1)$$

де $Area\ of\ Intersection\ (A,B)$ – площа перетину двох рамок *A* та *B*;

$Area\ of\ Union\ (A, B)$ – площа їх об’єднання.

BlazePalm використовує алгоритм прогнозування орієнтації долоні, який базується на відносних відстанях між ключовими точками в руці, що були виявлені раніше. Орієнтація долоні потім використовується для вибору відповідного орієнтаційного шаблону з набору, що дозволяє мережі точно визначати положення долоні на зображенні та її орієнтацію [28]. BlazePalm забезпечує високу точність та швидкість виявлення долонь на зображенні, що робить його популярним рішенням для задач розпізнавання жестів та інтерактивних застосунків. Вона може детектувати одну або обидві руки та повертає для кожної руки прямокутник, що охоплює її (рис. 2.6).

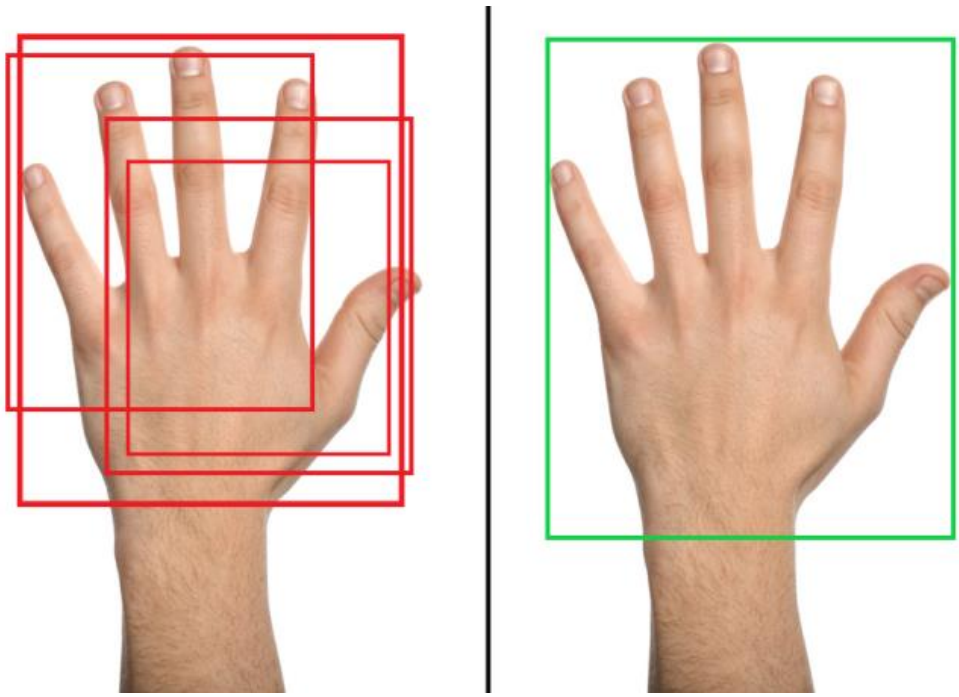
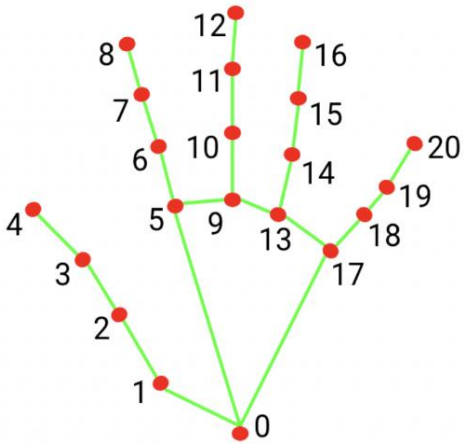


Рисунок 2.6 – Показ процесу розпізнавання BlazePalm

Для оцінки позицій ключових точок на руках використовується модель, яка містить 21 визначену ключову точку для кожної руки, зображена на рисунку 2.7 [29].



а)

0. WRIST
1. THUMB_CMC
2. THUMB_MCP
3. THUMB_IP
4. THUMB_TIP
5. INDEX_FINGER_MCP
6. INDEX_FINGER_PIP
7. INDEX_FINGER_DIP
8. INDEX_FINGER_TIP
9. MIDDLE_FINGER_MCP
10. MIDDLE_FINGER_PIP
11. MIDDLE_FINGER_DIP
12. MIDDLE_FINGER_TIP
13. RING_FINGER_MCP
14. RING_FINGER_PIP
15. RING_FINGER_DIP
16. RING_FINGER_TIP
17. PINKY_MCP
18. PINKY_PIP
19. PINKY_DIP
20. PINKY_TIP

б)

Рисунок 2.7 – Набір орієнтирів долоні моделі Hands:
а) положення орієнтирів; б) ідентифікаційні назви кожного з орієнтирів

Вона працює на основі архітектури глибокої нейронної мережі, яка використовує згорткові та повнозв’язні шари для визначення позицій ключових точок на зображенні руки. Основна структура моделі складається з двох згорткових шарів, двох повнозв’язних шарів та функції активації ReLU між шарами.

Вхідний шар нейронної мережі Hands приймає зображення руки розміром 256×256 пікселів. Кожен піксель може мати три значення: червоний, зелений, синій (R, G, B) або одне значення відтінку сірого, якщо колірні дані були перетворені відповідно до попереднього опису.

Вхідний шар складається з конволюційного шару та шару нормалізації за кожним каналом. Згортковий шар застосовує фільтри до вхідного зображення, що дає можливість вилучати його візуальні ознаки.

Основна формула згорткового шару може бути записана як:

$$Z_{i,j} = f(\sum w_{k,l} * X_{i+k-1,j+l-1} + b), \quad (2.2)$$

де $Z_{i,j}$ – вихідне зображення після згортки;

$X_{i+k-1,j+l-1}$– вхідне зображення;

$w_{k,l}$– вагові коефіцієнти фільтру;

b – зміщення;

f – функція активації, яка застосовується до суми вагових значень.

ReLu (Rectified Linear Unit) – це одна з найпоширеніших функцій активації в нейронних мережах, яка використовується для нелинійної трансформації вихідних значень нейронів. Функція ReLu повертає значення 0 для від'ємних вхідних значень і вхідне значення для додатніх вхідних значень (рис. 2.8), для досягнення кращих результатів у завданнях класифікації [30].

Формула застосування функції активації ReLu в нейронній мережі має вигляд:

$$Z_{i,j} = \max(0, Z_{i,j}) , \quad (2.3)$$

де $Z_{i,j}$ – вихідне значення після згорткового шару.

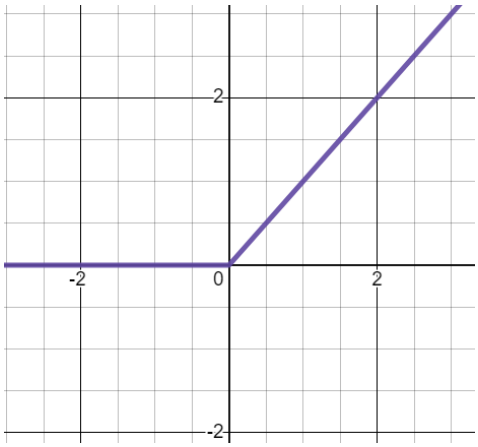


Рисунок 2.8 – Показ графіку функції ReLu

У контексті нейронної мережі функція ReLu використовується після згорткового шару для того, щоб прибрати від’ємні значення, які можуть виникнути після застосування фільтрів до зображення.

Після цього відбувається пулінг з максимальним значенням, щоб зменшити розмір зображення та сконцентруватися на найважливіших ознаках.

Отримані дані після аугментації подаються на вхід до нейронної мережі Hands. Ця мережа складається з двох головних компонентів: відносно простої згорткової мережі та більш складної рекурентної мережі LSTM. Після згорткової мережі використовується більш складна рекурентна мережа LSTM, яка дозволяє враховувати контекст даних з попередніх кадрів та забезпечує більш точне визначення положення рук та пальців у відео [31].

2.5 Застосування алгоритму класифікації

Random Forest – це алгоритм машинного навчання, який використовується як для класифікації, так і для регресії. Він є прикладом дерев рішень, який випадковим чином вибирає частину функцій для побудови дерев рішень та підбирає параметри, що робить його більш устійчивим до перенавчання та здатним до роботи з великими обсягами даних [32].

Алгоритм має кілька ключових параметрів. Один з них – це кількість дерев, що включаються в ліс. Цей параметр впливає на точність та швидкість алгоритму: більше дерев означає більшу точність, але й більший час обробки даних. Інший ключовий параметр – це глибина дерев, що включаються в ліс. Більша глибина може призвести до перенавчання моделі, тому рекомендується встановлювати обмеження на глибину дерев.

Алгоритм діє наступним чином:

Крок 1. З наявного набору даних випадковим чином вибирається підмножина з заміщенням, тобто кожен елемент може бути вибраний декілька

разів. Це дозволяє створити декілька незалежних вибірок для тренування дерев рішень.

Крок 2. На кожному вузлі дерева рішень з наявних ознак випадковим чином вибирається підмножина ознак для подальшого розгалуження. Це дозволяє зменшити кореляцію між деревами та зробити їх більш різноманітними.

Крок 3. Для кожної випадкової підмножини даних побудова дерева рішень проводиться шляхом рекурсивного розгалуження на основі вибраної підмножини ознак. Критерієм розгалуження є максимізація індексу Джині або ентропії, які відображають ступінь «чистоти» кожного вузла дерева.

Крок 4. Для запобігання перенавчання, дерева рішень зазвичай зрізаються до певної глибини або за критерієм мінімальної кількості зразків у листі.

Крок 5. Результатом класифікації є голосування всіх дерев. Клас, який набирає найбільшу кількість голосів, стає загальним прогнозом класифікатора.

Цей алгоритм є потужним інструментом для класифікації, оскільки дозволяє підтримувати високу точність навіть на дуже великих наборах даних. Додатково, він може використовуватись для визначення важливості ознак, що може бути корисним для подальшого аналізу даних. Однак, він може бути менш ефективним на даних з високою розмірністю та великою кількістю категорій.

3 РОЗРОБКА ТА ТЕСТУВАННЯ ЗАСТОСУНКУ

3.1 Розробка прототипу

Система для розпізнавання жестів рук повинна виявляти та розпізнавати різні жести рук, використовуючи вебкамеру. Така система може бути корисна для людей з фізичними обмеженнями, які не можуть спілкуватися за допомогою мови або клавіатури. Розпізнавання жестів дозволяє їм комунікувати з іншими людьми за допомогою мови тіла.

Для розробки прототипу такої системи потрібно виконати декілька кроків. Першим кроком є збір даних для тренування моделі машинного навчання. Для цього можна використовувати фотографії, на яких показані різні жести рук. Найкраще, якщо дані будуть зібрані з різних джерел та з різних людей, щоб модель була більш універсальною та точною, але у даній роботі будуть різноманітні фото зображення жестів у незмінній сцені.

Перед тим, як навчати модель, необхідно підготувати дані, щоб вони були придатні для роботи з ними. Зображення можуть бути різного розміру та якості, тому їх необхідно підготувати для подальшого використання. Одним з методів обробки зображень є зменшення їх розміру, яке дозволяє зменшити обсяг даних та прискорити обробку.

У етапі навчання моделі, після підготовки даних, треба навчити її розпізнавати жестів рук. Для цього можна використати різні алгоритми машинного навчання, такі як згорткові нейронні мережі або методи на основі випадкових лісів. При навчанні моделі слід враховувати те, що необхідно додатково обробляти зображення з вебкамери, щоб знизити рівень шуму та покращити якість даних.

Розробка інтерфейсу – це важлива складова процесу створення системи для розпізнавання жестів рук з вебкамери. Інтерфейс повинен бути зрозумілим та зручним для користувачів, які будуть взаємодіяти з системою. Для розробки

інтерфейсу буде використано інструменти та технології, такі як мова програмування Python та графічна бібліотека PyQt, бібліотеки для роботи з вебкамерою та бібліотеки для відображення зображень та відео.

Інтерфейс може містити такі елементи, як кнопки, поля введення, елементи відображення зображень та відео. Основна мета інтерфейсу – забезпечити користувачам зручну та інтуїтивно зрозумілу взаємодію з системою. Для цього можна використати прості та зрозумілі елементи управління, забезпечити відображення результатів розпізнавання та надати можливість збереження результатів роботи системи. Крім того, можна додати кнопки для запуску та зупинки розпізнавання жестів, а також можливість вибору режиму розпізнавання (наприклад, режим розпізнавання жестів з обмеженим набором дій)(рис. 3.1).

Після навчання моделі її потрібно протестувати на наборі тестових даних. Це дозволить переконатися, що модель добре працює на нових даних та може розпізнавати жести рук з високою точністю.

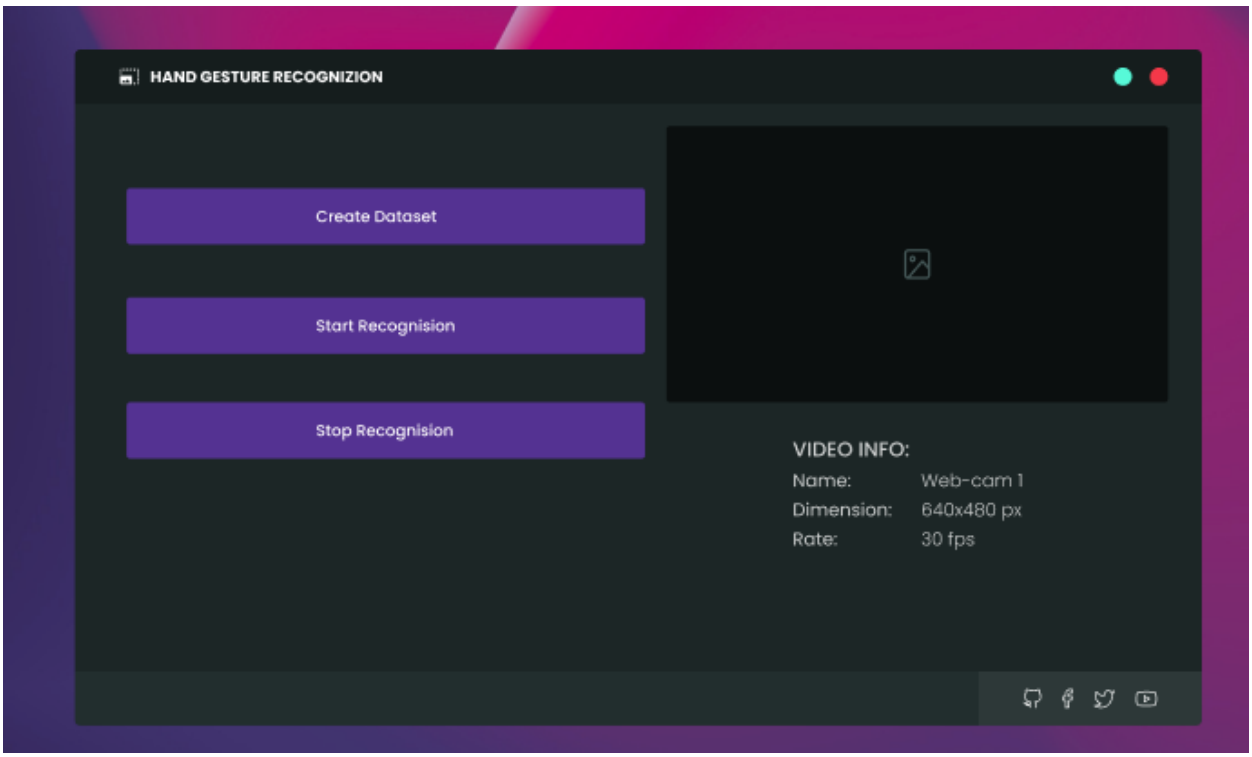


Рисунок 3.1 – Прототип інтерфейсу програмного застосунку

На етапі після тренування моделі необхідно імплементувати її в програмне забезпечення. Для цього можна використовувати мови програмування, такі як Python, та відповідні бібліотеки для машинного навчання. Програма повинна включати в себе код для зчитування відеопотоку з вебкамери, обробки зображень та передавання їх на вхід моделі для розпізнавання жестів рук. Після розпізнавання жесту програма повинна відповідно реагувати, наприклад, виконувати певну дію, виводити інформацію на екран або надсилати команду на зовнішнє пристрій.

Окрім того, слід звернути увагу на оптимізацію роботи програми, особливо на швидкість розпізнавання жестів. Це може включати в себе використання технік оптимізації моделі, таких як квантизація або обрізка, а також оптимізацію роботи програми в цілому, наприклад, за допомогою асинхронного програмування. Після імплементації моделі слід провести тестування програми на різних вхідних даних, щоб переконатися в її правильній роботі. Візуальний процесс може виглядати як наявне розпізнавання у реальному часі показано на рисунку 3.2.

Рисунок 3.2 – Прототип процесу візуального розпізнавання

Також варто передбачити можливість підтримки програми, наприклад, шляхом введення функції зворотного зв’язку для користувачів або планування майбутніх оновлень та вдосконалень.

3.2 Реалізація застосунку

Python була обрана для реалізації системи розпізнавання жестів рук з кількох причин. По-перше, мова Python має досить простий синтаксис, що дозволяє швидко писати і читати код, зменшуючи кількість можливих помилок.

Крім того, у Python є велика кількість різноманітних модулів для машинного навчання, що дозволяє легко використовувати популярні бібліотеки, такі як Mediapipe та OpenCV, для тренування моделі розпізнавання жестів рук.

Розробка проєкту відбувалася на операційній системі Windows 10 з використанням версії Python 3.9.5. Для написання та запуску коду було обрано інтегроване середовище розробки PyCharm від JetBrains. PyCharm забезпечує аналіз коду, графічний відлагоджувач, інтегровані засоби для запуску юніт-тестів, інтеграцію із системами контролю версій, а також підтримує веб-розробку з Django та аналіз даних з Anaconda. Це дозволило зосередитися на розробці самого застосунку, не витрачаючи час на додаткові інструменти розробки.

У застосунку, окрім стандартних бібліотек Python, використовуються наступні додаткові бібліотеки:

– OpenCV – бібліотека алгоритмів комп’ютерного зору, обробки зображень та чисельних алгоритмів загального призначення з відкритим кодом. Використовується для роботи з відеопотоками та обробки зображень руки з камери;

– NumPy – бібліотека для роботи з матрицями та багатомірними масивами, містить великий набір математичних функцій. Використовується для роботи з матрицями зображень руки та її розпізнавання;

– Matplotlib – бібліотека для побудови графіків. Використовується для візуалізації орієнтирів у процесі розпізнавання жестів рук;

– Mediapipe – бібліотека для роботи з обробкою відеопотоку та розпізнаванням об’єктів, у тому числі рук. Використовується для розпізнавання жестів рук та їх класифікації;

– Scikit-learn – бібліотека для машинного навчання та аналізу даних, яка містить вбудовані алгоритми для класифікації, кластеризації, регресії та інші;

– Pickle – бібліотека для серіалізації та десеріалізації об’єктів Python, яка дозволяє зберігати дані у файл та повертати їх у пізнішому використанні системою;

– PyQt – бібліотека для розробки графічного інтерфейсу користувача. Використовується для створення інтерфейсу застосунку та взаємодії з користувачем.

3.2.1 Mediapipe та OpenCV

Mediapipe Hands – це модуль бібліотеки Mediapipe, який забезпечує розпізнавання рук у відео або зображеннях з вебкамери чи файлу. Цей модуль використовує машинне навчання та нейронні мережі для точної ідентифікації кінцівок рук, а також визначення їхньої позиції та орієнтації. Mediapipe Hands працює шляхом виявлення ключових точок на зображенні, що відповідають конкретним частинам руки, таким як кінчики пальців, зап’ястя, долоня, основа пальців тощо. Для цього використовується глибока нейронна мережа, навчена на великому наборі даних, що включає зображення рук з різних кутів та позицій. Знайдені ключові точки використовуються для відновлення

положення та орієнтації рук у просторі. Зокрема, це дає змогу визначити, чи зігнуті пальці, які саме пальці зігнуті, кут стиснення кисті, а також визначити координати кінчиків пальців [33].

Для свого навчання Hands використовує датасет, що називається «FreiHAND dataset». FreiHAND є набором даних пози рук у 3D, який записує різні дії рук, виконані 32 людьми. Для кожного зображення руки надаються анотації пози рук у 3D на основі моделі для артикульованих рук. Набір даних містить 32 560 унікальних навчальних зразків та 3960 унікальних зразків для оцінки (рис. 3.3). Навчальні зразки записані на зеленому тлі для можливості видалення фону. Крім того, до навчальних зразків застосовуються три різних стратегії пост-обробки для розширення даних. Однак ці стратегії не застосовуються до зразків для оцінки [34].

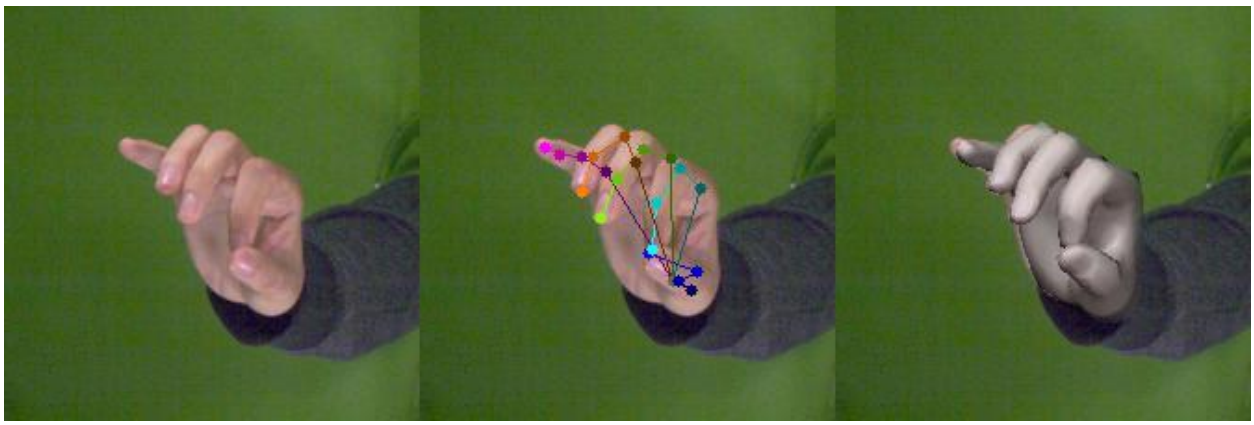


Рисунок 3.3 – Процедура використання FreiHAND

OpenCV (Open Source Computer Vision Library) – це відкрита бібліотека, що надає інструменти для розробки програм для обробки зображень та комп’ютерного зору. Бібліотека написана на C++ і має інтерфейси для взаємодії з іншими мовами програмування [35].

Однією з головних переваг OpenCV є те, що вона доступна для використання на різних платформах, таких як Windows, Linux та MacOS. Також OpenCV підтримує багато мов програмування, таких як C++, Python, Java та інші, що робить її дуже гнучкою для використання.

3.2.2 Алгоритм створення датасету

Написання скрипта було виконано у середовищі JetBrains PyCharm. На першому етапі, з використанням бібліотек os та OpenCV, були створені базові класи, що містять необхідні поля та методи, що дозволятимуть зберігати та відтворювати у подальшому файли у форматі JPG.

Фото мають такі параметри:

– зроблені з гарним освітленням;
– роздільна здатність: 640×480 px;
– кількість точок на дюйм: 96 dpi;
– глибина кольору: 24 bit depth.

При виконанні коду створюється база зображень, яку можна використовувати для навчання нейронних мереж та інших завдань машинного навчання. Для початку, імпортується дві бібліотеки – «os» та «cv2». Бібліотека «os» використовується для роботи з операційною системою, а бібліотека «cv2» (OpenCV) використовується для обробки зображень.

Клас «ImagesDB» містить у собі інформацію про базу зображень, зокрема – кількість класів і розмір бази. Конструктор класу приймає два аргументи – кількість класів та розмір бази. Клас «Directory» відповідає за створення директорії, де будуть зберігатися зображення. Змінна «DATA_DIR» містить шлях до директорії. Метод «create_directory()» перевіряє наявність директорії та створює її, якщо її не існує. Клас «ImagesFolderFiller» наслідує два класи – «ImagesDB» і «Directory».

Цей клас містить метод «make_images», який збирає дані з відео потоку, формує зображення та зберігає їх у відповідну директорію. Метод «create_folder» створює директорію для зображень для кожного класу. Якщо директорія вже існує, то він пропускає створення.

Лістінг 3.1 Реалізація процесу створення фото для набору даних:

def make_images(self, imagesdb):

```
cap = cv2.VideoCapture(0)
for j in range(imagesdb.get_number_of_classes()):
    self.create_folder(j)
    print('Collecting data for class {}'.format(j))
    self.get_frame(cap)
    self.fill_images(cap, j, imagesdb)
cap.release()
cv2.destroyAllWindows()
```

Метод «get_frame» використовується для зчитування відео потоку та відображення зображення на екрані. Перед початком збору даних пропонується користувачу натиснути клавішу «Q». Після натискання клавіші починається збір даних.

Метод «fill_images» відповідає за збір даних. Він зчитує зображення з відеокамери, показує його на екрані та зберігає у відповідну папку зображення, що містить інформацію про клас. У циклі він зберігає зображення поки їх кількість не досягне значення, яке визначається параметром «dataset_size» об’єкту ImagesDB. Кожне зображення отримує ім’я, що складається з порядкового номеру та розширення «jpg».

Лістінг 3.2 Реалізація функції зчитування та збереження кадрів з вебкамери:

```
def fill_images(self, cap, j, imagesdb):
counter = 0
while counter < imagesdb.get_dataset_size():
    ret, frame = cap.read()
    cv2.imshow('frame', frame)
    cv2.waitKey(25)
    cv2.imwrite(os.path.join(Directory.DATA_DIR,
str(j),'{}.jpg'.format(counter)), frame)
```

counter += 1

Після закінчення збору зображень метод виводить повідомлення про завершення збору для поточного класу. Кожен клас відзначається цифровим значенням (рис. 3.4).

Датасет, який буде використовуватися для навчання моделі класифікації жестів рук, створюється іншим скриптом. Для цього використовуються бібліотеки OpenCV, Mediapipe та Pickle. Клас Dataset містить методи для створення датасету та зберігання його у файл. Він також містить статичні поля, які містять шлях до директорії з даними та об’єкти, необхідні для виконання розпізнавання жестів рук з допомогою Mediapipe.

Рисунок 3.4 – Структура файлової системи класів датасету

Метод make_dataset має таку логіку:

Крок 1. Для кожного каталогу в «./data» директорії скануються всі файли з розширенням «.jpg».

Крок 2. Для кожного файлу на зображенні рук використовується Mediapipe для визначення координат ключових точок.

Крок 3. Координати кожної ключової точки відштовхуючись від крайньої лівої і верхньої точки області рук на зображенні нормалізуються і зберігаються як ознаки для кожного зображення у списку data.

Крок 4. Мітки, що вказують на те, який жест представлений на зображенні, зберігаються в списку labels.

Метод save_data зберігає датасет у файл форматі Pickle.

3.2.3 Вихідна класифікація

Орієнтири руки, що використовуються в Mediapipe Hands, це набір точок, що вказують на різні частини рук та пальців. Загалом, у Mediapipe Hands використовується 21 орієнтир, що розташовані на кожній руці.

Для руки базовими точками є:

– зап’ястя – орієнтир, що вказує на кисть руки;
– базовий вказівник – орієнтир, що вказує на базу пальця;
– середній вказівник – орієнтир, що вказує на середину пальця;
– кінчик пальця – орієнтир, що вказує на останню фалангу пальця.

Для роботи необхідно визначити орієнтири руки та робити класифікацію згідно з їх позицією в режимі реального часу (рис. 3.5).

Класифікатор – це інструмент машинного навчання, який використовується для класифікації об’єктів відповідно до їх характеристик. В контексті комп’ютерного зору, класифікатор використовується для розпізнавання образів, що знаходяться на зображеннях або відео [36].

У випадку з роботою з жестами рук, класифікатор може використовуватись для розпізнавання жестів рук, зроблених користувачем, та їх класифікації на відповідний набір дій [37]. Наприклад, якщо користувач зробив жест «кулак», система може розпізнати цей жест та виконати дію, пов’язану з цим жестом.

TrainClassifier проєкту містить методи для навчання та збереження класифікатора. Після завантаження даних з файлу data.pickle за допомогою методу load_data().

Лістінг 3.3 Фрагмент класифікатора набору даних:

```
self.x_train, self.x_test, self.y_train, self.y_test = train_test_split(self.data, self.labels, test_size=0.2, shuffle=True, stratify=self.labels )
model = RandomForestClassifier()
```

model.fit(self.x_train, self.y_train)

accuracy = self.check_accuracy(model)

self.save_model_data(model)

Метод classify() навчає модель Random Forest на навчальному наборі даних, який складається з 80% даних та відповідних міток з файлу data.pickle. Модель Random Forest є ансамблем дерев рішень, який показує добрі результати в багатьох задачах класифікації. Навчання моделі проводиться за допомогою методу fit() з параметрами *self.x_train* та *self.y_train*.

Наступним кроком є перевірка точності моделі на тестовому наборі даних. Метод train_test_split() розбиває дані на тестовий та навчальний набір даних. Параметр test_size встановлює розмір тестового набору даних (20% від загальної кількості даних). Параметр shuffle вказує, що дані будуть перемішані перед розбиттям.

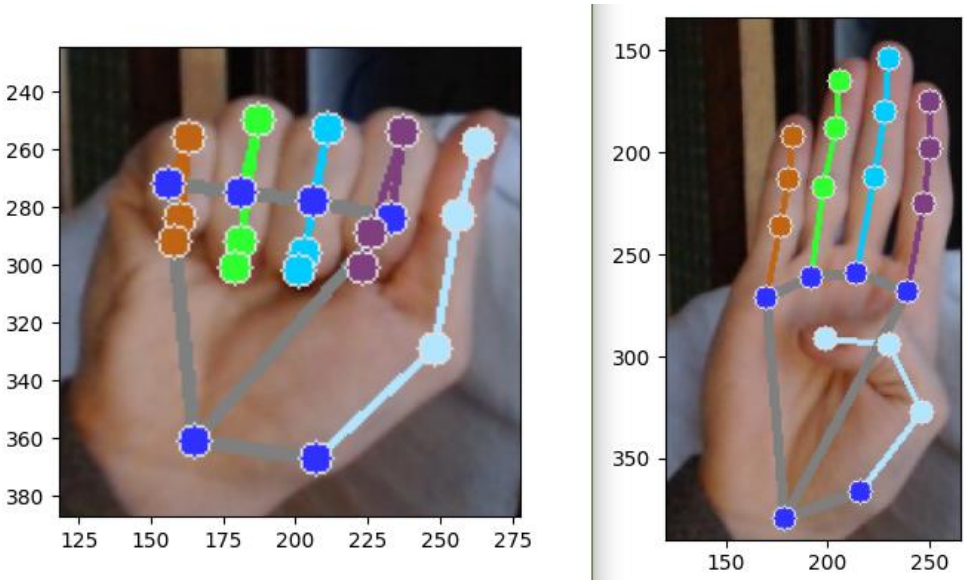


Рисунок 3.5 – Показ визначення орієнтирів на долоні

Перемішування даних допомагає уникнути можливого впливу порядку даних на навчання моделі. Якщо вхідні дані впорядковані за певними критеріями, наприклад, за часом, це може призвести до того, що модель буде

навчена на однотипних або відносно однакових даних, що може призвести до поганої прогнозувальної здатності моделі на нових невидимих даних.

Основний клас HandGestureRecognitionSystem містить методи, які використовують відеопотік з вебкамери для визначення жестів рук (рис. 3.6). Після запуску програма відкриває вебкамеру та чекає на вхідні кадри.

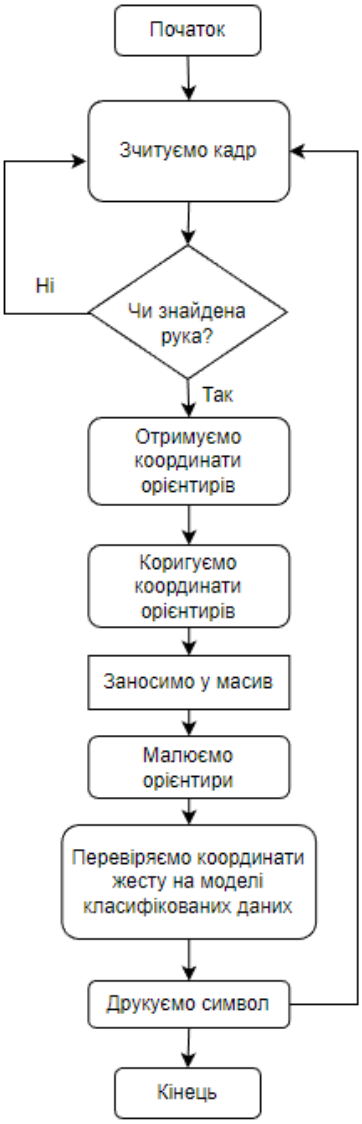


Рисунок 3.6 – Блок схема функції real_time_recognition()

Метод real_time_recognition є основним методом, який обробляє відеопотік і розпізнає жести рук в реальному часі. Спочатку метод отримує кадр з відеопотоку, потім конвертує його у формат RGB та передає до моделі.

Лістінг 3.4 Реалізація розпізнавання у режимі реального часу:

```
while True:
data_aux = []
x_ = []
y_ = []
ret, frame = self.cap.read()
H, W, _ = frame.shape
frame_rgb = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
results = self.hands.process(frame_rgb)
            for i in range(len(hand_landmarks.landmark)):
                x = hand_landmarks.landmark[i].x
                y = hand_landmarks.landmark[i].y
                data_aux.append(x - min(x_))
                data_aux.append(y - min(y_))
            x1 = int(min(x_) * W) - 30
            y1 = int(min(y_) * H) - 20
            x2 = int(max(x_) * W) + 20
            y2 = int(max(y_) * H) + 20
            prediction = self.model.predict([asarray(data_aux)])
            predicted_character = labels_dict[int(prediction[0])]
```

Якщо на кадрі виявлено руку, то проводиться розрахунок координат ключових точок та класифікація жесту. За допомогою методу draw_landmarks бібліотеки mediapipe на кадрі намальовані точки ключових точок руки.

Після отримання координат ключових точок руки, для класифікації жесту дані про координати зберігаються в списку *data_aux*. Для кожної ключової точки координати (*x*, *y*) переводяться відносно початку координат, знаходять різницю від координати точки з мінімальним значенням *x* та *y* і додаються до списку *data_aux*. Після збору координат усіх ключових точок,

шукаються координати мінімальної та максимальної точок, щоб виділити прямокутник, який містить усі ключові точки руки.

Далі проводиться передбачення жесту за допомогою навченої моделі. Метод predict застосовується до вектора, який містить дані про координати ключових точок руки, та повертає індекс класу, який найкраще підходить до вектора. Після отримання індексу класу відбувається використання словника labels_dict для перетворення індексу класу в символ жесту.

3.3 Тестування застосунку

Навчальна модель штучної нейронної мережі була тренована на фотографіях рук лише однієї людини, тому різноманіття може бути не великим. Це може призвести до неточностей у розпізнаванні жестів рук інших людей [38].

Отже, для перевірки точності роботи застосунку на зображеннях рук інших людей було обрано слово, яке містить різні літери та знаки, які можуть бути сплутані з іншими (рис. 3.7). Було обрано слово «літак» (рис. 3.8).

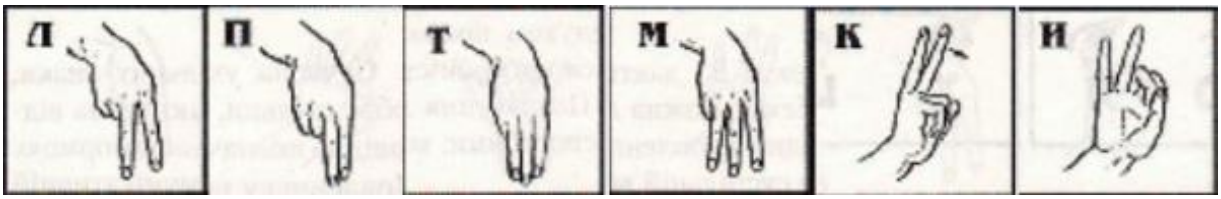


Рисунок 3.7 – Попарне порівняння «спірних» жестів

«Спірними» є жести, що можуть бути сплутані системою розпізнавання жестів рук через схожість їх форми, знаків та орієнтації. Одним з основних параметрів класифікації жестів рук є відстань між орієнтирами, що може створювати проблеми при класифікації жестів, що мають подібні форми. Наприклад, жести «ОК» та «Добре» мають досить подібну форму та

орієнтацію, тому система може сплутати їх. На прикладі порівняння через схожість у нашому слові можуть бути сплутані Л-П, Т-М, К-И.

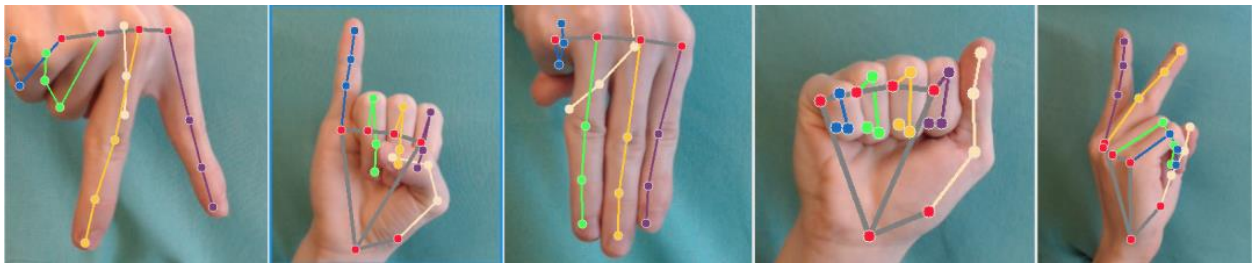


Рисунок 3.8 – Зображення тестового слова «Літак»

Згідно з результатами розпізнавання слово було надруковано правильно, але зі збільшенням спроб показана конфігурація букви «Т» у 20–30% набувала значення букви «М». Значно менше разів буква «Л» була розпізнана як «П».

Проведемо тестування за допомогою інших людей (рис. 3.9), фото конфігурацій рук яких класифікатор не бачив:

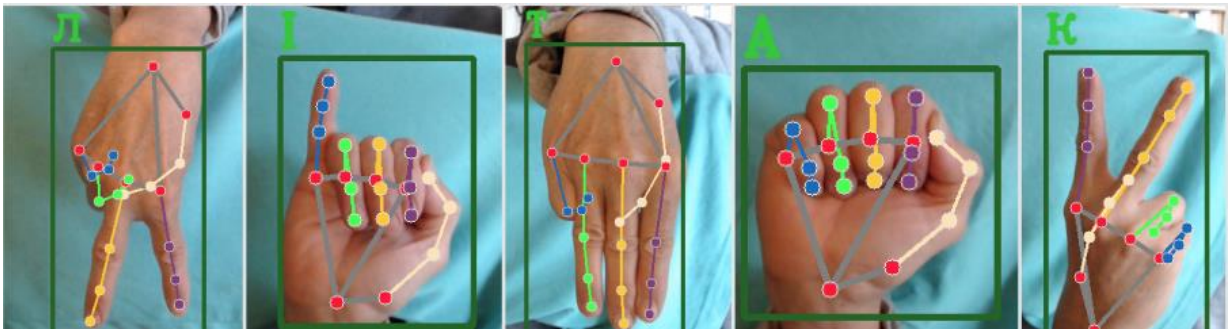


Рисунок 3.9 – Зображення тестового слова з показом конфігурацій іншою людиною

Візуально можна побачити різницю між двома сетами показу слова, однак класифікатор непогано знаходить схожі положення, що дає гарний результат розпізнавання. Проведення процесу переведення жестів в простий текст показало різницю у результатах.

Обрано протягом скільки секунд користувач повинен показувати жест, щоб його можна було розпізнати. Було призначено 2 секунди, тож у список буде записувати кожну літеру, проте, щоб відображалась лише одна літера.

Список очищується, якщо попередня літера не збігається з поточною. Коли в списку з’являється більше 60 (2 секунди × 30 кадрів в секунду) однакових літер, вона виводиться у текстове поле.

При тестуванні побачено, що упроцесі розпізнавання у режимі реального часу «спірні» жести впливають на результат більше через постійне зміщення руки, яке утворює відходження від класифікованої моделі класів (рис. 3.10).

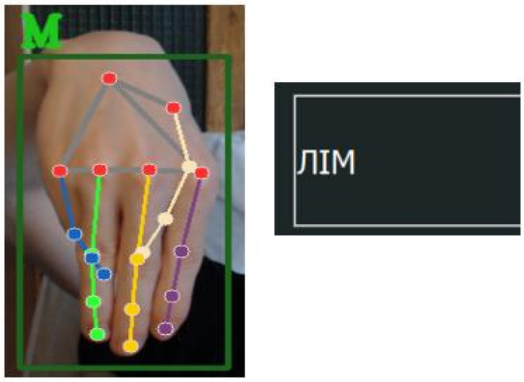


Рисунок 3.10 – Зображення неточності процесу розпізнавання у реальному часі на прикладі букви «Т»

Однак у своїй більшості система показала гарний результат. На основі тестувань можна стверджувати, що в цілому класифікатор правильно визначив більшість літер, за винятком літери декількох «спірних» жестів. Результати можна вважати успішними, оскільки з п’яти жестів було розпізнано чотири, якщо не змінювати положення руки.

Для того, щоб запобігти сплутуванню жестів, необхідно створити більш великий та різноманітний датасет для навчання системи розпізнавання жестів, що включатиме в себе різні форми, орієнтації та знаки жестів рук. Це означає, що треба збірати більше прикладів. Одним із способів збільшити розмаїтість датасету є залучення більшої кількості людей для запису жестів. Різні люди можуть мати відмінність в рухах рук, формі жестів або їх інтерпретації. Чим більше різних людей беруть участь у записі датасету, тим більше різноманітних жестів буде представлено.

3.4 Інструкція користувача

Застосунок було написано на версії інтерпретатора Python 3.9. Перед використанням необхідно завантажити необхідні для роботи модулі, які вказані у файлі requirements.txt, а саме:

– opencv-python==4.7.0.68;
– mediapipe==0.9.0.1;
– scikit-learn==1.2.0;
– pyqt5==5.15.9.

Важливо: шлях до проєкту не повинен мати символів кирилиці, бо це може викликати помилки про роботі з бібліотеками, такими як Mediapipe!

Для початку роботи з додатком (рис. 3.11) потрібно запустити його через інтегровану середу розробки або перейти до директорії проєкту та ввести у терміналі команду запуску.

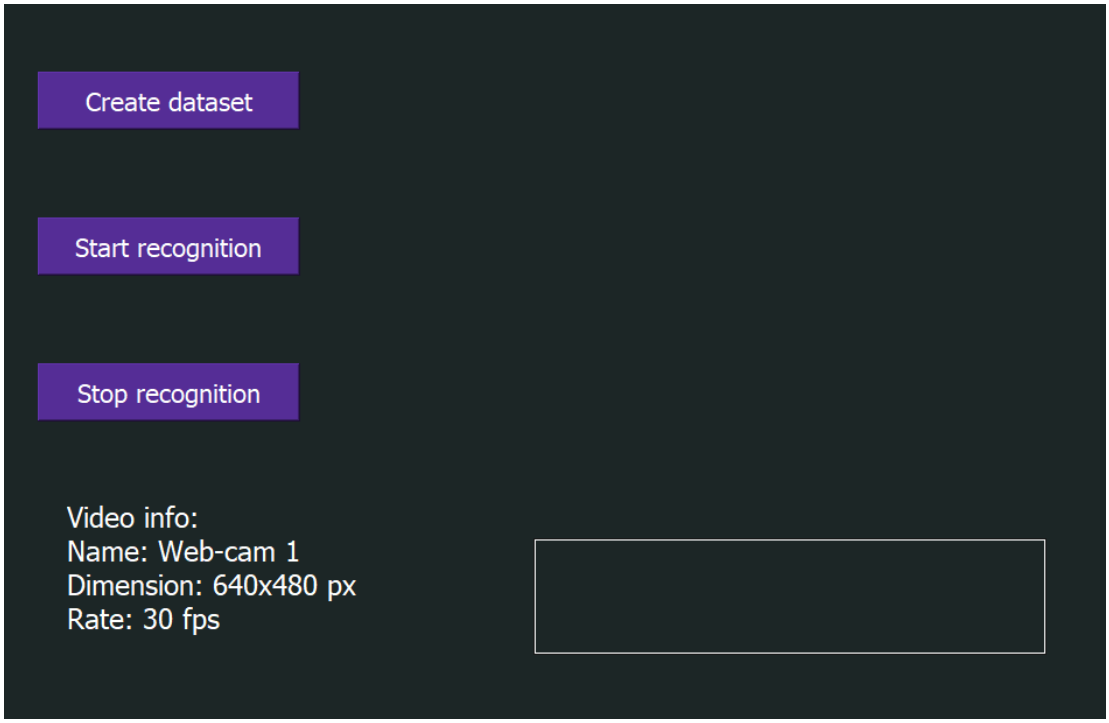


Рисунок 3.11 – Показ інтерфейсу користувача після запуску застосунку

При натисканні на кнопку «Create dataset» у графічному інтерфейсі програми викликається функція, яка запускає процес збору інформації на

цьому вікні. Від користувача потребується натискати англійську літеру «Q» для запуску циклу створення фото (рис. 3.12). Процесс нотується у терміналі як показано на рисунку 3.13. Після успішного збирання даних формується датасет, який на наступному передається у класифікатор та оброблюється. Вихідна інформація записується у файл «model.p».

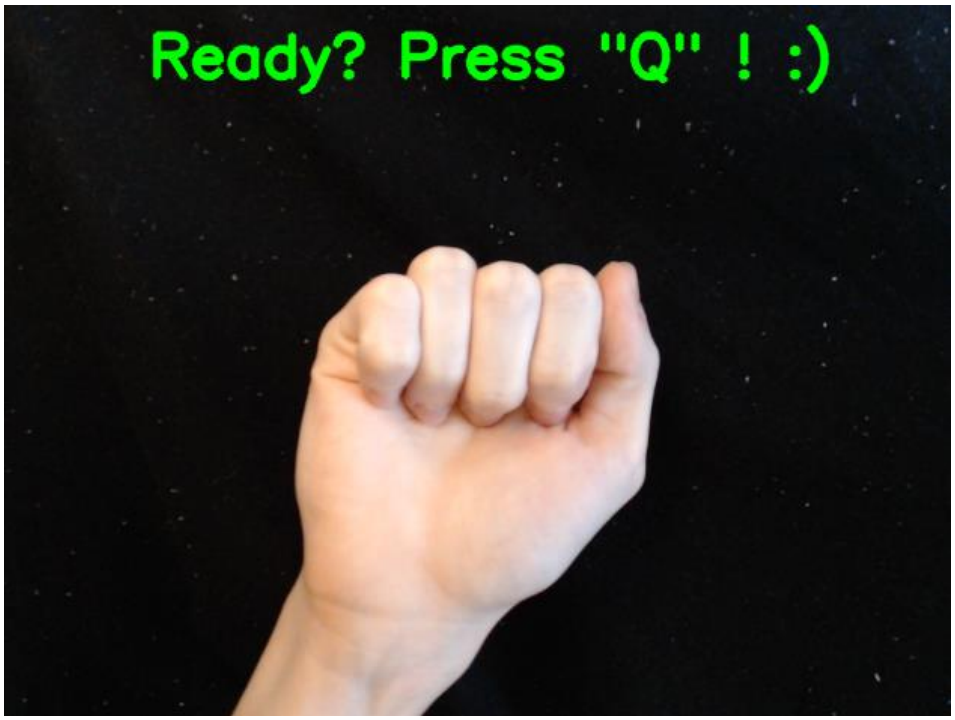


Рисунок 3.12 – Показ інтерфейсу процесу збору фотографій

```
Collecting data for class 0
92% of samples were classified correctly!
Dataset created!
```

Рисунок 3.13 – Показ прикладу покрокового нотування процесу

При натисканні на кнопку «Start recognition» у графічному інтерфейсі програми викликається функція, яка запускає процесс розпізнавання жестів рук у режимі реального часу на цьому вікні. Користувач може показувати жести, щоб програма перевела їх у текст.

Процес супроводжується показом знайденої літери або слова у кадру (рис. 3.14). Також на інтерфейсі є зображення характеристик відео, таких як роздільна здатність та кількість кадрів у секунду.

При натисканні на кнопку «Stop recognition» у графічному інтерфейсі програми викликається функція, яка прибирає зображення відео на цьому вікні. Кнопка «Start recognition» відновлює цей процес.

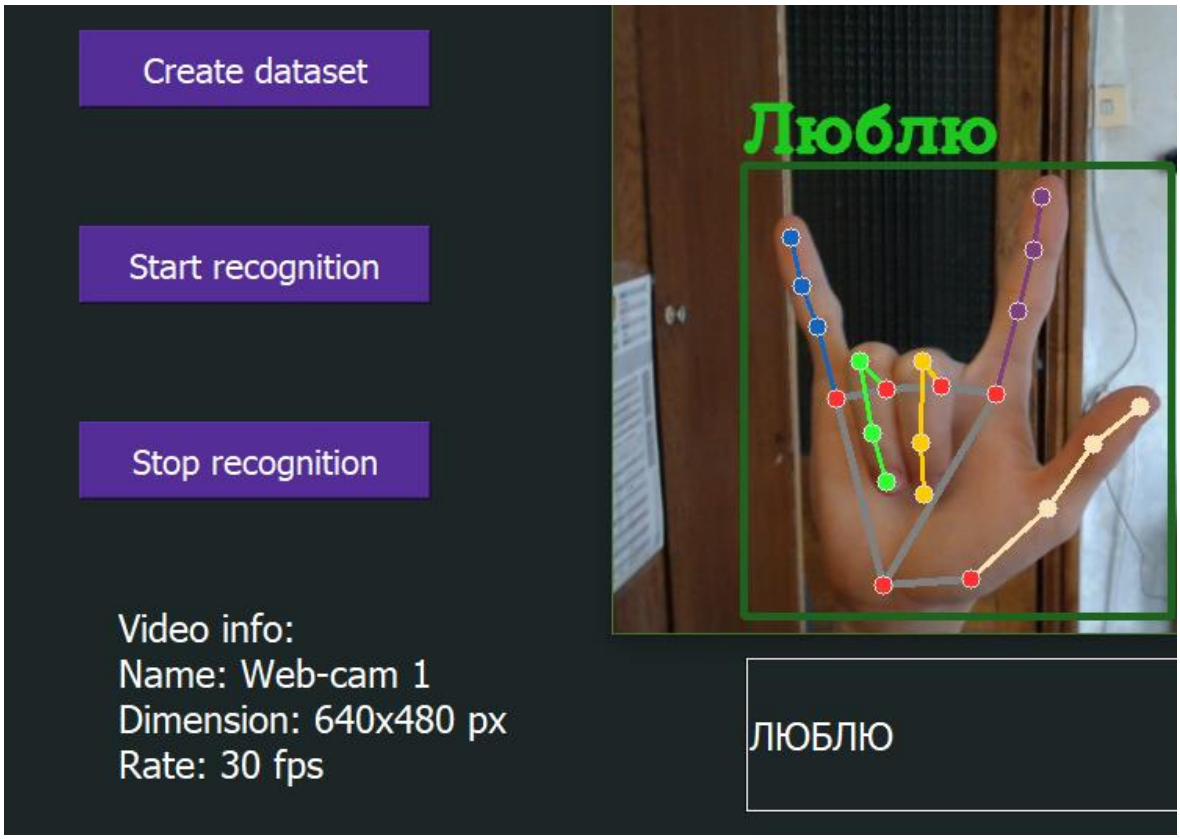


Рисунок 3.14 – Показ процесу розпізнавання слів у режимі реального часу

ВИСНОВКИ

У рамках кваліфікаційної роботи було розроблено застосунок та спроєктовано процес класифікації для системи розпізнавання жестів рук на мові програмування Python. Застосунок було розширено через додавання можливостей створення власного набору даних та перекладу розпізнаних жестів у простий текст.

Процес розпізнавання жестів рук є важливим для різних застосувань, таких як взаємодія з комп'ютером без клавіатури та миші, руховий керуванням пристроями, віртуальна реалість, медична діагностика та реабілітація, безпека та відеоспостереження, ігри тощо.

Розглянуті були теми, що стосуються мови жестів та саме української дактильної абетки. Структури систем та існуючі методології розпізнавання жестів було досліджено. Було вибрано мову програмування та сформовано процес розробки згідно з принципами SOLID. Також було зпроєктовано процес прозпізнавання жестів та розглянуто архітектуру нейронної мережі, яка використовується у застосунку. Проаналізовано алгоритм класифікації зображень з набору даних.

Розроблену систему розпізнавання жестів рук було успішно протестовано на різних користувачах, та проведено дослідження з використанням «спірних» жестів.

Інструкція користувача та результати роботи системи розпізнавання жестів рук було додано у кваліфікаційну роботу.

В результаті роботи було досягнуто мети – створення застосунку з системою розпізнавання жестів рук, який перекладає жести у простий текст.

Результати роботи апробовано у вигляді 1 тези доповіді «ОЦІНКА МЕТОДА ПРИХОВАНИХ МАРКОВСЬКИХ МОДЕЛЕЙ ДЛЯ РОЗПІЗНАВАННЯ ОБРАЗІВ» під час Міжнародного молодіжного форуму «РАДІОЕЛЕКТРОНІКА І МОЛОДЬ У XXI СТОЛІТТІ» [39].

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. What is sign language? URL: https://makeitfable.com/glossary-term/sign-language/ (дата звернення 12.04.2023).

2. Chetham's Library | Digiti-Lingua. URL: https://library.chethams.com/blog/digiti-lingua/ (дата звернення 12.04.2023).

3. «Я тебе (не) чую». Як в Україні живеться людям із порушеннями слуху. URL: https://lyuk.media/behind-city/hearing-loss/ (дата звернення 12.04.2023).

4. Українська дактильна абетка – Вікіпедія. Вікіпедія. URL: https://uk.wikipedia.org/wiki/Українська_дактильна_абетка (дата звернення 12.04.2023).

5. Camgoz, N. C., Koller, O., Hadfield, S., & Bowden, R. (2020). Sign language transformers: Joint end-to-end sign language recognition and translation. In Proceedings of the *IEEE*/CVF conference on computer vision and pattern recognition (pp. 10023-10033).

6. Understanding Sign Language: Expressions, movements and structure. URL: https://www.brombergtranslations.com/understanding-sign-language-part-one/ (дата звернення: 12.04.2023).

7. Tvoroshenko, I., & Kharchenko, A. (2021). Some aspects of modern development for sign language recognition systems.

8. Ganguly, B., Vishwakarma, P., & Biswas, S. (2020). Kinect Sensor Based Single Person Hand Gesture Recognition for Man–Machine Interaction. In Computational Advancement in Communication Circuits and Systems: Proceedings of ICCACCS 2018 (pp. 139-144). Springer Singapore.

9. O'Connor, T. F., Fach, M. E., Miller, R., Root, S. E., Mercier, P. P., & Lipomi, D. J. (2017). The Language of Glove: Wireless gesture decoder with low-power and stretchable hybrid electronics. PloS one, 12(7), e0179766.

10. CyberGlove III, CyberGlove Systems LLC. URL: http://www.cyberglovesystems.com/cyberglove-iii (дата звернення: 13.04.2023).

11. Kinect – Вікіпедія. Вікіпедія. URL: https://uk.wikipedia.org/wiki/Kinect (дата звернення: 13.04.2023).

12. Zhang, Z. (2012). Microsoft kinect sensor and its effect. IEEE multimedia, 19(2), 4-10.

13. Galván-Ruiz, J., Travieso-González, C. M., Tejera-Fettmilch, A., Pinan-Roescher, A., Esteban-Hernández, L., & Domínguez-Quintana, L. (2020). Perspective and evolution of gesture recognition for sign language: A review. Sensors, 20(12), 3571.

14. Gorokhovatskyi, V., Tvoroshenko, I., & Chmutov, Y. (2022). Застосування систем ортогональних функцій для формування простору ознак у методах класифікації зображень. Advanced Information Systems, 6(3), 5-12.

15. Computer Science. URL: https://web.cs.ucdavis.edu/~okreylos/ResDev/Kinect/MainPage.html (дата звернення: 13.04.2023).

16. What is Computer Vision? | IBM. URL: https://www.ibm.com/topics/computer-vision (дата звернення: 13.04.2023).

17. Gorokhovatskiy, V. A., Berestovsky, A. Y., & Putyatin, Y. P. (2015). Self-Learning methods in the space of image structural sign criterions. *Telecommunications and Radio Engineering*, 74(18).

18. Згорткова нейронна мережа – Вікіпедія. URL: https://uk.wikipedia.org/wiki/Згорткова_нейронна_мережа (дата звернення: 17.04.2023).

19. Gorokhovatskyi, O., Peredrii, O., Gorokhovatskyi, V., & Vlasenko, N. (2023). Explanation of CNN image classifiers with hiding parts. *In Explainable Deep Learning AI* (pp. 125-146). Academic Press..

20. Daradkeh, Y. I., Gorokhovatskyi, V., Tvoroshenko, I., & Al-Dhaifallah, M. (2022). Classification of images based on a system of hierarchical features. *Computers, Materials & Continua*, 72(1), 1785-1797.

21. Lin, M., Chen, Q., & Yan, S. (2013). Network in network. *arXiv preprint* arXiv:1312.4400.

22. Gu, J., Wang, Z., Kuen, J., Ma, L., Shahroudy, A., Shuai, B., ... & Chen, T. (2018). Recent advances in convolutional neural networks. Pattern recognition, 77, 354-377.

23. What is Python? Executive Summary. Python.org. URL: https://www.python.org/doc/essays/blurb/ (дата звернення: 24.04.2023).

24. Dmitrović, S. (2023). What is C++? In Modern C++ for Absolute Beginners: A Friendly Introduction to the C++ Programming Language and C++ 11 to C++ 23 Standards (pp. 3-3). Berkeley, CA: Apress.

25. What is Java technology and why do I need it? URL: https://www.java.com/en/download/help/whatis_java.html (дата звернення: 25.04.2023).

26. Tian, F., Liang, P., & Babar, M. A. (2019, March). How developers discuss architecture smells? an exploratory study on stack overflow. In *2019 IEEE international conference on software architecture (ICSA)* (pp. 91-100). IEEE.

27. Wadhawan, A., & Kumar, P. (2020). Deep learning-based sign language recognition system for static signs. *Neural computing and applications*, *32*, 7957-7968.

28. Gorokhovatskiy, V. A., Gorokhovatskiy, A. V., & Peredrii, E. O. (2017). Vector Quantization, Learning and Recognition in the Space of Descriptors of Structural Features of Images. *Telecommunications and Radio Engineering*, 76(19).

29. Tan, M., Pang, R., & Le, Q. V. (2020). Efficientdet: Scalable and efficient object detection. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (pp. 10781-10790).

30. Hand landmarks detection guide | MediaPipe | *Google Developers*. URL:https://developers.google.com/mediapipe/solutions/vision/hand_landmarker (дата звернення: 01.05.2023).

31. Xu, J., Li, Z., Du, B., Zhang, M., & Liu, J. (2020, July). Reluplex made more practical: Leaky ReLU. In *2020 IEEE Symposium on Computers and communications (ISCC)* (pp. 1-7). IEEE.

32. Zhang, H., Zimmerman, J., Nettleton, D., & Nordman, D. J. (2019). Random forest prediction intervals. *The American Statistician*.

33. Lugaresi, C., Tang, J., Nash, H., McClanahan, C., Uboweja, E., Hays, M., ... & Grundmann, M. (2019). Mediapipe: A framework for building perception pipelines. *arXiv preprint arXiv:1906.08172*.

34. Papers with Code - FreiHAND Dataset. *The latest in Machine Learning | Papers With Code*. URL: https://paperswithcode.com/dataset/freihand (дата звернення: 31.04.2023).

35. Gorokhovatskyi, V., Gadetska, S., Gorokhovatskyi, O., & Ponomarenko, R. (2020). Statistical Methods for Analyzing and Processing Data Components When Recognizing Visual Objects in the Space of Key Point Descriptors. *In Data Stream Mining & Processing: Third International Conference, DSMP 2020, Lviv, Ukraine, August 21–25, 2020, Proceedings 3* (pp. 241-252). Springer International Publishing.

36. Daradkeh, Y. I., Gorokhovatskyi, V., Tvoroshenko, I., Gadetska, S., & Al-Dhaifallah, M. (2021). Methods of classification of images on the basis of the values of statistical distributions for the composition of structural description components. *IEEE Access*, 9, 92964-92973.

37. Gadetska, S., Gorokhovatskyi, V., Stiahlyk, N., & Vlasenko, N. (2022). Aggregate Parametric Representation of Image Structural Description in Statistical Classification Methods.

38. Gorokhovatskyi, O., Peredrii, O., & Gorokhovatskyi, V. (2020, August). Interpretability of Neural Network Binary Classification with Part Analysis. *In 2020 IEEE Third International Conference on Data Stream Mining & Processing (DSMP)* (pp. 136-141). IEEE.

39. Слободенюк І. & Путятіна О. (2023). Оцінка метода прихованих марковських моделей для розпізнавання образів. *Матеріали 27-го міжнародного молодіжного форуму «Радіоелектроніка і молодь у xxi столітті». Конференція «Комп'ютерний зір, системний аналіз та математичне моделювання».* (pp. 88–89).