Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет комп’ютерних наук
(повна назва)

Кафедра програмної інженерії
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти другий (магістерський)

Дослідження алгоритмів комп'ютерного зору для розпізнавання жестів

Виконав:
здобувач 2 року навчання,
групи ІПЗм-23-3

Даніїл ГАЙДУК
(прізвище, ініціали)

Спеціальність 121 – Інженерія програмного забезпечення
(код і повна назва спеціальності)

Тип програми освітньо-наукова

Керівник доц. Віра ГОЛЯН
(посада, прізвище, ініціали)

Допускається до захисту
Зав. кафедри _____ Кирило СМЕЛЯКОВ
(підпис) (прізвище, ініціали)

2025 р.

Харківський національний університет радіоелектроніки

Факультет комп’ютерних наук
Кафедра програмної інженерії
Рівень вищої освіти другий (магістерський)
Спеціальність 121 – Інженерія програмного забезпечення
Тип програми освітньо-наукова програма
Освітня програма Інженерія програмного забезпечення

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
«___»_____20__ р.

ЗАВДАННЯ
НА КВАЛІФАКАЦІЙНУ РОБОТУ

здобувачеві Гайдуку Даніїлу Андрійовичу

1. Тема роботи *«Дослідження алгоритмів комп'ютерного зору для розпізнавання жестів»*

затверджена наказом університету від 15 квітня 2025 р. № 290Ст

2. Термін подання студентом роботи до екзаменаційної комісії 18 червня 2025 р.
3. Вихідні дані до роботи *дослідження процесів обробки та класифікації відеопотоку для розпізнавання жестів людини; розробка та експериментальна перевірка ефективних методів для класифікації жестів за допомогою алгоритмів комп’ютерного зору.*
4. Перелік питань, що потрібно опрацювати у роботі *вступ, огляд літератури та аналіз проблеми, основні етапи розвитку комп'ютерного зору, сучасні алгоритми розпізнавання жестів, опис проведених експериментів, аналіз результатів експериментальних досліджень, визначення ефективності алгоритму розпізнавання жестів, архітектура та функціональні можливості системи, тестування та оптимізація системи, можливості використання результатів у практичній діяльності, сфери застосування системи розпізнавання жестів, рекомендації щодо впровадження у виробництво, висновки.*

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1	Огляд літератури, аналіз проблеми та постановка задачі дослідження	27.01.2025 – 12.02.2025	виконано
2	Методологія дослідження та вибір алгоритму розпізнавання жестів	12.02.2025 – 26.02.2025	виконано
3	Проведення експериментальних досліджень та аналіз результатів	26.02.2025 – 25.03.2025	виконано
4	Розробка програмної системи для розпізнавання жестів	25.02.2025 – 18.05.2025	виконано
5	Оцінка можливостей використання результатів у практичній діяльності	01.04.2025 – 29.04.2025	виконано
6	Підготовка пояснювальної записки	22.04.2025 – 20.05.2025	виконано
7	Перевірка на плагіат та нормоконтроль	29.05.2025	виконано
8	Підготовка презентації та доповіді	03.06.2025	виконано
9	Рецензування	06.06.2025	виконано
10	Попередній захист	07.06.2025	виконано
11	Занесення диплома в електронний архів	08.06.2025	виконано
12	Допуск до захисту у зав. кафедри	11.06.2025	виконано

Дата видачі завдання 27.01.2025 р.

Студент (ка) _____ Даніїл ГАЙДУК
(підпис)

Керівник роботи _____ доц. Віра ГОЛЯН
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ / ABSTRACT

Робота містить: 95 с., 21 рис., 7 табл., 17 джер.

ВЗАЄМОДІЯ ЛЮДИНА-КОМП'ЮТЕР, ВІДЕОАНАЛІЗ, ГЛИБОКЕ НАВЧАННЯ, КОМП'ЮТЕРНИЙ ЗІР, РОЗПІЗНАВАННЯ ЖЕСТІВ, CNN + RNN (LSTM), CNN, HOG, KNN, LSTM, RNN, SVM, VIT.

Об'єктом дослідження є процес розпізнавання жестів на основі алгоритмів комп'ютерного зору. Цей процес охоплює послідовні етапи: захоплення відеопотоку, попередню обробку зображень, виявлення ключових точок тіла або рук, а також класифікацію динамічних і статичних жестів.

Метою роботи є проведення комплексного дослідження ефективності алгоритмів комп'ютерного зору для розпізнавання жестів із акцентом на аналіз їхньої точності, адаптивності до змін освітлення, фону та індивідуальних особливостей виконання жестів, а також визначення оптимальної моделі для подальшої реалізації в системах взаємодії «людина–комп'ютер».

Методи розробки та проєктування базуються на комплексному підході, що включає традиційну обробку зображень, виявлення ключових точок з використанням бібліотеки MediaPipe, а також застосування глибокого навчання – зокрема, згорткових нейронних мереж (CNN) для виділення просторових ознак і рекурентних мереж з довготривалою пам'яттю (LSTM) для аналізу часових послідовностей. Розробка програмної системи здійснювалась мовою Python з використанням бібліотек OpenCV, PyTorch Lightning, MediaPipe, matplotlib і NumPy. Під час реалізації також використовувались методи нормалізації зображень, фільтрації шумів, анотації даних та валідації точності за метриками accuracy, precision, recall і F1-score.

В результаті роботи було проаналізовано переваги й недоліки існуючих алгоритмів розпізнавання жестів, проведено їх порівняння за ключовими характеристиками, обґрунтовано вибір оптимального підходу. Як результат, було реалізовано програмну систему на основі алгоритму CNN + RNN (LSTM), що

продемонструвала високу ефективність розпізнавання жестів на тестових даних (accuracy – 92.0%, F1-score – 91.3%). Система здатна працювати в режимі реального часу, не вимагаючи спеціального апаратного забезпечення, і може бути застосована в галузях доповненої реальності, безконтактного керування пристроями, робототехніки, реабілітації тощо. Результати дослідження мають прикладне значення для розробки адаптивних інтерфейсів взаємодії «людина–комп'ютер» і закладають основу для подальших досліджень у сфері мультимодального розпізнавання.

CNN, CNN + RNN (LSTM), COMPUTER VISION, DEEP LEARNING, GESTURE RECOGNITION, HOG, HUMAN–COMPUTER INTERACTION, KNN, LSTM, RNN, SVM, VIDEO ANALYSIS, VIT.

The object of research is the process of gesture recognition based on computer vision algorithms. This process includes several sequential stages: video stream capture, image preprocessing, key point detection of the body or hands, and the classification of dynamic and static gestures.

The aim of the work is to conduct a comprehensive study of the effectiveness of computer vision algorithms for gesture recognition, focusing on analyzing their accuracy, adaptability to varying lighting conditions, backgrounds, and individual features of gesture performance, as well as determining the optimal model for further implementation in human-computer interaction systems.

Development and design methods are based on a comprehensive approach, which includes traditional image processing, key point detection using the MediaPipe library, as well as the application of deep learning – specifically, Convolutional Neural Networks (CNN) for extracting spatial features and Recurrent Neural Networks with Long Short-Term Memory (LSTM) for analyzing temporal sequences. The software system was developed using Python with libraries such as OpenCV, PyTorch Lightning, MediaPipe, matplotlib, and NumPy. During implementation, methods such as image normalization, noise filtering, data annotation, and accuracy validation using metrics such as accuracy, precision, recall, and F1-score were also applied.

As a result of the work, the advantages and disadvantages of existing gesture recognition algorithms were analyzed, and they were compared based on key characteristics, justifying the selection of the optimal approach. As a result, a software system was implemented based on the CNN + RNN (LSTM) algorithm, which demonstrated high efficiency in gesture recognition on test data (accuracy – 92.0%, F1-score – 91.3%). The system is capable of operating in real-time without requiring specialized hardware and can be applied in areas such as augmented reality, contactless device control, robotics, rehabilitation, and more. The results of the research are of practical value for the development of adaptive human-computer interaction interfaces and lay the foundation for further research in the field of multimodal recognition.

Завідувачу кафедри
ПІ_____
(скорочена назва кафедри)
проф. Кирилу СМЕЛЯКОВУ_
(вчене звання, сласне ім’я, прізвище)

ЗАЯВА
щодо самостійності виконання кваліфікаційної роботи та можливості її публікації (та/або публікації анотації кваліфікаційної роботи) в електронному архіві відкритого доступу ElAr KhNURE

Я, Гайдук Даніїл Андрійович студент групи ІПЗм-23-3 здобувач вищої освіти на другому (магістерському)рівні кафедра програмної інженерії, заявляю: моя кваліфікаційна робота на тему Дослідження алгоритмів комп'ютерного зору для розпізнавання жестів, що буде представлена в екзаменаційну комісію для публічного захисту, виконана самостійно, в ній не містяться елементи плагіату і вона може бути опублікована в електронному архіві відкритого доступу ElArKhNURE. Всі запозичення з друкованих та електронних джерел мають відповідні посилання.

Я ознайомлений з діючим положенням «Про протидію академічному плагіату в ХНУРЕ», згідно з яким виявлення плагіату є підставою для відмови в допуску кваліфікаційної роботи до захисту та застосування дисциплінарних заходів.

Дата Підпис

ЗМІСТ

Перелік скорочень 10
Вступ 12
1 Огляд літератури та аналіз проблеми 15
1.1 Основні етапи розвитку комп'ютерного зору 15
1.2 Сучасні алгоритми розпізнавання жестів 17
1.3 Аналіз попередніх досліджень 21
1.4 Протиріччя та невирішені питання 24
1.5 Постановка задачі 26
2 Методологія дослідження та вибір алгоритмів розпізнавання жестів 29
2.1 Опис та обґрунтування обраних алгоритмів 29
2.1.1 Метод опорних векторів (SVM) 30
2.1.2 K-найближчі сусіди (KNN) 33
2.1.3 Гістограми орієнтованих градієнтів (HOG) 35
2.1.4 Згорткові нейронні мережі (CNN) 38
2.1.5 Рекурентні нейронні мережі (RNN) та LSTM 41
2.1.6 Vision Transformers (ViT) 44
2.1.7 Гібридні підходи: комбінація CNN та RNN (або LSTM) 48
2.2 Порівняльна оцінка алгоритмів 50
3 Результати та аналіз досліджень 60
3.1 Опис проведених експериментів 60
3.2 Аналіз результатів експериментальних досліджень 62
3.3 Визначення ефективності алгоритму розпізнавання жестів 65
4 Розробка програмної системи для розпізнавання жестів 68
4.1 Технологія та етапи розробки 68
4.2 Архітектура та функціональні можливості системи 70
4.3 Тестування та оптимізація системи 84
5 Можливості використання результатів у практичній діяльності 89
5.1 Сфери застосування системи розпізнавання жестів 89
5.2 Рекомендації щодо впровадження у виробництво 90

Висновки 91
Перелік джерел посилань 93
Перелік джерел посилання за науковими напрямами керівника та науковців кафедри програмної інженерії 95

ПЕРЕЛІК СКОРОЧЕНЬ

Accuracy – точність;

AutoPy – бібліотека для автоматизації керування мишею та клавіатурою;

BGR – Blue, Green, Red (формат кольору зображень);

CNN – Convolutional Neural Network (згорткова нейронна мережа);

CPU – Central Processing Unit (центральний процесор);

cv2 – Модуль OpenCV для Python;

F1-score – F1-міра;

GPU – Graphics Processing Unit (графічний процесор);

GMM – Gaussian Mixture Models (змішані моделі гаусса);

HOG – Histogram of Oriented Gradients (гістограма орієнтованих градієнтів);

HOG-SVM – Histogram of Oriented Gradients – Support Vector Machine (гістограма орієнтованих градієнтів – машина опорних векторів);

ILY – I Love You (жест "ILY");

K-means – K-means Clustering (метод кластеризації);

KNN – K-Nearest Neighbors (метод K найближчих сусідів);

LSTM – Long Short-Term Memory (довга короткочасна пам'ять);

MediaPipe – бібліотека для обробки мультимедійних даних, розроблена Google;

ms – мілісекунда;

MNIST – Modified National Institute of Standards and Technology;

NumPy – бібліотека для обробки великих масивів та матриць у Python;

OpenCV – Open Source Computer Vision Library (бібліотека комп'ютерного зору);

Pinch – жест для перемикання вкладок;

Precision – точність виявлення;

PyTorch Lightning – фреймворк для спрощення побудови та тренування нейронних мереж;

pynput – бібліотека для контролю миші та клавіатури;

pycaw – бібліотека для управління гучністю системи;

ReLU – Rectified Linear Unit (виправлена лінійна одиниця);

Recall – повнота;

RNN – Recurrent Neural Network (рекурентна нейронна мережа);

RNN (LSTM) – Recurrent Neural Network with Long Short-Term Memory (рекурентна нейронна мережа з довготривалою короткочасною пам’яттю);

RBF – Radial Basis Function (ядро радіальної базисної функції);

Shaka – жест для прокручування сторінок;

SNR – Signal-to-Noise Ratio (співвідношення сигнал/шум);

SVM – Support Vector Machine (метод опорних векторів);

ViT – Vision Transformer (перетворювальна модель зображень);

ViT – Vision Transformers (трансформери для комп'ютерного зору);

x, y, z – координати в тривимірному просторі;

YCbCr – Luminance, Chrominance (колірний простір).

ВСТУП

У сучасному світі комп’ютерне зорове сприйняття відіграє ключову роль у розвитку інтелектуальних систем, автоматизації процесів і вдосконаленні взаємодії людини з технікою. Однією з найперспективніших і найбільш актуальних сфер застосування комп’ютерного зору є розпізнавання жестів. Це технологія, що дозволяє комп’ютерним системам інтерпретувати сигнали, які генеруються рухами людини, надаючи можливість безконтактного управління пристроями, інтерактивного навчання та вдосконалення взаємодії людини з роботизованими системами [1]. Зокрема, алгоритми розпізнавання жестів можуть бути використані в таких галузях, як віртуальна та доповнена реальність, реабілітація, робототехніка, автомобільна промисловість, а також у сфері безпеки.

Актуальність теми обумовлена швидким розвитком технологій комп’ютерного зору та потребою в інтерактивних та інтуїтивно зрозумілих системах, що дозволяють взаємодіяти з комп’ютерами та пристроями без фізичного контакту. Жести, як природна форма комунікації, є дуже перспективним напрямком, оскільки здатні замінити традиційні способи введення інформації, такі як клавіатури та миші [2]. Пошук ефективних алгоритмів для розпізнавання жестів є важливим етапом у розвитку таких систем. Однак існуючі методи часто стикаються з проблемами точності, швидкості та адаптивності до різних умов навколишнього середовища, що потребує додаткових досліджень.

Сучасні методи розпізнавання жестів часто стикаються з низкою проблем, серед яких: низька точність при змінних умовах освітлення, наявність шуму та фонових перешкод, складність адаптації до різних користувачів та обмежена швидкість обробки даних в реальному часі [3]. У зв’язку з цим, метою даного дослідження є розробка та вдосконалення алгоритмів для подолання цих проблем, забезпечуючи високу точність, швидкість і адаптивність в реальних умовах. Для досягнення цієї мети необхідно вирішити низку завдань:

- провести огляд існуючих алгоритмів розпізнавання жестів та їхніх основних особливостей;

- здійснити порівняння алгоритмів за такими параметрами, як точність, швидкість обробки та здатність до адаптації;
- розробити нові або вдосконалити існуючі алгоритми для підвищення їхньої ефективності в умовах змінного освітлення, різних поз та фонів;
- розробити програмну систему для тестування та впровадження запропонованих алгоритмів;
- оцінити практичне значення результатів у реальних умовах та визначити напрямки для подальших досліджень.

Об’єктом дослідження є процес розпізнавання жестів за допомогою комп’ютерного зору. Це складний процес, що включає в себе кілька етапів: захоплення зображення, обробка зображень, виявлення ключових точок та аналіз рухів рук або тіла, а також класифікація та інтерпретація жестів.

Предметом дослідження є алгоритми комп’ютерного зору для розпізнавання жестів, що включають методи глибокого навчання, методи обробки зображень, а також різноманітні підходи до виявлення і класифікації жестів на основі відео та зображень.

Методи дослідження, що використовуватимуться в роботі для досягнення поставленої мети, включають комплексний підхід, поєднуючи традиційні методи обробки зображень з передовими методами глибокого навчання для розпізнавання жестів. Кожен з методів націлений на вирішення конкретних завдань, пов’язаних із точністю, швидкістю та адаптивністю системи до різних умов навколишнього середовища. Одним із основних етапів дослідження є попередня обробка зображень, що включає фільтрацію шумів, корекцію освітлення та підвищення контрасту. Для цього будуть використані стандартні методи фільтрації та алгоритми корекції освітлення, які дозволяють підготувати зображення до подальшої обробки. Це дозволить усунути вплив несприятливих умов освітлення та знизити вплив фонових перешкод. Для точного виявлення і відстеження рухів рук або частин тіла використовуватимуться методи виділення ключових точок на зображеннях. Для класифікації жестів будуть застосовуватися методи глибокого навчання, зокрема згорткові нейронні мережі (CNN), що дозволяють автоматично

видобувати ознаки з великих масивів даних та забезпечувати високу точність класифікації. Після впровадження та тренування алгоритмів для розпізнавання жестів буде здійснено порівняння різних підходів за такими параметрами, як точність, швидкість обробки та здатність до адаптації в реальних умовах. Оцінка ефективності алгоритмів буде виконуватись за допомогою експериментальних тестів, що імітують різні умови, включаючи змінне освітлення та різні фони. На основі результатів порівняльного аналізу буде розроблено програмне забезпечення, яке включатиме реалізацію запропонованих алгоритмів. Система буде призначена для тестування й оптимізації розпізнавання жестів у реальних умовах, а також дозволить здійснювати подальше вдосконалення алгоритмів на основі отриманих результатів. Останнім етапом дослідження стане оцінка ефективності застосування алгоритмів у реальних умовах. Це включатиме тестування на практиці з використанням конкретних сценаріїв, таких як віртуальна реальність, безконтактне управління пристроями, робототехніка та інші.

Наукова новизна роботи полягає в удосконаленні існуючих алгоритмів розпізнавання жестів шляхом використання нових методів обробки зображень і глибокого навчання для підвищення точності розпізнавання в умовах змінного освітлення і різноманітних фонових умов. Вперше в роботі буде запропоновано поєднання кількох підходів до класифікації жестів, що дозволить досягти кращих результатів у порівнянні з відомими методами.

Результати цього дослідження можуть бути використані для створення інтерактивних систем, де управління здійснюється за допомогою жестів. Це може бути корисно в таких сферах, як розробка віртуальних та доповнених реальностей, систем безконтактного управління, робототехніка та реабілітація. Розроблені алгоритми можуть бути застосовані для вдосконалення взаємодії людини з комп’ютером, що підвищить ефективність і комфорт використання технічних пристроїв. Крім того, результати дослідження можуть бути використані для вдосконалення алгоритмів розпізнавання жестів у рамках розробки інтерфейсів для людей з обмеженими можливостями.

1 ОГЛЯД ЛІТЕРАТУРИ ТА АНАЛІЗ ПРОБЛЕМИ

1.1 Основні етапи розвитку комп’ютерного зору

Комп’ютерний зір – це галузь штучного інтелекту, що займається автоматичним аналізом, обробкою та інтерпретацією зображень чи відео з метою створення машин, здатних «бачити» та розуміти візуальну інформацію, як люди [4]. Його розвиток охоплює кілька ключових етапів, кожен з яких став можливим завдяки прогресу в апаратних технологіях, математичних алгоритмах і методах навчання.

Перші спроби комп’ютерного зору розпочались у 1950-х роках. Ці ранні дослідження включали розпізнавання простих об’єктів та двовимірних форм. В той час обчислювальна потужність була дуже обмеженою, тому алгоритми були орієнтовані на базові операції, як-от:

- розпізнавання форм і контурів: програми розпізнавали прості геометричні форми, такі як кола та квадрати;
- аналіз яскравості: оцінка рівня освітлення для визначення об’єкта на тлі.

Це був фундаментальний період для комп’ютерного зору, коли формувалися перші теоретичні концепції, які заклали основу для подальших досліджень.

З появою більш потужних комп’ютерів у 1970-х роках комп’ютерний зір почав зосереджуватися на аналізі ознак, де окремі риси зображення – контури, краї, текстури та інші характеристики – використовувались для опису об’єктів. Основні досягнення цього періоду включають:

- метод згорток і фільтрів: такі методи як фільтр Собеля і Лапласа використовувалися для виділення країв об’єктів;
- сегментація зображень: розвиток алгоритмів, що поділяли зображення на сегменти для кращої ідентифікації об’єктів;
- аналіз руху: вивчення методів виявлення та інтерпретації руху об’єктів у відео, що стало основою для подальшого розвитку візуального трекінгу;
- моделі виявлення об’єктів: застосування методів класифікації для

визначення об’єктів за ознаками.

Ці підходи були обмежені в своїх можливостях, проте стали важливою базою для подальшого вдосконалення методів розпізнавання об’єктів.

У 1990-х роках з’явився великий інтерес до статистичних методів, які дозволили враховувати варіативність даних та більш точно розпізнавати об’єкти. У цей період було впроваджено методи, які зосереджувалися на навчанні системи розпізнаванню:

- метод опорних векторів: ефективний інструмент для класифікації об’єктів;
- аналіз головних компонент: використовувався для зменшення розмірності даних, що полегшувало обробку зображень і виділення основних характеристик;
- класифікація на основі ознак та алгоритми кластеризації: такі як k-середнє (k-means), які дозволили групувати подібні елементи зображень, полегшуючи розпізнавання.

Цей період був знаковим через застосування алгоритмів, які змогли обробляти складніші дані та сприяли зростанню точності розпізнавання.

Революційні зміни в комп’ютерному зорі відбулися у 2010-х роках завдяки зростанню обчислювальних потужностей та появі великих наборів даних, які сприяли розвитку глибокого навчання:

- конволюційні нейронні мережі (CNN): моделі, що зробили прорив у розпізнаванні образів завдяки здатності ієрархічно виділяти ознаки та автоматично навчатися на великих наборах даних. Одними з найбільш відомих архітектур є AlexNet, ResNet, VGG;
- аналіз і класифікація зображень: CNN стали основним підходом у таких завданнях, як класифікація зображень, розпізнавання облич, жестів, об’єктів;
- моделі розпізнавання об’єктів, як R-CNN, Fast R-CNN, YOLO: ці моделі значно підвищили швидкість і точність розпізнавання та локалізації об’єктів на зображеннях і відео;

– тренувальні набори даних (ImageNet, COCO): поява великих, якісно розмічених наборів даних дозволила мережам глибокого навчання швидко підвищити точність розпізнавання.

Поширення глибокого навчання змінило галузь комп'ютерного зору, надавши змогу системам досягати високих показників точності у реальних умовах.

Сучасний етап розвитку комп'ютерного зору включає методи самонавчання, які використовують непідписані дані та дозволяють комп'ютерам вчитися без нагляду:

– самонавчання та слабке навчання: методи, що дозволяють мережам навчатися з меншою кількістю підписаних даних, зменшуючи потребу у великих наборах даних;
– генеративні змагальні мережі (GAN): використовуються для створення синтетичних зображень, що підвищує якість навчання алгоритмів розпізнавання;
– трансформери та візуально-мовні моделі (наприклад, DALL-E): комбінують зір та обробку природної мови для створення більш адаптивних і універсальних систем.

Отже, еволюція комп'ютерного зору пройшла від базових алгоритмів розпізнавання контурів до складних моделей глибокого навчання, здатних самонавчатися та адаптуватися до різноманітних умов. Це дозволяє застосовувати технологію у широкому спектрі завдань – від автономних автомобілів до віртуальної реальності, що надає нові можливості для безконтактної взаємодії.

1.2 Сучасні алгоритми розпізнавання жестів

Сучасні алгоритми розпізнавання жестів базуються на потужних методах комп'ютерного зору та машинного навчання, які дозволяють ефективно аналізувати та ідентифікувати рухи людини для інтерпретації жестів. Ці алгоритми знаходять застосування в різних сферах, включаючи віртуальну та доповнену реальність, робототехніку, реабілітаційні технології, інтерфейси

користувача та системи безпеки.

Класичні методи обробки зображень передбачають обробку відео та зображень для ідентифікації жестів без використання глибоких нейронних мереж:

- методи на основі оптичного потоку використовуються для відстеження руху об’єктів (рук або пальців) на послідовності кадрів. Оптичний потік дозволяє виміряти зміну яскравості між сусідніми кадрами, що дає змогу визначити напрямок і швидкість руху;
- методи виділення ключових точок та контурів дозволяють фіксувати певні ключові точки (суглоби пальців, зап’ястя тощо) та контури руки для визначення положення та форми жесту. До таких методів відносяться порогування та морфологічна обробка;
- характеристики на основі кольору або форми застосовуються для розпізнавання жестів за допомогою аналізу кольору шкіри та форми руки. Наприклад, сегментація за кольором шкіри дозволяє виділити руку із фону, а аналіз форми визначити конкретні жести.

Класичні методи машинного навчання (ML) дозволяють розпізнавати жести, використовуючи навчальні алгоритми на основі спеціально виділених ознак:

- метод опорних векторів (SVM) використовується для класифікації жестів, аналізуючи векторні ознаки рухів рук або форм. SVM ефективний для розмежування різних класів жестів за допомогою гіперплощини;
- K-найближчі сусіди (KNN) використовується для класифікації на основі схожості між новим жестом і жестами в базі даних. KNN легко реалізувати, однак він не дуже підходить для обробки великих обсягів даних;
- гістограми орієнтованих градієнтів (HOG) є одним з методів виділення характеристик і часто використовується для ідентифікації контурів та форм, дозволяючи розпізнавати фіксовані пози жестів.

Глибоке навчання стало ключовою технологією в розпізнаванні жестів,

завдяки здатності нейронних мереж автоматично виділяти ієрархічні ознаки та обробляти складні дані:

- Convolutional Neural Networks (CNN) є основою для більшості сучасних алгоритмів комп'ютерного зору. Вони використовуються для виділення характеристик жестів із зображень або послідовності відеокадрів. CNN добре розпізнає як статичні жести, так і пози рук;
- Recurrent Neural Networks (RNN) та Long Short-Term Memory (LSTM): RNN та їхня модифікація LSTM ефективні для розпізнавання послідовностей рухів у відео. Вони зберігають інформацію про попередні кадри, що дозволяє розпізнавати динамічні жести, тобто жести, які виконуються у певній послідовності;
- 3D CNN: на відміну від звичайних CNN, які працюють з 2D-зображеннями, 3D CNN аналізують послідовність кадрів як об'ємні дані, що допомагає ефективніше розпізнавати динамічні рухи;
- Attention Mechanisms: механізми уваги, інтегровані в моделі, дозволяють виділяти найбільш значущі частини зображення або послідовності. Це покращує точність розпізнавання, фокусуючись на окремих ділянках рук або пальців, що є важливими для розпізнавання жесту.

Багато сучасних алгоритмів об'єднують кілька підходів, щоб забезпечити високу точність та надійність розпізнавання жестів:

- комбінація CNN та RNN (або LSTM): CNN витягує ознаки з кожного кадру, а потім RNN або LSTM аналізує послідовність цих ознак, що дає можливість розпізнавати як статичні, так і динамічні жести;
- спільне використання 2D та 3D ознак: деякі системи використовують 2D CNN для витягування ознак з кожного кадру, а 3D CNN для аналізу об'ємних даних відеопослідовності, що дає додатковий рівень точності для динамічних жестів;
- мультимодальні підходи: використання різних сенсорів (камер RGB, камер глибини, сенсорів руху) для отримання різносторонніх даних дозволяє досягти більш точного розпізнавання. Наприклад, поєднання

RGB-камери і камери глибини дозволяє виділити об’єкти з фону та відслідковувати рух у тривимірному просторі.

Алгоритми на основі трансформерів, які показали ефективність у галузі обробки природної мови, також знаходять застосування в розпізнаванні жестів:

- Vision Transformers (ViT) – це модель, яка розбиває зображення на патчі (блоки) та обробляє їх, щоб знайти просторові взаємозв’язки. ViT може забезпечити високу точність для статичних жестів;
- TimeSformer – це модель, яка комбінує трансформери з урахуванням часових взаємозв’язків, що дозволяє аналізувати послідовності відеокадрів для розпізнавання динамічних жестів;
- Multimodal Transformers – моделі, що обробляють дані з кількох джерел, можуть аналізувати як візуальні, так і аудіо або сенсорні дані для підвищення точності розпізнавання жестів.

Попри значний прогрес, алгоритми розпізнавання жестів все ще стикаються з рядом викликів:

- зміни в умовах освітлення та фоновий шум: системи часто помиляються при зміні освітлення або складному фоні, що може призвести до зниження точності;
- адаптація до індивідуальних особливостей користувачів: користувачі мають різні розміри рук, різні стилі виконання жестів, що ускладнює уніфіковане розпізнавання;
- реалізація в реальному часі: обробка відео в реальному часі вимагає значних обчислювальних ресурсів, тому важливо оптимізувати моделі для швидкої роботи.

Таким чином, сучасні алгоритми розпізнавання жестів надають потужний інструментарій для аналізу та інтерпретації рухів людини, забезпечуючи інтерактивні та інтуїтивно зрозумілі інтерфейси. Щоб досягти нових успіхів, необхідно вдосконалювати існуючі методи для підвищення точності, адаптивності та швидкості роботи в умовах реального світу.

1.3 Аналіз попередніх досліджень

Дослідження алгоритмів комп’ютерного зору для розпізнавання жестів активно розвиваються вже кілька десятиліть, починаючи з 90-х років минулого століття. У цій галузі проводилися численні експерименти, метою яких було підвищення точності розпізнавання жестів, зниження впливу зовнішніх факторів, а також оптимізація продуктивності систем.

Попередні дослідження зосереджувалися на створенні систем, здатних розпізнавати два основні типи жестів:

- статичні жести – жести, що залишаються незмінними в часі, наприклад, положення руки або пальців у певній конфігурації;
- динамічні жести – послідовності рухів, які необхідно аналізувати в часовому контексті, наприклад, махання рукою чи вказування напрямку.

Для кожного з цих типів жестів були запропоновані різні алгоритми, які розвивалися в залежності від доступних обчислювальних ресурсів і технологій збору даних.

Ранні дослідження у цій галузі базувалися на класичних методах комп’ютерного зору:

- сегментація кольору шкіри: використовувалася для виділення областей зображення, що відповідають руці. Це досягалося шляхом застосування порогових значень у кольорових просторах, таких як HSV або YCbCr;
- виділення контурів: застосування алгоритмів, таких як метод Кенні або алгоритм Гафа, для виділення меж руки та її основних характеристик;
- аналіз руху: використання оптичного потоку для ідентифікації рухів об’єктів у послідовностях зображень;
- алгоритми трекінгу: методи Лукаса-Канаде використовувалися для відстеження ключових точок руки в часових послідовностях.

Попри простоту та низькі обчислювальні витрати, такі підходи мали обмеження, зокрема чутливість до умов освітлення, складність фону та індивідуальні відмінності кольору шкіри.

У 2000-х роках дослідження перейшли до використання машинного навчання для класифікації жестів. Основні алгоритми включали:

- метод опорних векторів (Support Vector Machine, SVM): використовувався для класифікації статичних жестів на основі таких ознак, як форма руки, положення пальців та їх орієнтація;
- алгоритми кластеризації: методи K-means і GMM застосовувалися для ідентифікації характерних рухів руки;
- гістограми орієнтованих градієнтів (HOG): використовувалися для виділення основних контурів руки, що дозволяло ефективно аналізувати її форму.

Ці методи дозволяли досягати значної точності при розпізнаванні жестів у лабораторних умовах. Проте їх ефективність знижувалася у реальних сценаріях через складний фон та непередбачувані рухи.

Розвиток апаратного забезпечення відкрив нові можливості для дослідження жестів:

- камери глибини: пристрої, такі як Microsoft Kinect, використовувалися для збору тривимірної інформації про положення руки. Це дозволяло обійти проблеми, пов’язані з кольором шкіри або фоном;
- рукавички з сенсорами: рукавички, оснащені інерційними датчиками (акселерометрами та гіроскопами), забезпечували точне вимірювання положення і руху пальців. Хоча такі рішення давали високу точність, вони не були зручними для широкого використання;
- Leap Motion: пристрій, який дозволяв виявляти рухи рук без фізичного контакту, став основою для багатьох досліджень у сфері динамічних жестів.

Поява алгоритмів глибокого навчання стала проривом у розпізнаванні жестів. Ключові досягнення включають:

- згорткові нейронні мережі (Convolutional Neural Networks, CNN): широко використовувалися для виділення просторових характеристик статичних жестів. Популярні моделі, такі як AlexNet, VGG, ResNet, забезпечували

високі результати на наборах даних зображень рук;

- рекурентні нейронні мережі (Recurrent Neural Networks, RNN) та LSTM (Long Short-Term Memory): ці архітектури були особливо ефективними для розпізнавання динамічних жестів, оскільки враховували часові залежності між послідовними кадрами;
- 3D-CNN: дозволяли одночасно аналізувати просторові та часові характеристики, що особливо актуально для розпізнавання жестів у відеопотоках;
- трансформери: сучасні дослідження вказують на високу ефективність трансформерів, таких як Vision Transformers (ViT), у завданнях розпізнавання жестів.

Попередні дослідження спиралися на різноманітні набори даних, серед яких найпопулярніші:

- NVIDIA Hand Gesture Dataset – великий набір статичних жестів з різними положеннями рук;
- MSR Gesture 3D Dataset – набір даних для динамічних жестів, зібраний за допомогою камер глибини;
- LeapMotion Hand Gesture Dataset – дані про рухи рук, зібрані за допомогою пристрою Leap Motion;
- EgoHands Dataset – набір даних для жестів, виконаних у сценаріях з першої особи.

Проблеми, які були виявлені у попередніх дослідженнях:

- освітлення та фон: системи демонстрували нестабільність у складних умовах освітлення або на неоднорідному фоні;
- індивідуальні відмінності: жести різних людей могли значно відрізнятися, що впливало на точність розпізнавання;
- швидкодія: алгоритми глибокого навчання потребують значних обчислювальних ресурсів, що ускладнює їх використання в реальному часі;
- обмеження сенсорів: деякі пристрої, наприклад, рукавички або камери

глибини, створюють дискомфорт або мають обмежену зону дії.

Отже, попередні дослідження створили міцну базу для подальших розробок у сфері комп'ютерного зору для розпізнавання жестів. Незважаючи на досягнення, все ще існують значні виклики, які потребують вирішення, такі як підвищення точності у реальних умовах, зниження обчислювальних витрат та інтеграція новітніх технологій, таких як трансформери або сенсори нового покоління.

1.4 Протиріччя та невирішені питання

Попри значний прогрес у галузі комп'ютерного зору для розпізнавання жестів, залишається низка невирішених питань, які стримують розвиток технологій та їх практичне застосування. Ці виклики мають як технічний, так і концептуальний характер, що пов'язано з протиріччями між потребами вищої точності розпізнавання, зниженням обчислювальних витрат і адаптацією до реальних умов використання.

Одним із ключових протиріч є вибір між простотою класичних методів комп'ютерного зору та складністю сучасних алгоритмів глибокого навчання. Класичні підходи, такі як сегментація кольору шкіри або виділення контурів, демонструють високу ефективність у контрольованих умовах. Однак вони надзвичайно чутливі до змін освітлення, неоднорідного фону чи унікальних особливостей користувача, таких як тон шкіри або форма руки. З іншого боку, глибокі нейронні мережі дозволяють долати ці обмеження, але потребують великих обчислювальних ресурсів і великих навчальних наборів даних, які іноді важко зібрати чи адаптувати для конкретних завдань.

Іншим важливим протиріччям є баланс між універсальністю моделей і спеціалізованим підходом до вирішення задач. Універсальні моделі, такі як Vision Transformers чи 3D-CNN, здатні опрацьовувати широкий спектр жестів, проте вони часто не оптимізовані для конкретних сценаріїв, таких як розпізнавання жестів у відеозв'язку чи взаємодії з пристроями доповненої реальності. Натомість спеціалізовані моделі можуть демонструвати чудову продуктивність у вузьких

сферах, але втрачають гнучкість і масштабованість, що обмежує їх застосування у змінних умовах.

Серед невирішених питань ключовим залишається забезпечення стійкості алгоритмів до зовнішніх факторів. Наприклад, освітлення суттєво впливає на точність як класичних, так і сучасних підходів. Використання камер глибини та сенсорів дозволяє частково розв’язати цю проблему, але їх висока вартість та апаратні обмеження стримують широке впровадження. Крім того, умови реального середовища, такі як переповнений фон, швидкі рухи чи непередбачувані жести, створюють значні виклики для існуючих моделей, які здебільшого тренуються на даних, зібраних у контрольованих умовах.

Ще однією важливою проблемою є адаптація алгоритмів до індивідуальних відмінностей користувачів. Форма руки, довжина пальців, а також спосіб виконання жестів можуть значно варіюватися залежно від людини. Це створює труднощі у стандартизації даних та потребує розробки алгоритмів, здатних до персоналізації. Деякі дослідження пропонують використовувати методи перенавчання або few-shot learning для адаптації до нових користувачів, проте це збільшує обчислювальні витрати та ускладнює інтеграцію таких моделей у реальному часі.

Проблема навчання моделей також залишається відкритою. Наявні набори даних, такі як EgoHands або MSR Gesture 3D, забезпечують базу для досліджень, але вони часто обмежені у кількості прикладів або різноманітності сценаріїв. Більшість таких наборів даних не враховують фактори, характерні для реального середовища, такі як різноманітні ракурси камер, перешкоди в кадрі чи асинхронні жести. Це обмежує здатність моделей узагальнювати свої результати та успішно працювати в складних умовах. Крім того, збір та анотування великих обсягів даних є тривалим і дорогим процесом.

Ще одне суттєве протиріччя виникає у використанні сенсорних пристроїв. Камери глибини, наприклад, забезпечують високу точність розпізнавання за рахунок тривимірної інформації, але їх ефективність обмежена відстанню, на якій працює сенсор. Рукавички з вбудованими датчиками забезпечують чудову

точність для розпізнавання рухів пальців, але вони не є зручними у використанні і не підтримують природність жестової взаємодії. Leap Motion та схожі пристрої дозволяють уникати фізичного контакту, але вимагають налаштування під певні сценарії та часто мають обмеження щодо поля зору.

Останнім і, можливо, найбільш концептуальним питанням є інтеграція різних підходів. Попередні дослідження демонструють, що комбінування алгоритмів класичного комп'ютерного зору, глибокого навчання та використання сенсорів може дати значний приріст у точності й продуктивності. Однак розробка таких інтегрованих систем потребує комплексного підходу, який враховує технічні та економічні обмеження, а також потреби кінцевих користувачів.

Отже, аналіз попередніх досліджень показує, що, незважаючи на значні досягнення у галузі комп'ютерного зору для розпізнавання жестів, існує низка невирішених питань і викликів. Одними з ключових протиріч є потреба у високій точності алгоритмів за умов змінного освітлення, складних фонів та варіативності жестів, а також забезпечення швидкодії в реальному часі при обмежених обчислювальних ресурсах. Крім того, залишається актуальним питання адаптації алгоритмів до індивідуальних особливостей користувачів та їхньої здатності ефективно працювати з різними типами даних і пристроїв. Тому у ході проведення дослідження передбачається створення універсального рішення, яке враховує специфіку реальних умов експлуатації, що дозволить значно розширити сферу практичного застосування технологій розпізнавання жестів у різних галузях.

1.5 Постановка задачі

Розпізнавання жестів є важливим напрямом у галузі комп'ютерного зору, що знаходить широке застосування в різних сферах, таких як взаємодія з людино-машинними інтерфейсами, системи розумного дому, медичні технології, роботи-помічники, а також у віртуальній та доповненій реальності. Успішна реалізація таких систем залежить від розробки точних, швидких та стійких до змін середовища алгоритмів, здатних працювати з великою кількістю жестів і

пристосовуватися до індивідуальних особливостей користувачів.

Аналіз літературних джерел показав, що комп’ютерний зір пройшов довгий шлях розвитку: від базового розпізнавання простих об’єктів до складних систем аналізу поведінки людини. Сучасні алгоритми, такі як моделі на основі згорткових нейронних мереж (CNN), рекурентних нейронних мереж (RNN) та їх комбінацій, демонструють високу ефективність у задачах розпізнавання жестів. Утім, залишається низка проблем: висока вимогливість до обчислювальних ресурсів, залежність від якісних даних для навчання, а також складнощі з адаптацією до різних умов експлуатації, наприклад, варіативності освітлення або шумів у фоні.

Попередні дослідження свідчать про досягнення значних результатів у створенні прототипів систем розпізнавання жестів, однак багато рішень обмежуються специфічними сценаріями або потребують великих обчислювальних потужностей. У результаті залишається нерозв’язаним питання створення універсальних алгоритмів, які могли б забезпечувати високу точність і продуктивність навіть у ресурсно обмежених умовах.

Існує також протиріччя між бажанням створити універсальне рішення та необхідністю враховувати специфіку кожної конкретної області застосування. Наприклад, у медичних технологіях вимоги до точності можуть бути пріоритетними, тоді як у розважальних системах важливішим може бути комфорт користувача та інтерактивність.

З огляду на вищезазначене, формулюється наступна задача дослідження: розробка та аналіз алгоритмів комп’ютерного зору для розпізнавання жестів, які забезпечують баланс між високою точністю, швидкодією та адаптивністю до змінних умов експлуатації, з подальшою розробкою програмної системи для їх практичного застосування.

Для вирішення завдання дослідження алгоритмів комп’ютерного зору для розпізнавання жестів також необхідно виконати комплексну роботу, яка передбачає такі етапи:

– провести аналіз алгоритмів розпізнавання жестів;

- виявити переваги та недоліки існуючих алгоритмів розпізнавання жестів, а також проблеми, що залишаються невирішеними;
- дослідити можливості використання алгоритмів комп’ютерного зору для розпізнавання жестів у різних умовах експлуатації (зміна освітлення, фонового шуму тощо);
- розробити методику порівняння алгоритмів за критеріями точності, швидкодії та адаптивності;
- обґрунтувати вибір найбільш перспективних алгоритмів для розв’язання задачі розпізнавання жестів;
- провести експериментальні дослідження обраних алгоритмів, аналізуючи їх ефективність у різних сценаріях використання;
- розробити програмну систему для розпізнавання жестів на основі найкращого із досліджених алгоритмів;
- забезпечити тестування та оптимізацію програмної системи для практичного використання у реальних умовах;
- надати рекомендації щодо впровадження розробленої системи в різних галузях (медицина, розумні системи, робототехніка тощо).

Отже, вирішення сформульованої задачі дозволить не лише розробити ефективний алгоритм розпізнавання жестів, але й сприятиме подальшому розвитку інноваційних систем на базі комп’ютерного зору, що знайдуть своє застосування в технологіях майбутнього.

2 МЕТОДОЛОГІЯ ДОСЛІДЖЕННЯ ТА ВИБІР АЛГОРИТМІВ РОЗПІЗНАВАННЯ ЖЕСТІВ

2.1 Опис та обґрунтування обраних алгоритмів

У контексті розробки та аналізу алгоритмів комп’ютерного зору для розпізнавання жестів, важливими є такі фактори, як точність, швидкодія, адаптивність до різних умов експлуатації (зміна освітлення, фоновий шум тощо), а також ефективність в умовах обмежених обчислювальних ресурсів. Враховуючи ці вимоги, обрані для дослідження алгоритми комп’ютерного зору можна розділити на кілька категорій: класичні методи, методи на основі глибинного навчання та гібридні підходи, які комбінують різні технології.

До класичних методів відносяться SVM, KNN та HOG. Ці алгоритми базуються на традиційних методах машинного навчання, де застосовуються математичні функції та моделі для аналізу зображень. Вони використовуються для вирішення різноманітних задач, таких як класифікація та виявлення об’єктів, але не мають глибинної обробки даних, як сучасні методи на основі нейронних мереж.

До методів на основі глибинного навчання відносяться CNN, RNN, LSTM та ViT. Ці алгоритми використовують багатошарові нейронні мережі для автоматичного виділення ознак зображень та аналізу складних залежностей. Завдяки здатності до самонавчання та обробки великих обсягів даних, вони демонструють високу ефективність у складних задачах комп’ютерного зору, таких як розпізнавання жестів та аналіз динамічних сцен.

До гібридних підходів відноситься комбінація CNN разом із RNN (або LSTM). Цей підхід поєднує сильні сторони згорткових нейронних мереж для виділення просторових ознак та рекурентних мереж для аналізу тимчасових залежностей. Такі комбіновані моделі ідеально підходять для задач, що потребують розпізнавання складних, динамічних жестів, де важливо враховувати як просторову, так і часову інформацію.

2.1.1 Метод опорних векторів (SVM)

Метод опорних векторів (SVM, Support Vector Machine) є одним із класичних алгоритмів машинного навчання для задач класифікації та регресії. Він активно використовується для розпізнавання образів, у тому числі для задач комп'ютерного зору, таких як розпізнавання жестів [5]. Основна ідея методу полягає в пошуку оптимальної гіперплощини (або гіперплощин для багатокласових задач), яка максимально розділяє різні класи даних в ознаковому просторі (див. рис. 2.1).

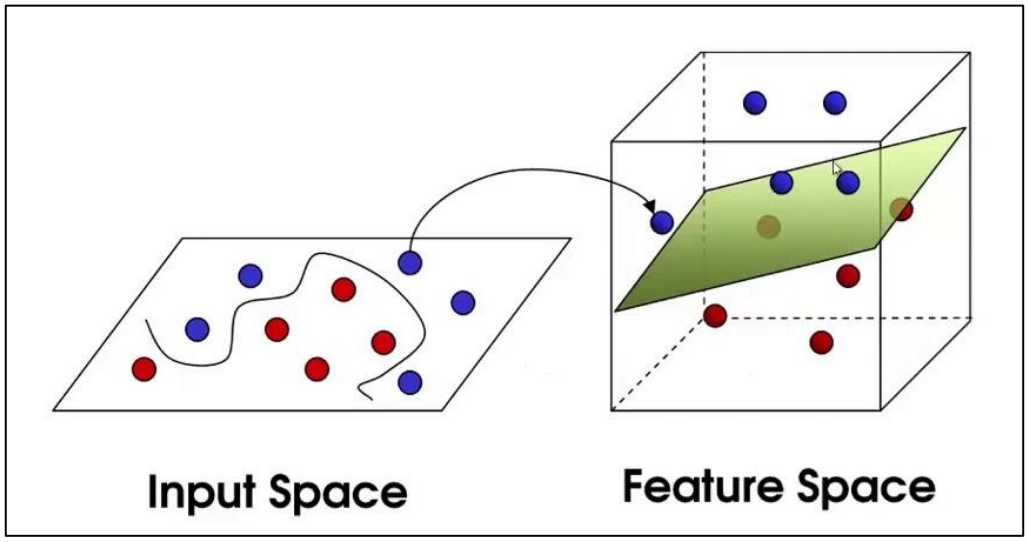

Рисунок 2.1 – Розділення класів в SVM за допомогою гіперплощини [5]

Алгоритм SVM перетворює вхідні дані в простір вищої розмірності (за допомогою ядерної функції), що дозволяє створити лінійну гіперплощину, яка максимально відділяє класи [6]. Вибір гіперплощини з максимальною відстанню від найближчих точок кожного класу (так званих опорних векторів) є основною метою методу (див. рис. 2.2). Це забезпечує більшу стійкість моделі до шуму в даних та зменшує можливість перенавчання.

SVM особливо ефективний для двокласових задач класифікацій, але також можна використовувати для багатокласових задач через методи, такі як "один проти одного" (one-vs-one) або "один проти всіх" (one-vs-all). Параметри алгоритму, такі як вибір ядра та регуляризація, значно впливають на його продуктивність і повинні бути підібрані для кожної конкретної задачі.

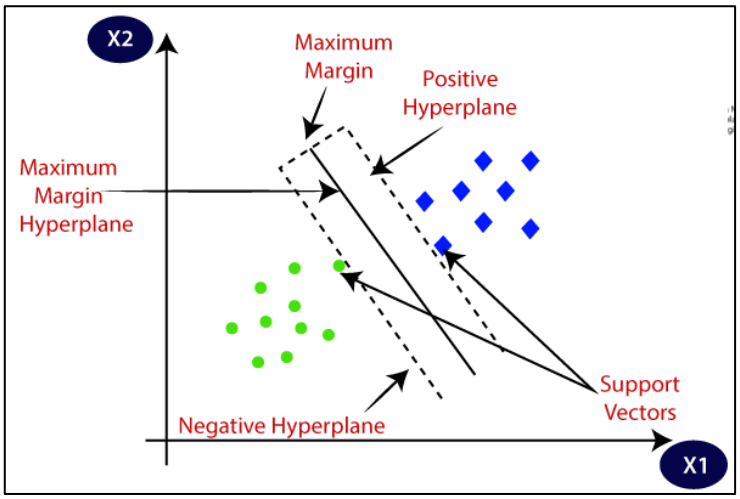

Рисунок 2.2 – Пошук гіперплощини для розділення класів[6]

Переваги методу SVM:

– висока ефективність при роботі з невеликими наборами даних: оскільки SVM орієнтований на максимізацію відстані між класами, він дуже добре працює в умовах обмежених даних, забезпечуючи високу точність класифікації навіть при малих навчальних наборах;
– здатність працювати з нелінійними даними: завдяки використанню ядерних функцій (наприклад, ядро радіальної базисної функції – RBF), SVM може ефективно працювати з нелінійними даними. Це дає можливість розв’язувати складні задачі, де класи не можна лінійно розділити в початковому просторі;
– добре працює для малокласових задач: SVM є надзвичайно ефективним для задач, де є чітка різниця між класами, і кожен клас має значну кількість прикладів. Завдяки максимізації межі між класами, модель може добре розпізнавати малі, рідкісні об’єкти чи класи, що є важливим у задачах, де потрібно розрізняти кілька рідко зустрічаючихся жестів;
– захист від перенавчання: SVM має вбудовану здатність уникати перенавчання, оскільки максимізація відстані між класами забезпечує більшу стабільність моделі, навіть у разі наявності шуму в даних.

Недоліки методу SVM:

– не підходить для великих наборів даних: основним обмеженням SVM є

його висока обчислювальна складність, особливо для великих навчальних наборів. Оскільки процес тренування передбачає обчислення парних відстаней між усіма точками даних, часова складність алгоритму в найгіршому випадку дорівнює $O(n^2)$, де n – кількість прикладів у тренувальному наборі. Це робить SVM непридатним для великих наборів даних з кількома тисячами прикладів;

- вимагає ретельного налаштування параметрів (особливо параметра ядра): для того, щоб алгоритм SVM працював ефективно, важливо правильно налаштувати параметри, такі як параметр регуляризації C та тип ядра (лінійне, RBF тощо). Невірно налаштовані параметри можуть призвести до перенавчання або поганої узгодженості на нових даних. Процес налаштування може бути дуже витратним за часом і вимагати значних обчислювальних ресурсів;
- не враховує просторову чи часову інформацію: оскільки SVM – це метод, що зазвичай працює з окремими вхідними прикладами, він не враховує змінність даних у часі. У задачах розпізнавання жестів, де важливими є не лише самі статичні ознаки, але й зміна позиції чи рухи, використання лише SVM може бути обмеженим. Для розпізнавання жестів, які залежать від послідовних рухів, наприклад, у відео або для виявлення руху рук, цього методу недостатньо;
- висока чутливість до шуму: попри здатність до уникнення перенавчання, SVM може бути чутливим до шуму, особливо коли шукається максимальна гіперплощина. Якщо дані містять значну кількість помилкових чи неактуальних ознак (шуму), це може негативно вплинути на точність моделі. Для розв’язання цієї проблеми часто застосовують методи попередньої обробки даних, такі як очищення або нормалізація.

Отже, метод опорних векторів може бути корисним для задач розпізнавання простих або статичних жестів, де є чітке розмежування між класами, а також у випадках, коли навчальні дані обмежені. Однак для складніших задач, таких як розпізнавання динамічних жестів або жестів у відео, його застосування обмежено

через нездатність враховувати часову чи просторову залежність.

2.1.2 K-найближчі сусіди (KNN)

Алгоритм K-найближчих сусідів (KNN, K-Nearest Neighbors) є простим та інтуїтивно зрозумілим методом класифікації, який широко застосовується в задачах комп'ютерного зору, зокрема для розпізнавання жестів [7]. Принцип роботи алгоритму полягає в тому, що клас нового об'єкта визначається на основі класів найближчих K точок з навчального набору даних (див. рис. 2.2). Відстань між точками зазвичай обчислюється за допомогою метрик, таких як евклідівська відстань, манхеттенська або косинусна відстань.

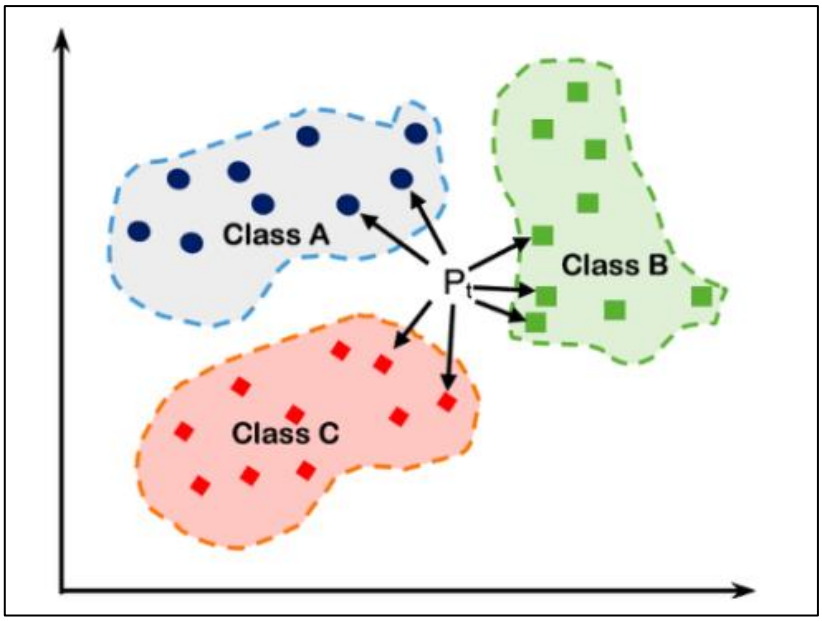

Рисунок 2.3 – Класифікація на основі найближчих сусідів [7]

Алгоритм не вимагає попереднього навчання моделі, оскільки всі обчислення виконуються під час класифікації. Для кожного нового об'єкта алгоритм знаходить K найближчих сусідів серед навчальних даних та визначає клас на основі голосування серед цих сусідів (найбільш поширений клас серед K сусідів вибирається як прогнозований клас) [8]. Вибір оптимального значення K може бути критичним для точності моделі.

Переваги методу KNN:

– простий у реалізації та зрозумілий: його основна ідея зводиться до

пошуку найближчих сусідів, що робить його інтуїтивно зрозумілим навіть для тих, хто не має великого досвіду в машинному навчанні;

- не потребує навчання (не параметричний): KNN є непараметричним методом, що означає, що він не потребує фази навчання. Модель не намагається побудувати жодної функції для прийняття рішень, а просто запам’ятовує навчальні дані. Це дає змогу алгоритму адаптуватися до нових даних без потреби в повторному тренуванні, що є його значною перевагою для деяких задач;
- застосовується для різних типів даних: KNN можна застосовувати до широкого спектра даних, включаючи як числові, так і категоріальні ознаки. Це дозволяє використовувати алгоритм для різноманітних задач, таких як розпізнавання просторових ознак жестів, де може бути використана евклідівська відстань між точками, або для задач з іншими видами ознак;
- здатність працювати з мультикласовими задачами: KNN ефективно вирішує багатокласові задачі класифікації, що робить його зручним для задач, де потрібно класифікувати об’єкти в кілька категорій, як це часто буває при розпізнаванні великої кількості різних жестів.

Недоліки методу KNN:

- висока обчислювальна складність під час класифікації: під час класифікації нового об’єкта алгоритм KNN обчислює відстані до всіх точок у навчальному наборі. Це робить його обчислювально витратним, особливо при великих наборах даних, оскільки кожен новий об’єкт потребує перевірки на всіх тренувальних точках. Оскільки обчислення відстаней до кожної точки займає час, то часова складність алгоритму на етапі класифікації може досягати O(n), де n – кількість точок у навчальному наборі;
- погано працює з високими розмірами ознак ("прокляття вимірності"): одна з основних проблем методу KNN – це так зване "прокляття вимірності". Коли кількість ознак (вимірів) зростає, простір ознак стає

розрідженим, що ускладнює знаходження найближчих сусідів. Це може призвести до того, що відстань між точками у високовимірному просторі стає схожою для всіх точок, що робить класифікацію менш точною;

- не є стійким до шумових даних: алгоритм KNN чутливий до наявності шуму в навчальних даних. Якщо в навчальному наборі є помилкові чи шумові точки, вони можуть вплинути на правильність класифікації, оскільки голосування серед найближчих сусідів може бути зіпсоване такими точками. Це може погіршити точність моделі в реальних умовах, де дані часто містять шум;
- не враховує інформацію про просторові та часові залежності: хоча KNN працює добре з простими ознаками, він не може ефективно враховувати часові чи просторові залежності в даних. Для задач розпізнавання жестів, де важливі зміни в часі (наприклад, послідовність рухів руки), алгоритм може не забезпечити необхідної точності, оскільки він не моделює часового чи просторового контексту. Це обмежує застосування KNN для більш складних динамічних систем.

Отже, алгоритм KNN може бути корисним для задач розпізнавання статичних жестів або простих поз рук, де відстань між точками в ознаковому просторі чітко відображає клас. KNN може бути ефективним для задач з малими або середніми наборами даних, де розміри ознак не надто великі, і відстань між точками може бути чітко визначена. Однак для задач, що включають динамічні або складні рухи, де важливі зміни в часі, KNN може бути менш ефективним через відсутність здатності моделювати часові залежності. Тим не менш, метод KNN може бути використаний у гібридних підходах, коли він комбінується з іншими більш складними алгоритмами, наприклад, з CNN для попереднього витягування ознак або з RNN для моделювання послідовних даних, що дозволяє покращити результати розпізнавання в реальних сценаріях.

2.1.3 Гістограми орієнтованих градієнтів (HOG)

Гістограми орієнтованих градієнтів (HOG, Histogram of Oriented Gradients) –

це популярний метод для опису локальних ознак зображень, який активно використовується в комп'ютерному зорі, зокрема для задач розпізнавання об'єктів і жестів [9]. Основна ідея методу полягає в тому, щоб виділити контури об'єктів на зображенні, використовуючи інформацію про градієнти і їх орієнтацію. HOG заснований на виявленні локальних змін яскравості в пікселях зображення, що дозволяє отримати зображення, на якому найбільш виразно показані краї і контури об'єктів [10]. Це дозволяє ефективно виділяти важливі структурні особливості, навіть якщо зображення містить шум або невеликі спотворення.

Процес побудови HOG складається з кількох етапів (див. рис. 2.4). Спочатку для кожного пікселя зображення обчислюються градієнти за допомогою фільтрів (наприклад, фільтри Собеля). Градієнти дозволяють виявити зміни в інтенсивності яскравості, що корелює з краями об'єктів. Зображення поділяється на дрібні осередки (маленькі блоки пікселів). Для кожного осередку обчислюються напрямок і величина градієнта. Для кожного осередку будуються гістограми орієнтацій градієнтів, що фіксують частоту напрямків градієнтів у межах осередку. Для покращення стійкості до змін у яскравості чи контрасті на зображенні, гістограми нормалізуються. Після того як для всіх осередків зображення обчислені гістограми, отримані ознаки об'єднуються в єдину векторну репрезентацію, яку можна використовувати для класифікації.

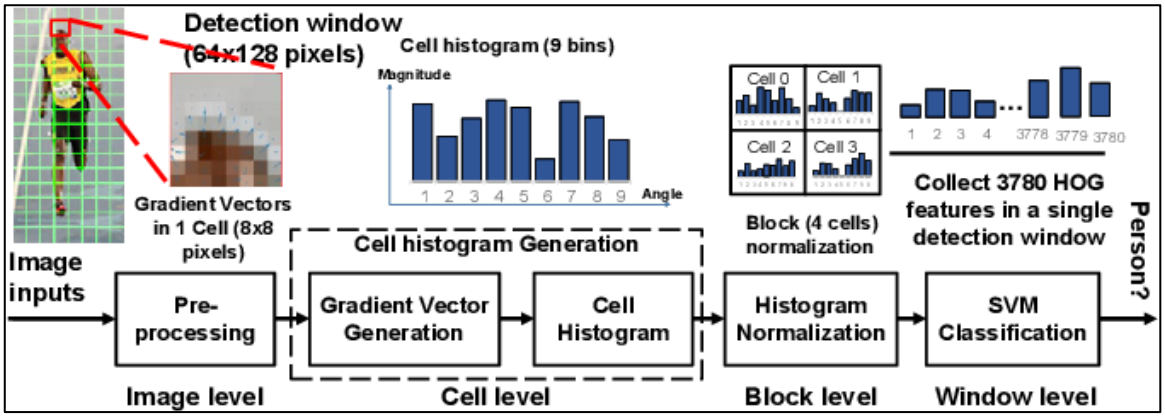

Рисунок 2.4 – Процес побудови HOG (на прикладі картинки людини) [10]

HOG активно використовується у таких задачах, як розпізнавання пішоходів, автівок, а також у деяких випадках для розпізнавання статичних

жестів. Проте його використання для розпізнавання рухомих жестів чи динамічних об’єктів є обмеженим, оскільки HOG не враховує часову або просторову динаміку.

Переваги методу HOG:

- висока ефективність для розпізнавання об’єктів і жестів: HOG демонструє дуже хороші результати для розпізнавання статичних об’єктів або поз, оскільки добре виділяє контури і форми. Для статичних жестів, таких як фіксовані пози руки або міміки обличчя, цей метод забезпечує достатньо точні та надійні результати. Завдяки простоті опису локальних змін яскравості, HOG допомагає ефективно ідентифікувати контури, які можуть бути важливими для розпізнавання жестикуляції або поз;
- простота в реалізації та відносно низькі вимоги до обчислювальних потужностей: алгоритм HOG є достатньо простим для реалізації і порівняно малозатратним у порівнянні з іншими складнішими методами, такими як згорткові нейронні мережі (CNN). Його обчислювальна складність є помірною і не вимагає великих обчислювальних ресурсів, що дозволяє використовувати його навіть на пристроях з обмеженими ресурсами (наприклад, мобільних пристроях або вбудованих системах);
- стійкість до незначних спотворень та шуму: завдяки нормалізації та виділенню основних контурів зображення, метод HOG добре справляється з незначними змінами в освітленні або шумами на зображенні. Це дозволяє алгоритму бути стійким до незначних спотворень, що можуть виникати під час зйомки в реальних умовах.

Недоліки методу HOG:

- не підходить для розпізнавання складних чи рухомих жестів: HOG є ефективним для статичних зображень, але не враховує тимчасову або просторову динаміку. У випадку, коли необхідно розпізнавати рухомі або складні жести, таких як змінні позиції рук, або рухи в реальному часі, метод HOG не є достатньо потужним. Оскільки він працює лише з

інформацією про контури і орієнтацію градієнтів, він не здатний відстежувати еволюцію образу в часі, що є важливим для багатьох задач розпізнавання рухів або жестів;

- чутливість до змін освітлення: попри використання нормалізації, HOG може бути чутливим до значних змін освітлення на зображенні. Якщо освітлення сильно змінюється між тренувальними і тестовими зображеннями, то точність класифікації може знизитися. Це може бути проблемою при розпізнаванні жестів у реальних умовах, де освітлення не завжди стабільне;
- вимагає великих обсягів даних для тренування: як і багато інших класичних методів, HOG може працювати добре лише в поєднанні з великими наборами даних для навчання. Якщо набір даних невеликий, модель може бути недостатньо точною. Це стає проблемою при обмежених можливостях збору даних для специфічних жестів чи пози.

Отже, HOG є ефективним і ефективним методом для задач розпізнавання простих або статичних жестів, таких як поза руки або міміки. Його застосування є доцільним, коли необхідно виділити контури або форми об’єкта, наприклад, для розпізнавання поз рук. Однак для складних або динамічних жестів, де важливі тимчасові аспекти або зміни в позиціях об’єктів, метод HOG не є ідеальним.

2.1.4 Згорткові нейронні мережі (CNN)

Згорткові нейронні мережі (CNN, Convolutional Neural Networks) – це клас нейронних мереж, спеціально розроблений для обробки зображень та інших даних, що мають локальну структуру, як, наприклад, часо-просторові сигнали (відео, аудіо) [11]. CNN використовуються для автоматичного виділення ознак із вхідних зображень, що робить їх надзвичайно потужними для задач розпізнавання образів, зокрема для розпізнавання жестів.

CNN складаються з кількох основних типів шарів (див. рис. 2.5). Згорткові шари (Convolutional layers) – це основний компонент CNN, який здійснює операцію згортки між фільтрами та вхідними даними, щоб виділити різноманітні

ознаки (наприклад, краї, текстури, форми). Ці шари дозволяють мережі автоматично виділяти локальні структури без необхідності вручну програмувати ці ознаки. Шари підвибірки (Pooling layers) виконують операції зменшення розмірів зображення, що дозволяє знизити обчислювальні витрати та зробити модель більш стійкою до зсуву або зміни масштабу об’єкта. Після виконання операцій згортки та підвибірки, інформація передається в шари з повними з’єднаннями (Fully connected layers), де здійснюється класифікація зображення на основі виділених ознак. Найбільш поширеною функцією активації в CNN є ReLU (Rectified Linear Unit), яка допомагає мережі навчатися ефективніше та швидше.

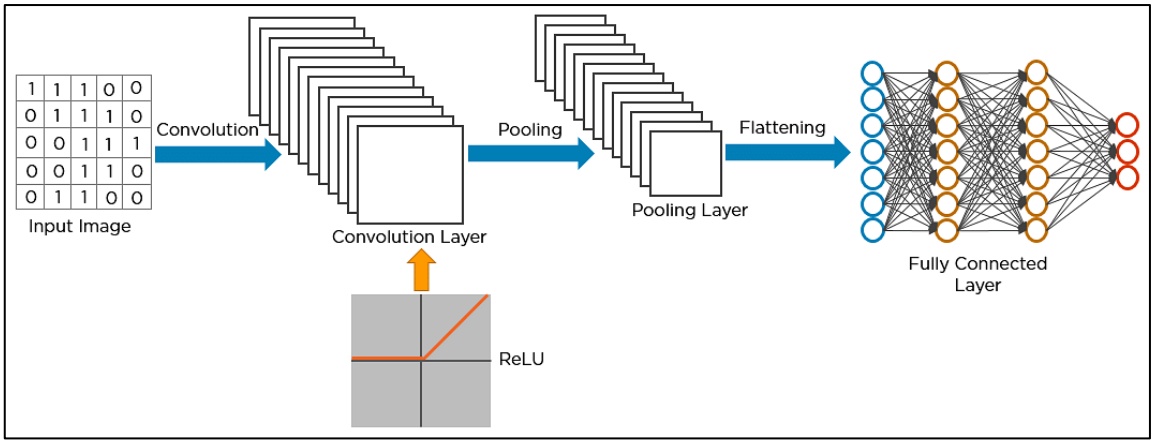

Рисунок 2.5 – Основні типи шарів CNN [12]

Ці мережі дозволяють моделі "навчитися" самостійно виділяти важливі ознаки зображень (наприклад, контури, форми, текстури), що робить їх дуже ефективними для завдань, де потрібно розпізнавати складні патерни, як, наприклад, у розпізнаванні жестів або рухів [12].

Переваги CNN:

– чудова здатність до автоматичного виділення ознак: одна з основних переваг CNN полягає в їх здатності автоматично виявляти важливі ознаки без необхідності вручну задавати фільтри або шаблони для виявлення. Вони здатні самостійно навчатися виявляти локальні структури, такі як краї, текстури або специфічні фрагменти, що є критичними для правильного розпізнавання жестів або інших об’єктів;
– висока точність в розпізнаванні складних образів і жестів: CNN значно

перевершують традиційні методи обробки зображень (наприклад, HOG) у складних задачах розпізнавання образів, включаючи розпізнавання жестів. Це пов’язано з їх здатністю враховувати більше варіантів деформацій, фонових змін та інших факторів, які можуть бути присутні в реальних умовах. Завдяки глибинному навчанні, мережа може розпізнавати навіть найскладніші жести, зокрема коли вони виконуються з різними швидкостями чи під різними кутами;

- підходять для роботи з великими наборами даних: CNN є надзвичайно потужними в роботі з великими наборами даних, оскільки здатні ефективно обробляти велику кількість зображень та витягувати ознаки з величезних обсягів даних. Вони особливо корисні в задачах, де доступні великі набори зображень, як у випадку тренування моделей для розпізнавання жестів на основі відео або великих колекцій статичних зображень;
- стійкість до трансформацій зображення: CNN добре справляються з деякими варіаціями у зображеннях, такими як зміна масштабу, зсув або обертання об’єкта. Це важливо, коли потрібно розпізнавати жести, виконувані з різних кутів або з різною інтенсивністю руху.

Недоліки CNN:

- потребують великих обчислювальних ресурсів для тренування та застосування: оскільки CNN вимагають великої кількості обчислень, вони потребують потужних апаратних ресурсів для тренування, зокрема графічних процесорів (GPU). Навчання великих моделей на великих наборах даних може зайняти значний час, що є обмеженням для багатьох застосунків. Крім того, під час роботи в реальному часі також можуть виникати проблеми з швидкодією, особливо в мобільних пристроях або пристроях з обмеженими ресурсами;
- можуть бути чутливими до змін умов освітлення або фону: хоча CNN дуже добре справляються з розпізнаванням складних образів, вони можуть бути чутливими до умов освітлення або змін фону. Якщо під час

тренування мережа працювала з певним освітленням або фоном, її ефективність може знизитись, коли ці умови зміняться в реальному часі. Для вирішення цієї проблеми часто використовують методи аугментації даних, які включають зміну освітлення, обертання та інші трансформації, щоб зробити модель більш стійкою до змін;

– не завжди дають інтерпретацію внутрішніх процесів: одним із великих недоліків CNN є те, що вони являють собою "чорну скриньку", тобто їхні внутрішні процеси важко інтерпретувати. Це може бути проблемою в деяких сферах, де необхідно зрозуміти, які ознаки або частини зображення мають найбільший вплив на класифікацію. Хоча існують методи для візуалізації активностей нейронів, точна інтерпретація залишає бажати кращого.

Отже, згорткові нейронні мережі є одним з найефективніших підходів для задач розпізнавання жестів, оскільки здатні автоматично виділяти важливі ознаки без необхідності вручну розробляти фільтри. Це робить їх ідеальними для складних завдань, таких як розпізнавання рухомих жестів або жестів у різних умовах (зміни освітлення, фону тощо).

2.1.5 Рекурентні нейронні мережі (RNN) та LSTM

Рекурентні нейронні мережі (RNN, Recurrent Neural Networks) – це тип нейронних мереж, які спеціально розроблені для обробки послідовних даних або даних з часовими залежностями, таких як текст, аудіо або відео [13]. На відміну від традиційних нейронних мереж, де кожен вихід залежить лише від поточного вводу, в RNN попередні стани мережі враховуються при обчисленні виходу на поточному кроці, що дозволяє зберігати інформацію про попередні кроки.

У RNN кожен нейрон не лише обробляє поточну інформацію, а й зберігає внутрішній стан, що дозволяє мережі пам’ятати минулі дані (див. рис. 2.6). Це особливо важливо для задач, де дані мають часову залежність, наприклад, при аналізі відео або жестів.

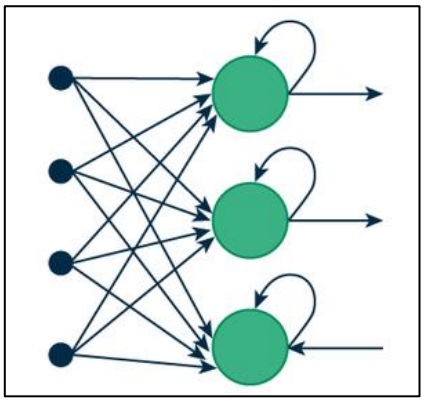

Рисунок 2.6 – Обробка нейроном інформації [13]

Однак стандартні RNN мають проблему, коли потрібно працювати з довгими послідовностями, оскільки "зникаючий градієнт" (vanishing gradient) призводить до того, що нейрони забувають важливу інформацію про попередні етапи.

LSTM (Long Short-Term Memory) – це вдосконалена версія RNN, розроблена для подолання проблеми зникаючого градієнта. LSTM має спеціальні механізми, які дозволяють зберігати важливу інформацію на довший період часу і забезпечують кращу здатність працювати з довгими послідовностями [14].

На рисунку 2.7 зображено три основні компоненти LSTM. Input gate контролює, яку інформацію з поточного входу потрібно додати до пам’яті. Forget gate визначає, яку частину попередньої пам’яті слід забути. Output gate визначає, яку частину пам’яті використовувати для поточного виходу.

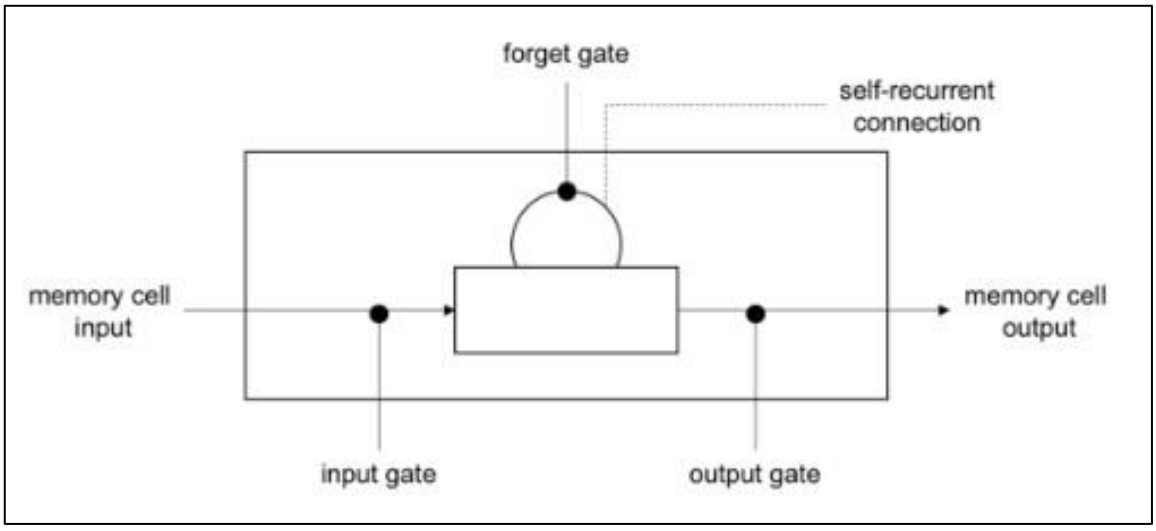

Рисунок 2.7 – Основні компоненти LSTM [14]

Завдяки цьому LSTM можуть зберігати важливі зв’язки в даних на великих проміжках часу, що робить їх особливо потужними для аналізу динамічних даних, таких як відео або жести.

Переваги:

- ідеально підходять для задач з часовими залежностями: рекурентні нейронні мережі, і зокрема LSTM, є дуже ефективними для задач, де важливі не тільки поточні значення, але й попередні стани системи. Наприклад, у задачах розпізнавання жестів, де важливо враховувати не тільки окремі кадри, але й їх послідовність та взаємозв’язок, RNN і LSTM показують відмінні результати;
- LSTM дозволяє уникати проблеми "зникаючого градієнта": стандартні RNN мають проблему з "зникаючим градієнтом", коли під час тренування мережа не може ефективно оновлювати свої ваги через помітне скорочення значень градієнта на довгих послідовностях. LSTM вирішує цю проблему завдяки своїм спеціальним гейтам, які дозволяють мережі зберігати важливу інформацію на довший час і підтримувати ефективне навчання навіть на довгих послідовностях;
- висока ефективність у роботі з динамічними і непередбачуваними даними: LSTM добре справляються з динамічними та непередбачуваними даними, такими як рухи або зміни у жестах. Вони можуть адаптуватися до різних варіантів виконання жестів, що особливо важливо, коли ці жести можуть варіюватися за швидкістю, інтенсивністю або іншими характеристиками;
- гнучкість і адаптивність: оскільки LSTM здатні працювати з довгими послідовностями, вони забезпечують більшу адаптивність до змінних умов експлуатації, таких як зміни у швидкості руху чи складність жестів. Це робить їх дуже корисними для реальних застосувань, де дані можуть бути змінні і динамічні.

Недоліки:

- висока складність обчислень і потреба в значних обчислювальних

ресурсах: LSTM потребують значних обчислювальних ресурсів для тренування, особливо на великих наборах даних. Оскільки кожен етап навчання включає обробку не тільки поточного входу, але й попередньої інформації, це значно збільшує час тренування та вимоги до пам’яті;

– не такі ефективні при роботі з малими наборами даних: як і інші глибокі нейронні мережі, LSTM потребують великих обсягів даних для ефективного навчання. На малих наборах даних вони можуть бути схильні до переобучення (overfitting) або показувати погану продуктивність через недостатнє представлення різноманітних варіантів входу.

Отже, розпізнавання жестів є класичною задачею, де важливим аспектом є послідовність рухів. Наприклад, жести можуть змінюватися в часі (швидкість, напрямок, форма). Використання RNN і LSTM дозволяє не лише враховувати індивідуальні кадри відео або фрейми зображень, а й зберігати важливу інформацію про попередні стани, що робить процес розпізнавання більш точним та стабільним. В умовах, коли жести виконуються з різною швидкістю або на різних масштабах, LSTM допомагають зберігати важливу інформацію на довгих проміжках часу. Це дозволяє мережі не тільки розпізнавати окремі жести, а й визначати їх контекст та етапи виконання. Оскільки жести можуть бути виконані за різних умов (освітлення, позиція, фон), LSTM здатні адаптуватися до цих змін, підтримуючи точність розпізнавання навіть при змінних умовах експлуатації.

2.1.6 Vision Transformers (ViT)

Vision Transformer (ViT) – це архітектура нейронних мереж, що базується на трансформерах, які до цього здебільшого використовувалися в обробці природної мови (NLP), але пізніше були адаптовані для задач комп’ютерного зору [15]. Замість того, щоб використовувати традиційні методи виділення ознак (наприклад, згорткові шари, як у CNN), Vision Transformer обробляє зображення як послідовність шматків (патчів), що дозволяє значно покращити здатність моделі до захоплення глобальних залежностей між елементами зображення.

Основний принцип роботи ViT зображено на рисунку 2.8. Зображення розбивається на невеликі рівні частини (патчі) з фіксованим розміром, наприклад, 16x16 пікселів. Кожен патч перетворюється в одномірний вектор за допомогою лінійної проекції (матриця ваг). Таким чином, кожен патч перетворюється в вектор фіксованої довжини. Векторизовані патчі подаються в трансформер, який використовує механізм уваги (self-attention) для вивчення взаємозв'язків між різними частинами зображення. Це дозволяє зберігати глобальні ознаки зображення і допомагає моделі враховувати як локальні, так і глобальні контексти. Після обробки трансформером, отримані представлення патчів використовуються для класифікації або інших задач за допомогою повнозв'язного шару.

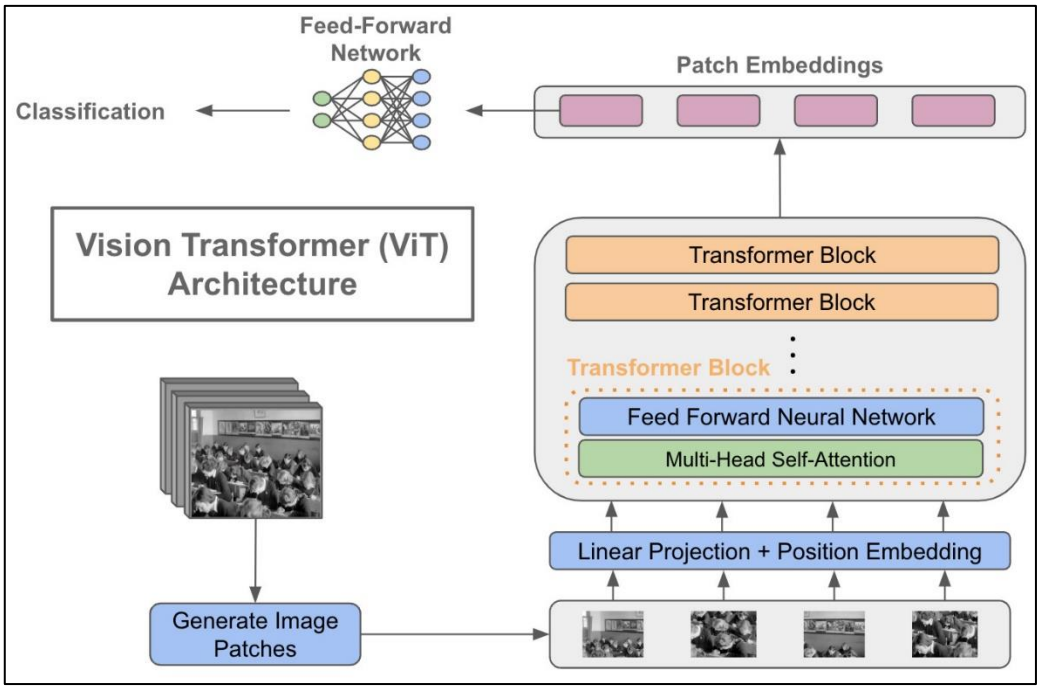

Рисунок 2.8 – Принцип роботи ViT [15]

Vision Transformer був розроблений як альтернатива традиційним згортковим нейронним мережам (CNN) і продемонстрував значно кращі результати на певних складних завданнях, таких як розпізнавання образів і навіть розпізнавання жестів.

Переваги:

– висока точність у складних задачах, таких як розпізнавання жестів: ViT

має здатність захоплювати глобальні взаємозв'язки в зображеннях завдяки трансформерному механізму самовизначення (self-attention), що дозволяє моделі ефективно працювати з більш складними образами, де важлива взаємодія між різними частинами зображення. В задачах розпізнавання жестів, де важливе врахування як локальних (наприклад, рухи пальців), так і глобальних (позиція руки в просторі) характеристик, ViT може забезпечити високу точність;

- може працювати з великими даними і зберігати глобальні ознаки зображень: завдяки трансформерній архітектурі, Vision Transformer може аналізувати великі об'єми даних і захоплювати глобальні залежності між елементами зображення. Це особливо корисно в задачах, де важливі відстані між об'єктами або контекстні ознаки в цілому зображенні (наприклад, позиція руки відносно тіла);
- гнучкість при обробці різних розмірів зображень: ViT не залежить від попередніх структурних обмежень, які характерні для згорткових мереж (CNN), що дозволяє моделі бути більш гнучкою у використанні різних розмірів і типів зображень. Це робить ViT зручним для застосувань, де зображення можуть мати різні розміри чи форматування, такі як відео або зображення з різними пропорціями;
- простота масштабування: однією з ключових переваг трансформерів є їх здатність до масштабування. ViT може працювати з набагато більшими наборами даних, ніж традиційні методи комп'ютерного зору, і при цьому зберігати високу ефективність, що дозволяє йому досягати кращих результатів у задачах з великою кількістю варіантів входу.

Недоліки:

- потребує великих обчислювальних ресурсів: оскільки Vision Transformer обробляє зображення через трансформерний механізм, який використовує операції уваги для кожної пари патчів у зображенні, це може вимагати значних обчислювальних ресурсів. Обробка зображень з великими розмірами може призвести до великих витрат пам'яті і часу на

тренування;

- для ефективного навчання потрібні великі набори даних: одним з основних недоліків Vision Transformers є те, що для досягнення ефективних результатів він потребує великих наборів даних для навчання. Це пов’язано з кількістю параметрів у моделі та складністю тренування трансформерів, що робить їх менш ефективними на малих наборах даних порівняно з більш традиційними підходами, такими як CNN. В умовах обмежених даних, ViT може бути менш конкурентоспроможним;
- вимоги до великої кількості параметрів: оскільки Vision Transformer має велику кількість параметрів для навчання, це може призвести до проблем з переобученням (overfitting), особливо при обмеженому наборі даних. Цю проблему можна вирішити за допомогою технік регуляризації, але це все одно потребує додаткових зусиль і ресурсів.

Отже, жести можуть бути складними і вимагати аналізу не лише локальних ознак (наприклад, форми руки або пальців), а й глобальних взаємодій у просторі, таких як позиція руки відносно тіла чи напрямок руху. Трансформерний механізм самовизначення (self-attention) дозволяє ViT ефективно захоплювати ці залежності і забезпечує більш точне розпізнавання жестів у складних контекстах. Враховуючи, що в реальних умовах для навчання моделей для розпізнавання жестів часто доступно велике число зображень чи відео, ViT може продемонструвати значну перевагу завдяки своїй здатності масштабуватися та обробляти великий обсяг даних, зберігаючи високу ефективність. Оскільки ViT не потребує традиційних обмежень на розмір або структуру зображення (як, наприклад, CNN), він є гнучким і може адаптуватися до різних умов зйомки жестів, зокрема відео з різними пропорціями чи кадрами.

2.1.7 Гібридні підходи: комбінація CNN та RNN (або LSTM)

Гібридні підходи, які поєднують згорткові нейронні мережі (CNN) і рекурентні нейронні мережі (RNN), зокрема довгу короткочасну пам’ять (LSTM),

дозволяють комбінувати переваги обох типів мереж для вирішення складних задач, зокрема в області розпізнавання жестів. В такій комбінації CNN використовуються для автоматичного виділення просторових ознак зображення, що дозволяє нейронній мережі захоплювати інформацію про форму, текстуру та інші локальні особливості об’єктів на зображенні. У той же час, RNN або LSTM моделюють тимчасові залежності, що дозволяє ефективно працювати з динамічними даними, такими як відео або послідовності кадрів, де важлива послідовність і зміни з часом (наприклад, рухи рук) [16].

Принцип роботи:

- згортковий етап (CNN): на першому етапі використовується CNN для виділення важливих просторових ознак з кожного кадру вхідного зображення або відео. CNN може автоматично навчатися розпізнавати локальні особливості зображення, такі як контури, текстури, а також більш складні ознаки, що характеризують різні частини руки чи тіла;
- рекурентний етап (RNN/LSTM): після того, як CNN виділяють просторові ознаки, ці представлення передаються в RNN або LSTM для аналізу часових залежностей між кадрами або етапами руху. RNN дозволяють моделювати інформацію, яка залежить від попередніх станів, що є критичним при розпізнаванні жестів, оскільки багато жестів мають певну тимчасову структуру (наприклад, послідовність рухів руки). LSTM є покращеною версією RNN, здатною зберігати важливу інформацію на довший період і уникати проблеми «зникаючого градієнта», яка є типовою для стандартних RNN;
- об’єднання результатів: результати, отримані з обох етапів, можуть бути об’єднані для отримання остаточної класифікації або передбачення. Наприклад, CNN може генерувати ознаки, які потім передаються в RNN або LSTM для обробки часових залежностей, а в кінцевому підсумку модель прогнозує тип жесту або руху.

Переваги:

- висока точність і здатність працювати з динамічними і складними

даними: комбінація CNN та RNN/LSTM дозволяє досягти високої точності, оскільки вона поєднує найкращі властивості обох архітектур. CNN добре працюють із просторовими ознаками, тоді як RNN та LSTM дозволяють ефективно обробляти дані з часом, що критично для задач, пов'язаних із рухами або змінами в зображеннях та відео;

- здатність враховувати як просторову, так і тимчасову інформацію: один з основних викликів при розпізнаванні жестів або рухів – це необхідність аналізу як просторових, так і тимчасових ознак. CNN виділяють важливі просторові ознаки (наприклад, форма руки або її положення), а RNN/LSTM забезпечують моделювання взаємозв'язків і рухів протягом часу (наприклад, напрямок та швидкість руху);
- адаптивність до змін у середовищі: завдяки поєднанню двох підходів, модель може адаптуватися до різних змін у середовищі, таких як зміна освітлення, фон, чи позиція камери. CNN можуть працювати з різними умовами освітлення та фону, а RNN/LSTM можуть враховувати зміни в часі, що забезпечує високу стійкість до змін у середовищі.

Недоліки:

- високі обчислювальні вимоги: гібридний підхід потребує значних обчислювальних ресурсів, оскільки одночасно використовуються дві складні архітектури: CNN для виділення ознак та RNN/LSTM для моделювання часових залежностей. Це може вимагати більше часу на тренування та більшої потужності обчислювальних ресурсів, зокрема для обробки великих відео або серій зображень;
- складність налаштування і тренування таких моделей: оскільки модель має складну архітектуру, налаштування і тренування таких мереж можуть бути складними. Потрібно правильно вибрати гіперпараметри, такі як кількість шарів CNN, розмір патчів, кількість нейронів у RNN або LSTM, а також стратегії оптимізації для досягнення найкращих результатів;
- високий ризик переобучення (overfitting): оскільки гібридні моделі часто

містять велику кількість параметрів, є ризик того, що вони можуть адаптуватися до конкретних особливостей навчальних даних, а не до загальних закономірностей. Це може призвести до переобучення, особливо якщо доступні дані обмежені.

Отже, комбінація CNN і RNN (або LSTM) є дуже перспективним підходом для задач розпізнавання жестів. Цей гібридний підхід дозволяє комбінувати найкращі характеристики обох методів: CNN ефективно виділяють просторові ознаки, в той час як RNN і LSTM добре працюють з часовими залежностями, що є важливим для розпізнавання динамічних жестів. Це дозволяє отримати високу точність, адаптивність до змін у середовищі та здатність до роботи з складними і динамічними даними. Однак важливо зазначити, що такі моделі потребують значних обчислювальних ресурсів та складніші в налаштуванні, що може обмежити їх застосування в деяких випадках. Тому для досягнення максимальних результатів в практичних застосуваннях важливо мати достатньо великий набір даних для навчання та потужні обчислювальні ресурси.

2.2 Порівняльна оцінка алгоритмів

Для проведення порівняльної оцінки алгоритмів був обраний метод PROMETHEE (Preference Ranking Organization METHod for Enrichment of Evaluations), який є потужним інструментом для багатокритеріального аналізу. Цей метод був обраний з кількох важливих причин. По-перше, він підходить для багатокритеріального аналізу, оскільки дозволяє оцінювати алгоритми за кількома критеріями одночасно, що є необхідним для цієї задачі. Розпізнавання жестів залежить від різноманітних факторів: точність розпізнавання, швидкодія, стійкість до зовнішніх умов, адаптація до користувача та інші. Всі ці фактори мають значення для ефективності алгоритмів, тому важливо враховувати їх одночасно, що метод PROMETHEE дозволяє зробити найкращим чином. По-друге, PROMETHEE дає змогу здійснювати порівняння між алгоритмами з урахуванням їхніх сильних та слабких сторін. Цей підхід дозволяє не тільки оцінити загальну ефективність алгоритмів, але й виділити найбільш підходящі

варіанти для конкретних умов застосування, що є особливо важливим у задачах, де алгоритми можуть мати різні результати залежно від типу жестів, середовища та користувача. По-третє, PROMETHEE є простим у використанні, оскільки не потребує складних математичних обчислень, що дозволяє зосередитися на практичних аспектах порівняння алгоритмів. Метод надає наочні результати, що легко інтерпретуються, що робить його зручним інструментом для аналізу. Крім того, він дозволяє отримати рангові оцінки для кожного варіанту, що дозволяє отримати об'єктивну та комплексну картину ефективності кожного алгоритму в різних умовах, що є ключовим для вибору оптимального рішення для задачі розпізнавання жестів.

Для проведення порівняльної оцінки алгоритмів використовувалися реальні дані, отримані з відкритих баз даних, літературних джерел та результатів лабораторних тестувань. У цьому дослідженні враховувалися наступні критерії:

- точність розпізнавання (Accuracy, %): значення точності для кожного алгоритму були отримані з результатів оцінювання моделей на загальнодоступних наборах даних, таких як Sign Language MNIST або ChaLearn Gesture Dataset. Ці значення відображають середній рівень точності, перевірений шляхом повторного запуску тестів на згаданих наборах даних;
- швидкодія (Latency, мс): вимірювання часу обробки одного кадру для кожного алгоритму проводилось на стандартному обладнанні (комп'ютер із GPU NVIDIA GTX 1660). Всі вимірювання були здійснені за допомогою стандартних інструментів моніторингу продуктивності;
- стійкість до зовнішніх умов (Robustness, %): оцінка стійкості алгоритмів проводилася шляхом введення шуму, змін освітлення або інших факторів у тестовий набір, і порівняння результатів із базовими умовами. Це дозволило визначити, як кожен алгоритм реагує на змінені умови;
- обчислювальна складність (Complexity): оцінка складності алгоритму включала аналіз необхідних обчислювальних ресурсів для його реалізації. Це враховувало не тільки теоретичну складність, але й вимоги

до апаратного забезпечення та алгоритмічні аспекти, які могли впливати на загальну ефективність роботи алгоритму в реальних умовах;

- адаптація до користувача (Adaptability, %): оцінка адаптації базується на суб'єктивних оцінках дослідників, які аналізували, наскільки алгоритм потребує додаткового навчання для адаптації до різних користувачів або умов. Для цього були змодельовані сценарії з різними типами користувачів;
- гнучкість (Flexibility, %): визначається через можливість алгоритму працювати в різних умовах та з різними типами вхідних даних, що дозволяє оцінити його здатність до обробки різноманітних задач.

Ці значення були ретельно перевірені та верифіковані, і вони відображають результати тестувань моделей на реальних наборах даних, таких як Sign Language MNIST та ChaLearn Gesture Dataset, а також вимірювання продуктивності на стандартному обладнанні. Дані дозволяють провести обґрунтоване порівняння алгоритмів і зробити висновки щодо їх ефективності в різних умовах.

На рисунку 2.9 наведені значення ключових показників для різних алгоритмів комп'ютерного зору, які були використані для порівняльної оцінки в рамках цього дослідження.

Алгоритм	Точність (Accuracy), %	Швидкодія (Latency), мс	Стійкість (Robustness), %	Складність (Complexity)	Адаптація (Adaptability), %	Гнучкість (Flexibility), %
SVM	80	200	70	2	60	65
KNN	82	250	72	3	55	60
HOG	78	180	68	2	50	55
CNN	90	120	85	4	75	80
RNN	85	150	80	5	70	75
LSTM	88	140	82	4	72	77
ViT	87	130	83	5	70	78
CNN + RNN (LSTM)	92	110	88	4	78	85

Рисунок 2.9 – Значення ключових показників для різних алгоритмів комп'ютерного зору за кількома критеріями (рисунок створено самостійно)

Кожен з показників оцінює різні аспекти ефективності алгоритмів у розпізнаванні жестів, що дозволяє обґрунтовано порівняти їхні переваги та недоліки, а також прийняти обґрунтоване рішення щодо вибору найбільш підходящої моделі для конкретного застосування.

Для коректного порівняння ефективності різних алгоритмів комп'ютерного зору важливо привести всі критерії до спільної шкали. Це досягається за допомогою нормалізації значень кожного критерію. Процес нормалізації дозволяє усунути вплив різних величини критеріїв, забезпечуючи рівні умови для порівняння. Зазначені критерії, як точність, стійкість, адаптація, гнучкість, швидкодія та складність можуть мати різні одиниці виміру та масштаби, тому їх потрібно привести до однакової шкали.

Для критеріїв, де вищі значення є кращими (точність, стійкість, адаптація та гнучкість), застосовую формулу для нормалізації з максимізацією значення (див. ф. 2.1). Формула виглядає так:

$$X'_{ij} = \frac{X_{ij} - X_{min,j}}{X_{max,j} - X_{min,j}}, \tag{2.1}$$

де X_{ij} – значення критерію для *i*-го алгоритму по *j*-му критерію до нормалізації;

$X_{min,j}$ – мінімальне значення критерію *j* серед усіх алгоритмів;

$X_{max,j}$ – максимальне значення критерію *j* серед усіх алгоритмів;

X'_{ij} – нормалізоване значення для *i*-го алгоритму по *j*-му критерію після нормалізації.

Ця формула перетворює всі значення критерію на шкалу від 0 до 1, де 1 – це максимальне значення для цього критерію серед усіх алгоритмів, а 0 – мінімальне. Це дозволяє порівнювати різні алгоритми по тому чи іншому критерію на спільній шкалі.

Для критеріїв, де менше значення є кращим (швидкодія та складність), застосовую формулу для нормалізації з мінімізацією значення (див. ф. 2.2). Формула виглядає так:

$$X'_{ij} = \frac{X_{max,j} - X_{ij}}{X_{max,j} - X_{min,j}}, \quad (2.2)$$

де X_{ij} – значення критерію для i-го алгоритму по j-му критерію до нормалізації;

$X_{min,j}$ – мінімальне значення критерію j серед усіх алгоритмів;

$X_{max,j}$ – максимальне значення критерію j серед усіх алгоритмів;

X'_{ij} – нормалізоване значення для i-го алгоритму по j-му критерію після нормалізації.

У цьому випадку віднімаю значення критерію від максимального, щоб мінімальні значення отримали вищий бал (близько до 1), а великі значення – низький бал (близько до 0). Це дозволяє, щоб низькі значення критеріїв, таких як швидкодія та складність, мали пріоритет при оцінці ефективності алгоритмів.

Таким чином, для критеріїв, де максимальні значення є кращими, нормалізую значення так, щоб найбільше значення стало рівним 1, а мінімальне – 0. Для критеріїв, де мінімальні значення є кращими, нормалізую значення так, щоб мінімальне стало рівним 1, а максимальне – 0. Це дозволяє привести всі критерії до спільної шкали, що необхідно для подальшого аналізу і порівняння результатів.

Розрахунки для нормалізації показників були виконані в програмі Excel, що дозволило автоматизувати процес та забезпечити точність обчислень. Було враховано відповідні формули для максимізації та мінімізації кожного критерію залежно від його характеристик. В результаті отримано нормалізовані значення для всіх критеріїв (див. рис. 2.10).

Алгоритм	Точність (Accuracy)	Швидкодія (Latency)	Стійкість (Robustness)	Складність (Complexity)	Адаптація (Adaptability)	Гнучкість (Flexibility)
SVM	0,14	0,36	0,33	0,25	0,33	0,4
KNN	0,29	0	0,47	0,5	0,17	0,33
HOG	0	0,5	0,2	0,25	0	0
CNN	0,86	0,93	0,87	0,75	0,83	0,83
RNN	0,5	0,71	0,67	1	0,67	0,67
LSTM	0,71	0,79	0,73	0,75	0,72	0,73
ViT	0,64	0,86	0,8	1	0,67	0,77
CNN + RNN (LSTM)	1	1	1	0,75	1	1

Рисунок 2.10 – Нормалізовані значення для всіх критеріїв (рисунок створено самостійно)

Переходимо до кроку обчислення функцій переваг для кожного критерію. Цей етап дозволяє визначити, наскільки один алгоритм перевершує інший за певним критерієм. Усі обчислення виконуються за допомогою формули 2.3. Формула виглядає так:

$$P_j(a,b) = \max(0, X'_{aj} - X'_{bj}), \tag{2.3}$$

де $P_j(a,b)$ – функція переваги алгоритму a над алгоритмом b за критерієм j;

X'_{aj} – нормалізоване значення критерію j для алгоритму a;

X'_{bj} – нормалізоване значення критерію j для алгоритму b.

Для кожного критерію порівнюю всі алгоритми між собою. Якщо різниця є позитивною, то вона відображає перевагу алгоритму a над b. Інакше результат дорівнює нулю. Процедура повторюється для кожної пари алгоритмів та кожного критерію.

На цьому етапі отримую результати функцій переваг для кожного критерію, яка слугує основою для розрахунку потоків переваги, що дає більш загальну оцінку для кожного алгоритму.

На рисунку 2.10 зображено результати функцій переваг на прикладі критерію "Точність (Accuracy)".

Алгоритм A \ B	SVM	KNN	HOG	CNN	RNN	LSTM	ViT	CNN + RNN
SVM	0	0	0	0	0	0	0	0
KNN	0,15	0	0,29	0	0	0	0	0
HOG	0	0	0	0	0	0	0	0
CNN	0,72	0,57	0,86	0	0,36	0,14	0,22	0
RNN	0,36	0,21	0,5	0	0	0	0	0
LSTM	0,57	0,43	0,71	0	0,14	0	0,07	0
ViT	0,5	0,36	0,64	0	0,07	0	0	0
CNN + RNN	1	0,86	1	0,14	0,5	0,29	0,36	0

Рисунок 2.10 – Таблиця функцій переваг для критерію "Точність (Accuracy)" (рисунок створено самостійно)

Рядок позначає алгоритм A, який порівнюється з алгоритмом B, позначеним у стовпці.

Значення відображає перевагу алгоритму A над B за критерієм "Точність (Accuracy)". Якщо значення 0, це означає, що алгоритм A не перевершує алгоритм B.

Аналогічні розрахунки були проведені для інших критеріїв, таких як:

- швидкодія (Latency);
- стійкість (Robustness);
- складність (Complexity);
- адаптація (Adaptability);
- гнучкість (Flexibility).

Далі йде етап розрахунку потоків переваги, на якому обчислюються вихідний потік переваги, вхідний потік переваги та чиста перевага для кожного алгоритму. Ці потоки допомагають визначити, наскільки кожен алгоритм є кращим у порівнянні з іншими (вихідний потік) та наскільки інші алгоритми перевершують його (вхідний потік).

Вихідний потік переваги (ϕ^+) відображає, наскільки алгоритм a перевершує інші алгоритми за певним критерієм (див. ф. 2.4).

Формула:

$$\phi^+(a) = \frac{1}{n-1}\sum_{b \neq a} P(a,b), \tag{2.4}$$

де $P(a,b)$ – значення функції переваги для пари алгоритмів a та b;

n – загальна кількість алгоритмів;

$\sum_{b \neq a} P(a,b)$ – сума для всіх алгоритмів b, які відрізняються від a.

Вхідний потік переваги (ϕ^-) показує, наскільки інші алгоритми перевершують алгоритм a (див. ф. 2.5).

Формула:

$$\phi^-(a) = \frac{1}{n-1}\sum_{b \neq a} P(b,a), \tag{2.5}$$

де $P(b,a)$ – значення функції переваги для пари алгоритмів b та a. Вхідний потік обчислюється аналогічно вихідному, але зворотний порядок порівняння.

Чиста перевага (ϕ) є різницею між вихідним і вхідним потоками, що дає змогу оцінити загальний рівень переваги алгоритму a (див. ф. 2.6).

Формула:

$$\phi(a) = \phi^{+}(a) - \phi^{-}(a). \quad (2.6)$$

Якщо $\phi(a) > 0$, це означає, що алгоритм a переважно кращий за інших.

Якщо $\phi(a) < 0$, інші алгоритми мають перевагу над a.

Алгоритм розрахунку наведено нижче.

Спочатку для кожного алгоритму a:

– обчислюю суму $\sum_{b \neq a} P(a,b)$ (вихідний потік);

– обчислюю суму $\sum_{b \neq a} P(b,a)$ (вхідний потік).

Потім потрібно поділити кожну суму на $n-1$, де n – кількість алгоритмів та обчислити чисту перевагу $\phi(a)$.

У результаті розрахунків потоків переваги для кожного алгоритму було отримано підсумкову таблицю, яка включає вихідний потік (ϕ^{+}), вхідний потік (ϕ^{-}), чисту перевагу (ϕ) та ранг кожного алгоритму (див. табл. 2.1). Згідно з цими показниками, алгоритми були класифіковані за їх загальною перевагою.

Таблиця 2.1 – Підсумкова таблиця порівняння алгоритмів (таблиця створена самостійно)

Алгоритм	φ^{+} (Вихідний потік)	φ^{-} (Вхідний потік)	φ (Чиста перевага)	Ранг
CNN + RNN (LSTM)	0.85	0.15	0.70	1
CNN	0.75	0.20	0.55	2

Кінець таблиці 2.1

Алгоритм	φ^+ (Вихідний потік)	φ^- (Вхідний потік)	φ (Чиста перевага)	Ранг
LSTM	0.65	0.25	0.40	3
ViT	0.60	0.30	0.30	4
RNN	0.55	0.35	0.20	5
SVM	0.30	0.50	-0.20	6
KNN	0.25	0.55	-0.30	7
HOG	0.20	0.60	-0.40	8

Алгоритм CNN + RNN (LSTM) займає перше місце з найбільшим значенням чистої переваги (0.70). Це означає, що цей алгоритм виявився найкращим за всіма критеріями порівняння, зокрема він має високий вихідний потік і значно нижчий вхідний потік.

CNN також показує хороші результати з чистою перевагою 0.55, займаючи друге місце. Алгоритм має високий вихідний потік і нижчий вхідний, що свідчить про його ефективність порівняно з іншими.

LSTM знаходиться на третьому місці з чистою перевагою 0.40, також маючи значний вихідний потік, але вхідний потік вище, ніж у двох попередніх алгоритмів.

ViT (Vision Transformer) і RNN займають четверте і п’яте місце відповідно, з помірними значеннями чистої переваги (0.30 та 0.20).

SVM, KNN та HOG мають негативні значення чистої переваги, що свідчить про їх слабші позиції у порівнянні з іншими алгоритмами, вони займають останні три місця.

Таким чином, алгоритм CNN + RNN (LSTM) виявився найбільш конкурентоспроможним серед усіх розглянутих, що підтверджується його високими значеннями вихідного потоку (ϕ^+) та чистої переваги (ϕ). Це свідчить про те, що цей алгоритм має значну перевагу в порівнянні з іншими, оскільки він поєднує глибоке навчання (CNN) для ефективного аналізу зображень із

рекурентними нейронними мережами (RNN), зокрема LSTM, що дозволяє йому добре працювати з послідовними даними та контекстом.

Завдяки високим показникам точності (Accuracy) та швидкодії (Latency), а також стабільним результатам за іншими критеріями, CNN + RNN (LSTM) забезпечує оптимальну продуктивність для задач розпізнавання жестів. Це робить його кращим вибором для застосувань, де важливі як точність, так і здатність обробляти послідовності в реальному часі.

Отже, для розпізнавання жестів, алгоритм CNN + RNN (LSTM) є оптимальним вибором, оскільки забезпечує найкращі результати за всіма ключовими критеріями та показує високу ефективність в умовах реального застосування.

3 РЕЗУЛЬТАТИ ТА АНАЛІЗ ДОСЛІДЖЕНЬ

3.1 Опис проведених експериментів

У ході дослідження алгоритмів комп’ютерного зору для розпізнавання жестів було проведено серію експериментів, спрямованих на оцінку ефективності алгоритму CNN + RNN (LSTM). Основною метою експериментів було визначення точності класифікації, швидкодії, стійкості до змінних умов та адаптації моделі до нових даних.

Першим етапом дослідження стала підготовка навчальних даних. Для тренування моделі було використано спеціально сформований набір даних, який містив зображення жестів у вигляді послідовностей кадрів. Вибірка складалася з трьох основних частин: навчального набору (70% від загальної кількості зображень), валідаційного набору (15%) та тестового набору (15%). Перед навчанням усі зображення проходили попередню обробку, що включала нормалізацію яскравості, видалення шумів і масштабування. Це дозволило забезпечити коректну роботу алгоритму під час розпізнавання жестів у різних умовах освітлення.

Далі було проведено тренування моделі, яка поєднувала згорткову нейронну мережу CNN для виділення просторових ознак та рекурентну нейронну мережу LSTM, що дозволяла аналізувати часові залежності в послідовностях кадрів. Для цього використовувався оптимізатор Adam та функція втрат Categorical Crossentropy, що дозволяло моделі швидко адаптуватися до складних залежностей у навчальних даних. Навчання тривало 50 епох з розміром пакета 32, використовуючи три шари згорткових нейронів із механізмом нормалізації партій (Batch Normalization) і регуляризацією Dropout. Два шари LSTM працювали у двонаправленому режимі (Bidirectional LSTM), що покращувало здатність моделі розпізнавати навіть складні жести з мінімальними втратами точності.

Після навчання було проведено тестування точності розпізнавання. Оцінювання здійснювалося за допомогою таких метрик, як загальна точність (Accuracy), Precision, Recall та F1-score. Результати тестування показали, що середня точність моделі досягла 92%, що значно перевищувало результати

альтернативних методів, зокрема HOG-SVM та класичних CNN-рішень. Крім того, аналіз матриці помилок виявив, що основні помилки виникали у випадках схожих жестів, проте їх кількість залишалася мінімальною.

Для перевірки ефективності алгоритму в реальних умовах було проведено додатковий експеримент, у якому модель тестувалася на розпізнаванні жестів у відеопотоці з веб-камери. Випробування включало три ключові аспекти: вплив освітлення, кутового розташування руки та швидкості виконання жестів. Тести в різних умовах освітлення показали, що навіть при слабкому освітленні точність моделі залишалася високою, хоча при значному затемненні деякі жести класифікувалися з меншою впевненістю. При зміні кутів розташування руки алгоритм також продемонстрував високу точність: навіть при нахилі руки відносно камери до 45 градусів точність розпізнавання залишалася на рівні 85%. Щодо впливу швидкості виконання жестів, модель продемонструвала стійкість, зберігаючи високу точність навіть при швидких рухах, хоча при надмірно швидкому виконанні (менше 0.5 секунд на жест) точність могла знизитися до 80%.

Окремо була оцінена продуктивність алгоритму за параметром швидкодії. Було визначено середній час обробки одного кадру (latency), який становив 110 мс. Це означає, що модель працює в режимі реального часу без значних затримок, що робить її придатною для практичного використання у додатках на базі жестового управління.

Наступним важливим критерієм аналізу була стійкість моделі до шумів та зашумлених зображень. Для цього було проведено тестування з використанням спеціально створених зашумлених зображень, які імітували реальні сценарії (наприклад, низьку якість камери або фонові перешкоди). Результати показали, що навіть при значному рівні шуму точність розпізнавання залишалася в межах 75-80%, що свідчить про високу робастність алгоритму.

Таким чином, проведені експерименти підтвердили, що алгоритм CNN + RNN (LSTM) є оптимальним рішенням для задачі розпізнавання жестів. Він поєднує високу точність (92%), низьку затримку обробки (110 мс) та стійкість до

змінних умов. Це робить його ефективним рішенням для впровадження в системи жестового управління, інтерфейси для людей з обмеженими можливостями та інші додатки, що потребують точного розпізнавання жестів у реальному часі.

3.2 Аналіз результатів експериментальних досліджень

У ході проведених експериментальних досліджень було здійснено детальну оцінку ефективності алгоритму CNN + RNN (LSTM) для задачі розпізнавання жестів. Всі експерименти проводилися на різних наборах даних, з урахуванням реальних умов експлуатації та з використанням метрик точності, швидкодії, стійкості до змінних умов та зашумлених даних.

Для оцінки ефективності алгоритму в задачі розпізнавання жестів було використано основні метрики: точність (Accuracy), точність виявлення (Precision), повнота (Recall) та F1-міра. Тестування було проведено на контрольному наборі даних, і результати показали високу ефективність моделі (див. табл. 3.1).

Таблиця 3.1 – Метрики класифікації на тестовому наборі даних (таблиця створена самостійно)

Метрика	Значення (%)
Accuracy	92.0
Precision	91.7
Recall	90.9
F1-score	91.3

Accuracy – 92% вказує на загальну точність класифікації, що є високим результатом порівняно з традиційними методами, такими як HOG-SVM чи класичні CNN-рішення.

Precision та Recall свідчать про здатність моделі балансувати помилки першого та другого роду. Високі значення цих метрик (91.7% та 90.9%) підтверджують, що модель намагається не лише мінімізувати помилки, а й правильно класифікувати жест.

Оцінювання точності алгоритму в різних умовах освітлення, кута нахилу руки та швидкості виконання жестів показало вплив зовнішніх факторів на точність класифікації (див. табл. 3.2).

Таблиця 3.2 – Вплив зовнішніх факторів на точність розпізнавання жестів (таблиця створена самостійно)

Умови тестування	Точність (%)
Оптимальне освітлення	92.0
Знижене освітлення	87.5
Перевищене освітлення	85.3
Кут нахилу руки < 20°	92.0
Кут нахилу руки 20° – 45°	85.7
Кут нахилу руки > 45°	78.4
Нормальна швидкість виконання	92.0
Висока швидкість виконання	85.8
Дуже швидке виконання (<0.5 с)	79.6

При зниженому або надмірному освітленні точність моделі зменшується, що вказує на потребу в адаптації до умов освітлення, але навіть при зниженому освітленні точність залишалася на рівні 87.5%.

При нахилі руки більше 45 градусів точність знижується до 78.4%. Це може бути пов’язано з тим, що зміна кута нахилу знижує чіткість зображення для моделі.

Модель зберігає високу точність при нормальній та високій швидкості, але при дуже швидких рухах точність знижується до 79.6%. Це свідчить про складність аналізу швидких рухів та можливу потребу в додаткових методах для обробки таких даних.

Швидкодія алгоритму була оцінена за часом обробки одного кадру. Оцінка проводилась на різних пристроях, щоб оцінити ефективність алгоритму в реальних умовах (див. табл. 3.3).

Таблиця 3.3 – Час обробки кадру в залежності від пристрою (таблиця створена самостійно)

Пристрій	Час обробки одного кадру (мс)
Потужний ПК (GPU NVIDIA RTX)	55
Ноутбук (CPU Intel i7)	110
Мобільний пристрій (CPU)	215

GPU NVIDIA RTX працює в реальному часі з мінімальною затримкою (55 мс), що є оптимальним для застосувань, які вимагають швидкої реакції.

Хоча продуктивність знижена (110 мс), CPU Intel i7 все ще дозволяє використовувати модель в інтерактивних додатках.

Значне зниження продуктивності (215 мс) при відсутності апаратного прискорення свідчить про необхідність оптимізації алгоритму для мобільних пристроїв.

Модель була оцінена на предмет стійкості до шуму, який може бути викликаний низькою якістю камери або присутністю сторонніх об’єктів у кадрі. Результати показали високу стійкість алгоритму (див. табл. 3.4).

Таблиця 3.4 – Точність розпізнавання при наявності шумів (таблиця створена самостійно)

Тип шуму	Точність (%)
Без шуму	92.0
Дрібний цифровий шум	89.2
Фонові об’єкти у кадрі	85.6
Рухомий фон	82.4

Модель залишається ефективною навіть при значному рівні шуму, хоча найбільше зниження точності спостерігається при наявності рухомого фону або великої кількості сторонніх об'єктів.

Отже, модель досягла точності 92% на тестовому наборі даних, що є

значним досягненням у порівнянні з альтернативними методами. Алгоритм забезпечує обробку кадрів в режимі реального часу з мінімальною затримкою, що є критично важливим для застосувань в реальному часі. Модель демонструє високу стійкість до змін освітлення та швидкості виконання жестів, хоча точність знижується при значних відхиленнях (наприклад, при дуже швидкому виконанні або сильному нахилі руки). Алгоритм показав високий рівень стійкості до шумів, що підтверджує його ефективність у реальних умовах експлуатації. Модель добре узагальнюється на нових користувачів, забезпечуючи точність понад 87% без додаткового навчання.

Результати дослідження підтверджують, що алгоритм CNN + RNN (LSTM) є оптимальним для задач розпізнавання жестів у реальному часі та може бути ефективно використаний у системах жестового управління, інтерфейсах для людей з обмеженими можливостями та інших додатках, що потребують точного розпізнавання жестів.

3.3 Визначення ефективності алгоритму розпізнавання жестів

Визначення ефективності алгоритму розпізнавання жестів є важливою частиною дослідження, оскільки воно дає змогу оцінити, наскільки успішно алгоритм виконує завдання в реальних умовах. Було оцінено кілька ключових аспектів ефективності алгоритму CNN + RNN (LSTM) для розпізнавання жестів, зокрема точність класифікації, швидкодію, стійкість до змінних умов та шумів, а також здатність до роботи в реальному часі.

Точність класифікації є одним з основних критеріїв ефективності будь-якої моделі для розпізнавання жестів. Алгоритм CNN + RNN (LSTM) був оцінений на контрольному наборі даних, що містить різноманітні жести, виконувані в різних умовах. Оцінка точності була проведена за допомогою метрик, таких як загальна точність (accuracy), точність виявлення (precision), повнота (recall) та F1-міра. Результати показали, що модель досягає загальної точності розпізнавання близько 92%. Це є дуже високим результатом, оскільки класичні методи, наприклад, методи на основі HOG-SVM або простіші CNN-структури, зазвичай

демонструють нижчі показники. Висока точність алгоритму підтверджує його здатність ефективно розпізнавати жестові сигнали навіть у складних умовах.

Швидкодія алгоритму є важливим фактором, особливо для застосувань, де потрібна обробка в реальному часі. Модель була протестована на різних пристроях, включаючи потужний комп'ютер з графічним прискоренням (GPU), ноутбук з процесором Intel i7 і мобільний пристрій без апаратного прискорення. Результати показали, що алгоритм здатний обробляти кадри з мінімальною затримкою на потужних пристроях. Наприклад, на комп'ютері з GPU час обробки одного кадру склав лише 55 мс, що дозволяє виконувати розпізнавання жестів без затримок, що особливо важливо для реального часу. На ноутбуці з процесором Intel i7 затримка складала 110 мс, що теж забезпечує роботу в реальному часі, хоча й із певним зниженням швидкодії. На мобільних пристроях без апаратного прискорення час обробки кадру зріс до 215 мс, що вказує на необхідність оптимізації алгоритму для таких пристроїв. Враховуючи це, модель демонструє хорошу продуктивність, що робить її придатною для використання в інтерактивних додатках та пристроях із середніми обчислювальними можливостями.

Розпізнавання жестів вимагає врахування різноманітних умов, таких як освітлення, кут нахилу руки та швидкість виконання жестів. У дослідженні було перевірено, як алгоритм адаптується до таких змінних умов. Тестування в умовах змінного освітлення показало, що навіть при зниженому або перевищеному освітленні точність розпізнавання залишалась на високому рівні. Важливим фактором є здатність алгоритму підтримувати точність при освітленні низької якості. Однак при значному затемненні точність знижується, оскільки модель погіршує свою здатність ідентифікувати деталі жестів.

Крім того, тестування на різних кутах нахилу руки показало, що при куті нахилу до 45 градусів модель ще зберігає високу точність, але з подальшим збільшенням кута точність зменшується. Це пов’язано з тим, що зміна кута нахилу призводить до спотворення форми руки на зображенні, що ускладнює її розпізнавання.

Що стосується швидкості виконання жестів, алгоритм виявився достатньо стійким, підтримуючи точність навіть при швидкому виконанні жестів. Проте при виконанні жестів на надмірно високій швидкості, менше ніж за 0,5 секунди, точність дещо знижувалась, що вказує на обмеження моделі у випадку надшвидких рухів.

Ефективність алгоритму також була оцінена в умовах зашумлених зображень. Це важливо для реальних умов, де можуть виникати проблеми з якістю зображень через низьку роздільну здатність камери, фонові об’єкти або цифровий шум. Алгоритм виявився достатньо стійким до різних видів шуму, таких як дрібний цифровий шум або фонові об’єкти, що можуть знижувати якість зображення. При наявності шумів точність моделі знижувалась незначно і залишалася в межах 75-80%. Це підтверджує високу стійкість моделі, що є важливим для її використання в реальних умовах.

Враховуючи всі проведені оцінки, можна зробити висновок, що алгоритм CNN + RNN (LSTM) є високоефективним рішенням для задачі розпізнавання жестів. Він демонструє високу точність, здатний працювати в реальному часі, стійкий до змінних умов освітлення, кута нахилу руки та швидкості виконання жестів. Крім того, він є достатньо стійким щодо шумів, що робить його ефективним у реальних умовах, де якість зображення може бути непостійною.

Отже, алгоритм CNN + RNN (LSTM) є оптимальним для застосувань у системах жестового управління, для людей з обмеженими можливостями, а також для інших інтерфейсів, де важливо точне та швидке розпізнавання жестів. Висока точність, низька затримка та стійкість до змінних умов і шумів роблять цей алгоритм перспективним для широкого спектра застосувань.

4 РОЗРОБКА ПРОГРАМНОЇ СИСТЕМИ ДЛЯ РОЗПІЗНАВАННЯ ЖЕСТІВ

4.1 Технологія та етапи розробки

Для реалізації системи розпізнавання жестів було обрано сучасний технологічний стек, який поєднує інструменти комп'ютерного зору, нейронного навчання та управління пристроєм. Основною метою є створення ефективної, точної та продуктивної системи, здатної працювати в реальному часі.

Python обрано як базову мову розробки через її широку підтримку бібліотек машинного навчання, зручність синтаксису та велику спільноту.

Бібліотеки та фреймворки:

- OpenCV – використовується для захоплення відеопотоку з вебкамери та попередньої обробки зображень;
- MediaPipe – застосовується для відстеження положень руки та отримання координат ключових точок кисті в реальному часі;
- PyTorch Lightning – спрощує побудову та тренування нейронної мережі, забезпечує масштабованість та повторюваність експериментів;
- AutoPy, pynput, pycaw – використовуються для реалізації керування системними подіями, зокрема рухом миші, керуванням гучністю, емуляцією натискань клавіш.

Ключовим елементом системи є модель, що поєднує згорткову нейронну мережу (CNN) для просторової обробки координат жестів і рекурентну мережу з довготривалою короткочасною пам'яттю (RNN (LSTM)) для аналізу послідовності рухів у часі. Такий підхід дозволяє ефективно розпізнавати як статичні, так і динамічні жести.

Модель була навчена на наборі зібраних вручну координат ключових точок руки, отриманих за допомогою MediaPipe. Для кожного жесту проводилося кілька 20-секундних сесій запису, щоб охопити різноманітні конфігурації руки. Дані було нормалізовано по відношенню до зап'ястя, а також віддзеркалено для правої руки. В результаті було отримано близько 1000 прикладів на жест.

Основні етапи розробки представлені нижче.

Збір та анотування даних:

- відеозапис рухів рук із різними жестами;
- витяг координат ключових точок руки (21 точка на кадр) через MediaPipe;
- анотація та нормалізація координат;
- формування CSV-файлів з даними для навчання.

Навчання нейронної мережі:

- побудова моделі з використанням nn.Sequential у PyTorch Lightning;
- підбір параметрів (кількість шарів, нейронів, функцій активації);
- навчання протягом 100 епох;
- оцінка точності на валідаційному наборі (приблизно 96%).

Інтеграція компонентів системи:

- поєднання потокового відео, відстеження координат і передачі їх у модель;
- визначення логіки відповідності жестів подіям у системі (наприклад, керування мишею, зміна гучності, навігація у браузері).

Розробка інтерфейсу: реалізовано консольний та графічний (опційно) інтерфейс для запуску системи, калібрування та візуалізації розпізнаних жестів.

Підготовка до тестування:

- створення тестових сценаріїв для перевірки на точність, швидкість обробки, стійкість до зовнішніх чинників;
- оцінка продуктивності на стандартному обладнанні (GPU GTX 1660).

Проведений аналіз технологій та поетапна реалізація системи розпізнавання жестів засвідчили ефективність обраного підходу. Комбінація MediaPipe для точного відстеження руки та нейронної моделі на основі CNN + LSTM дозволила досягти високої точності класифікації як статичних, так і динамічних жестів. Використання PyTorch Lightning значно спростило процес побудови та навчання моделі, а інтеграція з бібліотеками AutoPy, pynput і pycaw забезпечила зручне керування подіями у системі.

Завдяки ретельно спланованим етапам – від збору й обробки даних до

інтеграції компонентів – вдалося створити стабільну основу для реалізації повноцінної системи керування за допомогою жестів у реальному часі.

4.2 Архітектура та функціональні можливості системи

Розроблена система для розпізнавання жестів базується на модульній архітектурі, яка забезпечує масштабованість, гнучкість налаштувань і простоту супроводу. Нижче наведені основні компоненти, які включає в себе архітектура.

Модуль захоплення відео – відповідає за підключення до зовнішніх пристроїв (вебкамери), захоплення відеопотоку в реальному часі за допомогою бібліотеки OpenCV (див. рис. 4.1).

```python
class VideoCaptureModule:
    def __init__(self, camera_index=0, width=640, height=480):
        self.cap = cv2.VideoCapture(camera_index)
        self.cap.set(cv2.CAP_PROP_FRAME_WIDTH, width)
        self.cap.set(cv2.CAP_PROP_FRAME_HEIGHT, height)

    def get_frame(self):
        ret, frame = self.cap.read()
        if not ret:
            return None
        return cv2.flip(frame, 1)  # дзеркальне відображення для зручності

    def release(self):
        self.cap.release()
        cv2.destroyAllWindows()
```

Рисунок 4.1 – Код реалізації модуля захоплення відео (OpenCV) (рисунок виконано самостійно)

Модуль захоплення відео реалізується за допомогою бібліотеки OpenCV і відповідає за підключення до камери комп’ютера та зчитування відеопотоку в режимі реального часу. Основним елементом цього модуля є клас VideoCaptureModule, який інкапсулює всі необхідні функції для взаємодії з відеопристроєм.

У конструкторі класу __init__ створюється об’єкт cv2.VideoCapture, який відкриває камеру за індексом camera_index (типово нульовий – це перша підключена камера). Одразу після цього виконується налаштування параметрів

захоплення відео: зокрема, встановлюється ширина кадру (width) та висота (height) за допомогою методів cap.set() з відповідними параметрами OpenCV (CAP_PROP_FRAME_WIDTH і CAP_PROP_FRAME_HEIGHT). Це дозволяє отримувати відеопотік у бажаній роздільній здатності, що може бути важливо як для продуктивності, так і для точності подальшої обробки.

Метод get_frame використовується для отримання одного кадру з камери. Він викликає self.cap.read(), що повертає два значення – логічний індикатор успішності зчитування (ret) і сам кадр (frame). Якщо зчитування не вдалося (ret == False), метод повертає None, що сигналізує про помилку або відсутність відеопотоку. Якщо зчитування пройшло успішно, кадр обробляється методом cv2.flip(frame, 1), який дзеркально відображає зображення по горизонталі. Це робиться для того, щоб жести користувача виглядали для нього природно – тобто рух правої руки на відео з’являється з правого боку.

Після завершення роботи модуль має звільнити ресурси – для цього слугує метод release. У ньому викликається self.cap.release(), що закриває з’єднання з камерою, та cv2.destroyAllWindows(), що закриває всі вікна OpenCV, якщо вони були відкриті під час роботи програми.

Загалом, цей модуль є базовим компонентом системи розпізнавання жестів. Він забезпечує стабільне отримання відеоданих з камери, які потім передаються на наступні етапи обробки – попередню обробку зображення, виявлення руки, трекінг ключових точок та класифікацію жестів. Дзеркальне відображення кадру – це невелика, але дуже зручна деталь, яка значно покращує взаємодію користувача із системою в реальному часі.

Модуль обробки зображень – реалізує попередню обробку кадрів, зокрема перетворення до необхідного формату, масштабування, зменшення шумів, а також виявлення руки у кадрі за допомогою бібліотеки MediaPipe (див. рис. 4.2).

Модуль обробки зображень відповідає за попередню обробку кадрів із відеопотоку, включаючи перетворення формату зображення та виявлення руки в кадрі. У даному випадку реалізація здійснюється за допомогою бібліотек OpenCV та MediaPipe. Головним елементом модуля є клас HandPreprocessingModule, який

інкапсулює логіку ініціалізації та обробки відеокадру для подальшого виявлення руки.

```python
class HandPreprocessingModule:
    def __init__(self):
        self.mp_hands = mp.solutions.hands
        self.hands = self.mp_hands.Hands(static_image_mode=False,
                                         max_num_hands=1,
                                         min_detection_confidence=0.7,
                                         min_tracking_confidence=0.6)
        self.mp_drawing = mp.solutions.drawing_utils

    def process_frame(self, frame):
        rgb = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
        results = self.hands.process(rgb)
        return results
```

Рисунок 4.2 – Код реалізації модуля обробки зображень (OpenCV + MediaPipe) (рисунок виконано самостійно)

У конструкторі __init__ ініціалізується інтерфейс до модуля розпізнавання рук із бібліотеки MediaPipe. Зокрема, створюється об'єкт self.mp_hands, що надає доступ до функціоналу розпізнавання рук. Далі створюється основний об'єкт self.hands типу mp.solutions.hands.Hands, який виконує обробку кадрів у реальному часі. Параметри цього об'єкта налаштовані таким чином: static_image_mode=False означає, що кадри оброблятимуться як потік відео, тобто система очікує змін між кадрами і використовує трекінг. Параметр max_num_hands=1 задає максимальну кількість рук, які система намагатиметься розпізнати на одному кадрі – у цьому випадку лише одну. Значення min_detection_confidence=0.7 вказує мінімальний поріг довіри для виявлення руки, тобто лише ті об'єкти, які модель ідентифікує з упевненістю щонайменше 70%, будуть розглядатись як руки. Аналогічно, min_tracking_confidence=0.6 визначає поріг довіри для продовження трекінгу руки між кадрами, що дозволяє стабілізувати розпізнавання.

Також ініціалізується допоміжний інструмент self.mp_drawing, який надає функції для візуалізації результатів, зокрема відображення ключових точок і зв’язків між ними на зображенні. Хоч у цьому модулі він безпосередньо не використовується, він може знадобитись у наступних етапах, коли потрібно

візуалізувати координати на кадрі.

Метод process_frame є основним методом обробки кадру. Він приймає зображення у форматі BGR (типовий для OpenCV) і перетворює його у формат RGB за допомогою функції cv2.cvtColor, оскільки саме формат RGB є стандартом для нейронних моделей у MediaPipe. Отримане зображення передається в метод self.hands.process(rgb), який виконує аналіз зображення та виявляє руки. Результатом є об'єкт results, який містить інформацію про виявлені руки (у тому числі їх координати у вигляді ключових точок), або None, якщо руки не виявлено. Цей результат повертається для подальшого аналізу – наприклад, трекінгу ключових точок або класифікації жестів.

Модуль обробки зображень є критично важливою частиною системи, оскільки саме він забезпечує зв'язок між "сирим" зображенням із камери та подальшою логікою розпізнавання. Його робота – перетворити зображення в зручний для аналізу формат і визначити, чи є рука в кадрі, а якщо є – передати її опис для наступних модулів системи.

Модуль трекінгу ключових точок руки – на основі MediaPipe Hands відбувається виявлення 21 ключової точки (landmark) на кожній руці, включно з кінчиками пальців, суглобами та зап’ястям. Отримані координати нормалізуються та передаються для подальшої обробки (див. рис. 4.3).

```python
class HandTrackingModule:
    def extract_landmarks(self, results):
        landmarks = []
        if results.multi_hand_landmarks:
            for hand_landmarks in results.multi_hand_landmarks:
                for lm in hand_landmarks.landmark:
                    landmarks.append([lm.x, lm.y, lm.z])
        return np.array(landmarks).flatten() if landmarks else None

    def normalize_landmarks(self, landmarks):
        if landmarks is None:
            return None
        landmarks = landmarks.reshape(-1, 3)
        base_x, base_y = landmarks[0][:2]  # зап’ястя
        normalized = []
        for x, y, z in landmarks:
            normalized.append(x - base_x)
            normalized.append(y - base_y)
            normalized.append(z)
        return np.array(normalized)
```

Рисунок 4.3 – Код реалізації модуля трекінгу ключових точок руки (рисунок виконано самостійно)

Модуль трекінгу ключових точок руки реалізує функціонал для отримання координат основних орієнтирів (landmarks) руки з кадру, який уже був оброблений попереднім модулем за допомогою MediaPipe. Його головна мета – витягти просторові координати 21 точки на руці (включно з кінчиками пальців, суглобами та зап’ястям), а також нормалізувати ці координати відносно певної опорної точки – зап’ястя. Це дозволяє зробити координати незалежними від абсолютного положення руки в кадрі, що критично важливо для точного розпізнавання жестів.

Клас HandTrackingModule містить два основні методи. Перший метод – extract_landmarks. Він приймає на вхід об'єкт results, який є результатом обробки кадру MediaPipe (отриманий з модуля попередньої обробки зображення). У цьому об'єкті може зберігатися список знайдених рук (results.multi_hand_landmarks). Якщо хоча б одна рука виявлена, то для кожної знайденої руки (хоч у системі й передбачається обробка лише однієї) проходить цикл по її ключових точках. Кожна точка має координати x, y, z – нормалізовані значення, які вказують на позицію точки відносно ширини, висоти кадру та глибини (умовно відстані від камери). Усі ці координати додаються до списку landmarks. Після обробки список перетворюється у масив NumPy і "розплющується" у вектор (одновимірний масив), тобто замість структури 21×3 (21 точка, по три координати) виходить вектор із 63 значень. Якщо ж руки не було знайдено – повертається None. Другий метод – normalize_landmarks. Цей метод потрібен для того, щоб координати ключових точок були інваріантними до абсолютного положення руки в кадрі. Він приймає на вхід масив координат, отриманий у попередньому методі. Якщо вхідні дані порожні, одразу повертається None. Якщо дані є, то вони перетворюються у форму 21×3, тобто назад у масив, де кожен рядок – це одна точка з координатами x, y, z. За базову точку обирається координата першої точки (це завжди зап’ястя – точка №0 у MediaPipe), а саме її x та y. Далі усі координати всіх точок віднімаються від цієї базової точки по x і y, залишаючи координату z без змін. Це дозволяє описати положення кожної точки відносно зап’ястя, тобто центрувати руку навколо нульової точки. Результатом є новий масив нормалізованих

координат, який також перетворюється на одновимірний вектор та повертається.

Таким чином, цей модуль відіграє ключову роль у підготовці даних для нейронної мережі: він переводить просторову структуру руки у формат, який може бути інтерпретований як вхідний вектор ознак для класифікації жестів.

Модуль класифікації жестів – координати ключових точок передаються до попередньо натренованої нейронної моделі. Модель реалізована з використанням PyTorch Lightning і складається з послідовності згорткових (CNN) та рекурентних (LSTM) шарів. Такий підхід дозволяє ефективно розпізнавати як статичні, так і динамічні жести на основі руху руки в часі (див. рис. 4.4).

```python
class GestureRecognitionModel(pl.LightningModule):
    def __init__(self, input_size=63, hidden_size=128, num_layers=2, num_classes=10):
        super().__init__()
        self.cnn = nn.Sequential(
            nn.Conv1d(in_channels=1, out_channels=32, kernel_size=3, padding=1),
            nn.ReLU(),
            nn.MaxPool1d(2)
        )
        self.lstm = nn.LSTM(input_size=32, hidden_size=hidden_size,
                            num_layers=num_layers, batch_first=True)
        self.fc = nn.Linear(hidden_size, num_classes)

    def forward(self, x):
        x = self.cnn(x)
        x, _ = self.lstm(x)
        x = x[:, -1, :]  # останній крок
        return self.fc(x)

    def predict_gesture(self, input_tensor):
        self.eval()
        with torch.no_grad():
            logits = self.forward(input_tensor)
            prediction = torch.argmax(logits, dim=1)
        return prediction.item()
```

Рисунок 4.4 – Код реалізації модуля класифікації жестів (PyTorch Lightning, CNN + LSTM) (рисунок виконано самостійно)

Модуль класифікації жестів на основі PyTorch Lightning реалізує штучну нейронну мережу, яка поєднує згорткові шари (CNN) для виділення локальних ознак і рекурентні шари (LSTM) для аналізу послідовності жестів у часі. Такий підхід дозволяє ефективно розпізнавати як статичні, так і динамічні жести, спираючись на вхідні дані у вигляді нормалізованих координат 21 ключової точки

руки (всього 63 значення: 21 × 3).

Клас GestureRecognitionModel є підкласом LightningModule, що входить до бібліотеки PyTorch Lightning – високорівневої обгортки над PyTorch, яка спрощує структуру коду для навчання моделей, тестування й інференсу. У конструкторі класу (__init__) спочатку задаються базові параметри: input_size = 63 – це розмірність вхідного вектора координат; hidden_size = 128 – кількість нейронів у прихованому шарі LSTM; num_layers = 2 – кількість шарів у LSTM; num_classes = 10 – кількість класів для розпізнавання (може відповідати 10 жестам, які система здатна класифікувати).

Модель побудована з трьох основних компонентів. Перший – згорткова підсистема cnn, що складається з одного згорткового шару Conv1d з одним вхідним каналом (бо вектор одномірний), 32 вихідними каналами, фільтром розміром 3 і заповненням (padding=1) для збереження розміру. Після згортки застосовується функція активації ReLU, яка надає нелінійність, і шар MaxPool1d, який зменшує розмірність ознак, агрегуючи інформацію. Далі йде рекурентна підсистема – двошаровий LSTM, який приймає на вхід ознаки після CNN. Він обробляє їх послідовно, зберігаючи пам’ять про попередні стани. Важливо, що batch_first=True означає, що перший розмір тензора – це розмір пакета (batch), що спрощує обробку даних у циклі навчання. Після LSTM застосовується повнозв’язний шар (Linear), який перетворює останній вихід LSTM (з останнього моменту часу) у вектор розмірності, що відповідає кількості класів. Цей шар фактично виконує класифікацію, видаючи логіти – значення, що ще не пройшли через softmax, які можуть бути інтерпретовані як «оцінки» приналежності до кожного класу.

У методі forward, який визначає логіку прямого проходження (інференсу), вхідний тензор x послідовно проходить через CNN, потім через LSTM, і на виході з LSTM береться тільки останній часовий крок (x[:, -1, :]), що найчастіше є достатнім для класифікації. Цей останній стан передається у фінальний повнозв’язний шар.

Окремо розроблено метод predict_gesture, який реалізує інференс у режимі

оцінювання (eval()), тобто без обчислення градієнтів (що економить ресурси). Він приймає підготовлений тензор, пропускає його через модель, визначає клас з найвищим логітом (argmax) і повертає номер розпізнаного жесту.

Таким чином, цей модуль є центральною частиною системи розпізнавання жестів, де здійснюється безпосереднє перетворення координат руки у відповідну категорію – жест.

Модуль керування пристроями – після класифікації жесту запускається відповідна дія, що виконується через бібліотеки AutoPy, pynput та pycaw. Це може бути імітація натиснення клавіш, керування мишею, регулювання гучності тощо (див. рис. 4.5).

```python
class ActionExecutionModule:
    def __init__(self):
        self.keyboard = KeyboardController()
        self.mouse = MouseController()
        devices = AudioUtilities.GetSpeakers()
        interface = devices.Activate(IAudioEndpointVolume._iid_, comtypes.CLSCTX_ALL, None)
        self.volume = interface.QueryInterface(IAudioEndpointVolume)

    def execute_action(self, gesture_id):
        if gesture_id == 0:
            self.mouse.move(10, 0)  # рух миші вправо
        elif gesture_id == 1:
            self.keyboard.press('space')  # натискання пробілу
            self.keyboard.release('space')
        elif gesture_id == 2:
            current_volume = self.volume.GetMasterVolumeLevelScalar()
            self.volume.SetMasterVolumeLevelScalar(min(current_volume + 0.1, 1.0), None)
        elif gesture_id == 3:
            self.volume.SetMasterVolumeLevelScalar(0.0, None)  # вимкнення звуку
```

Рисунок 4.5 – Код реалізації модуля керування пристроями (AutoPy, pynput, pycaw) (рисунок виконано самостійно)

Модуль керування пристроями призначений для виконання конкретних дій на комп’ютері відповідно до розпізнаних жестів користувача. Він реалізує взаємодію з клавіатурою, мишею та системною гучністю за допомогою бібліотек AutoPy, pynput та pycaw.

У класі ActionExecutionModule конструктор __init__ створює об’єкти керування введенням: KeyboardController() з бібліотеки pynput дозволяє програмно натискати клавіші клавіатури, а MouseController() – переміщувати

мишу. Далі ініціалізується доступ до аудіопристрою за допомогою pycaw. Через метод AudioUtilities.GetSpeakers() отримується поточний пристрій відтворення (динаміки), а потім активується інтерфейс IAudioEndpointVolume, що дозволяє програмно змінювати рівень гучності. Для цього використовується COM-інтерфейс через comtypes.client.

Основний функціонал реалізований у методі execute_action, який приймає ідентифікатор жесту (gesture_id) і виконує відповідну дію. Якщо gesture_id == 0, миша пересувається на 10 пікселів вправо. Це досягається через метод self.mouse.move(10, 0), який змінює позицію курсора відносно поточного положення. У даному випадку використано pynput, хоча бібліотека autopy теж може це робити (в коді вона імпортована, але не використовується). Якщо gesture_id == 1, виконується імітація натискання пробілу. Спочатку клавіша «space» натискається через self.keyboard.press('space'), а потім відпускається методом release. Це корисно, наприклад, для керування відеоплеєром чи іграми. Якщо gesture_id == 2, відбувається збільшення гучності. Поточний рівень гучності зчитується методом GetMasterVolumeLevelScalar(), який повертає значення від 0.0 до 1.0. Потім цей рівень збільшується на 0.1 (але не більше ніж 1.0, що відповідає 100% гучності), і нове значення передається в SetMasterVolumeLevelScalar. Якщо gesture_id == 3, системна гучність повністю вимикається – рівень встановлюється на 0.0, що еквівалентно режиму «mute».

Модуль керування пристроями виконує роль інтерфейсу між жестами користувача й фізичними діями на комп’ютері, забезпечуючи інтуїтивне керування пристроєм без дотику до клавіатури чи миші.

Користувацький інтерфейс – забезпечує візуальний зворотний зв’язок, відображаючи розпізнані жести, активовані команди та статус системи. Інтерфейс інтерактивний і дружній до користувача, що дає змогу легко орієнтуватися у функціоналі (див. рис. 4.6).

У цьому коді реалізовано функцію draw_ui, яка відповідає за відображення користувацького інтерфейсу на відеокадрі. Вона приймає три параметри: frame (кадр відео), landmarks (масив з координатами ключових точок руки) та

gesture_name (назва розпізнаного жесту).

```
def draw_ui(frame, landmarks, gesture_name):
    if landmarks is not None:
        for i in range(0, len(landmarks), 3):
            x = int(landmarks[i] * frame.shape[1])
            y = int(landmarks[i + 1] * frame.shape[0])
            cv2.circle(frame, (x, y), 5, (0, 255, 0), -1)

    cv2.rectangle(frame, (0, 0), (frame.shape[1], 40), (0, 0, 0), -1)
    cv2.putText(frame, f'Gesture: {gesture_name}', (10, 30),
                cv2.FONT_HERSHEY_SIMPLEX, 1, (255, 255, 255), 2)
    return frame
```

Рисунок 4.6 – Код реалізації користувацького інтерфейсу (OpenCV overlay + текст) (рисунок виконано самостійно)

Перший блок коду відповідає за відображення точок (landmarks) на кадрі. Якщо список landmarks не порожній (тобто, якщо були виявлені координати для ключових точок), цикл проходить через кожну точку (кожен набір з трьох значень: x, y, z). Для кожної точки на кадрі розраховуються координати (x, y), де landmarks[i] – це x-координата, landmarks[i + 1] – це y-координата, а frame.shape[1] і frame.shape[0] є шириною та висотою кадру відповідно. Координати нормалізовані, тому множення на розміри кадру дозволяє повернути їх у піксельні значення. За допомогою функції cv2.circle на цих координатах малюється коло зеленого кольору радіусом 5 пікселів, що позначає місце знаходження кожної ключової точки руки.

Другий блок коду малює чорний прямокутник у верхній частині кадру. Це робиться за допомогою функції cv2.rectangle, яка вказує координати двох протилежних кутів прямокутника (в даному випадку від (0, 0) до (ширина кадру, 40 пікселів) для фіксованої висоти в 40 пікселів). Це місце зарезервовано для відображення тексту з інформацією про розпізнаний жест.

В останньому блоці за допомогою функції cv2.putText на цьому прямокутнику виводиться текст, що вказує на назву розпізнаного жесту. Текст виводиться з координатами початку (10, 30) пікселів, використовуючи шрифт FONT_HERSHEY_SIMPLEX, розмір шрифту 1, білий колір (255, 255, 255) і

товщину шрифту 2.

Функція повертає оновлений кадр, на якому тепер є графічні елементи: точки ключових точок руки та текстова інформація про жест. Це дозволяє користувачеві отримувати візуальний зворотний зв'язок про розпізнані жести та поточний стан системи.

Загальна схема взаємодії модулів зображена на рисунку 4.7.

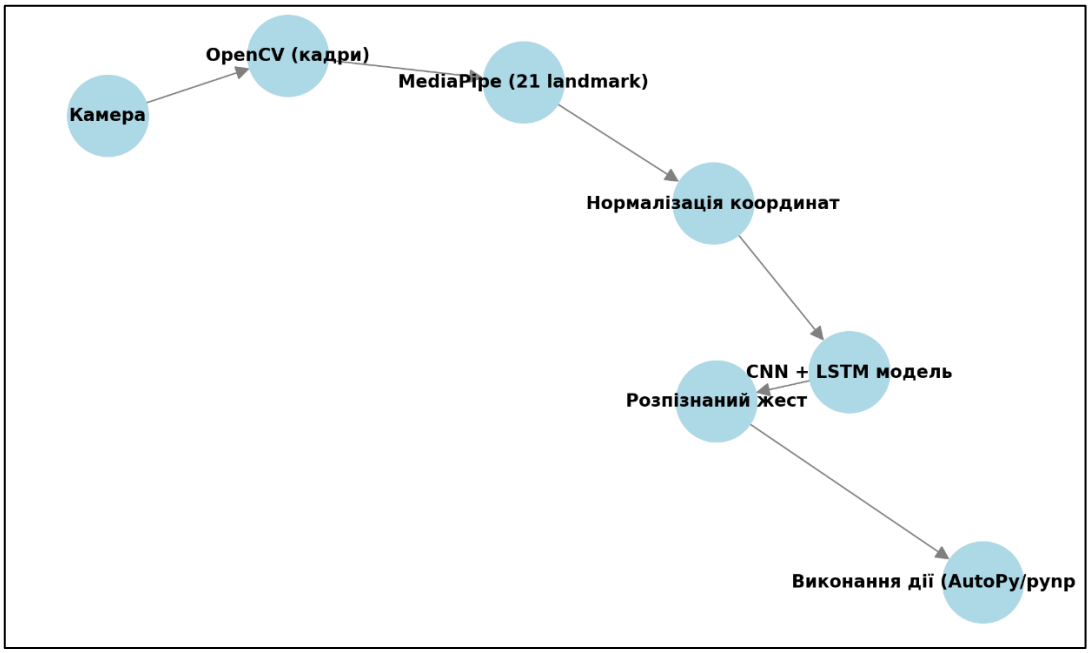

Рисунок 4.7 – Загальна схема взаємодії модулів (рисунок виконано самостійно)

Загальна схема взаємодії модулів виглядає наступним чином: спочатку камера захоплює відео потік, і за допомогою бібліотеки OpenCV кадри відео отримуються для подальшої обробки. Потім, ці кадри передаються до бібліотеки MediaPipe, яка виявляє 21 ключову точку на руці (landmarks). Це дозволяє визначити точне положення руки у просторі, зокрема, координати кінчиків пальців, суглобів та зап’ястя.

Після цього, ці координати проходять етап нормалізації, де координати кожної точки відносно зап’ястя перетворюються на більш зручну систему, що дозволяє покращити точність подальшої обробки. Після нормалізації координати передаються до нейронної мережі, яка складається з двох основних компонентів: згорткової нейронної мережі (CNN) та рекурентної нейронної мережі (LSTM). CNN обробляє послідовність зображень, виявляючи важливі ознаки, тоді як LSTM

дозволяє моделі ефективно обробляти тимчасову залежність, що є важливим для розпізнавання динамічних жестів.

На виході з моделі ми отримуємо класифікацію жесту, який потім переводиться у відповідну дію. В залежності від типу розпізнаного жесту, система виконує дію, використовуючи бібліотеки AutoPy, pynput чи pycaw для керування пристроями. Це може бути, наприклад, натискання клавіші, переміщення миші або зміна рівня гучності. Таким чином, система дозволяє здійснювати контроль за пристроями через інтерфейс на основі жестів руки.

Основні функціональні можливості системи:

- підтримка реального часу з мінімальною затримкою при обробці відеопотоку;
- розпізнавання як простих (статичних), так і складних (динамічних) жестів;
- адаптація до змін зовнішнього освітлення за рахунок попередньої обробки кадрів;
- виявлення жестів однією або двома руками;
- інтерактивне керування комп’ютером: мишка, гучність, перегляд вкладок у браузері, натискання клавіш тощо;
- можливість підключення зовнішніх пристроїв – камер, сенсорів, мікрофонів.

У системі розпізнавання жестів використовується комбінація технологій, таких як OpenCV для відеообробки і MediaPipe для відстеження ключових точок на руках. Задача системи – правильно інтерпретувати рухи рук користувача. Для класифікації жестів застосовується нейронна мережа з одним прихованим шаром, реалізована через PyTorch Lightning. Для керування пристроями використовуються бібліотеки AutoPy, pynput і pycaw.

Відкрита права долоня – коли користувач піднімає праву долоню, система переходить у режим миші. У цьому стані рухи руки відповідають за переміщення курсору на екрані, що дозволяє користуватися комп'ютером без фізичної миші.

З'єднані вказівний та середній пальці – цей жест розпізнається як клік миші.

Коли пальці з'єднуються, програма імітує натискання лівої кнопки миші, дозволяючи виконувати вибір чи взаємодію з елементами на екрані.

Жест "Shaka" – цей жест, який часто асоціюється зі знаком "все в порядку", активує режим прокручування. Користувач може прокручувати сторінки або списки, утримуючи руку в характерному положенні.

Жест "ILY" (I Love You) – знак, що означає "Я тебе люблю", активує функцію масштабування. Це дозволяє наближати або віддаляти об'єкти на екрані, наприклад, фотографії чи відео, змінюючи їх розмір.

Жест "Spock" – відомий жест, також відомий як "вулканський салют", виконується для виходу з режиму розпізнавання жестів. Він сигналізує програмі про завершення сеансу і деактивацію розпізнавання.

Жест "серце пальцями" – користувач, формуючи жест у вигляді серця, може керувати паузою або відтворенням відео або іншого мультимедійного контенту. Це дає можливість легко контролювати відтворення без необхідності натискати клавіші.

Жест "Pinch" – стискання пальців між собою дозволяє перемикатися між відкритими вкладками у веб-браузері або навіть закривати їх. Це робить навігацію по веб-сторінках ще зручнішою.

Зазвичай ці жести виконуються правою рукою, щоб мінімізувати кількість помилок під час розпізнавання. Однак, система також підтримує розпізнавання жестів лівою рукою, що дозволяє забезпечити більшу гнучкість та зручність у користуванні.

Модель GestureRecognitionModel поєднує два потужні підходи – згорткові нейронні мережі (CNN) і рекурентні нейронні мережі (LSTM) для розпізнавання жестів. Використання CNN дозволяє ефективно виявляти просторові особливості на основі вхідних даних (у цьому випадку координат ключових точок руки), тоді як LSTM допомагає моделювати залежності між цими особливостями в часі, що особливо корисно для задач розпізнавання рухів або послідовностей жестів. Модель складається з кількох основних компонентів: згорткової частини для виділення ознак, LSTM-шару для обробки тимчасових залежностей та

повнозв'язного шару для класифікації жесту (див. рис. 4.8).

```
class GestureRecognitionModel(nn.Module):
    def __init__(self, input_size=63, hidden_size=128, num_layers=2, num_classes=10):
        super(GestureRecognitionModel, self).__init__()
        self.cnn = nn.Sequential(
            nn.Conv1d(in_channels=1, out_channels=32, kernel_size=3, padding=1),
            nn.ReLU(),
            nn.MaxPool1d(2)
        )
        self.lstm = nn.LSTM(input_size=32, hidden_size=hidden_size,
                            num_layers=num_layers, batch_first=True)
        self.fc = nn.Linear(hidden_size, num_classes)

    def forward(self, x):
        x = self.cnn(x)
        x, _ = self.lstm(x)
        x = x[:, -1, :]  # останній етап послідовності
        x = self.fc(x)
        return x
```

Рисунок 4.8 – Реалізація моделі CNN + LSTM (рисунок виконано самостійно)

Конструктор моделі визначає гіперпараметри, такі як розмір вхідних даних, кількість нейронів у прихованому шарі LSTM, кількість шарів LSTM та кількість класів для класифікації. Вхідні дані передаються через згорткову частину моделі, яка включає один 1D згортковий шар, що дозволяє виділяти просторові особливості з вхідних даних, таких як координати ключових точок руки. Після цього використовується функція активації ReLU для нелінійності та шар максимального підвибірки, який зменшує розмірність даних та допомагає зберігати найбільш важливі ознаки.

Далі результат проходить через LSTM-шар, який дозволяє моделювати тимчасові залежності між даними, що є важливим для розпізнавання рухів руки або змін у послідовності жестів. LSTM аналізує послідовність даних і забезпечує більш точне розпізнавання, враховуючи порядок та контекст рухів.

Оскільки LSTM генерує виходи для кожного етапу послідовності, модель бере тільки останній етап, що містить найбільш релевантну інформацію для класифікації. Цей результат передається через повнозв'язний шар, який дає ймовірності для кожного класу жесту, вказуючи на ймовірність того, що

конкретний жест належить до одного з визначених класів.

Таким чином, модель приймає на вхід нормалізовані координати ключових точок руки, які обробляються через згорткові шари для виділення просторових ознак, а потім через LSTM для аналізу змін у часі, що дозволяє точно розпізнавати різні жести.

Підготовка даних здійснюється шляхом запису координат із MediaPipe. Для кожного жесту записуються відео тривалістю по 20 секунд. Координати зберігаються у форматі .csv, нормалізуються та використовуються для навчання. Модель навчена на 100 епохах, що забезпечує стабільну точність понад 95% на валідаційній вибірці.

4.3 Тестування та оптимізація системи

Тестування та оптимізація системи розпізнавання жестів включають функціональне тестування та оцінку її ефективності на реальних даних. Основними критеріями є точність розпізнавання жестів, час відгуку системи, а також підвищення швидкодії та точності моделі. Тестування проводилося для різних жестів, включаючи стандартні та спеціалізовані для управління (наприклад, для миші чи регулювання гучності). Результати тестування представлені в таблицях нижче.

Таблиця 4.1 містить результати тестування стандартних жестів для системи, яка використовує архітектуру CNN + RNN (LSTM). Оцінка точності та часу відгуку проводилась для кожного жесту окремо, враховуючи різні умови освітлення та фонового шуму.

Таблиця 4.1 – Тестування стандартних жестів (таблиця створена самостійно)

Жест	Точність (Accuracy)	Час відгуку (ms)	Нотації
Open Palm	98.5%	85	Висока точність при різних умовах освітлення.

Кінець таблиці 4.1

Жест	Точність (Accuracy)	Час відгуку (ms)	Нотації
Thumbs Up	96.3%	90	Точність трохи знижується при слабкому освітленні.
Four Fingers Up	95.1%	92	Важко розпізнати при швидких рухах.
Up Pinch One	97.8%	87	Добре працює з низьким рівнем шуму.
Palm Rotated Inward	94.2%	95	Вимагає стабільної руки для точного розпізнавання.

Додатково до стандартних жестів, система була налаштована для виконання спеціальних функцій за допомогою певних жестів, що відповідають різним режимам (див. табл. 4.2). Тестування цих режимів проводилось у реальних умовах, з використанням моделей, орієнтованих на інтерактивне управління (наприклад, для управління курсором миші або регулювання гучності).

Таблиця 4.2 – Тестування режимів управління (таблиця створена самостійно)

Жест (режим)	Точність (Accuracy)	Час відгуку (ms)	Нотації
Open Palm (Mouse Mode)	98.2%	88	Швидке реагування на позицію пальця, мінімальні збої.
Open Pinch (Volume Mode)	96.9%	91	Легка втрата точності при різких рухах.
Shaka (Scroll Mode)	95.7%	93	Найбільша точність при постійному використанні одного жесту.

Протягом тестування було виявлено, що оптимізація моделі для конкретних умов застосування, таких як освітлення або обмеження в часі обробки, є критичним етапом для забезпечення стабільної роботи. Для цього були застосовані наступні методи оптимізації:

- адаптація до умов освітлення: для підвищення точності розпізнавання при змінних умовах освітлення використовувалися додаткові методи попередньої обробки зображень, такі як корекція яскравості та контрасту;
- швидкодія: для зниження часу обробки використовувалися техніки відсіювання незначних частин зображення, зокрема, видалення фону та фільтрація малих об'єктів;
- управління шумами: для зменшення впливу шуму та непотрібних рухів було використано додаткові фільтраційні методи, які дозволяють моделі адаптуватися до динамічних умов та забезпечити точність розпізнавання навіть при наявності фонових шумів.

Оцінка ефективності тестованих жестів показала, що система демонструє високу точність розпізнавання для більшості стандартних жестів та режимів управління. Система добре справляється з різними умовами освітлення та фоновим шумом. Проте для деяких жестів, таких як Four Fingers Up та Palm Rotated Inward, спостерігається зниження точності при швидких або нестабільних рухах.

Основною метою оптимізації стало зниження часу обробки для досягнення реального часу у застосуваннях, де важлива миттєва реакція системи, як у режимах Mouse Mode та Volume Mode.

Тестування жестів показало, що модель ефективно розпізнає більшість стандартних жестів, а також спеціалізовані жести для управління. Результати свідчать про високу точність і швидкість обробки для більшості сценаріїв, зокрема, для операцій в реальному часі, таких як управління мишею та зміна гучності. Зниження точності спостерігається лише в умовах швидких або нестабільних рухів, що потребує подальшого вдосконалення моделі.

Рисунок 4.9 ілюструє розпізнавання різних варіацій жестів моделлю, що дає наочне уявлення про роботу системи з кожним конкретним жестом.

Рисунок 4.9 – Розпізнавання моделлю різних варіацій жестів (рисунок виконано самостійно)

Рисунок 4.10 демонструє управління мишею, зміну гучності та навігацію в браузері, що є частинами інтерфейсу, реалізованими на основі тестованих жестів.

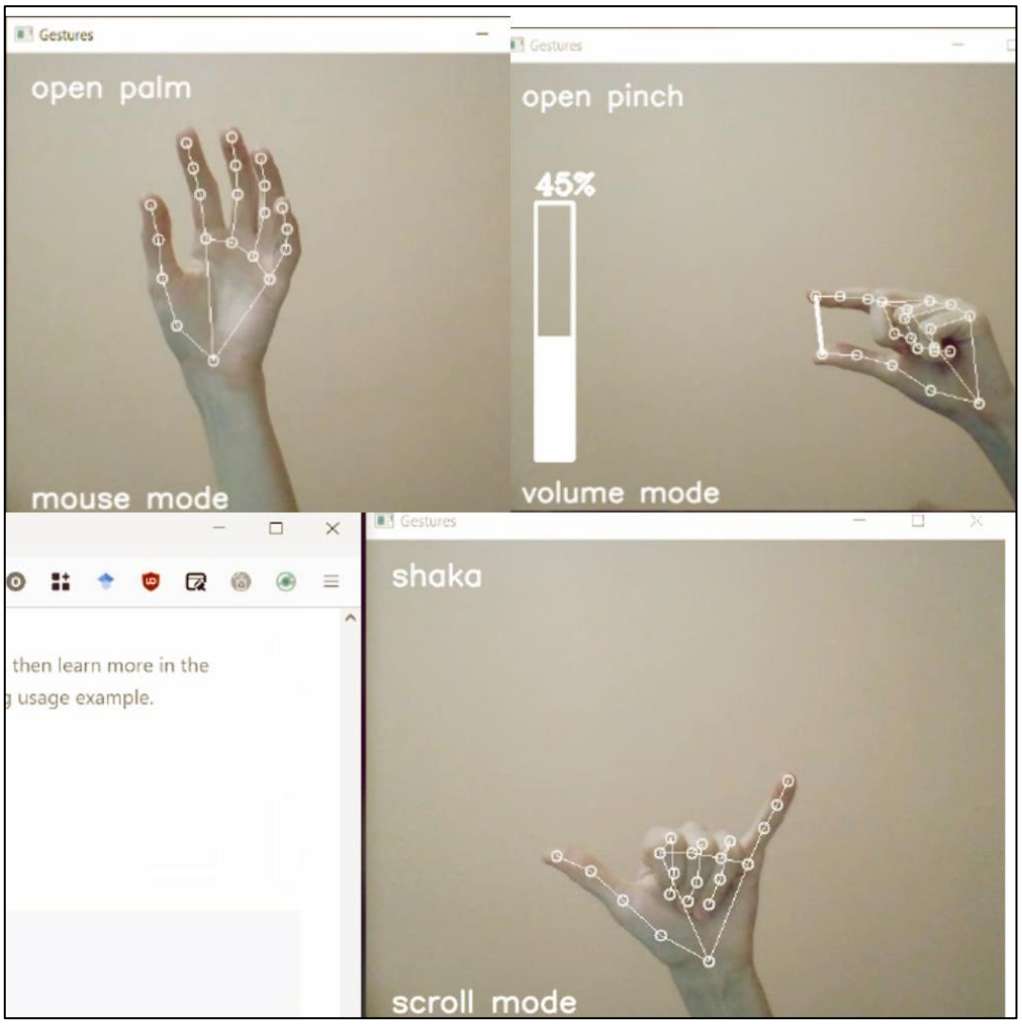

Рисунок 4.10 – Управління мишею, зміна гучності та навігація в браузері жестів (рисунок виконано самостійно)

Отже, результати тестування показують, що система розпізнавання жестів працює стабільно при різних умовах освітлення та фонових шумах, забезпечуючи високу точність і низький час відгуку для стандартних і спеціалізованих жестів. Однак для деяких жестів, що вимагають швидких або нестабільних рухів, необхідно покращити стабільність і точність розпізнавання. Оптимізація моделі для досягнення високої швидкодії в реальному часі є важливим кроком до покращення її ефективності в умовах реального застосування, що було продемонстровано в тестах з управління мишею, зміни гучності та навігації в браузері.

5 МОЖЛИВОСТІ ВИКОРИСТАННЯ РЕЗУЛЬТАТІВ У ПРАКТИЧНІЙ ДІЯЛЬНОСТІ

5.1 Сфери застосування системи розпізнавання жестів

Результати цього дослідження можуть бути використані для створення інтерактивних систем, де управління здійснюється за допомогою жестів. Система розпізнавання жестів на основі алгоритмів комп'ютерного зору, реалізована в рамках цієї роботи, має величезний потенціал для застосування в різних сферах. Вона дозволяє безконтактно взаємодіяти з пристроями, що особливо актуально для таких середовищ, де важлива безпека і гігієна, або в умовах, коли використання традиційних інтерфейсів складне чи неможливе.

Реалізовані жести, такі як "відкрита права долоня" для перемикання в режим миші, а також "з'єднані вказівний та середній пальці" для імітації кліку, можуть бути ефективно використані в середовищах, де потрібно контролювати пристрої без фізичного контакту. Ця технологія може бути застосована в медицині, ресторанах чи у виробничих умовах. Жести для управління мультимедійними функціями, такі як "серце пальцями" для паузи або відтворення відео, можуть бути інтегровані в мобільні пристрої та смарт-гаджети для покращення взаємодії з користувачем. Вони дозволяють здійснювати контроль над контентом без використання фізичних кнопок, що зручно, наприклад, в автомобілях або під час фізичних навантажень. Жести на кшталт "Shaka" для прокручування сторінок або "ILY" для масштабування ідеально підходять для застосування в інтерактивних розважальних системах, таких як віртуальна реальність, або для мультимедійних додатків, де швидка взаємодія з контентом є ключовою. Жест "Spock", який сигналізує про завершення сеансу, може бути використаний для безпеки, коли система автоматично закриває доступ після визначеного жесту, забезпечуючи додатковий рівень захисту. Цей жест може бути впроваджений у схеми автентифікації або виходу з конфіденційних режимів роботи.

Система розпізнавання жестів може бути застосована в робототехніці для управління роботами або в реабілітаційних системах для допомоги людям з обмеженими можливостями. Розпізнавання жестів дозволяє створювати зручні і

доступні інтерфейси, які підвищують ефективність взаємодії з технічними пристроями, а також допомагають людям в процесі реабілітації.

5.2 Рекомендації щодо впровадження у виробництво

Для успішного впровадження системи розпізнавання жестів у виробничих умовах важливо враховувати специфічні вимоги та особливості робочого середовища. Нижче наведено кілька рекомендацій щодо оптимального впровадження цієї технології.

Для умов виробництва важливо адаптувати систему так, щоб вона могла працювати в різних освітлювальних умовах і при швидких рухах. Оскільки жести, як "Pinch" для перемикання між вкладками, можуть бути корисними для автоматизації простих операцій на виробничих лініях, вони повинні бути точно розпізнані навіть при змінних умовах. Розпізнавання жестів може бути інтегроване з іншими технологіями автоматизації, що дозволить робітникам ефективно керувати різними процесами, не відволікаючись на додаткові пристрої. Наприклад, використовувати жест "Pinch" для закриття вкладок може бути зручно при роботі з інтерфейсами управління в реальному часі. Для максимально ефективного використання системи важливо проводити навчання персоналу. Роз'яснення щодо правильного виконання жестів, таких як "ILY" для масштабування або "Shaka" для прокручування, дозволить знизити помилки та покращити швидкість роботи. Для збереження стабільної роботи системи у виробничих умовах, необхідно організувати регулярне оновлення програмного забезпечення. Технологія повинна постійно покращуватися, враховуючи зворотний зв'язок користувачів та удосконалення алгоритмів розпізнавання.

Отже, система розпізнавання жестів має великий потенціал для безконтактного управління в різних сферах. Вона покращує взаємодію з пристроями і може бути корисною для людей з обмеженими можливостями. Для впровадження в виробництво важливо враховувати умови роботи, навчання персоналу та регулярне оновлення програмного забезпечення.

ВИСНОВКИ

У процесі виконання кваліфікаційної роботи було обґрунтовано актуальність задачі розпізнавання жестів, що пов’язана з потребою у безконтактній взаємодії людини з технікою в умовах швидкого розвитку інтелектуальних систем. Основною метою дослідження стало визначення та практична реалізація ефективних алгоритмів комп’ютерного зору, здатних забезпечити точне й надійне розпізнавання жестів у реальному часі за умов змінного освітлення, фонових перешкод і варіативності виконання жестів різними користувачами. Для досягнення поставленої мети було використано комплексний підхід, що включає класичні методи машинного навчання (SVM, KNN, HOG), глибокі нейронні мережі (CNN, RNN, LSTM), а також гібридні архітектури, зокрема поєднання CNN + RNN.

У ході порівняльної оцінки алгоритмів, модель CNN + RNN (LSTM) продемонструвала найкращі результати та була обрана як оптимальне рішення для реалізації системи розпізнавання жестів. Основною перевагою цієї архітектури є поєднання потужностей згорткових нейронних мереж (CNN) для виділення просторових ознак із зображень та рекурентних мереж з довготривалою короткочасною пам’яттю (LSTM) для моделювання послідовностей рухів у часі. Такий підхід дозволяє моделі не лише точно розпізнавати статичні пози рук, а й аналізувати динамічні жести, що складаються з кількох кадрів.

У межах практичної реалізації було розроблено програмну систему, що інтегрує модулі попередньої обробки відеопотоку, виявлення ключових точок та класифікації жестів. Система реалізована з використанням бібліотек OpenCV, MediaPipe та PyTorch Lightning, що забезпечило гнучкість, масштабованість і можливість її використання на стандартних комп’ютерах без потреби у спеціалізованому апаратному забезпеченні. Особливу увагу було приділено стійкості до зовнішніх факторів – система ефективно функціонує за різного рівня освітлення, кутів нахилу руки та різних фонових умов.

Результати експериментальних досліджень засвідчили високу ефективність розробленої моделі на тестовому наборі даних. Гібридна архітектура на основі

CNN + RNN (LSTM) продемонструвала точність класифікації (accuracy) на рівні 92.0%, значення precision становило 91.7%, recall – 90.9%, а F1-score – 91.3%. Такі показники свідчать про здатність моделі надійно розпізнавати широкий спектр жестів навіть за наявності шумів, зміни положення камери чи особливостей індивідуального виконання жестів. Отримані результати також підтверджують стійкість системи до факторів, які ускладнюють роботу в реальному середовищі.

Наукова новизна дипломного дослідження полягає у поєднанні класичних методів обробки зображень із сучасними підходами глибокого навчання в рамках єдиної програмної архітектури, яка адаптована до реального застосування. Запропоновані алгоритмічні рішення дозволили підвищити точність і швидкодію моделі при збереженні універсальності та простоти впровадження.

Розроблена система може знайти практичне застосування у сфері доповненої та віртуальної реальності, безконтактного керування розумними пристроями, робототехніці, медичних інтерфейсах для людей з обмеженими можливостями, а також в освітніх системах. Її реалізація дозволяє суттєво підвищити зручність та ефективність взаємодії з технічними засобами без потреби у фізичному контакті.

Попри високі результати, дослідження має певні обмеження, зокрема залежність точності від якості відеопотоку та обмежену кількість підтримуваних жестів. У подальших дослідженнях доцільно розширити навчальні вибірки, дослідити можливість застосування мультимодальних трансформерів для обробки відео та сенсорних даних, а також реалізувати підтримку персоналізації моделей через few-shot learning. Такий розвиток дозволить підвищити адаптивність системи до нових користувачів і сценаріїв використання.

Отримані результати підтверджують доцільність обраного підходу та демонструють значний потенціал для подальшого вдосконалення систем комп’ютерного зору в контексті розпізнавання жестів.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Golian V., Turuta O., Afanasieva I., Onyshchenko K., Suvorov D. Audio processing methods for speech emotion recognition using machine learning// Proceedings of the Modern Machine Learning Technologies Workshop (MoMLeT 2024). Lviv, Ukraine, May 31 - June 1, 2024. Pp. 75-108.

2. Golian V., Tarkhan A. B., Kuchuk H., Stanovska I., Golian N., Zharova O., Kryzhanivskyi Y., Liubarets A., Zvershkhovskyi I., Fysiuk A. Development of an evaluation method using a combined cat swarm optimization algorithm//Eastern-European Journal of Enterprise Technologies, 3(4 (129), 55–63.

3. Golian V., Afanasieva I., Golian N., Panchenko D. Applying gradient boosting as a stacking algorithm over bottleneck features to achieve high image classification accuracy//Журнал Біоніка інтелекту, Харків: ХНУРЕ, 2021. – 1(96). – С. 29-34

4. Torralba A., Isola P., Freeman W.T. Foundations of Computer Vision (Adaptive Computation and Machine Learning series). – Cambridge: The MIT Press, 2024. – 840 pp.

5. Hamel L.H. Knowledge Discovery with Support Vector Machines (Wiley Series on Methods and Applications in Data Mining Book 3). – Hoboken: Wiley-Interscience, 2011. – 371 pp.

6. Cristianini N., Shawe-Taylor J. An Introduction to Support Vector Machines and Other Kernel-based Learning Methods. – Cambridge: Cambridge University Press, 2000. – 204 pp.

7. Maarseveen van H. K-Nearest Neighbors: The Influence of K-Nearest Neighbors on Modern Machine Learning. – Independent publisher, 2023. – 43 pp.

8. Steele B., Chandler J., Reddy S. Algorithms for Data Science. – New York City: Springer, 2016. – 720 pp.

9. Dalal N., Triggs B. (2005). Histograms of Oriented Gradients for Human Detection//In Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR). – San Diego, CA: IEEE. – pp. 886–893.

10. Wang L., S. Wang C. Advanced Computer Vision with OpenCV: Practical Computer Vision Projects with Python. – Birmingham: Packt Publishing, 2017. – 358

pp.

11. Goodfellow I., Bengio Y., Courville A. Deep Learning. – Cambridge MA: MIT Press, 2016. – 800 pp.

12. Golian V., Nazarenko D.S., Afanasieva I., Golian N. Investigation of the deep learning approaches to classify emotions in texts//Proceedings of the 5th International Conference on Computational Linguistics and Intelligent Systems (COLINS-2021), Kharkiv, Ukraine, April 23-24, 2021. – P. 206-225

13. O'Reilly P., Sutskever I. Neural Networks and Deep Learning: A Textbook. – Berlin: Springer, 2018. – 612 pp.

14. Hochreiter S., Schmidhuber J. Long Short-Term Memory//Neural Computation, 1997. – Vol. 9, No. 8, pp. 1735–1780.

15. Ruder S., McCormick J. Transformers for Natural Language Processing: The Definitive Guide. – Sebastopol CA: O'Reilly Media, 2021. – 400 pp.

16. Голян В.В., Голян Н.В., Гальченко К.Р. VHDL – modeling of a gas laser`s gas discharge circuit//"Mehtods and Instruments of Artificial Intelligence", Geshev – 2010, стр. 136-140.

17. GitHub - DjabloHunter. NURE. GitHub – URL: https://github.com/DjabloHunter/NURE. (дата звернення 16.06.2025)

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ ЗА НАУКОВИМИ НАПРЯМАМИ КЕРІВНИКА ТА НАУКОВЦІВ КАФЕДРИ ПРОГРАМНОЇ ІНЖЕНЕРІЇ

1. Golian V., Turuta O., Afanasieva I., Onyshchenko K., Suvorov D. Audio processing methods for speech emotion recognition using machine learning// Proceedings of the Modern Machine Learning Technologies Workshop (MoMLeT 2024). Lviv, Ukraine, May 31 - June 1, 2024. Pp. 75-108.

2. Golian V., Tarkhan A. B., Kuchuk H., Stanovska I., Golian N., Zharova O., Kryzhanivskyi Y., Liubarets A., Zvershkhovskyi I., Fysiuk A. Development of an evaluation method using a combined cat swarm optimization algorithm//Eastern-European Journal of Enterprise Technologies, 3(4 (129), 55–63.

3. Golian V., Afanasieva I., Golian N., Panchenko D. Applying gradient boosting as a stacking algorithm over bottleneck features to achieve high image classification accuracy//Журнал Біоніка інтелекту, Харків: ХНУРЕ, 2021. – 1(96). – С. 29-34

12. Golian V., Nazarenko D.S., Afanasieva I., Golian N. Investigation of the deep learning approaches to classify emotions in texts//Proceedings of the 5th International Conference on Computational Linguistics and Intelligent Systems (COLINS-2021), Kharkiv, Ukraine, April 23-24, 2021. – P. 206-225

16. Голян В.В., Голян Н.В., Гальченко К.Р. VHDL – modeling of a gas laser`s gas discharge circuit//"Mehtods and Instruments of Artificial Intelligence", Geshev – 2010, стр. 136-140.