Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра обчислювальної техніки

БАКАЛАВРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему:

«Комп'ютерна система розпізнавання мови жестів з використанням штучного інтелекту»

08-54.БКР.017.00.000 ПЗ

Виконала: студентка 4 курсу, групи 1КІ-21б
спеціальності 123 – «Комп'ютерна інженерія»
освітня програма – «Комп'ютерна інженерія»
Шпачинська А. С.
Керівник к.т.н., доц. каф. ОТ
Городецька О. С.
Рецензент д.т.н., проф. каф. ПЗ
Ліщинська Л. Б.

Допущено до захисту
Завідувач кафедри ОТ
д.т.н., проф. Азаров О. Д.
«13» 06 2025 р.

Вінниця ВНТУ – 2025 рік

Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра обчислювальної техніки
Освітньо – кваліфікаційний рівень бакалавр
Галузь знань — 12 Інформаційні технології
Спеціальність — 123 Комп'ютерна інженерія
Освітня програма — Комп'ютерна інженерія

ЗАТВЕРДЖУЮ
Завідувач кафедри ОТ
д.т.н., проф. _____ Азаров О. Д.
«21» березня 2025 року

З А В Д А Н Н Я
НА БАКАЛАВРСЬКУ КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ
Шпачинській Анастасії Степанівні

1 Тема роботи: Комп'ютерна система розпізнавання мови жестів з використанням штучного інтелекту, керівник роботи Городецька Оксана Степанівна к.т.н., доцент кафедри ОТ, затверджено наказом вищого навчального закладу від «20» березня 2025 року №97.

2 Строк подання студентом роботи 13.05.2025

3 Вихідні дані до роботи: відеозаписи або зображення з жестами, що виконуються руками, навчальні приклади жестів української жестової мови, координати ключових точок: рук, пальців, отримані з камер або сенсорів за допомогою технологій комп'ютерного зору.

4 Зміст розрахунково-пояснювальної записки: вступ; аналіз сучасного стану нейронних мереж та методів розпізнавання жестів; вибір технологій та архітектури системи розпізнавання мови жестів; розробка та навчання нейронної мережі; розробка комп'ютерної системи розпізнавання мови жестів на основі нейронної мережі та її тестування; висновки.

5 Перелік графічного матеріалу: архітектура комп'ютерної системи

розпізнавання мови жестів, структурна схема комп'ютерної системи розпізнавання мови жестів.

Перелік ілюстративного матеріалу: графіки точності та функції втрат моделей ансамблю, матриця плутанини.

6 Консультанти розділів роботи приведені в таблиці 1.

Таблиця 1 — Консультанти роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1-4	к.т.н., доц. каф. ОТ Городецька О. С.	21.03.25	13.05.25
Нормоконтроль	ас.. каф. ОТ Швець С. І.	14.05.25	30.05.25

7 Дата видачі завдання 21.03.2025 р.

8 Календарний план приведений в таблиці 2.

Таблиця 2 — Календарний план

№ з/п	Назва етапів дипломної роботи	Строк виконання	Підпис
1	Постановка задачі роботи	21.03.25	вик.
2	Пошук літературних джерел та аналіз методів розпізнавання жестів	21.03.25 — 28.03.25	вик.
3	Дослідження технологій комп'ютерного зору та підходів до зчитування координат руки	29.03.25 — 06.04.25	вик.
4	Збір і підготовка власного датасету, навчання моделі для класифікації жестів	07.04.25 — 14.04.25	вик.
5	Розробка вебінтерфейсу та інтеграція моделі розпізнавання в режимі реального часу	15.04.25 — 27.04.25	вик.
6	Оформлення пояснювальної записки та підготовка ілюстративного матеріалу	28.04.25 — 11.05.25	вик.
7	Перевірка якості виконання бакалаврської кваліфікаційної роботи та усунення недоліків	13.05.25	вик.

Студент Анастасія ШПАЧИНСЬКА

Керівник роботи к.т.н., доц. Оксана ГОРОДЕЦЬКА

АНОТАЦІЯ

УДК 004.8

Шпачинська А. С. Комп’ютерна система розпізнавання мови жестів з використанням штучного інтелекту. Бакалаврська дипломна робота зі спеціальності 123 — Комп’ютерна інженерія, освітня програма — Комп’ютерна інженерія. Вінниця: ВНТУ, 2025. 76 с.

На укр.мові. Бібліогр.: 30 назв; рис.: 26.

У бакалаврській кваліфікаційній роботі розроблено програмний засіб для розпізнавання жестів української жестової мови в реальному часі. Система використовує технології комп’ютерного зору для зчитування координат рук і методи глибинного навчання для класифікації жестів на основі послідовності кадрів. Для навчання нейронної мережі зібрано власний датасет відеожестів із подальшим виділенням координат ключових точок рук за допомогою бібліотеки MediaPipe. Програмну реалізацію виконано мовою Python з використанням фреймворку Flask, бібліотек OpenCV, TensorFlow і Keras. Результати розпізнавання жестів виводяться у вебінтерфейсі. Проведено тестування працездатності системи та базову оцінку її точності.

Ключові слова: розпізнавання жестів, жестова мова, нейронні мережі, MediaPipe, TensorFlow, GRU, LSTM, комп’ютерний зір, машинне навчання.

ABSTRACT

Shpachynska A. S. Computer system for sign language recognition using artificial intelligence. Bachelor's thesis in specialty 123 — Computer engineering, educational program — Computer engineering. Vinnytsia: VNTU, 2025. 76 p.

Ukrainian language. Bibliogr.: 30 titles; figs.: 26.

In the bachelor's qualification work, a software tool for real-time recognition of Ukrainian sign language gestures was developed. The system uses computer vision technology to read the coordinates of hands and deep learning methods to classify gestures based on a sequence of frames. To train the neural network, I collected our own dataset of video gestures and then extracted the coordinates of key points of the hands using the MediaPipe library. The software implementation was done in Python using the Flask framework, OpenCV, TensorFlow and Keras libraries. The results of gesture recognition are displayed in the web interface. The system performance is tested and a basic assessment of its accuracy is performed.

Keywords: gesture recognition, sign language, neural networks, MediaPipe, TensorFlow, GRU, LSTM, computer vision, machine learning.

ЗМІСТ

ВСТУП

У сучасних умовах розвитку інформаційних технологій зростає потреба у створенні рішень, що забезпечують рівний доступ до комунікації для всіх категорій населення. Особливої уваги потребують люди з порушенням слуху, для яких жестова мова є основним засобом спілкування. Водночас більшість населення не володіє жестовою мовою, що значно ускладнює комунікацію та соціальну інтеграцію таких осіб. У зв'язку з цим розробка інтелектуальних систем автоматичного розпізнавання жестів з подальшим перекладом у текст чи голосове повідомлення є вкрай актуальним завданням.

Актуальність даної роботи полягає у необхідності створення доступних та ефективних технологій взаємодії для людей з порушенням слуху, що сприятиме їхній активнішій участі в суспільному житті. Завдяки стрімкому розвитку штучного інтелекту, машинного навчання та комп'ютерного зору стало можливим реалізувати системи, здатні розпізнавати жести в реальному часі. Водночас важливим напрямом є адаптація таких рішень до особливостей української жестової мови, оскільки більшість наявних розробок орієнтовані на англомовне середовище.

Метою даної роботи є розширення функціональних можливостей комп'ютерної системи розпізнавання жестів української жестової мови завдяки інтеграції з модулем штучного інтелекту.

Для досягнення поставленої мети передбачається виконання таких завдань:

— здійснити огляд сучасного стану розвитку нейронних мереж;

— проаналізувати методи комп'ютерного зору для обробки зображень та відео;

— обґрунтувати вибір технологій, інструментів та архітектури системи;

— сформувати датасет жестів української жестової мови для навчання моделі;

— здійснити навчання нейронної мережі для розпізнавання жестів;

— розробити комп'ютерну систему розпізнавання мови жестів з використанням штучного інтелекту;

— здійснити перевірку працездатності системи.

Об’єктом дослідження є процес розпізнавання жестової мови.

Предметом дослідження є методи комп’ютерного зору та машинного навчання, що використовуються для класифікації жестів у реальному часі.

Апробація результатів бакалаврської роботи: зроблено доповідь на LIV науково-технічної конференції факультету інформаційних технологій та комп’ютерної інженерії [1]:

Інформаційна система розпізнавання мови жестів з використанням штучного інтелекту / А. С. Шпачинська, О. С. Городецька // Матеріали LIV науково-технічна конференція факультету інформаційних технологій та комп’ютерної інженерії. Вінниця 2025 р. [Електронний ресурс]. Режим доступу: https://conferences.vntu.edu.ua/index.php/all-fitki/all-fitki-2025/paper/view/23924/19786

1 АНАЛІЗ СУЧАСНОГО СТАНУ НЕЙРОННИХ МЕРЕЖ ТА МЕТОДІВ РОЗПІЗНАВАННЯ ЖЕСТІВ

1.1 Штучні нейронні мережі. Історія розвитку та сучасний стан

Штучна нейронна мережа (ШНМ) — це математична модель або її програмна чи апаратна реалізація, побудована за принципом організації біологічних нейронних мереж, що складається з просто влаштованих штучних нейронів, з'єднаних між собою. Спільно вони здатні виконувати складні обчислювальні завдання, зокрема розпізнавання, класифікацію, прогнозування, тощо. В основі ШНМ лежить здатність до навчання шляхом зміни параметрів зв'язків між нейронами, що дозволяє системі адаптуватися до нових даних [2].

Штучна нейронна мережа моделює принцип роботи нервової системи. Вона складається з великої кількості взаємопов'язаних елементів – нейронів, які приймають, обробляють та передають інформацію. Кожен нейрон має вхідні з'єднання, які передають сигнали з певним числовим коефіцієнтом, що визначає вплив конкретного сигналу на результат роботи мережі [3].

Процес обробки інформації у ШНМ розподіляється на декілька етапів: отримання даних (пікселі зображення, числові параметри, тощо), послідовна обробка й перетворення, остаточний результат (клас об'єкта, числове значення, тощо) [3]. Структуру штучної нейронної мережі показано на рисунку 1.1.

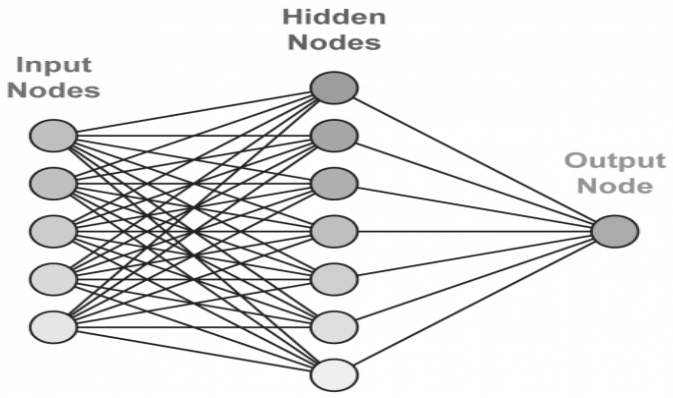

Рисунок 1.1 — Базова структура штучної нейромережі

На відміну від традиційних алгоритмів, які працюють за чітко заданими правилами, нейромережі навчаються на прикладах. Під час навчання мережа

отримує вхідні дані разом із правильними відповідями (навчальні вибірки або датасети) і поступово коригує свої вагові коефіцієнти так, щоб зменшити помилку. Цей процес відбувається за допомогою спеціального алгоритму — зворотного поширення помилки, що дозволяє мережі навчитися узагальнювати знання, а не лише запам’ятовувати приклади [3].

Перші концепції штучних нейронних мереж з’явилися ще у середині XX століття. У 1957 році Френк Розенблатт запропонував модель перцептрона — найпростішої нейронної мережі, здатної навчатися розпізнаванню простих образів. Це стало важливим кроком у розвитку машинного навчання. Однак через обмеження цієї архітектури інтерес до нейромереж поступово згас у 1970-х роках, коли дослідники Мінський та Паперт критично оцінили потенціал перцептронів [4].

Варто зазначити, що Україна також має свої вагомі досягнення у розвитку нейромереж. У 1969 — 1971 роках в Київському НДІ гідроприладів під керівництвом Олександра Різника було створено експериментальні моделі АДАМ-А та АДАМ-Д. Перший працював на базі електрохімічних елементів пам’яті і мав 16 нейронів, а другий став першим у світі повністю цифровим нейрокомп’ютером на 512 елементів [5].

Після тривалого періоду застою, з початку 2000-х років, завдяки розвитку алгоритмів та потужності обчислювальних систем, ШНМ знову привернули увагу дослідників. Особливу роль у цьому відіграли глибокі нейронні мережі, які змогли навчатися на великих масивах даних і демонструвати високу точність у складних завданнях класифікації, прогнозування та розпізнавання [4].

В сучасному світі нейронні мережі мають широкий спектр застосувань. Їх використовують від розпізнавання мовлення, обробки зображень і генерації тексту до автономного керування автомобілями. Нові системи комп’ютерного зору здатні виявляти об’єкти на відео в реальному часі, а мовні моделі чудово ведуть діалог на рівні, близькому до людського [4].

Завдяки високій адаптивності та здатності навчатися на великій кількості даних, нейронні мережі стали ключовим компонентом сучасних технологій штучного інтелекту. Основна популярність ШНМ полягає в тому, що вона може

правильно реагувати на нові, раніше не бачені приклади, розпізнаючи закономірності та узагальнюючи знання. Саме ця особливість зумовила надзвичайну ефективність нейронних мереж у різноманітних прикладних задачах.

1.2 Методи обробки зображень та відео в комп'ютерному зорі

Комп'ютерний зір (Computer Vision) — це галузь штучного інтелекту, яка займається автоматичним отриманням, обробкою, аналізом та інтерпретацією візуальної інформації з цифрових зображень, відео або інших джерел візуальних даних з метою прийняття рішень або виконання певних дій [6].

Основною метою комп'ютерного зору є автоматизація процесів візуального контролю, що традиційно виконувалися людиною. Технологія забезпечує обробку візуальної інформації в режимі реального часу, надаючи комп'ютерним системам здатність сприймати та інтерпретувати зображення подібно до людського зору [6].

Алгоритми комп'ютерного зору ефективно працюють із даними, отриманими з відеокамер, сенсорів та інших цифрових пристроїв, що дає змогу створювати гнучкі й адаптивні системи моніторингу та управління. Це сприяє зменшенню впливу людського фактора, підвищенню точності аналізу, своєчасності реагування на критичні події, а також оптимізації виробничих процесів та підвищенню рівня технологічної безпеки [6].

Методи обробки зображень та відео в комп'ютерному зорі поділяються на декілька ключових груп, кожна з яких орієнтована на конкретні завдання та типи оброблюваних даних.

До першої групи можна віднести методи просторової обробки, які працюють з піксельними значеннями зображення та застосовуються для покращення його якості. Вони охоплюють фільтрацію, згладжування, підвищення різкості, зміну контрасту, а також локальну обробку і морфологічні операції, що дозволяють змінювати форму об'єктів або виділяти їхні характерні риси. Обробка здійснюється безпосередньо в області пікселів за допомогою масок, які переміщуються по зображенню та обчислюють нові значення на основі локального оточення. Такі методи готують зображення до подальшого аналізу, зменшують шуми, покращують

видимість об'єктів і сприяють якіснішій роботі систем комп’ютерного зору [7].

Просторові методи поділяються на лінійні та нелінійні. Лінійні, зокрема згладжувальні фільтри, дозволяють зменшити шум, хоча при цьому можуть трохи розмивати контури. Нелінійні фільтри, навпаки, ефективно усувають імпульсні шуми, зберігаючи чіткість країв, що робить їх більш придатними для багатьох завдань обробки зображень [7].

Наступною групою є методи частотної обробки, які дозволяють аналізувати спектральні компоненти зображень та відео. В основі цих методів лежать перетворення Фур’є та вейвлет-перетворення, що дають змогу виявляти періодичні структури, текстури та аномалії. Застосування цих методів сприяє покращенню якості зображень, а також оптимізації їхнього зберігання і передачі [8].

Перетворення Фур’є дозволяє розкласти зображення на складові частоти, що сприяє ефективному фільтруванню та стисненню даних. Вейвлет-перетворення, на відміну від Фур’є, забезпечує аналіз як у частотній, так і у часово-просторовій області, що підвищує точність виявлення локальних особливостей та текстур у зображенні [8].

Методи сегментації відіграють ключову роль у комп’ютерному зорі, оскільки дозволяють розділяти зображення на окремі смислові області, такі як об’єкти, фон або інші важливі елементи, що полегшує їхній подальший аналіз, класифікацію та обробку. Завдяки цьому точність розпізнавання об’єктів значно підвищується навіть у складних та зашумлених зображеннях, що робить методи сегментації незамінними у багатьох сферах застосування комп’ютерного зору, зокрема в медичній діагностиці, відеоспостереженні та автономних транспортних системах [9].

До основних методів сегментації належать порогова обробка, кластеризація, регіональне зростання та графові методи. Порогова обробка дозволяє відокремити об’єкти від фону на основі визначеного порогу інтенсивності пікселів, що робить цей метод простим і ефективним у випадках з чітким контрастом. Кластеризація групує пікселі за схожими характеристиками, що дає змогу виділяти області з подібними ознаками. Регіональне зростання базується на об’єднанні пікселів, які

мають схожі властивості, поступово розширюючи сегменти. Графові методи моделюють зображення як граф, де вузли відповідають пікселям або їх групам, а ребра — зв’язкам між ними, що дозволяє ефективно вирішувати задачі розподілу областей [9].

Методи виявлення та опису ознак дають змогу знаходити на зображеннях характерні точки, контури, текстури або інші унікальні елементи, що сприяють подальшій ідентифікації об’єктів. Завдяки їм можна не лише виявляти важливі об’єкти в кадрі, а й створювати їх детальні описи, які застосовуються для зіставлення та порівняння з іншими зображеннями. Це особливо важливо для завдань розпізнавання, відстеження руху, 3D-реконструкції та виявлення змін. Крім того, ці методи забезпечують стійкість до змін ракурсу, освітлення, масштабу та часткового перекриття об’єктів, що значно підвищує точність і надійність алгоритмів комп’ютерного зору в реальних умовах експлуатації. Такі підходи широко використовуються у сфері безпеки, робототехніки, медичної діагностики та автономних транспортних систем [10].

Методи аналізу руху орієнтовані на роботу з відеопотоками або послідовностями зображень і мають на меті виявлення динамічних змін у сцені. Завдяки здатності працювати з часовою складовою, ці методи дозволяють не лише фіксувати факт наявності руху, але й відстежувати об’єкти в часі та аналізувати їхню поведінку: рух, зміну форми, появу або зникнення. У практичних застосуваннях аналіз руху широко використовується в системах відеоспостереження, автономному транспорті, у сфері робототехніки та доповненої реальності [11].

Одним із базових підходів до виявлення руху є метод різниці кадрів, за якого від поточного кадру віднімається попередній, а отримане різницеве зображення використовується для ідентифікації змін. Хоч цей підхід простий у реалізації, він чутливий до шумів, тіней та змін освітлення. Іншим важливим методом є аналіз оптичного потоку, що відображає переміщення кожного пікселя між двома послідовними кадрами. Алгоритми оптичного потоку дозволяють точно визначати напрямок і швидкість руху, що є надзвичайно важливим для задач трекінгу [11].

Крім класичних підходів, сучасні методи аналізу руху активно використовують глибокі згорткові нейронні мережі, здатні автоматично навчатися виявляти та описувати шаблони руху в складних сценах. Такі підходи дозволяють системам враховувати контекст, фонові умови та складні взаємодії між об’єктами [11].

Статичні та стохастичні методи аналізу зображень ґрунтуються на використанні математичних моделей та ймовірнісних підходів для виявлення та інтерпретації структур у візуальних даних. Вони дозволяють не лише знаходити закономірності, але й оцінювати ймовірність належності елементів зображення до певних класів. Це особливо важливо у випадках, коли зображення є зашумленими або містять неоднозначні фрагменти. Такі методи активно застосовуються для класифікації, порівняння та відновлення зображень і відео, забезпечуючи адаптивність до складних і варіативних вхідних даних [12].

Усі розглянуті методи активно застосовуються в сучасних технологіях комп’ютерного зору. Їх поєднання дозволяє створювати потужні інтелектуальні системи, здатні ефективно аналізувати та інтерпретувати візуальні дані в найрізноманітніших сферах.

1.3 Типи нейронних мереж для розпізнавання жестів

Розпізнавання жестів потребує чіткого виділення ознак та ефективної інтерпретації руху і положення рук. Для досягнення високої точності у подібних задачах застосовують спеціалізовані види нейронних мереж, адаптовані під обробку вхідних зображень або відеопотоків.

Часто, для такого типу завдань, використовують згорткові нейронні мережі (CNN). Їхня архітектура побудована таким чином, щоб ефективно виявляти просторові ознаки з вхідного зображення, що робить CNN ідеальними для розпізнавання статичних жестів. Згорткові шари таких мереж застосовують фільтри для виявлення візуальних патернів, таких як краї, форми або положення пальців. Ці мережі здатні автоматично навчатись розпізнавати важливі для класифікації ознаки без потреби у ручному виділенні, що значно підвищує точність та узагальнювальну

здатність моделі [13].

Крім того, CNN добре масштабується до великих наборів даних та різних рівнів складності. Завдяки своїй модульній структурі, яка включає згорткові, підвибіркові та повнозв'язні шари, такі мережі здатні витягувати ознаки на різних рівнях абстракції. На перших шарах виявляються базові елементи, як-от краї або кути, а на глибших — більш складні структури, наприклад, комбінації пальців або форма руки [13].

У поєднанні з сучасними техніками глибинного навчання CNN досягають високої продуктивності в задачах класифікації зображень, навіть при наявності шуму або складного фону. Саме тому CNN є основою більшості сучасних систем для обробки зображень, зокрема і в контексті розпізнавання жестів. Архітектуру згорткових нейронних мереж показано на рисунку 1.2.

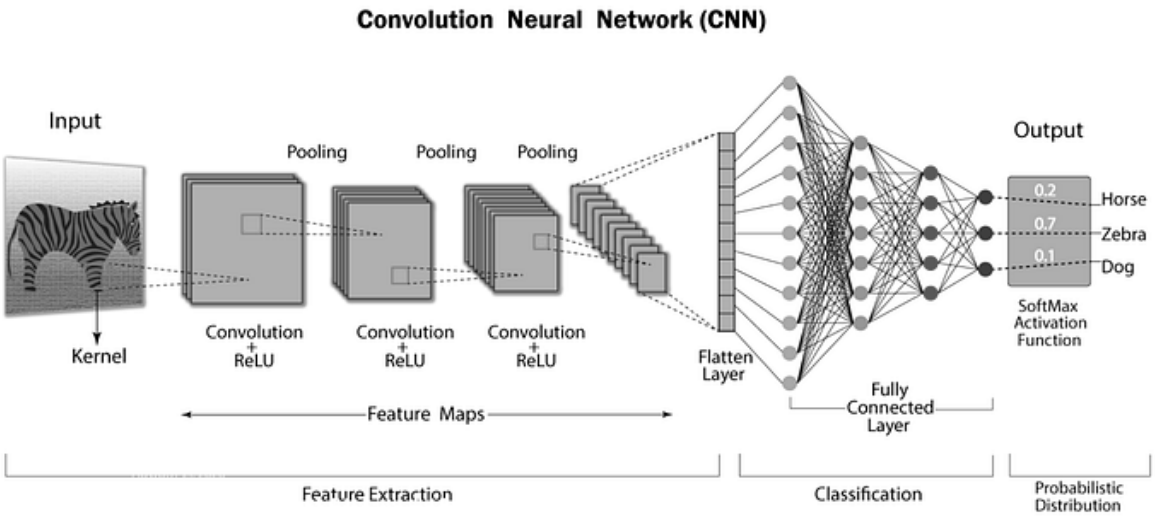

Рисунок 1.2 — Архітектура CNN

У випадках коли важливе значення має послідовність рухів у часі, використовуються рекурентні нейронні мережі (RNN). На відміну від згорткових мереж, RNN мають здатність обробляти послідовні дані, зберігаючи інформацію про попередні стани у внутрішній пам'яті. Це дає змогу аналізувати залежності між поточним і попередніми кадрами, що особливо важливо для відстеження змін у русі, інтерпретації жестів або мови тіла [13].

Проте базові RNN мають проблему затухаючого градієнта, через що їм важко

зберігати довготривалу інформацію. Для подолання цієї проблеми були розроблені вдосконалені архітектури, такі як LSTM (Long Short-Term Memory) і GRU (Gated Recurrent Unit). Обидва підходи використовують спеціальні механізми, які дозволяють зберігати релевантну інформацію довше і фільтрувати непотрібну [13]. Базову архітектуру рекурентних нейронних мереж показано на рисунку 1.3.

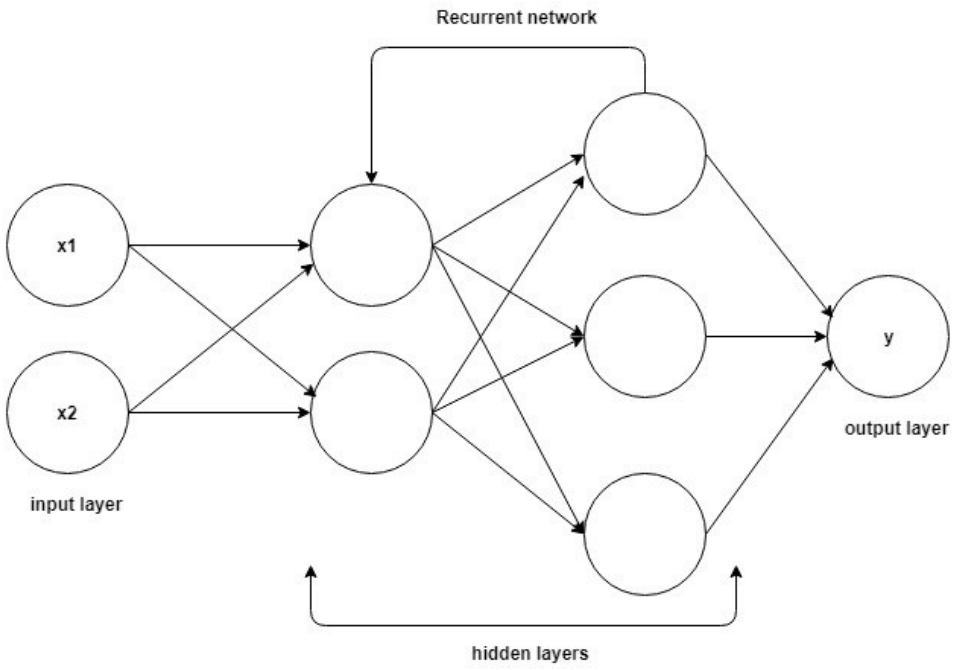

Рисунок 1.3 — Архітектура RNN

Ще одним підходом до обробки, як просторової, так і часової інформації є тривимірні згорткові нейронні мережі (3D CNN). На відміну від класичних згорткових мереж, які працюють лише з двовимірними зображеннями (ширина та висота), 3D CNN опрацьовують тривимірні вхідні дані, де третім виміром виступає час. Такий підхід дозволяє одночасно аналізувати форму жесту та його динаміку, тобто не лише зафіксовану позицію руки, а й послідовність її змін протягом короткого відеофрагмента [14].

У 3D CNN згорткові фільтри додатково переміщуються уздовж часової осі, що дає змогу виявляти просторово-часові залежності у вхідних даних. Такі мережі виявляють складні патерни рухів, що можуть бути недоступні для традиційних 2D CNN. У контексті розпізнавання жестів вони дозволяють будувати моделі, здатні не лише ідентифікувати конкретну позу, а й розрізняти, яким чином та у якій

послідовності ця поза була досягнута, що суттєво покращує точність класифікації жестів [14]. Базову архітектуру 3D згорткових нейронних мереж показано на рисунку 1.4.

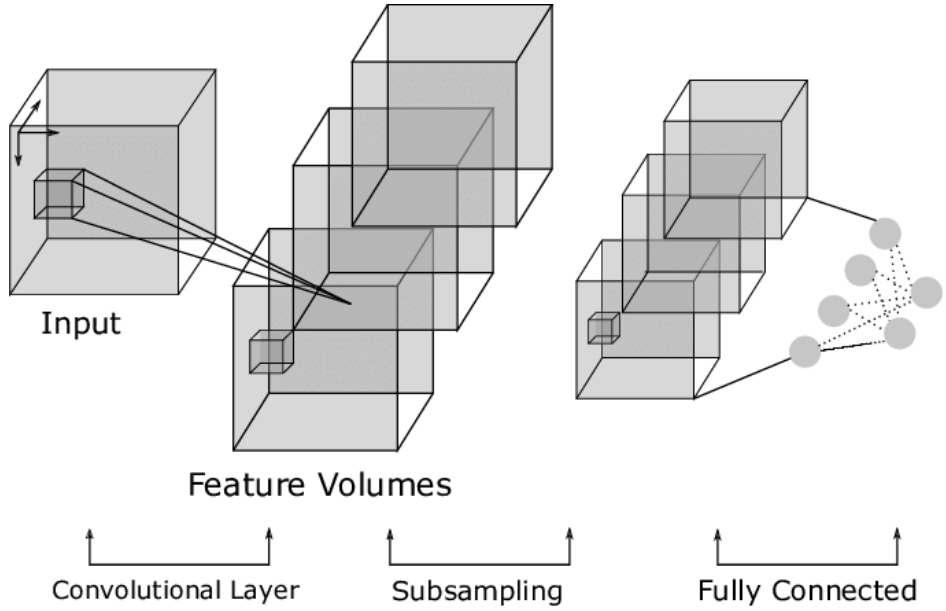

Рисунок 1.4 — Архітектура 3D CNN

Також в задачах класифікації жестів широко застосовуються глибокі нейронні мережі (DNN). Ці моделі складаються з великої кількості послідовно з’єднаних шарів з одним або кількома прихованими шарами між ними. Основна ідея DNN полягає у тому, що кожен наступний шар може вивчати все складніші абстрактні ознаки, базуючись на вихідних даних попереднього шару [13].

DNN ефективно застосовуються тоді, коли вхідні ознаки вже були попередньо оброблені або виділені іншими мережами, наприклад CNN або за допомогою алгоритмів виділення ознак. У контексті розпізнавання жестів DNN можуть використовуватися для остаточної класифікації рухів, після того як просторові та часові характеристики жесту були витягнуті згортковими або рекурентними мережами. Завдяки багатошаровій структурі, модель може поступово переходити від простих візуальних ознак до складних семантичних інтерпретацій жесту. Крім того, DNN добре підходять для обробки векторів ознак фіксованої довжини, що робить їх сумісними з різними типами попередньо обробленої інформації. Такий підхід покращує точність розпізнавання та забезпечує гнучкість моделі,

дозволяючи адаптувати її до різних форматів вхідних даних [13]. Базову архітектуру глибоких нейронних мереж показано на рисунку 1.5.

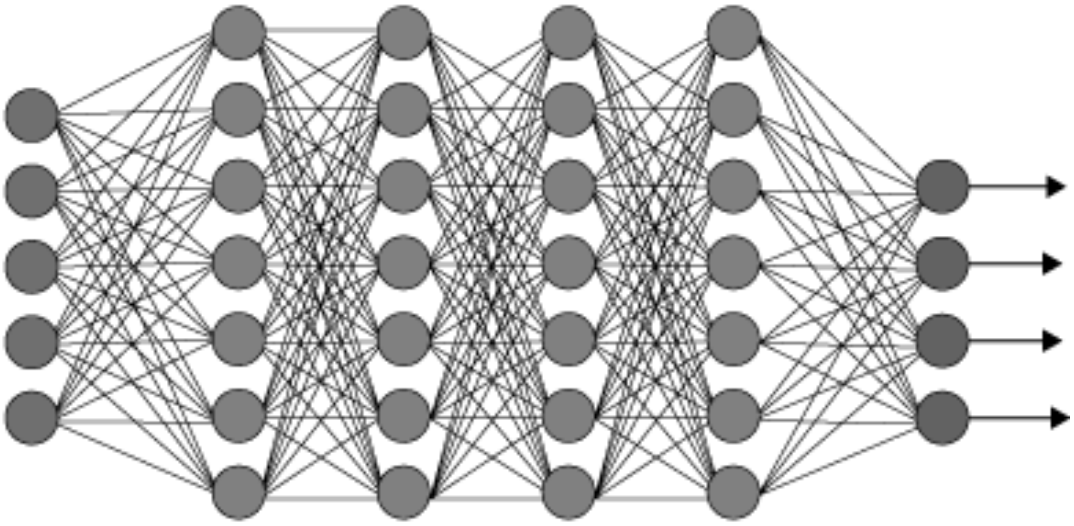

Рисунок 1.5 — Архітектура DNN

Для підвищення ефективності часто використовуються комбінації різних типів нейронних мереж, що дозволяє об’єднати їх сильні сторони та отримати високу точність розпізнавання.

1.4 Українська жестова мова: структура, особливості, нюанси в розпізнаванні

Українська жестова мова (УЖМ) – це природна візуально-жестова мовна система з власною лексико-граматичною структурою, яка використовується при спілкуванні глухих і слабочуючих людей в Україні. Вона не є похідною від усної української мови, а має власну лексичну систему, граматичні правила, синтаксис та морфологію. УЖМ є повноцінною мовою, яка має таку ж складність та виразність, як і будь-яка інша мова [15].

Основним, проте не єдиним, компонентом УЖМ є жести. Вони можуть бути одноручними або дворучними, а також поділяються на статичні і динамічні. Також важливу роль у комунікації відіграє міміка, яка допомагає передати емоції, зробити акценти та відмітити нюанси у реченнях. Для передачі власних назв, термінів, запозичених слів або тих, що не мають відповідника у жестовій лексиці, використовується пальцева абетка — дактиль, за допомогою якого кожна літера

вимовляється окремим жестом руки [15]. Українську дактильну абетку показано на рисунку 1.6.

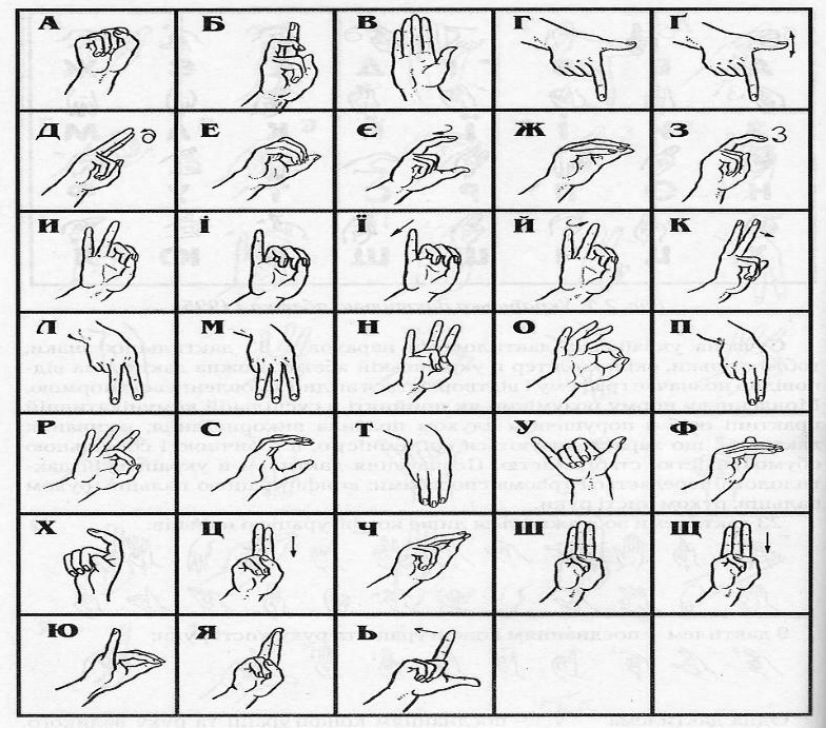

Рисунок 1.6 — Українська дактильна абетка

Українська жестова мова має свої цікаві особливості, що проявляються у власному порядку слів, відмінному від словесної української мови. Варто також зазначити відсутність відмінків та чоловічого і жіночого родів, а спільнокореневі слова можуть позначатися одним і тим самим жестом. Така структура дозволяє пришвидшити комунікацію, проте водночас ускладнює письмове спілкування для носіїв УЖМ, які з дитинства не використовують звукову мову [16].

Розпізнавання української жестової мови за допомогою технічних засобів є складним завданням, що зумовлено рядом лінгвістичних та індивідуальних факторів. Основна складність полягає в тому, що значення жестів може залежати від теми розмови, попередніх жестів або міміки. Крім того, існує велика варіативність виконання жестів, яка залежить від індивідуальних манер, регіональних діалектів та фізичних особливостей людини. Також важливо враховувати, що для більш коректного аналізу потрібно відстежувати не лише рух пальців і кистей, а й положення ліктів, плечей, нахил голови та загальну поведінку,

оскільки ці елементи часто мать додаткове або навіть ключове значення для розуміння змісту.

Така кількість важливих деталей вимагає від систем комп’ютерного зору обробки великого обсягу координат і взаємозв’язків між частинами тіла, що значно ускладнює створення точних і швидких систем розпізнавання.

1.5 Технології та інструменти для розпізнавання жестів

Розробка систем розпізнавання жестів передбачає використання різноманітних технологій і інструментів, кожен з яких має свої сильні сторони та обмеження. Вибір конкретних засобів залежить від особливостей задачі, вимог до швидкодії, складності реалізації та доступних ресурсів. Розглянемо основні інструменти, які широко застосовуються або можуть бути використані у цій сфері.

Одним з важливих інструментів для забезпечення можливості розпізнавання жестів у реальному часі є кросплатформений фреймворк MediaPipe від Google. Його архітектура оптимізована для швидкої обробки відеопотоків на різних типах пристроїв, від мобільних телефонів до вбудованих систем. Головна перевага MediaPipe полягає у наявності готових, високоефективних рішень для відстеження ключових частин тіла, зокрема розпізнавання рук, що значно прискорює розробку систем комп'ютерного зору без необхідності тривалого навчання власних моделей [17]. Попередньо навчені моделі здатні відстежувати руки навіть в умовах складного фону або часткового перекриття.

Для попередньої обробки зображень і відеопотоків часто застосовують бібліотеку OpenCV. Завдяки широкому спектру функцій, OpenCV забезпечує гнучкі можливості для покращення якості вхідного відеопотоку. Наприклад, OpenCV може використовуватися для кадрової стабілізації відео, усунення шумів, регулювання освітлення та вирізання області інтересу, що підвищує точність подальшого розпізнавання [18]. Нерідко OpenCV використовується у зв'язці з MediaPipe для оптимізації процесу виявлення та відстеження рук.

Для побудови та навчання нейронних мереж широко використовуються платформи TensorFlow та PyTorch. TensorFlow відомий своєю масштабованістю та

багатофункціональністю, дозволяючи створювати складні архітектури з підтримкою як CPU, так і GPU. PyTorch, у свою чергу, цінується за зручність у дослідженнях і гнучкість, особливо в розробці нових моделей завдяки своїй динамічній природі. Обидва фреймворки мають інтеграції з бібліотеками для комп'ютерного зору і підтримують роботу з тензорами, що дозволяє ефективно обробляти просторові та часові характеристики жестів [19].

Що стосується створення серверної частини для взаємодії з моделями розпізнавання, серед легких і гнучких рішень популярний Flask — мікрофреймворк, що дозволяє швидко розгорнути веб-додатки або API з мінімальними зусиллями. Альтернативами є Django, який пропонує більший набір готових функцій і підходить для масштабних проектів, та FastAPI — сучасний фреймворк, орієнтований на високу швидкість роботи та асинхронність, що особливо актуально для сервісів з інтенсивним обробленням даних [20].

Щодо роботи з числовими даними та багатовимірними масивами, NumPy є стандартом у Python-спільноті. Вона забезпечує ефективні операції з матрицями, що є основою для більшості алгоритмів машинного навчання. Хоча альтернативою може бути CuPy для прискорення на GPU, або JAX з підтримкою автоматичного диференціювання та виконання на GPU, NumPy лишається найпоширенішим вибором завдяки простоті і стабільності [21].

Кожен з цих інструментів відіграє важливу роль у забезпечені надійного та зручного розпізнавання жестів. Їх комбіноване використання відкриває широкі можливості для подальшого розвитку технологій розпізнавання жестової мови та їх інтеграції в різні сфери життя.

2 ВИБІР ТЕХНОЛОГІЙ ТА АРХІТЕКТУРИ СИСТЕМИ РОЗПІЗНАВАННЯ МОВИ ЖЕСТІВ

2.1 Вибір мови програмування та середовища розробки

Для реалізації системи розпізнавання мови жестів було обрано мову програмування Python.

Python — це інтерпретована, об’єктно-орієнтована мова програмування високого рівня, створена Гвідо ван Россумом та вперше випущена у 1991 році. Її популярність стрімко росте завдяки поєднанню потужних можливостей та простого, інтуїтивно зрозумілого синтаксису, що значно підвищує продуктивність розробників та знижує витрати на підтримку коду [22].

Синтаксис Python, окрім того, що він дуже схожий на англійську мову з математичним впливом, має ще деякі особливості. Замість використання фігурних дужок для визначення блоків коду, Python використовує відступи, що примушує розробників дотримуватися чіткої структури. Для завершення команди використовується символ нового рядка, а не крапка з комою, що також сприяє чистоті коду [22].

Серед вагомих переваг Python варто відзначити його кросплатформність, адже розроблені програми можуть без значних змін виконуватися на різних операційних системах. Простота синтаксису дозволяє писати програми з меншою кількістю рядків коду порівняно з багатьма іншими мовами, що прискорює розробку та зменшує потенційну кількість помилок. Завдяки інтерпретованій природі мови, код виконується одразу після написання, що робить процес прототипування надзвичайно швидким [23].

Поряд з перевагами, варто відмітити деякі недоліки Python, серед яких часто згадують, складність підтримки великих проєктів. Через динамічну типізацію одна й та сама команда може працювати по-різному залежно від контексту. Це ускладнює пошук помилок і розуміння коду, особливо якщо проєкт стрімко розширюється. Ще одним недоліком є швидкість виконання певних операцій. Гнучкість динамічної типізації вимагає додаткових дій під час виконання коду для визначення типів, що

може призводити до дещо повільнішої роботи порівняно з компільованими мовами. Проте, для багатьох завдань, особливо в галузі машинного навчання, де основне навантаження лягає на оптимізовані бібліотеки, написані на низькорівневих мовах, цей недолік не є критичним. Крім того, існують альтернативні інтерпретатори Python, такі як PyPy, які можуть значно підвищити швидкість виконання певних типів коду [22].

Python чудово поєднується зі штучним інтелектом завдяки своїй простоті. Його універсальність забезпечує охоплення усіх стадій розробки ШІ-рішень, забезпечуючи єдиний підхід від аналізу даних до створення та навчання моделей. Ключову роль відіграє наявність потужних та зручних бібліотек, таких як TensorFlow, PyTorch, OpenCV та Scikit-Learn, які надають необхідні інструменти для реалізації складних алгоритмів машинного навчання [24].

Всі ці фактори, в підсумку, стали визначальними при виборі мови програмування для розробки комп’ютерної системи розпізнавання мови жестів з використанням штучного інтелекту.

Для ефективної реалізації проєкту та комфортної роботи з Python необхідно обрати середовище розробки. Проаналізувавши можливі варіанти, було обрано PyCharm.

PyCharm — це потужне інтегроване середовище розробки (IDE), спеціально розроблене для Python компанією JetBrains, яке значно спрощує та підвищує ефективність процесу написання, налагодження та тестування коду [25]. Його популярність серед Python-розробників, особливо в галузі машинного навчання та штучного інтелекту, зумовлена широким спектром ключових особливостей та переваг.

Одним з ключових особливостей PyCharm є інтелектуальний редактор коду, який забезпечує автодоповнення, підсвічування синтаксису, перевірку помилок у реальному часі та швидку навігацію по коду. Серед вагомих переваг, також, варто виділити його потужні інструменти для налагодження. IDE дозволяє встановлювати точки зупинки, переглядати значення змінних та покроково виконувати код, що значно полегшує пошук та усунення помилок у програмі. Крім того, PyCharm тісно

інтегрований з системами контролю версій, такими як Git, що спрощує командну розробку та управління змінами в проєкті [25].

Варто згадати також про недоліки PyCharm. Одним із них є його значна вимогливість до системних ресурсів. IDE, особливо його професійна версія з розширеним набором функцій, може споживати значну кількість оперативної пам'яті та ресурсів центрального процесора. Це може призвести до зниження продуктивності та загальної швидкодії системи, особливо при роботі на комп'ютерах зі слабким апаратним забезпеченням або з великими проєктами. Також для розробників-початківців інтерфейс PyCharm може здатися перевантаженим через велику кількість доступних функцій, меню та налаштувань. Загалом PyCharm є чудовим інструментом для розробки на Python, проте, його підтримка інших мов програмування є обмеженою, тому у випадках, коли робота передбачає часте переключення між різними мовами програмування, використання окремих IDE або редакторів, краще оптимізованих для кожної конкретної мови, може бути більш ефективним [25].

Незважаючи на певні недоліки, PyCharm є надзвичайно зручним інструментом завдяки потужній підтримці Python та інтеграції з бібліотеками машинного навчання та комп'ютерного зору. Його використання робить процес створення та налагодження складних систем значно ефективнішим.

2.2 Вибір архітектури нейронної мережі

Серед різноманітних архітектур нейронних мереж, здатних обробляти послідовні дані, для ефективного розпізнавання динамічних жестів у даній роботі було обрано гібридну рекурентну нейронну мережу (RNN) з двонаправленими шарами GRU та LSTM.

Рекурентна нейронна мережа або RNN — це глибока нейронна мережа , навчена на послідовних або часових рядах даних для створення моделі машинного навчання, яка може робити прогнози або висновки на основі послідовних вхідних даних [26]. Під час прийняття рішення система враховує, як поточні вхідні дані, так і те, що вона дізналася з попередньо отриманих вхідних даних (рис. 2.1). На відміну

від CNN, які аналізують статичні зображення, RNN здатні моделювати часову динаміку та враховувати послідовність рухів.

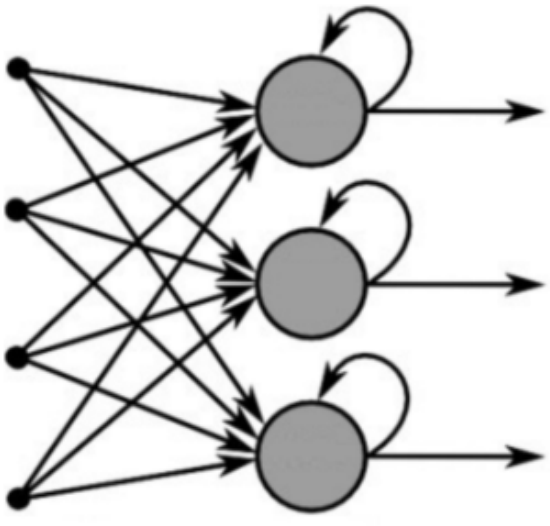

Рисунок 2.1 — Базова архітектура рекурентної нейронної мережі

LSTM (Long Short-Term Memory) є популярною архітектурою RNN, розробленою для вирішення проблеми зниклого градієнта та ефективної обробки довгострокових залежностей у послідовностях. LSTM використовує, так звані, комірки в прихованих шарах та три вентилі: вхідний, вихідний і забуття, які контролюють потік інформації, необхідної для прогнозування [26].

GRU (Gated Recurrent Unit) є схожою на LSTM архітектурою, яка також спрямована на вирішення проблеми короткочасної пам'яті в RNN, але замість стану комірки використовує приховані стани та має лише два вентилі: скидання та оновлення. Завдяки своїй простішій архітектурі, GRU є обчислювально ефективнішими [26].

Важливим уточненням є те, що використовуються саме двонаправлені шари LSTM та GRU.

Двонаправлені шари (Bidirectional Layers) в рекурентних нейронних мережах є архітектурним розширенням, що передбачає паралельне оброблення вхідної послідовності двома незалежними рекурентними шарами: один рухається вперед у часі, а інший – назад. Вихідні стани цих шарів комбінуються, надаючи моделі доступ до контексту як попередніх, так і наступних елементів послідовності. Такий

підхід використовується для задач, де розуміння контексту всієї послідовності є критично важливим [26]. У контексті розпізнавання жестів, це дозволяє враховувати як вже виконані, так і майбутні рухи для кращого розуміння поточного жесту. Загальну структуру двонаправлених рекурентних нейронних мереж показано на рисунку 2.2.

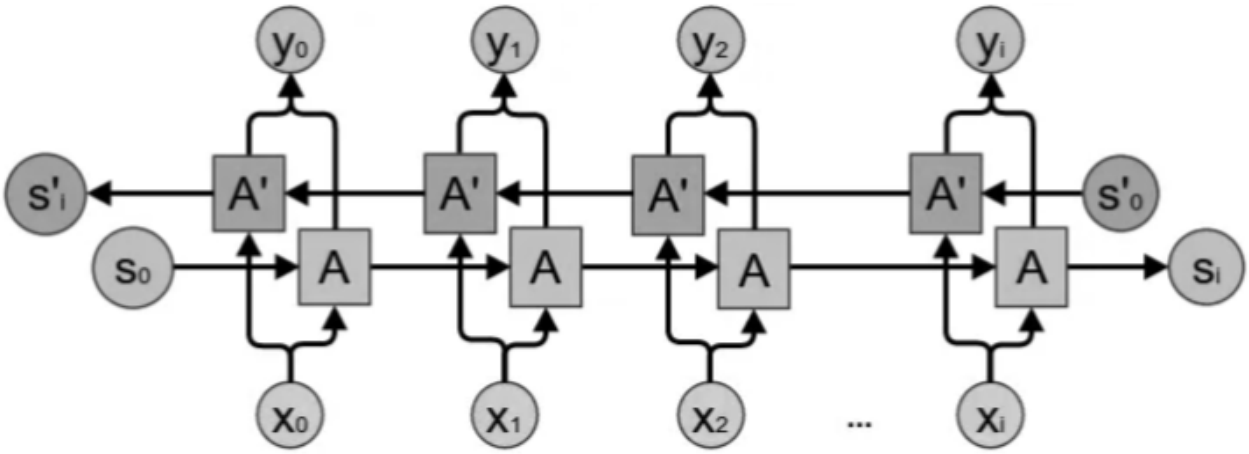

Рисунок 2.2 — Загальна структура двонаправленої рекурентної нейронної мережі

Використання гібридної рекурентної нейронної мережі з двонаправленими шарами LSTM та GRU, незважаючи на потенційну обчислювальну складність, є оптимальним рішенням, оскільки дозволяє поєднати їх переваги. LSTM відповідає за ефективне запам'ятовування та використання інформації з довготривалих часових залежностей у послідовності жестів, що дозволяє враховувати контекст, який може бути рознесений у часі. В свою чергу GRU сприяє кращому розумінню напрямку та швидкості руху рук, зокрема при зміні глибини (вперед-назад) відносно камери, аналізуючи часові зміни у відеопотоці.

Окрім рекурентних шарів, у моделі також застосовуються повнозв’язні шари (Dense).

Dense-шар — це тип шару в нейронній мережі, в якому кожен нейрон з'єднаний з кожним нейроном попереднього шару. Така щільна зв'язність дозволяє моделі враховувати всі ознаки, отримані з попередніх етапів обробки [26].

У кожному вузлі Dense-шару відбувається обчислення на основі зважених вхідних значень, а отриманий результат передається далі по мережі. Саме ці ваги

змінюються під час навчання, дозволяючи мережі виявляти складні закономірності у даних [27].

Попри те, що Dense-шари можуть використовуватись на будь-якому етапі нейронної мережі, найчастіше їх застосовують на фінальних етапах, тобто після згорткових або рекурентних шарів. Вони слугують для консолідації отриманих ознак та формування кінцевого виходу моделі, наприклад, для класифікації або регресії.

Таким чином, поєднання двонаправлених GRU і LSTM дозволяє точно моделювати часові залежності в рухах рук, а Dense-шари відповідають за прийняття класифікаційного рішення на основі всього вивченого контексту. Така архітектура забезпечує високу точність та гнучкість при розпізнаванні жестів у реальному часі.

2.3 Обґрунтування вибору бібліотек для обробки жестів

Окрім типу нейронної мережі, також було обрано ряд ключових бібліотек, кожна з яких використовується на різних етапах обробки даних та навчання моделі.

TensorFlow — це платформа з відкритим кодом для машинного навчання з використанням графів потоків даних, що спрощує створення та навчання нейромереж [19].

В даному проекті TensorFlow забезпечує гнучкий контроль над обчисленнями, ефективно використовує апаратні ресурси та реалізує основні алгоритми оптимізації, зворотного поширення помилки і роботи з тензорами. Вона служить базовим рушієм, який виконує всі внутрішні обчислення моделі.

Keras — це бібліотека нейронних мереж з відкритим кодом, яка працює поверх Theano або Tensorflow. Вона розроблена для швидкого та простого використання користувачем [19].

Keras використовується, як високорівнева надбудова над TensorFlow, що значно спрощує опис і структурування нейронної мережі. Замість ручного визначення обчислювальних графів, Keras дозволяє описати архітектуру моделі у зручному форматі через послідовне додавання шарів. Цей підхід робить процес створення та тестування моделей інтуїтивно зрозумілим і більш прозорим. У роботі

така комбінація використовується для досягнення балансу між гнучкістю та зручністю. TensorFlow виконує важкі обчислювальні задачі, тоді як Keras забезпечує швидку побудову та налаштування архітектури моделі.

OpenCV — це потужна бібліотека з відкритим кодом, розроблена для реалізації алгоритмів комп’ютерного зору та обробки зображень у реальному часі. Вона підтримує понад 2500 оптимізованих алгоритмів, зокрема для виявлення та відстеження об'єктів, розпізнавання обличь, обробки контурів, фільтрації, морфологічних операцій, конвертації кольорових просторів, а також роботи з відеопотоком. Бібліотека має багатоплатформену підтримку (Windows, Linux, macOS, Android) та забезпечує високу продуктивність, що дозволяє поєднувати ефективність із простотою у використанні. [18].

У рамках цього проєкту OpenCV використовується для зчитування відео з вебкамери, попередньої обробки зображень та передачі кадрів для подальшого аналізу жестів. Завдяки своїй здатності працювати в реальному часі та широким можливостям взаємодії з іншими бібліотеками, OpenCV стала оптимальним вибором. У порівнянні з альтернативами, такими як PIL або FFmpeg, OpenCV забезпечує повний набір інструментів саме для завдань, пов’язаних із виявленням і розпізнаванням об'єктів у кадрі в режимі реального часу.

MediaPipe — це кросплатформенний фреймворк для створення власних рішень для моделей машинного навчання, розроблений Google. Він також надає готові до використання моделі для різних задач комп'ютерного зору, включаючи відстеження рук [17].

У контексті даного проєкту фреймворк використовується для виділення 21 ключової точки кожної руки з відеопотоку з вебкамери. Ці координати надалі використовуються як вхідні дані для рекурентної нейронної мережі, що виконує класифікацію жестів. Однією з ключових переваг MediaPipe є його висока швидкість обробки кадрів, що досягається завдяки оптимізації під GPU і можливості виконання обчислень на мобільних пристроях у реальному часі. На відміну від альтернатив, таких як OpenPose, MediaPipe не потребує встановлення зовнішніх глибоких фреймворків або складного конфігурування середовища, що

спрощує інтеграцію в легкі системи. Крім того, MediaPipe забезпечує стабільні результати навіть при частковому перекритті рук або змінах освітлення, що особливо важливо для забезпечення надійності системи розпізнавання жестів у реальних умовах.

NumPy — це фундаментальна бібліотека Python для наукових обчислень, що забезпечує підтримку багатовимірних масивів та ефективних математичних операцій над ними [20].

У даному проєкті вона використовується для зберігання, обробки та стандартизації координат ключових точок, отриманих із фреймів відео, зокрема для формування вхідних послідовностей однакової довжини перед подачею їх у модель нейронної мережі. NumPy дозволяє ефективно вирівнювати всі відеопослідовності шляхом доповнення коротших серій нульовими значеннями, що критично важливо для рекурентних архітектур, таких як LSTM або GRU, які потребують фіксованої кількості кроків часу. Порівняно з Pandas, яка орієнтована переважно на табличні дані з гнучкою структурою, NumPy є більш ефективною при роботі з великими масивами чисел у вигляді фіксованих структур. Її низькорівнева реалізація на C дозволяє досягти високої швидкості обробки даних і мінімізувати споживання оперативної пам’яті, що особливо важливо при обробці десятків тисяч кадрів відеопотоку.

Flask — це легковаговий вебфреймворк на мові Python, що забезпечує реалізацію серверної логіки в клієнт-серверній архітектурі застосунків [20].

Flask дозволяє створювати вебінтерфейси, REST API та системи обробки запитів без надлишкових залежностей, зберігаючи при цьому високу гнучкість структури. У межах даного проєкту Flask виступає основою серверної частини, яка відповідає за керування потоками даних між фронтендом та модулями машинного навчання. Зокрема, за допомогою цього фреймворку реалізовано механізм приймання відеопотоку від клієнта, його обробку в режимі реального часу та передачу результатів розпізнавання жестів назад у вебінтерфейс. Flask також забезпечує інтеграцію з WebSocket-протоколом через розширення Flask-SocketIO, що дозволяє уникнути традиційної моделі запит-відповідь і досягти безперервної

двосторонньої комунікації між клієнтом і сервером, необхідної для плавного виведення результатів розпізнавання у реальному часі. Крім того, через маршрутизацію Flask координується запуск моделей машинного навчання на TensorFlow, а також сервіси передобробки відео з використанням OpenCV та MediaPipe. Завдяки своїй мінімалістичній природі, Flask не створює надлишкового навантаження на систему та легко масштабується під потреби проєкту, що критично важливо у контексті інтерактивних застосунків з високою частотою оновлення даних.

Звичайно, для системи розпізнавання жестової мови використовується значно ширший набір інструментів, проте описані вище бібліотеки є ключовими та визначають основну функціональність даної роботи.

2.4 Вибір технологій реалізації вебінтерфейсу

Для забезпечення зручної та функціональної взаємодії користувача з системою розпізнавання мови жестів важливим етапом є розробка вебінтерфейсу. В якості основних технологій для його реалізації було обрано HTML, CSS та JavaScript.

HTML (HyperText Markup Language) — це стандартна мова розмітки, що використовується для створення структури та семантичного опису вмісту вебсторінок [28].

У даному проєкті HTML відіграє ключову роль у створенні каркасу клієнтської частини застосунку. Зокрема, за її допомогою визначається просторове розміщення та взаємозв’язок між основними елементами інтерфейсу. Завдяки своїй декларативній природі, HTML забезпечує логічну ієрархію представлення даних на сторінці та спрощує інтеграцію зі скриптовими і стилізуючими мовами. Високий рівень сумісності HTML з усіма сучасними браузерами, а також його універсальність у поєднанні з іншими вебтехнологіями, робить його незамінним компонентом при розробці інтерфейсів для систем реального часу, що базуються на клієнт-серверній архітектурі.

CSS (Cascading Style Sheets) — це спеціальна мова стилів, що

використовується для опису зовнішнього вигляду сторінок, написаних мовами розмітки даних. CSS дозволяє контролювати кольори, шрифти, розміри, анімації та інші візуальні аспекти вебсторінки [28].

CSS відповідає за візуальне оформлення вебінтерфейсу, включаючи стилізацію інтерактивних компонентів. Завдяки механізмам каскадності та спадковості, CSS забезпечує ефективне застосування єдиних стилістичних шаблонів до численних елементів, що суттєво спрощує підтримку й модифікацію інтерфейсу. Крім того, мова дозволяє реалізувати адаптивний дизайн, що автоматично підлаштовується під розміри та орієнтацію екранів різних пристроїв. Використання CSS є ключовим для досягнення інтуїтивної зручності, естетичної привабливості та логічної структури вебсторінки.

JavaScript — це мова програмування високого рівня, яка використовується для додання інтерактивності та динамічної поведінки вебсторінкам [28].

У межах даного проєкту JavaScript відіграє ключову роль в обробці подій, керуванні захопленням і потоковою передачею відео з вебкамери, а також у реалізації механізмів асинхронного обміну даними з сервером без необхідності повного оновлення сторінки. Завдяки підтримці DOM-маніпуляцій, мова дозволяє динамічно змінювати вміст сторінки відповідно до результатів роботи моделі чи дій користувача. Сучасні браузери забезпечують повну сумісність із JavaScript, що гарантує стабільну роботу інтерфейсу на різних платформах. Крім того, розвинена екосистема мови значно пришвидшує розробку інтерактивних вебзастосунків, зокрема таких, що потребують обробки відеопотоку в режимі реального часу або взаємодії зі сторонніми API. Таким чином, JavaScript виступає критичним компонентом клієнтської частини даної інформаційної системи, забезпечуючи її динамічність, інтерактивність та швидкодію.

На основі обраних мов програмування розроблено надійний вебінтерфейс, що забезпечує зручну взаємодію користувача з системою розпізнавання мови жестів.

3 РОЗРОБКА ТА НАВЧАННЯ НЕЙРОННОЇ МЕРЕЖІ

3.1 Формування власного датасету для навчання нейромережі

Набір даних (dataset) — це структурований набір прикладів, які можуть бути представлені в різних форматах, на основі яких модель здатна навчитися розпізнавати певні закономірності або об’єкти [29].

Для ефективного розпізнавання жестової мови набір даних відіграє ключову роль, оскільки саме від його якості та обсягу залежить кінцева точність розробленої моделі. Існує велика кількість загальнодоступних датасетів жестових мов, проте більшість з них зосереджені на американському або міжнародному варіантах, тоді як ресурси для української жестової мови практично відсутні. Крім того, існуючі датасети зазвичай мають недостатню кількість зразків, обмежену якість або не охоплюють всі потрібні жести, що робить їх непридатними для повноцінного навчання глибоких нейронних мереж. Враховуючи ці обставини, було прийнято рішення про створення власного датасету, що надає повний контроль над процесом збору та анотації даних, дозволяючи врахувати всі особливості розроблюваної моделі та забезпечити необхідний обсяг і якість навчального матеріалу, адаптованого до української жестової мови.

На етапі вибору формату даних для навчання моделі розпізнавання жестів розглядалися два основні варіанти: окремі статичні зображення та відео. Обидва варіанти мають свої переваги та недоліки, проте вибір було зроблено на користь відеоформату, з огляду на кілька важливих аспектів.

Перш за все, жестова мова є динамічною системою комунікації, у якій значення передається не лише формою руки або положенням пальців у певний момент, а й послідовністю, швидкістю та траєкторією рухів у часі. Тобто, для правильного розпізнавання важливо враховувати контекстний часовий вимір, що неможливо забезпечити при роботі лише зі статичними зображеннями.

Також збір даних у вигляді відео є значно зручнішим та ефективнішим з практичної точки зору. Замість створення великої кількості окремих фотографій, достатньо записати короткий відеофрагмент, де демонструється жест, і система

автоматично зафіксує послідовність кадрів. Це не лише прискорює процес збору даних, а й зменшує ймовірність людських помилок та неточностей під час ручного фотографування.

Ще однією важливою перевагою вибору відеоформату для збору даних є можливість застосування аугментації — автоматичного розширення навчального набору шляхом генерації різноманітних варіацій на основі наявних відеозаписів. Цей процес передбачає створення нових прикладів жестів шляхом модифікації оригінальних відео, що суттєво підвищує різноманітність даних без необхідності збирати додаткові записи вручну. Наприклад, один короткий відеофрагмент із жестом можна трансформувати, змінюючи такі параметри, як фон, умови освітлення, кут зйомки або положення руки в просторі. Це дозволяє моделі навчатися на більш широкому спектрі варіацій одного й того самого жесту, що підвищує її здатність коректно розпізнавати жести в реальних умовах.

Таким чином, для успішної реалізації проєкту необхідно розробити зручний, інтуїтивно зрозумілий та функціональний графічний інтерфейс користувача. Він має забезпечити швидкий і стандартизований процес збору відеоданих, що дозволить користувачам без зайвих складнощів записувати необхідні жести. Такий підхід значно спростить подальші етапи навчання та тестування нейронної мережі, а також зробить процес збору даних більш організованим і продуктивним.

3.1.1 Розробка графічного інтерфейсу для створення набору даних

З метою покращення та автоматизації процесу збору відеоданих було реалізовано вебінтерфейс, що дозволяє швидко записувати відеофрагменти з жестами української жестової мови. Такий підхід мінімізує допущення помилки в анотації даних та забезпечує структуровану підготовку набору кадрів із чітким маркуванням.

Графічний інтерфейс, призначений для збору навчального матеріалу та подальшого навчання нейронної мережі, було реалізовано у вигляді інтерактивної вебсторінки. Для цього використано стандартні вебтехнології — HTML, CSS та JavaScript, що забезпечують кросплатформеність і широке охоплення серед

користувачів.

HTML відповідає за побудову логічної структури сторінки. Саме за допомогою HTML були створені базові елементи інтерфейсу: текстові поля для введення даних, вікна для виведення інформації, інтерактивні кнопки для керування процесами, а також контейнери для відображення відеопотоку з вебкамери.

CSS використовується для оформлення зовнішнього вигляду сторінки, забезпечуючи привабливий, сучасний і візуально зручний інтерфейс. За допомогою каскадних таблиць стилів реалізовано адаптивний дизайн, що дозволяє коректно відображати інтерфейс на пристроях з різними розмірами екранів, від мобільних телефонів до великих моніторів. Також через CSS налаштовано стилізацію кнопок, шрифтів, кольорової гами, відступів та інших елементів, що сприяє кращому користувацькому досвіду.

JavaScript виступає основною мовою для реалізації логіки функціонування вебінтерфейсу. Зокрема, саме JavaScript забезпечує: доступ до вебкамери користувача, виведення відеопотоку на сторінку, інтеграцію з бібліотекою MediaPipe, обробку подій користувача та анотацію зібраних даних.

Для виявлення положення рук у відеопотоці використовується бібліотека MediaPipe Hands, розроблена компанією Google. Вона забезпечує високу точність трекінгу положення рук навіть за наявності часткового перекриття або змін освітлення.

MediaPipe Hands виконує повноцінну обробку відеопотоку у режимі реального часу, автоматично виявляючи на зображенні одну або дві руки та будуючи скелетну модель кожної з них. Основною особливістю цієї бібліотеки є те, що вона визначає 21 ключову точку для кожної руки. Така кількість ключових точок розподіляються на кінчики пальців, центр долоні та суглоби між фалангами (рис. 3.1). Кожна з цих точок представлена координатами x, y та z, де x і y визначають положення на двовимірній площині кадру, а z вказує на відносну глибину, тобто визначається положення точки у тривимірному просторі. Під час запису навчального матеріалу ці координати дозволяють користувачу бачити,

наскільки точно система сприймає показаний жест і за потреби, повторити його для досягнення вищої якості навчального датасету.

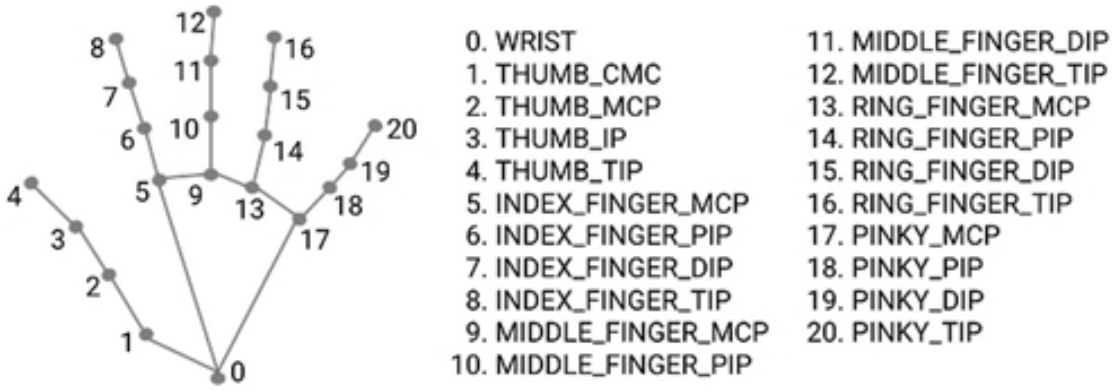

Рисунок 3.1 — Розміщення ключових точок кисті руки

На рисунку 3.2 показано загальний вигляд розробленого інтерфейсу збору даних для навчання нейромережі.

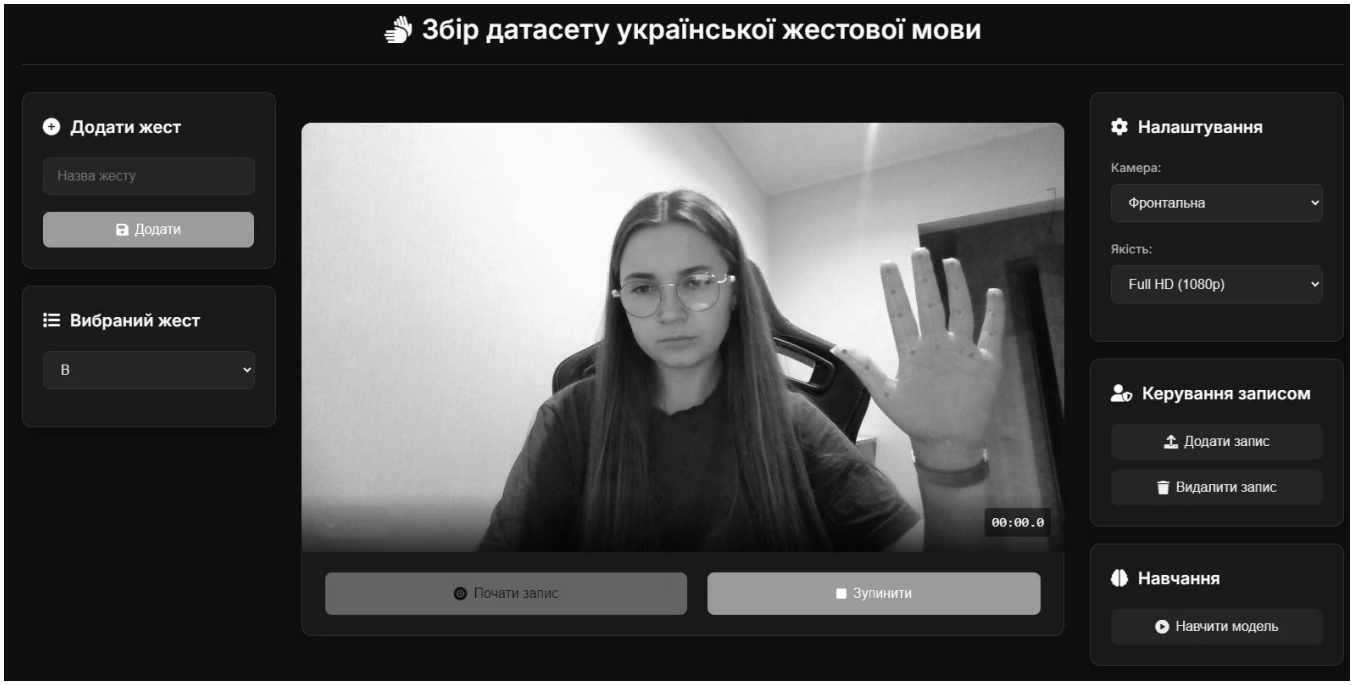

Рисунок 3.2 — Загальний вигляд інтерфейсу збору даних

Інтерфейс складається з трьох основних блоків: лівого, центрального та правого.

Лівий блок призначений для введення назв жестів, які будуть використовуватись під час запису відео для подальшого навчання моделі.

Користувач може створити нову категорію жесту, ввівши її назву в текстове поле розділу «Додати жест» та натиснувши кнопку «Додати». Після цього випадаючий список у блоці «Вибраний жест» автоматично оновлюється, і в ньому з’являється щойно додана назва (рис. 3.3).

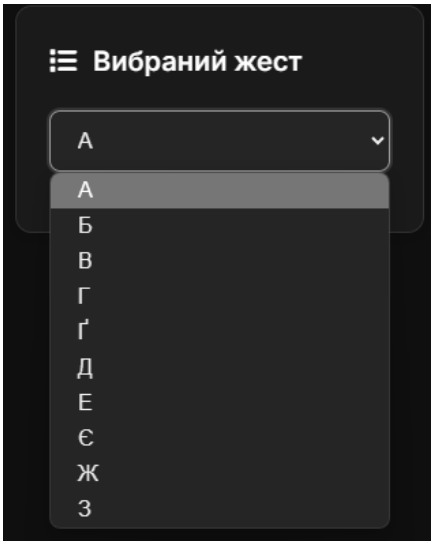

Рисунок 3.3 — Випадаючий список назв жестів

Центральний блок є основним і у ньому відображається відеопотік з вебкамери, яка автоматично активується під час завантаження сторінки. У реальному часі на відео накладаються ключові точки долоні, що дозволяє контролювати положення рук та пальців при записі. Нижче розташовані дві кнопки для керування записом. Кнопка «Почати запис» запускає процес фіксації кадрів, а кнопка «Зупинити» — завершує процес запису. Додатково у цьому блоці відображається статус запису та тривалість поточного або останнього запису у форматі таймера.

Правий блок містить корисну додаткову функціональність. У секції «Налаштування» користувач може обрати якість відеозапису (HD або Full HD) та активну камеру (фронтальну або задню), що є актуальним при записі зі смартфона. Секція «Керування записом» містить кнопку «Додати запис», яка відкриває файловий провідник і дозволяє завантажити попередньо записане відео та кнопку «Видалити запис», яка дозволяє скасувати збереження останнього записаного відео у датасет, що є корисним при допущені помилки під час запису. За умови успішного

видалення запису, користувач отримує сповіщення, як показано на рисунку 3.4.

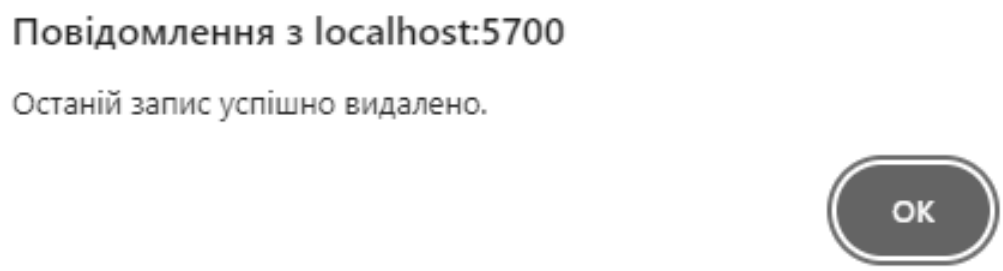

Рисунок 3.4 — Повідомлення після видалення останнього запису

У разі, якщо не вдалося видалити запис — виводиться повідомлення про помилку.

В нижній частині правого блоку інтерфейсу знаходиться секція «Навчання», яка містить кнопку «Навчити модель». Ця кнопка взаємодіє з серверною частиною і викликає функцію для навчання моделі.

Таким чином інтерфейс забезпечує повний контроль над процесом збору даних та можливість швидко розширювати датасет.

3.1.2 Анотація та структуризація даних

Після запису відео, важливим етапом є їх анотація та структуризація для подальшого використання в системі розпізнавання жестів. Основна мета цього процесу полягає у впорядкуванні відеофайлів відповідно до класів жестів, які вони представляють, а також у забезпеченні можливості автоматизованої обробки цих даних під час навчання нейронної мережі.

Під час запуску програми автоматично перевіряється наявність спеціальної директорії з назвою «dataset», яка є основним сховищем для всіх відеозаписів, що використовуються у процесі навчання нейронної мережі. Якщо така папка ще не існує, вона створюється програмно. Це дозволяє одразу підготувати структуру файлової системи до зберігання великого обсягу навчальних даних, без потреби втручання користувача.

У момент додавання нового жесту через графічний інтерфейс користувача програма створює вкладену папку, ім’я якої відповідає назві жесту, що була вказана

користувачем. Таким чином, кожен жест має окрему директорію, що дозволяє логічно структурувати дані за категоріями та спрощує подальшу обробку. Якщо папка з відповідною назвою вже існує, вона не створюється повторно, а нові відеозаписи автоматично додаються до неї.

Для уникнення дублювання назв файлів використовується мітка поточного часу, з точністю до секунди (рис. 3.5). Це забезпечує унікальність імен файлів, навіть якщо користувач виконує запис кілька разів поспіль у дуже короткі проміжки часу. Такий підхід гарантує, що кожен файл буде збережений окремо, не пошкоджуючи попередні, і забезпечить цілісність навчального набору.

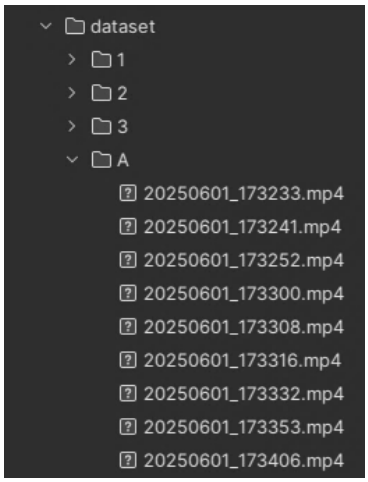

Рисунок 3.5 — Структуризація набору даних

Кожен відеофайл у датасеті містить запис одного повного виконання відповідного жесту. Такий підхід до структурування забезпечує зручну навігацію по датасету та спрощує процес зчитування даних для подальшого аналізу та навчання моделі. Крім того, організація даних за класами у вигляді окремих директорій є стандартною практикою, що підтримується більшістю сучасних бібліотек глибокого навчання. Таким чином, розміщення відеофайлів у папках із відповідними назвами вважається достатньою анотацією класу.

3.2 Підготовка набору даних для навчання нейромережі

Процес побудови системи розпізнавання жестів потребує наявності не лише структурованого, але й однорідного набору даних, тому перед навчанням

нейромережі необхідно виконати деякі етапи обробки.

3.2.1 Попередня обробка відео

Після структуризації даних необхідно перетворити відеофайли у формат, придатний для обробки нейронною мережею.

Відео покадрово зчитується за допомогою бібліотеки OpenCV. Для кожного кадру виконується перевірка на наявність руки за допомогою бібліотеки MediaPipe Hands. Якщо у кадрі ключові точки не виявлено, кадр замінюється вектором з нулями, що дозволяє зберігати однакову довжину вхідної послідовності незалежно від якості розпізнавання. Такий підхід дозволяє уникати втрати синхронізації під час навчання моделі.

Для забезпечення однорідності вхідних даних застосовується етап зміни розміру кадру. Для кожного кадру визначається його ширина та висота та обчислюється коефіцієнт масштабування. Коефіцієнт масштабування використовується для визначення, наскільки потрібно змінити розмір кадру відео, щоб більша сторона кадру після масштабування дорівнювала заданому target_size (640 пікселів). Обчислення коефіцієнта масштабування наведено у лістингу 3.1.

Лістинг 3.1 — Обчислення коефіцієнта масштабування

```
target_size = 640
scale = target_size / max(frame.shape[0], frame.shape[1])
if scale < 1:
        frame = cv2.resize(frame, (0, 0), fx=scale, fy=scale)
```

Якщо значення коефіцієнта більше або дорівнює 1, це означає, що оригінальний кадр менший або дорівнює target_size по обох осях, тому масштабування не відбувається. Якщо ж коефіцієнт менший за 1, то розмір кадру зменшується шляхом множення оригінальної ширини та висоти на цей коефіцієнт. При зміні розміру кадру зберігаються його оригінальні пропорції сторін.

Оскільки бібліотека MediaPipe, яка використовується для виявлення ключових точок руки, функціонує в колірному просторі RGB, перед передачею

кадру на обробку необхідно виконати його попереднє перетворення. Це зумовлено тим, що бібліотека OpenCV, за допомогою якої здійснюється зчитування зображень із відеопотоку, працює за замовчуванням у колірному просторі BGR. Таким чином, кожен кадр, отриманий із вебкамери, потребує конвертації у формат, з яким коректно працює MediaPipe. Це здійснюється за допомогою функції cv2.cvtColor(), яка приймає в якості параметрів зображення та код перетворення колірного простору.

Варто також зазначити, що MediaPipe Hands містить низку внутрішніх алгоритмів оптимізації: фільтрацію шумів, стабілізацію трекінгу, вирівнювання світлових умов тощо. Завдяки цьому підвищується точність виявлення контурів рук і їхніх ключових точок у реальному часі, навіть якщо відео знімається в динамічному середовищі або при неідеальному освітленні.

3.2.2 Зчитування координат ключових точок рук

Для обчислення координат основних точок руки використовується бібліотека MediaPipe та модуль Hands, який реалізує детекцію рук у кадрі та побудову 21 тривимірної точки для кожної виявленої руки. Ці точки формують скелет руки, починаючи від зап’ястя й закінчуючи кінчиками пальців (див. рис. 3.1).

Координати кожної точки визначаються по трьох площинах — x, y та z, тому кожен кадр однієї руки містить 63 значення.

Формування вхідних даних здійснюється у функції extract_keypoints_from_video() (лістинг 3.2). Її основне завдання полягає у зчитувані ключових точок з відеофайлу та підготовці їх у форматі, придатному для подачі на вхід моделі. На першому етапі функція відкриває відео за допомогою cv2.VideoCapture(), після чого поетапно зчитує кадри та обробляє їх. Аби забезпечити сталу довжину вхідної послідовності, незалежно від тривалості відео, аналізується не більше 30 кадрів. Якщо відео містить менше кадрів, решта позицій у масиві заповнюється нульовими векторами, що імітують відсутні жести. Це дозволяє зберігати однаковий розмір масиву ключових точок для кожного відео, що є важливою вимогою для стабільної роботи нейронної мережі.

Лістинг 3.2 — Формування вхідних даних

```
def extract_keypoints_from_video(video_path, max_frames=30):
    cap = cv2.VideoCapture(video_path)
    frame_width = int(cap.get(cv2.CAP_PROP_FRAME_WIDTH))
    frame_height = int(cap.get(cv2.CAP_PROP_FRAME_HEIGHT))
        raw_keypoints = []
        while len(raw_keypoints) < max_frames:
            ret, frame = cap.read()
            if not ret:
                break
            keypoints = process_frame(frame, hands)
            raw_keypoints.append(keypoints)
    cap.release()
    while len(raw_keypoints) < max_frames:
        raw_keypoints.append(np.zeros(21 * 3 * 2))
    keypoints = np.array(raw_keypoints[:max_frames])
    keypoints = normalize_keypoints(keypoints, frame_width, frame_height)
    keypoints = smooth_keypoints(keypoints)
    return keypoints
```

Функція process_frame() (лістинг 3.3) відповідає за обробку одного кадру. Вона викликає метод hands.process() для обробки зображення та витягує координати ключових точок обох рук. Координати подаються у вигляді списку з 63 значень для кожної руки, а якщо руку не виявлено — значення заповнюються нулями. Такий підхід дозволяє зберігати цілісність послідовності та уникнути зміщення кадрів при подальшій обробці.

Лістинг 3.3 — Визначення координат для кадру

```
def process_frame(frame, hands):
    try:
        frame_rgb = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
        results = hands.process(frame_rgb)
        lh = np.zeros(21 * 3)
        rh = np.zeros(21 * 3)
        if results.left_hand_landmarks:
            lh = np.array([[lm.x, lm.y, lm.z] for lm in
results.left_hand_landmarks.landmark]).flatten()
        if results.right_hand_landmarks:
            rh = np.array([[lm.x, lm.y, lm.z] for lm in
```

```
results.right_hand_landmarks.landmark]).flatten()
        return np.concatenate([lh, rh])
```

Після виконання функції process_frame(), отримані вектори записуються до списку raw_keypoints та відеофайл закривається. Якщо кількість оброблених кадрів менша за необхідну, список доповнюється векторами з нульовими значеннями, допоки довжина списку не буде дорівнювати 126-ти елементам. Далі список перетворюється у NumPy-масив, який проходить етап нормалізації відносно розміру кадру та згладжування з метою зменшення шуму. В результаті функція повертає масив розміром 30 на 126, що представляє послідовність координат ключових точок кистей обох рук для всього відеофрагмента.

Процес збереження координат у файли відбувається у функції save_data() (лістинг 3.4). Вона приймає на вхід повний набір даних X, вектор міток Y та словник labels, що відображає відповідність числових класів реальним назвам жестів. Координати зберігаються у спеціальну папку «coordinates», а labels у папку «result».

Лістинг 3.4 — Збереження координат

```
def save_data(X, Y, labels):
    np.save(os.path.join(OUTPUT_DIR, 'X.npy'), X)
    np.save(os.path.join(OUTPUT_DIR, 'Y.npy'), y)
    np.save('../result/labels.npy', labels)
```

Після збереження у відповідній папці з’являється два файли: X.npy та Y.npy, які містять масиви координат та відповідних їм міток жестів (рис. 3.6).

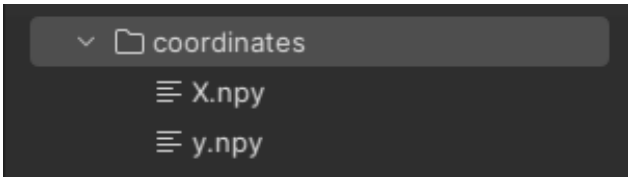

Рисунок 3.6 — Збережені файли з визначеними координатами

Набір даних повністю готовий до використання в навчанні моделі розпізнавання мови жестів.

3.3 Навчання нейронної мережі

3.3.1 Архітектура моделі

Для навчання моделі використовується послідовна архітектура рекурентної нейронної мережі, побудована з використанням високорівневої бібліотеки Keras, яка функціонує поверх фреймворку TensorFlow. Архітектура розробленої моделі включає кілька шарів, кожен з яких виконує певну функцію в процесі аналізу послідовностей жестів (лістинг 3.5).

Лістинг 3.5 — Архітектура моделі

```
def build_model(input_shape, num_classes):
    model = Sequential([
        Bidirectional(GRU(256, return_sequences=True), input_shape=input_shape),
        BatchNormalization(),
        Dropout(0.1),
        Bidirectional(LSTM(256)),
        BatchNormalization(),
        Dropout(0.1),
        Dense(128, activation='relu'),
        Dense(64, activation='relu'),
        Dense(num_classes, activation='softmax')
    ])
```

Першим елементом цієї моделі є двонаправлений рекурентний шар типу GRU з кількістю нейронів 256. Така структура дозволяє аналізувати як попередній, так і наступний контекст руху, що особливо важливо для правильного трактування динамічних змін у послідовності координат жесту.

Після шару GRU додається шар пакетної нормалізації, який нормалізує вихідні значення з попереднього шару, зменшуючи вплив зміни розподілу активацій і тим самим стабілізуючи процес навчання. Це сприяє швидшій збіжності моделі.

Далі застосовується шар Dropout з коефіцієнтом 0.1, який слугує для регуляризації та запобігання перенавчанню. Dropout випадковим чином вимикає певну частину нейронів у процесі тренування, що змушує модель не покладатися на окремі ознаки, а шукати загальні закономірності.

Після цього розташовано ще один двонаправлений рекурентний шар, цього разу типу LSTM з тією ж кількістю нейронів — 256. Завдяки своїй внутрішній структурі цей шар здатен краще зберігати довготривалі залежності в послідовностях і поглиблювати аналіз часових зв’язків між ключовими точками.

Як і після першого рекурентного шару, тут знову застосовуються пакетна нормалізація та Dropout, що забезпечує стабільну роботу моделі навіть за наявності значної варіативності в даних.

Далі слідують два повнозв’язні шари з 128 і 64 нейронами відповідно. Вони виконують функцію остаточної обробки ознак, які були виділені на попередніх рівнях мережі. Застосування функції активації ReLU на цих шарах дозволяє моделі ефективно моделювати складні залежності між ознаками, що сприяє точному розмежуванню жестів різних класів.

Завершальним елементом є вихідний Dense-шар із кількістю нейронів, що дорівнює кількості класів, які має розпізнавати модель. У цьому шарі використовується функція активації softmax, яка трансформує результати моделі у вектор ймовірностей. Кожен елемент цього вектора вказує на ймовірність того, що вхідні дані належать до певного класу жестів, що дозволяє здійснювати остаточну класифікацію.

3.3.2 Процес навчання моделі

Навчання моделі здійснюється у відповідь на натискання кнопки «Навчити модель» у вебінтерфейсі користувача. Після натискання формується запит, який передається на серверну частину додатку, де обробляється за допомогою функцій process_dataset() та training_progress_stream(), реалізацію яких наведено у лістингу Ґ.1.

Функція process_dataset() відповідає за підготовку вхідних даних: зчитування відеофайлів, витягування ключових точок за допомогою алгоритму MediaPipe Hands, формування масивів фіксованої довжини та розподіл даних на навчальну й тестову вибірки. Після цього запускається процес тренування моделі, під час якого викликається генератор training_progress_stream(). Його завдання — поступово

надсилати на фронтенд оновлення про поточний етап навчання, втрати, точність тощо.

Для зручного відстеження процесу навчання користувачу надається спеціальне інформаційне вікно, показане на рисунку 3.7.

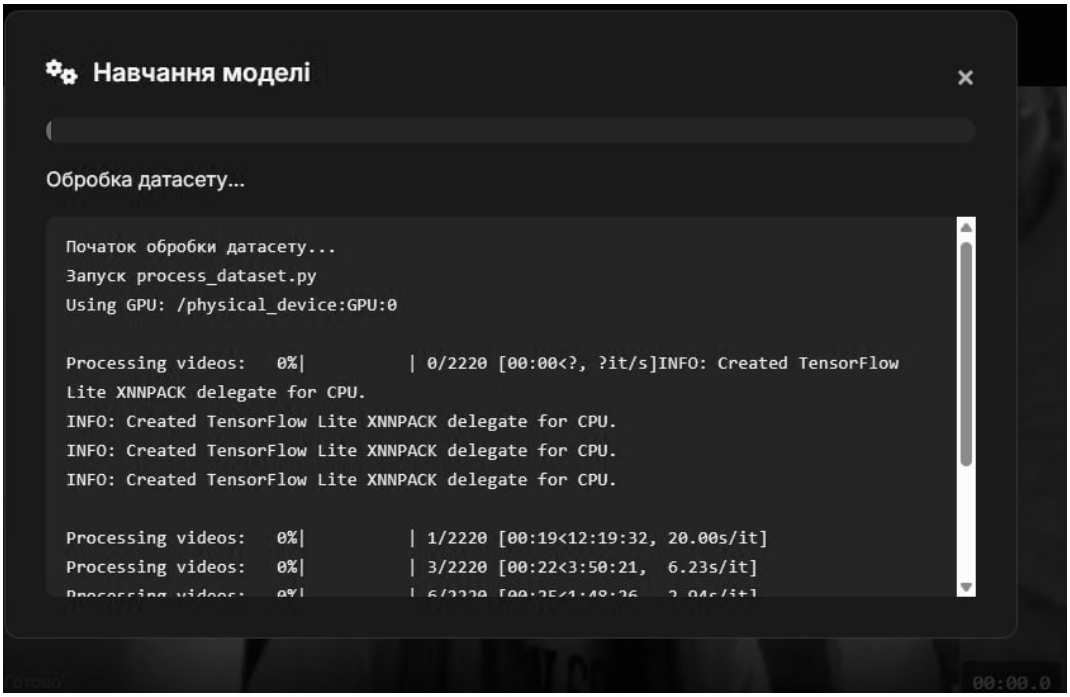

Рисунок 3.7 — Інформаційне вікно навчання моделі

Основна архітектура моделі побудована за допомогою бібліотеки Keras. На вхід подається послідовність координат, отриманих з відео, для подальшої класифікації на відповідний жест. Компіляція моделі відбувається з використанням функції втрат categorical_crossentropy, яка є стандартною для багатокласової класифікації при застосуванні one-hot кодування міток. Оптимізація параметрів моделі здійснюється за допомогою оптимізатора Adam, він адаптує швидкість навчання окремо для кожного параметра, що сприяє швидкій та стабільній збіжності моделі. Якість навчання оцінюється за допомогою метрики точність, яка відображає відсоток правильно класифікованих жестів.

З метою покращення процесу навчання та запобігання перенавчанню застосовуються спеціальні інструменти — колбеки. Колбек EarlyStopping контролює значення функції втрат на валідаційній вибірці та зупиняє навчання,

якщо протягом 10 послідовних епох не спостерігається її покращення. Це дозволяє уникнути перенавчання моделі на тренувальних даних. Колбек ReduceLROnPlateau динамічно зменшує коефіцієнт навчання, якщо протягом 5 епох валідаційна втрата не покращується. Це може допомогти моделі вийти з плато та знайти більш оптимальні значення параметрів. Колбек ModelCheckpoint відповідає за збереження найкращої версії моделі на основі мінімального значення функції втрат на валідаційній вибірці, забезпечуючи збереження найбільш ефективних ваг.

З метою підвищення точності та стійкості системи застосовується ансамбль з п’яти ідентичних моделей, які навчаються незалежно на одних і тих самих даних. Під час прогнозування результати п’яти моделей усереднюються, що зменшує вплив випадкових помилок та підвищує точність.

Важливо відзначити, що для досягнення якісного навчання моделі необхідно забезпечити достатню кількість прикладів для кожного класу жестів. У ході експериментального дослідження було встановлено, що для кожного жесту бажано мати щонайменше 60 відеофрагментів. Такий обсяг даних дозволяє моделі краще узагальнювати інформацію та зменшує ймовірність переадаптації до окремих прикладів.

На рисунку 3.8 показано процес створення ансамблю з п’яти моделей та тренування першої з них.

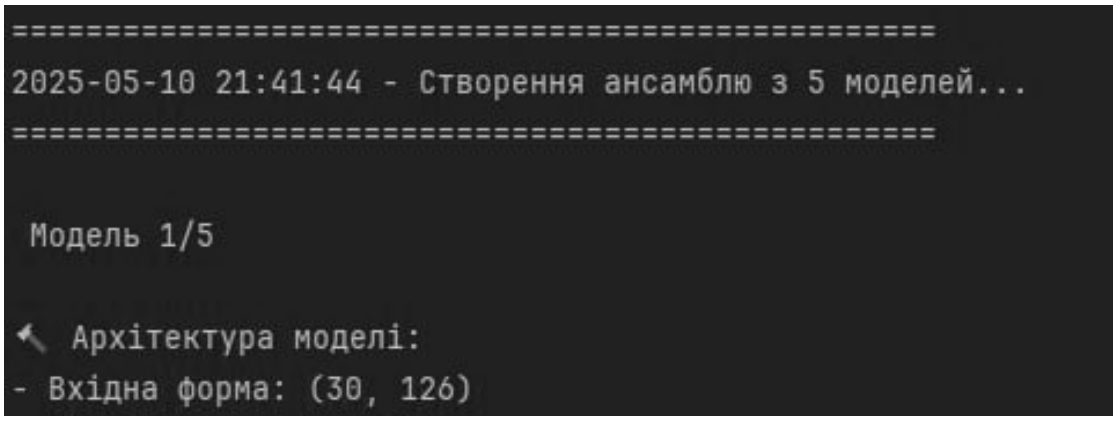

Рисунок 3.8 — Процес навчання ансамблю моделей

На рисунку 3.9 показано фрагмент процесу навчання однієї з моделей ансамблю. Зафіксовано, що значення функції втрат на валідаційній вибірці не

покращилося порівняно з попереднім найкращим значенням. Також наведено значення функції втрат та точності на навчальній вибірці, а також значення функції втрат та точності на валідаційній вибірці за цю епоху. Ключовим елементом рисунка є повідомлення про застосування колбеку ранньої зупинки**,** що свідчить про те, що навчання моделі було автоматично зупинено через відсутність покращення на валідаційній вибірці протягом заданої кількості епох. Це демонструє механізм запобігання перенавчанню в дії.

Рисунок 3.9 — Фрагмент процесу навчання моделі

Таким чином, процес навчання побудований з урахуванням найкращих практик глибинного навчання. Запровадження ансамблю з кількох ідентичних моделей, що функціонують незалежно, додатково підвищує надійність розпізнавання за рахунок зменшення впливу випадкових шумів або локальних аномалій у даних. Такий підхід забезпечує більш стійке та точне розпізнавання жестів навіть у складних або неоднозначних сценаріях.

3.4 Аналіз отриманих результатів

Після завершення навчання ансамблю з п'яти нейронних мереж важливо провести оцінку його ефективності на незалежній тестовій вибірці. Метою цього етапу є визначення здатності навченої системи до узагальнення знань на нових, раніше не бачених даних.

На рисунку Г.1 представлено графіки точності та функції втрат для кожної з п'яти моделей ансамблю протягом усього процесу навчання.

Аналізуючи графіки точності, можна спостерігати тенденцію до зростання цього показника як на навчальній, так і на валідаційній вибірках для кожної з моделей. Це свідчить про те, що моделі успішно засвоюють основні закономірності та ефективно навчаються розпізнавати жести. У деяких випадках на графіках

простежуються незначні коливання в точності або втратах на валідаційній вибірці. Такі коливання є типовими при роботі з реальними даними і не мають суттєвого негативного впливу на загальну якість моделі, оскільки компенсуються використанням ансамблю з декількох моделей. Графіки функції втрат підтверджують зниження значень втрат у ході навчання, що вказує на успішну оптимізацію ваг мережі. Зменшення функції втрат на валідаційній вибірці свідчить про збереження здатності моделі до узагальнення без ознак перенавчання. Таким чином, ансамбль моделей демонструє позитивну динаміку навчання та здатність до ефективного розпізнавання жестів.

Точність обробки датасету можна побачити за допомогою матриці плутанини, яка показана на рисунку Г.2.

Матриця плутанини відображає кількість фактичних прикладів для кожного класу жестів та кількість прикладів, які були спрогнозовані моделлю для кожного класу. Значення на головній діагоналі показують кількість правильно класифікованих прикладів, тоді як позадіагональні елементи вказують на кількість помилкових передбачень.

Аналізуючи матрицю плутанини, можна виявити, що для більшості жестів досягнуто високої точності класифікації, про що свідчать значення на відповідних діагональних елементах та відсутність або незначна кількість помилкових передбачень для інших класів. Помилки класифікації є поодинокими та не мають системного характеру. Їхня поява здебільшого обумовлена схожими конфігураціями жестів або варіативністю виконання рухів у різних прикладах, однак вони становлять лише незначний відсоток від загальної кількості передбачень.

Отримані результати свідчать про високу точність та надійність розробленої системи розпізнавання жестів. Мінімальна кількість помилок підтверджує ефективність використаного підходу до навчання моделі, а також достатню якість і репрезентативність підготовленого датасету.

4 РОЗРОБКА КОМП'ЮТЕРНОЇ СИСТЕМИ РОЗПІЗНАВАННЯ МОВИ ЖЕСТІВ НА ОСНОВІ НЕЙРОННОЇ МЕРЕЖІ ТА ЇЇ ТЕСТУВАННЯ

4.1 Розробка структурної схеми комп'ютерної системи

Розробка комп'ютерної системи розпізнавання жестової мови потребує чіткого розуміння її архітектури, складових елементів та взаємозв'язку між ними. Архітектура комп'ютерної системи включає 2 функціональних рівні: рівень обробки даних та рівень пристроїв, як показано на рисунку 4.1

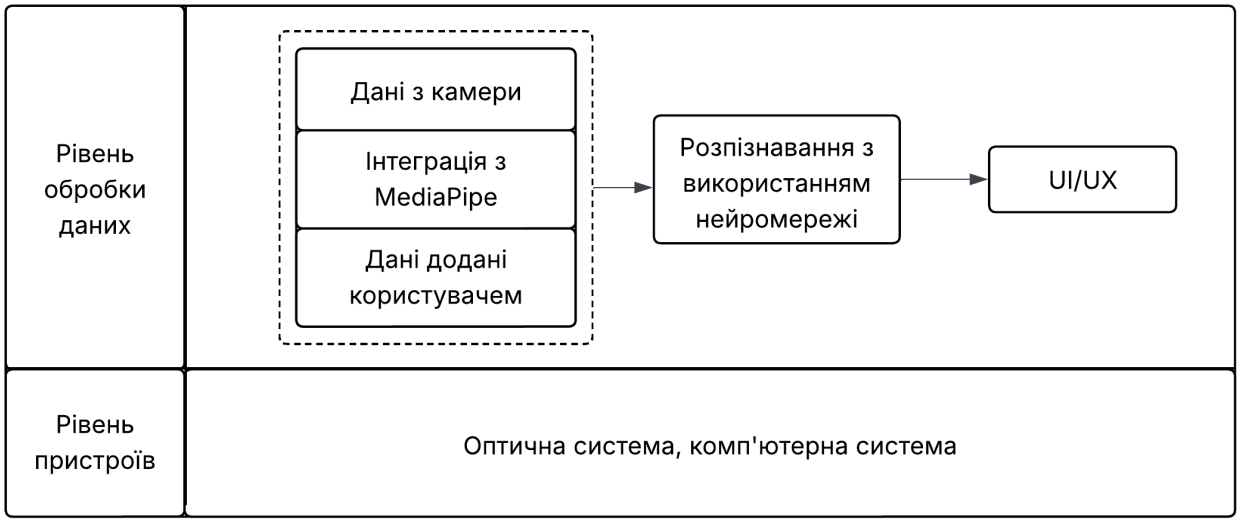

Рисунок 4.1 — Архітектура комп'ютерної системи

Рівень пристроїв відповідає за первинне отримання інформації про жести користувача. До нього входить оптична система, представлена камерою, яка захоплює відеопотік рухів рук. Якість камери має вирішальне значення, адже роздільна здатність та частота кадрів безпосередньо впливають на деталізацію й плавність фіксації жестів, що є критично важливим для точного розпізнавання. Крім того, до цього рівня належить комп'ютерна система, яка забезпечує необхідні обчислювальні ресурси для подальшої обробки отриманих даних.

Рівень обробки даних відповідає за аналіз отриманої інформації, розпізнавання жестів та взаємодію з користувачем. Рівень починається з отримання відеопотоку, захопленого на рівні пристроїв. Далі відбувається інтеграція з бібліотекою MediaPipe, яка забезпечує інструменти для виявлення ключових точок

рук та їх відстеження у часі. Дані передаються на етап розпізнавання з використанням нейромережі, де навчена модель аналізує вилучені ознаки та класифікує жест. Завершальним етапом рівня обробки даних є інтерфейс користувача, який забезпечує візуалізацію результатів роботи системи.

4.1.1 Апаратна частина

Апаратна частина є первинною складовою системи розпізнавання жестової мови, від якої безпосередньо залежить якість вхідних даних, швидкість обробки та загальна продуктивність системи. Структурна схема комп'ютерної системи розпізнавання мови жестів показана на рисунку В.2. Апаратна частина, своєю чергою, реалізується у вигляді клієнтської та серверної компонент, що відповідає сучасній клієнт-серверній архітектурі вебзастосунків.

Клієнтська частина представлена пристроєм користувача з камерою, яка забезпечує захоплення відео для подальшої обробки. Для стабільної роботи системи рекомендується використовувати камеру з роздільною здатністю щонайменше 1280×720 (HD), що дозволяє чітко фіксувати позицію рук і пальців. Частота знімання повинна становити не менше 10 кадрів за секунду для коректного відображення швидких рухів без розмиття. Для взаємодії з системою клієнт повинен мати доступ до сучасного браузера з підтримкою HTTPS-протоколу. Наявність автофокусу та широкого кута огляду не обов'язкова, проте значно підвищить зручність використання системи.

Серверна частина системи виконує основні обчислювальні операції, пов'язані з обробкою відеопотоку, виділенням ключових ознак, проведенням класифікації за допомогою нейронної мережі та виведенням результатів у зручному для користувача форматі. Вона виступає ядром комп'ютерної системи, на яке покладається основне навантаження при роботі з алгоритмами машинного навчання, зокрема з глибинними моделями розпізнавання. З технічної точки зору, сервер повинен бути оснащений достатньо потужним центральним процесором (CPU) із багатьма обчислювальними ядрами (наприклад, Intel Core i7 або AMD Ryzen 7), що забезпечить паралельну обробку відеоданих від кількох клієнтів.

Особливої уваги потребує обсяг оперативної пам’яті. Рекомендовано щонайменше 16 ГБ, адже кожне підключення створює нові буфери для зберігання координат ключових точок, попередньої обробки зображення, проміжних результатів класифікації тощо. Ці буфери активно оновлюються в режимі реального часу, і нестача пам’яті може призвести до затримок або помилок в обробці.

Ключову роль у продуктивності відіграє графічний процесор (GPU), який забезпечує апаратне прискорення при навчанні та застосуванні нейронної мережі. Підтримка технологій CUDA або OpenCL дозволяє ефективно використовувати потужності відеокарти для обробки багатьох паралельних операцій над масивами даних, зокрема матричних множень, згорток, активаційних функцій тощо. У рамках даного проєкту оптимальним варіантом є використання GPU з архітектурою NVIDIA Turing або новішою. Завдяки цьому значно скорочується час обробки кадрів, що особливо важливо при інтерактивному розпізнаванні жестів у реальному часі.

Сервер повинен мати стабільне підключення до мережі з високою пропускною здатністю, що особливо критично у випадках обслуговування кількох клієнтів одночасно. У таких умовах доцільним є впровадження механізмів балансування навантаження, попереднього кешування результатів обробки, а також глибокої оптимізації програмного коду з урахуванням архітектурних особливостей апаратного забезпечення. Це дозволяє мінімізувати затримки та забезпечити високу стабільність системи в умовах змінного навантаження.

Отже, серверна частина виконує ключову роль у функціонуванні комп’ютерної системи розпізнавання жестової мови. Від її обчислювальної потужності, надійності та масштабованості безпосередньо залежать точність, швидкодія та здатність обробляти дані в реальному часі, особливо у багатокористувацькому середовищі. У свою чергу, до клієнтської частини системи висуваються значно простіші вимоги.

4.1.2 Програмна частина

Програмна частина системи розпізнавання жестової мови відповідає за

основний процес аналізу відеоданих, отриманих від апаратної частини, виявлення та інтерпретацію жестів, а також взаємодію з користувачем.

Центральним елементом програмної обробки є використання бібліотеки MediaPipe (див. рис. В.2), яка реалізує високоточний аналіз відеозображення в реальному часі. Це включає в себе складні алгоритми для виділення ознак, таких як ключові точки на пальцях та загальне положення рук у кожному кадрі відеопотоку. Трекінг рук забезпечує відстежування рухів рук, аналізуючи послідовність кадрів.

Отримані координати ключових точок передаються до модуля розпізнавання. Його основою є нейронна мережа, яка була попередньо навчена на датасеті жестів української жествоої мови. Модель аналізує просторові та часові патерни рухів і виконує класифікацію жестів відповідно до навченої множини.

Після розпізнавання відбувається етап інтерпретації, під час якого розпізнаний жест трансформується у відповідний текстовий вивід. Для забезпечення зручності роботи система також передбачає збереження історії розпізнаних жестів, що дозволяє користувачу переглядати попередні результати.

Вся взаємодія користувача з системою здійснюється через графічний інтерфейс (UI/UX). Він забезпечує доступ до ключових функцій. Завдяки використанню вебтехнологій, програмна частина може працювати у браузері, що підвищує доступність системи для користувачів незалежно від операційної системи чи типу пристрою.

4.2 Розробка клієнтської частини та графічного інтерфейсу

Клієнтська частина комп’ютерної системи розпізнавання жестової мови реалізована, як вебзастосунок, що забезпечує користувачеві візуалізацію процесу розпізнавання та взаємодію з системою (рис. 4.2).

Графічний інтерфейс розроблено з використанням стандартних вебтехнологій, що забезпечує його кросплатформність та доступність через браузер. Основу інтерфейсу складають HTML — для визначення структури сторінки, CSS — для стилізації та оформлення візуальних елементів, а також JavaScript — для забезпечення динамічної поведінки та обміну даними з сервером.

Такий підхід гарантує коректну роботу інтерфейсу на більшості сучасних пристроїв, незалежно від операційної системи.

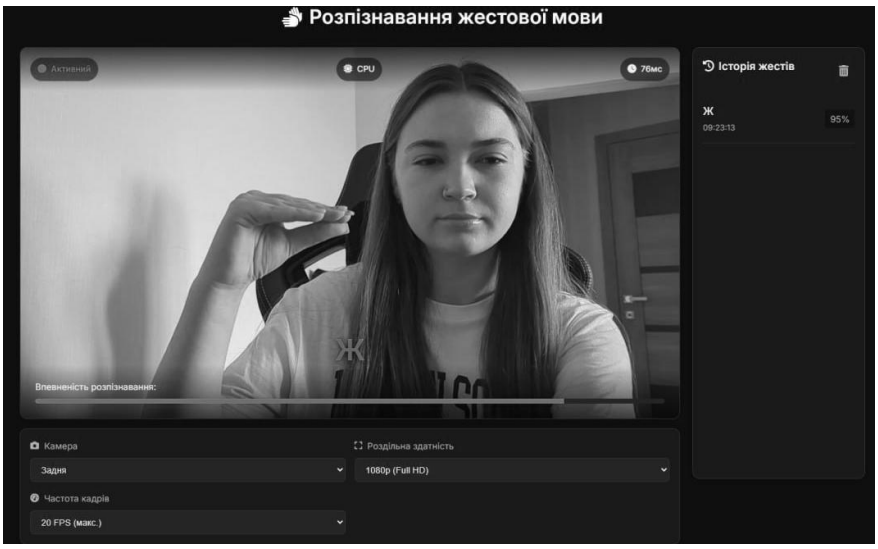

Рисунок 4.2 — Графічний інтерфейс розпізнавання жестів

Центральним елементом інтерфейсу є відеосекція, яка транслює зображення з вебкамери користувача в реальному часі. У межах цього блоку також виводиться поточний статус системи (активна чи не активна) та середній час обробки одного кадру, що дозволяє користувачеві оцінити продуктивність системи.

Для забезпечення миттєвого зворотного зв'язку під відеопотоком виводиться текстовий результат розпізнавання жесту. Нижче розміщено графічний індикатор впевненості розпізнавання, який демонструє, наскільки нейронна мережа впевнена у класифікації жесту. Це важливо в ситуаціях, коли розпізнавання може бути неоднозначним або жест виконано нечітко.

Для надання користувачеві можливості налаштування роботи системи передбачена окрема секція елементів керування. Вона включає випадаючі списки для вибору камери між фронтальною та задньою, а також для налаштування роздільної здатності відео та частоти кадрів, відповідно до потреб користувача. Це дозволяє адаптувати систему до різних пристроїв і умов використання, наприклад знизити навантаження на слабші комп’ютери або підвищити якість для точнішого розпізнавання.

У правій частині інтерфейсу розташовано панель історії, що зберігає

хронологічний список останніх розпізнаних жестів. Кожен запис містить інформацію про сам жест, час його фіксації та рівень впевненості системи у результаті розпізнавання. Ця функція надає користувачеві можливість відстежувати роботу системи та переглядати історію своїх взаємодій. Записи в історії можна очистити за допомогою відповідної кнопки в правому верхньому кутку блоку.

Загалом графічний інтерфейс створено з акцентом на простоту, інформативність та зручність у використанні, що робить систему інтуїтивно зрозумілою для широкого кола користувачів.

4.3 Реалізація серверної логіки та взаємодія з нейромережею

Серверна логіка реалізована з використанням фреймворку Flask, який спрощує обробку HTTP-запитів та передачу даних між клієнтською і серверною частинами.

Після запуску серверної частини ініціалізується захоплення відеопотоку з вебкамери користувача. Кожен кадр обробляється за допомогою бібліотеки MediaPipe, яка виконує детекцію рук і повертає координати 21 ключової точки для кожної виявленої руки. Отримані координати перетворюються у вектор ознак довжиною 126, який виступає вхідними даними для нейронної мережі.

З метою підвищення ефективності обчислень та зменшення навантаження на систему, вхід до моделі формується не з окремих кадрів, а з послідовності попередніх кадрів. Такий підхід дозволяє нейронній мережі враховувати часовий контекст руху та покращує точність класифікації динамічних жестів. Коли послідовність кадрів досягає довжини 30, з них формується спеціальний числовий масив послідовності ключових точок, який є вхідним форматом для нейромережі. Масив передається на вхід навченої моделі, яка виконує передбачення. Результатом роботи нейронної мережі є вектор ймовірностей для кожного класу жестів. За допомогою спеціальної функції обирається індекс класу з найвищою ймовірністю, який через словник трансформується у назву розпізнаного жесту. Додатково фіксується рівень впевненості моделі у передбаченні та час, витрачений на обробку. Отримані значення зберігаються у глобальній змінній, що згодом передається

клієнтській частині для відображення результату. Реалізація описаного алгоритму наведена у лістингу Ґ.2.

Таким чином, повний цикл роботи системи виглядає так: відео зчитується, кадри передаються на сервер, обробляються за допомогою MediaPipe, формується послідовність ключових точок, яка передається до навченої моделі, а результат відображається у вебінтерфейсі. Завдяки використанню Flask та чітко структурованої серверної логіки, програмний застосунок є масштабованим, гнучким у налаштуванні, і може бути адаптований до інших задач розпізнавання.

4.4 Виведення результатів розпізнавання у текстовому форматі

Виведення результатів розпізнавання жестів у текстовому форматі реалізоване за принципом динамічного оновлення інтерфейсу користувача. Після кожного передбачення модель повертає вектор імовірностей для кожного класу, і жест із найвищим значенням ймовірності визначається як розпізнаний. Однак, для уникнення хибних спрацьовувань використовується механізм фільтрації за рівнем впевненості.

Система виводить результат лише у тому випадку, коли впевненість розпізнавання моделі перевищує 60% (лістинг 4.1). Такий пороговий механізм дозволяє фільтрувати низькоякісні або випадкові передбачення, що можуть виникати через шум, часткове перекриття руки або неправильну постановку жесту. Встановлення мінімального рівня впевненості є досить важливим при взаємодії з реальними користувачами, оскільки дозволяє забезпечити надійний і зрозумілий зворотний зв'язок, зменшуючи ймовірність хибних спрацювань та підвищуючи довіру до системи.

Лістинг 4.1 — Фільтрація результатів за порогом впевненості

```
MIN_CONFIDENCE = 0.6
if confidence >= MIN_CONFIDENCE:
                    current_gesture["gesture"] = inverse_labels[class_idx]
                    current_gesture["confidence"] = confidence
                else:
```

```
current_gesture["gesture"] = None
current_gesture["confidence"] = 0
```

Крім того, система фіксує поточний активний жест і порівнює його з попереднім. Якщо розпізнаний жест не змінився порівняно з попереднім, вивід не оновлюється, що дозволяє уникати надмірної кількості тексту на екрані при демонстрації одного і того ж жесту протягом певного часу. Відповідна логіка, реалізовується за допомогою простої перевірки, яка наведена у лістингу 4.2.

Лістинг 4.2 — Умова виведення розпізнаного жесту

```
if (historyItems.length === 0 || historyItems[0]?.gesture !== data.gesture) {
   addToHistory(data.gesture, confidencePercent);
   lastGesture = data.gesture; }
```

Фронтенд періодично звертається до серверного API з запитами та отримує у відповідь JSON-об'єкт, що містить назву розпізнаного жесту, значення впевненості моделі у відсотках, а також час, витрачений на обробку передбачення. Ці параметри виводяться у текстовому вигляді в спеціально відведених блоках, що забезпечує миттєвий зворотний зв’язок для користувача.

Окрім поточного жесту, система також веде історію останніх 20 успішно розпізнаних жестів (лістинг 4.3).

Лістинг 4.3 — Ведення історії розпізнавання

```
function addToHistory(gesture, confidence) {
      const now = new Date();
      const timeString = now.toLocaleTimeString();
      historyItems.unshift({
         gesture,
         confidence,
         time: timeString,
         timestamp: now.getTime()
      });
      if (historyItems.length > 20) {
         historyItems.pop();
      }
      renderHistory(); }
```

У історії фіксується назва кожного жесту, точність передбачення та час, витрачений на його обробку. Історія оновлюється лише у випадку зміни поточного жесту, що дозволяє уникнути дублювань і підвищує інформативність інтерфейсу.

4.5 Перевірка працездатності системи

Для перевірки базової працездатності системи було проведено тестування її основних функціональних компонентів у середовищі реального запуску. Розгортання відбувалося локально на комп’ютері з використанням веббраузера.

Першочергово здійснено запуск серверної частини застосунку. Було перевірено коректність ініціалізації Flask-додатку, завантаження HTML-інтерфейсу та доступність вебінтерфейсу за локальною IP-адресою. Сервер успішно запустився з підтримкою SSL-сертифікатів, що забезпечило захищене з’єднання HTTPS.

Далі проведено тестування підключення та обробки відеопотоку з вебкамери. Система стабільно захоплює відео з камери користувача та відображає його в режимі реального часу у вікні браузера. Це підтверджує надійний обмін даними між клієнтською і серверною частинами та справність модуля відеообробки. Загальну працездатність моделі показано на рисунку 4.3.

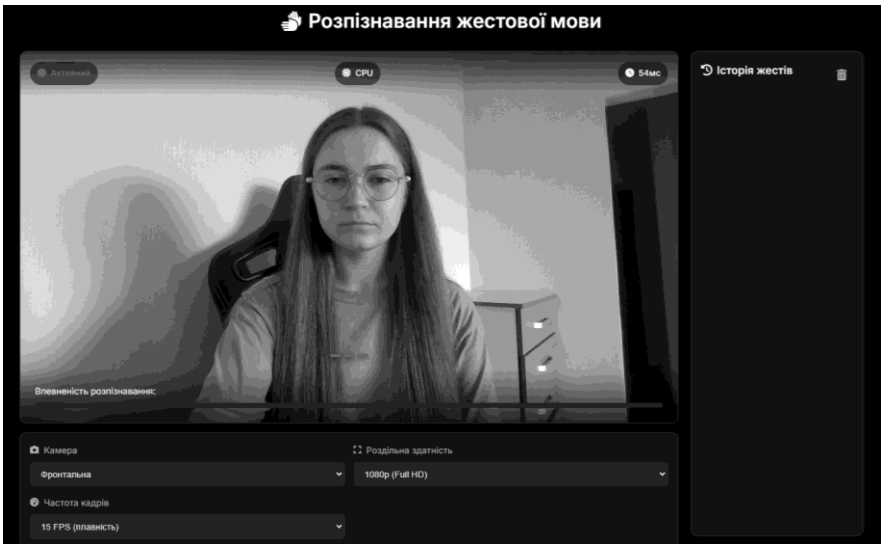

Рисунок 4.3 — Загальна перевірка працездатності системи

Також здійснено перевірку функціональності користувацького інтерфейсу.

Усі елементи керування працюють згідно з очікуваннями. Кнопки вибору активної камери, зміни роздільної здатності, а також очищення історії жестів виконують свій функціонал. Результат перевірки вибору роздільної здатності показано на рисунку 4.4.

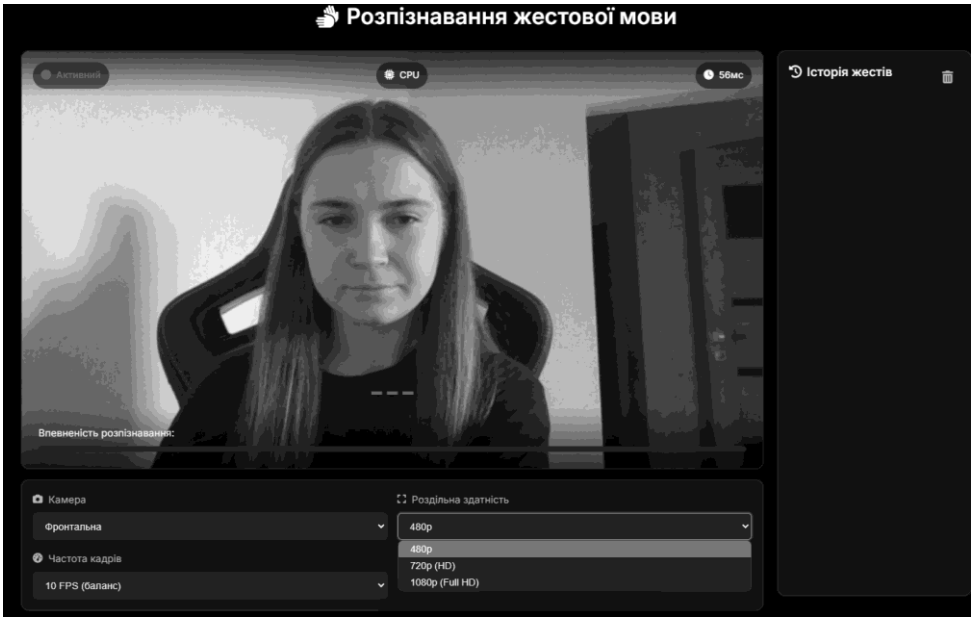

Рисунок 4.4 — Демонстрація зміни роздільної здатності відео

Після аналізу відеопотоку модель виконує розпізнавання жесту та виводить його назву у вебінтерфейсі. Виведення результату відбувається лише за умови, якщо рівень впевненості моделі у передбаченні перевищує встановлений поріг у 60%. Одночасно система веде історію останніх розпізнаних жестів, де для кожного з них фіксується назва, рівень точності та час обробки. Ці дані відображаються у спеціальному блоці вебінтерфейсу, що дозволяє користувачу переглядати результати попередніх розпізнавань. Працездатність виведення поточного жесту разом з історією розпізнавань показано на рисунку 4.5.

Для перевірки ефективності розробленої системи розпізнавання жестової мови було проведено комплексне тестування. Оскільки система призначена для роботи в інтерактивному середовищі, особлива увага приділялась перевірці її працездатності в умовах, максимально наближених до реального використання. Це дозволило оцінити стабільність взаємодії між клієнтською та серверною

частинами, точність розпізнавання жестів, швидкодію системи та зручність користувацького інтерфейсу.

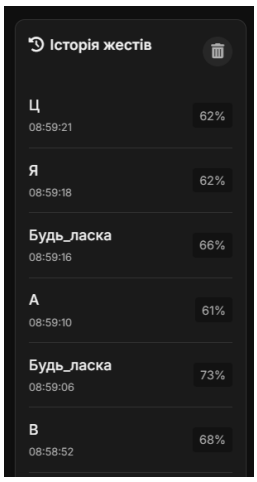

Рисунок 4.5 — Перевірка коректного ведення історії

Тестування проводилося на окремій тестовій вибірці, яка не використовувалася під час навчання. До її складу увійшли відеофрагменти із жестами, які виконані з різних ракурсів, при змінному освітленні та на різній відстані до камери. На рисунку 4.6 показано результат розпізнавання динамічного жесту.

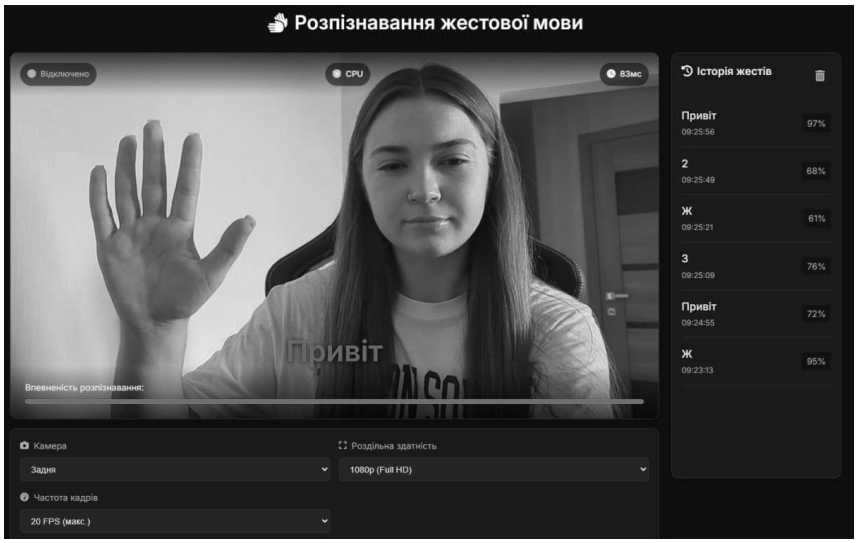

Рисунок 4.6 — Перевірка працездатності моделі на прикладі жесту «Привіт»

На рисунку 4.7 показано результат розпізнавання статичного жесту за умови не ідеального освітлення.

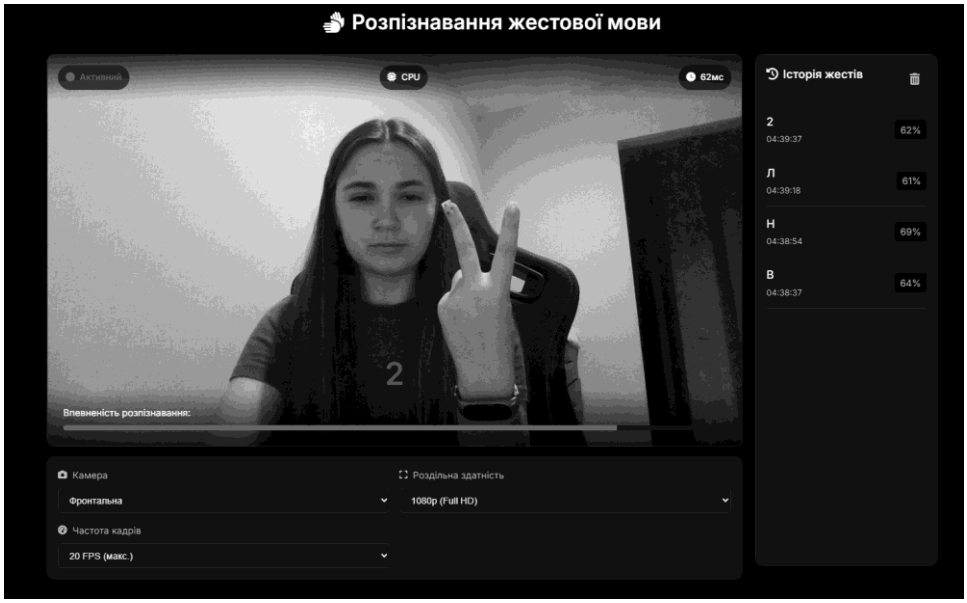

Рисунок 4.7 — Перевірка працездатності системи за умови поганого освітлення

Також система здатна розпізнавати жести показані під кутом, що демонструє хороший рівень навчання (рис. 4.8).

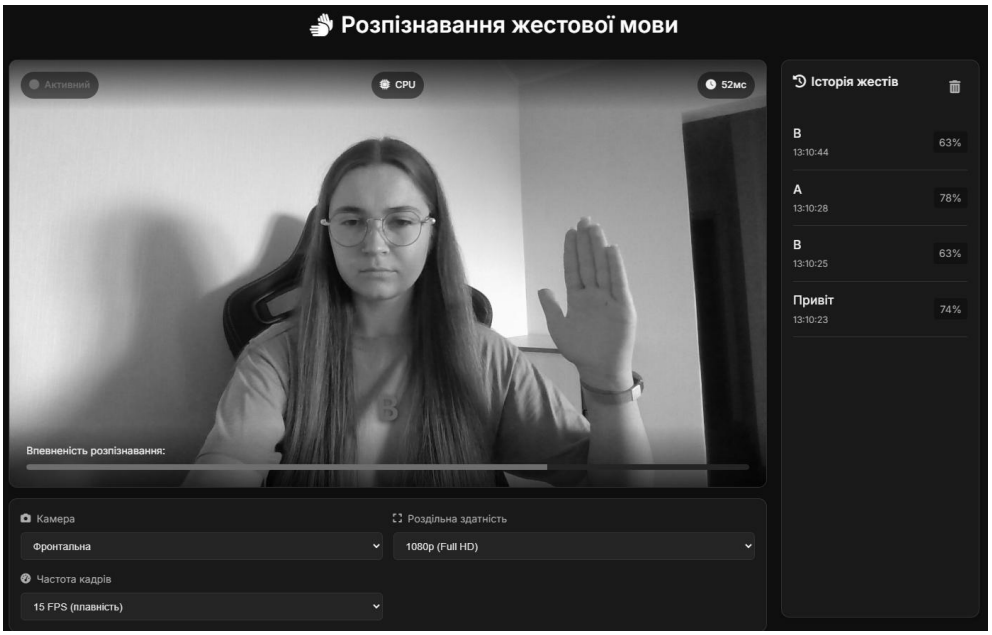

Рисунок 4.8 — Розпізнавання жестів, показаних під кутом

У результаті проведеного комплексного тестування було встановлено, що система демонструє стабільну та високу якість розпізнавання жестів за різних умов зйомки. Модель успішно класифікує жести, виконані як з близької відстані (приблизно 30 см до камери), так і на більшій відстані — до одного метра (рис. 4.9).

Це підтверджує гнучкість архітектури системи та її здатність до коректного функціонування незалежно від положення користувача в кадрі.

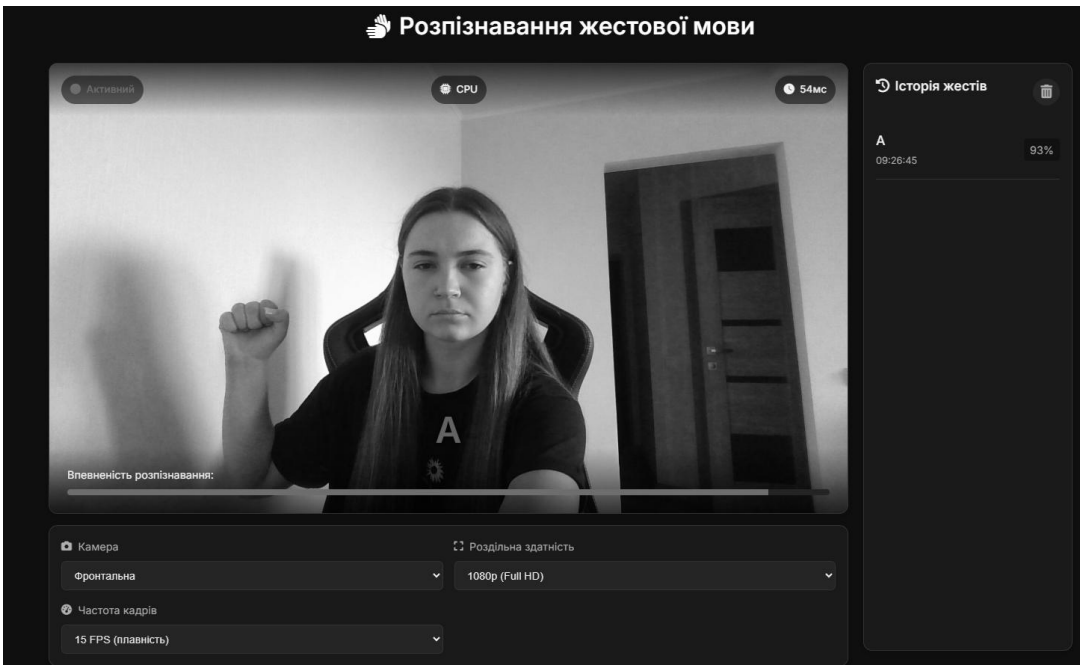

Рисунок 4.9 — Перевірка працездатності системи на більшій відстані

Додатково було перевірено вплив зміни умов освітлення та фону. У більшості випадків навіть при значному затемненні сцени або появі нових об'єктів у фоні точність розпізнавання залишалась на достатньому рівні. Це свідчить про адаптивність системи до умов реального середовища, а також про ефективність використаної архітектури нейронної мережі та якість попередньої обробки даних.

Особливу увагу приділено швидкодії системи. Незважаючи на обробку відеопотоку в реальному часі, модель демонструє низьку затримку між виконанням жесту та його відображенням у вебінтерфейсі. Середній час обробки одного передбачення становить менше 100 мілісекунд.

Загалом результати тестування підтверджують не лише працездатність розробленої системи, а й її здатність до узагальнення, стійкість до зовнішніх факторів і ефективне функціонування у реальному часі. Така система може бути застосована, як засіб комунікації для людей з порушеннями слуху, як навчальний інструмент для опанування української жестової мови або як база для подальшого розвитку в напрямку розпізнавання складніших динамічних фраз.

ВИСНОВКИ

У бакалаврській кваліфікаційній роботі було розроблено та реалізовано програмний засіб для розпізнавання жестів української жестової мови у реальному часі на основі нейронної мережі. Метою роботи було створення системи, здатної розпізнавати жести за відеопотоком з вебкамери та відображати результати у зручному інтерфейсі для кінцевого користувача.

У процесі виконання роботи було проведено аналіз сучасних підходів до комп’ютерного зору та глибинного навчання, досліджено можливості бібліотеки MediaPipe для виділення ключових точок рук та обґрунтовано вибір архітектури нейронної мережі. На основі аналізу було реалізовано архітектуру, яка використовує двонаправлені рекурентні шари GRU та LSTM для аналізу послідовностей координат рук. Для підвищення точності моделі використано ансамбль із 5 незалежно навчених моделей.

Окрему увагу приділено створенню власного датасету української жестової мови. Загалом датасет включає 40 жестів, кожен з яких представлений не менше ніж 60 відеофрагментами. Всі відео було оброблено з використанням MediaPipe, а з нього виділено координати ключових точок обох рук, що дозволило сформувати повноцінну навчальну вибірку для моделі. Також реалізовано зручний інтерфейс для додавання нових жестів, збору даних та навчання моделі.

Серверна частина застосунку реалізована мовою Python, а клієнтський інтерфейс забезпечує зворотний зв’язок із користувачем у реальному часі. Система дозволяє запускати процес розпізнавання жестів у браузері, обирати роздільну здатність камери та зберігати історію розпізнавань.

Проведене тестування системи підтвердило її працездатність, стабільність та здатність до реального розпізнавання жестів з прийнятною точністю.

Отримані результати свідчать про доцільність подальшого розвитку системи, зокрема, у напрямку розпізнавання не лише окремих літер та фраз, а й повноцінних речень, а також адаптації системи під мобільні платформи.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Інформаційна система розпізнавання мови жестів з використанням штучного інтелекту / А. С. Шпачинська, О. С. Городецька // Матеріали LIV науково-технічна конференція факультету інформаційних технологій та комп'ютерної інженерії. Вінниця 2025 р. [Електронний ресурс]. Режим доступу: https://conferences.vntu.edu.ua/index.php/all-fitki/all-fitki-2025/paper/view/23924/19786

2. С.В.Ткаліченко. Штучні нейронні мережі: Навчальний посібник. – Кривий Ріг: Державний університет економіки і технологій, 2023. –150 с.

3. Що таке нейронна мережа, як вона працює та які завдання може вирішувати. *MC.today, Media for Creators*. URL: https://mc.today/uk/shho-take-nejronna-merezha/ (дата звернення: 01.05.2025).

4. Усе, що ви хотіли знати про нейромережі, та чим вони можуть бути корисні. *Mind.ua*. URL: https://mind.ua/publications/20271107-use-shcho-vi-hotili-znati-pro-nejromerezhi-ta-chim-voni-mozhut-buti-korisni (дата звернення: 01.05.2025).

5. Дослідник нейромереж – про півсотлітню історію штучного інтелекту, свідомість AI та розвиток людства. *DOU.* URL: https://dou.ua/lenta/interviews/riznyk-intelligence/ (дата звернення: 01.05.2025).

6. Технології комп'ютерного зору – Computer Vision MetinvestDigital. *IT компанія - METINVEST.DIGITAL - Айті послуги в Києві та Україні*. URL: https://metinvest.digital/ua/page/1028 (дата звернення: 02.05.2025).

7. GeeksforGeeks. Spatial Filtering and its Types - GeeksforGeeks. *GeeksforGeeks*. URL: https://www.geeksforgeeks.org/spatial-filtering-and-its-types/?utm_source=chatgpt.com (дата звернення: 04.05.2025).

8. McCoy R. Exploring Image Compression with Fourier and Wavelet Transformations using Python. *Medium*. URL: https://medium.com/@ryanwmccoy/exploring-image-compression-with-fourier-and-wavelet-transformations-using-python-0b3c15a72703 (дата звернення: 04.05.2025).

9. GeeksforGeeks. Explain Image Segmentation : Techniques and Applications - GeeksforGeeks. *GeeksforGeeks*. URL: https://www.geeksforgeeks.org/explain-image-segmentation-techniques-and-applications/ (дата звернення: 06.05.2025).

10. ScienceDirect. Feature Detector. *ScienceDirect.* URL: https://www.sciencedirect.com/topics/computer-science/feature-detector?utm_source=chatgpt.com (дата звернення: 06.05.2025).

11. Berrios I. Introduction to Motion Detection: Part 1. *Medium*. URL: https://medium.com/@itberrios6/introduction-to-motion-detection-part-1-e031b0bb9bb2 (дата звернення: 06.05.2025).

12. Yu S., Weigl K. Performance comparison of a deterministic and a stochastic method for image classification. *SpringerLink*. URL: https://link.springer.com/chapter/10.1007/3-540-60268-2_388?utm_source=chatgpt.com (дата звернення: 06.05.2025).

13. 5 Types of Neural Networks: An Essential Guide for Analysts. *Data Science Dojo*. URL: https://datasciencedojo.com/blog/5-main-types-of-neural-networks/ (дата звернення: 07.05.2025).

14. Massobrio A. 3D Convolutional Neural Network — A Guide for Engineers. Neural Concept. URL: https://www.neuralconcept.com/post/3d-convolutional-neural-network-a-guide-for-engineers#link1 (дата звернення: 08.05.2025).

15. Українська жестова мова. *АлиБаба.* URL: https://www.alibabaru.com/ukrayinska-zhestova-mova/ (дата звернення: 09.05.2025).

16. Українська правда. Життя. Говоримо руками: що варто знати про жестову мову, добірка фраз на щодень. *Українська правда. Життя*. URL: https://life.pravda.com.ua/society/shcho-treba-znati-pro-zhestovu-movu-ta-chomu-vazhlivo-jiji-opanuvati-307769/ (дата звернення: 09.05.2025).

17. MediaPipe: A Guide Google's Open Source Framework - viso.ai. *viso.ai*. URL: https://viso.ai/computer-vision/mediapipe/ (дата звернення: 09.05.2025).

18. GeeksforGeeks. What is OpenCV Library? - GeeksforGeeks. *GeeksforGeeks*. URL: https://www.geeksforgeeks.org/opencv-

overview/ (дата звернення: 09.05.2025).

19. GeeksforGeeks. Difference between TensorFlow and Keras - GeeksforGeeks. *GeeksforGeeks*. URL: https://www.geeksforgeeks.org/difference-between-tensorflow-and-keras/ (дата звернення: 09.05.2025).

20. GeeksforGeeks. Flask Tutorial - GeeksforGeeks. *GeeksforGeeks*. URL: https://www.geeksforgeeks.org/flask-tutorial/ (date of access: 09.05.2025).

21. What is NumPy? – NumPy v2.2 Manual. *NumPy*. URL: https://numpy.org/doc/stable/user/whatisnumpy.html (дата звернення: 10.05.2025).

22. What is Python? | Teradata. *Cloud-Based Data Analytics and Trusted AI | Teradata*. URL: https://www.teradata.com/glossary/what-is-python (дата звернення: 10.05.2025).

23. W3Schools.com. *W3Schools Online Web Tutorials*. URL: https://www.w3schools.com/python/python_intro.asp (дата звернення: 10.05.2025).

24. PNN Soft. Використання Python для алгоритмів на основі штучного інтелекту. *Компанія розробки програмного забезпечення.* URL: https://pnn.com.ua/ua/blog/detail/unleashing-the-potential-of-ai-a-step-by-step-guide-to-implementing-ai-algorithms-in-python (дата звернення: 10.05.2025).

25. GeeksforGeeks. What is PyCharm? - GeeksforGeeks. *GeeksforGeeks*. URL: https://www.geeksforgeeks.org/what-is-pycharm/ (дата звернення: 10.05.2025).

26. IBM. What is a Recurrent Neural Network (RNN)? | IBM. *IBM - United States*. URL: https://www.ibm.com/think/topics/recurrent-neural-networks (дата звернення: 11.05.2025).

27. Dense Neural Networks: Understanding Their Structure and Function. *DataScientest*. URL: https://datascientest.com/en/dense-neural-networks-understanding-their-structure-and-function (дата звернення: 11.05.2025).

28. What is JavaScript? - Learn web development | MDN. *MDN Web Docs*. URL: https://developer.mozilla.org/en-US/docs/Learn_web_development/Core/Scripting/What_is_JavaScript (дата

звернення: 11.05.2025).

29. What is a Dataset: Types, Features, and Examples - GeeksforGeeks. *GeeksforGeeks*. URL: https://www.geeksforgeeks.org/what-is-dataset/ (дата звернення: 12.05.2025).

30. Методичні вказівки до виконання бакалаврських кваліфікаційних робіт для студентів спеціальності 123 «Комп'ютерна інженерія» (освітня програма «Комп'ютерна інженерія») / Уклад. О. Д. Азаров, С. В. Богомолов, С. І. Швець, — Вінниця : ВНТУ, 2024. – 106 с.

ДОДАТОК А

Технічне завдання

Міністерство освіти та науки України
Вінницький національний технічний університет
Факультет інформаційних технологій та комп’ютерної інженерії

ЗАТВЕРДЖУЮ
Завідувач кафедри ОТ
проф., д.т.н., Азаров О.Д.
«_21_» ___03_____2025 р.

ТЕХНІЧНЕ ЗАВДАННЯ
на виконання бакалаврської кваліфікаційної роботи
«Комп’ютерна система розпізнавання мови жестів з використанням штучного інтелекту»
08-54.БКР.017.00.000 ПЗ

Керівник роботи к.т.н., доц. каф. ОТ
_____ Городецька О. С.

Студентка групи 1КІ–21б
_____ Шпачинська А. С.

ВНТУ 2025

1 Підставою для виконання комплексної бакалаврської кваліфікаційної роботи (БКР)

1.1 Актуальність роботи обумовлена необхідністю створення доступних та ефективних технологій взаємодії для людей з порушенням слуху, що сприятиме їхній активнішій участі в суспільному житті. Завдяки стрімкому розвитку штучного інтелекту, машинного навчання та комп'ютерного зору стало можливим реалізувати системи, здатні розпізнавати жести в реальному часі. Водночас важливим напрямом є адаптація таких рішень до особливостей української жестової мови, оскільки більшість наявних розробок орієнтовані на англомовне середовище.

1.2 Наказ про затвердження теми БКР.

2 Мета БКР і призначення розробки

2.1 Мета роботи — розширення функціональних можливостей комп'ютерної системи розпізнавання жестів української жестової мови завдяки інтеграції з модулем штучного інтелекту.

2.2 Призначення розробки — інформаційна система призначена для розпізнавання української жестової мови в реальному часі з використанням технологій штучного інтелекту. Розробка повинна забезпечити точність розпізнавання, стабільну роботу, зручний інтерфейс і можливість подальшого розширення функціоналу.

3 Вихідні дані для виконання БКР

3.1 Наявність навчального датасету у вигляді відеофрагментів із жестами української жестової мови.

3.2 Отримані координати ключових точок рук.

3.3 Перелік жестів української дактильної абетки, які мають бути розпізнані системою.

3.4 Браузерне середовище виконання (Google Chrome, Mozilla Firefox) з підтримкою доступу до вебкамери, стабільної роботи Flask-сервера та дотримання вимог зручності інтерфейсу для користувача.

4 Вимоги до виконання БКР

4.1 Здійснити огляд сучасного стану розвитку нейромереж.

4.2 Проаналізувати методи комп’ютерного зору для обробки відео.

4.3 Вибрати та обґрунтувати технології, інструменти та архітектуру для розробки інформаційної системи.

4.4 Сформувати датасет жестів української жестової мови для навчання.

4.5 Здійснити навчання нейронної мережі для розпізнавання жестів.

4.6 Розробити комп’ютерну систему розпізнавання мови жестів з використанням штучного інтелекту.

4.7 Здійснити перевірку працездатності системи.

5 Етапи БКП та очікувані результати

Етапи роботи та очікувані результати приведено в Таблиці А.1.

Таблиця А.1 — Етапи БКП

№ етапу	Назва етапу	Термін виконання		Очікувані результати
		початок	кінець	
1	Пошук літературних джерел та аналіз методів розпізнавання жестів	21.03.25	28.03.25	Вступ, Розділ 1
2	Дослідження технологій комп’ютерного зору та підходів до зчитування координат рук	28.03.25	06.04.25	Розділ 1, Розділ 2
3	Збір і підготовка власного датасету, навчання моделі для класифікації жестів	07.04.25	14.04.25	Розділ 3
4	Розробка вебінтерфейсу та інтеграція моделі розпізнавання в режимі реального часу	15.04.25	27.04.25	Розділ 4
5	Оформлення пояснювальної записки та підготовка ілюстративного матеріалу	28.04.25	11.05.25	ПЗ, графічний матеріал
6	Перевірка якості виконання бакалаврської кваліфікаційної роботи та усунення недоліків	13.05.25	13.05.25	Внесення правок до ПЗ, презентація

6 Матеріали, що подаються до захисту БКР

До захисту подаються: пояснювальна записка БКР, графічні і ілюстративні матеріали, протокол попереднього захисту БКР на кафедрі, відгук наукового керівника, відгук рецензента, анотації до БКР українською та іноземною мовами.

7 Порядок контролю виконання та захисту БКР

Виконання етапів графічної та розрахункової документації БКР контролюється науковим керівником згідно зі встановленими термінами. Захист БКР відбувається на засіданні Екзаменаційної комісії, затвердженої наказом ректора.

8 Вимоги до оформлювання та порядок виконання БКР

8.1 При оформлюванні БКР використовуються:

— ДСТУ 3008: 2015 «Звіти в сфері науки і техніки. Структура та правила оформлювання»;

— ДСТУ 8302: 2015 «Бібліографічні посилання. Загальні положення та правила складання»;

— ГОСТ 2.104–2006 «Єдина система конструкторської документації. Основні написи»;

— методичні вказівки до виконання бакалаврських кваліфікаційних робіт зі спеціальності 123 — «Комп’ютерна інженерія»;

— документи на які посилаються у вище вказаних.

8.2 Порядок виконання БКР викладено в «Положення про кваліфікаційні роботи на першому (бакалаврському) рівні вищої освіти СУЯ ВНТУ-03.02.02-П.001.01:21.

ДОДАТОК Б

ПРОТОКОЛ ПЕРЕВІРКИ КВАЛІФІКАЦІЙНОЇ РОБОТИ

Назва роботи: __Комп'ютерна система розпізнавання мови жестів з_____ використанням штучного інтелекту_____

Тип роботи: _____бакалаврська кваліфікаційна робота_____
(бакалаврська кваліфікаційна робота / магістерська кваліфікаційна робота)

Підрозділ _____кафедра обчислювальної техніки_____
(кафедра, факультет, навчальна група)

Коефіцієнт подібності текстових запозичень, виявлених у роботі системою StrikePlagiarism __2__%

Висновок щодо перевірки кваліфікаційної роботи (відмітити потрібне)

- ✓ Запозичення, виявлені у роботі, є законними і не містять ознак плагіату, фабрикації, фальсифікації. Роботу прийняти до захисту
- ☐ У роботі не виявлено ознак плагіату, фабрикації, фальсифікації, але надмірна кількість текстових запозичень та/або наявність типових розрахунків не дозволяють прийняти рішення про оригінальність та самостійність її виконання. Роботу направити на доопрацювання.
- ☐ У роботі виявлено ознаки плагіату та/або текстових маніпуляцій як спроб укриття плагіату, фабрикації, фальсифікації, що суперечить вимогам законодавства та нормам академічної доброчесності. Робота до захисту не приймається.

Експертна комісія:

_____Азаров О.Д., зав. каф. ОТ_____ _____
(прізвище, ініціали, посада) (підпис)

____Крупельницький Л.В., доц. каф ОТ_____ _____
(прізвище, ініціали, посада) (підпис)

Особа, відповідальна за перевірку _____ __Захарченко С.М.__
(підпис) (прізвище, ініціали)

З висновком експертної комісії ознайомлений(-на)

Керівник _____ __к.т.н., доц. каф. ОТ_Городецька О. С.
(підпис) (прізвище, ініціали, посада)

Здобувач _____ _____Шпачинська А. С.
(підпис) (прізвище, ініціали)

ДОДАТОК В

Графічна частина

КОМП’ЮТЕРНА СИСТЕМА РОЗПІЗНАВАННЯ МОВИ ЖЕСТІВ З ВИКОРИСТАННЯМ ШТУЧНОГО ІНТЕЛЕКТУ

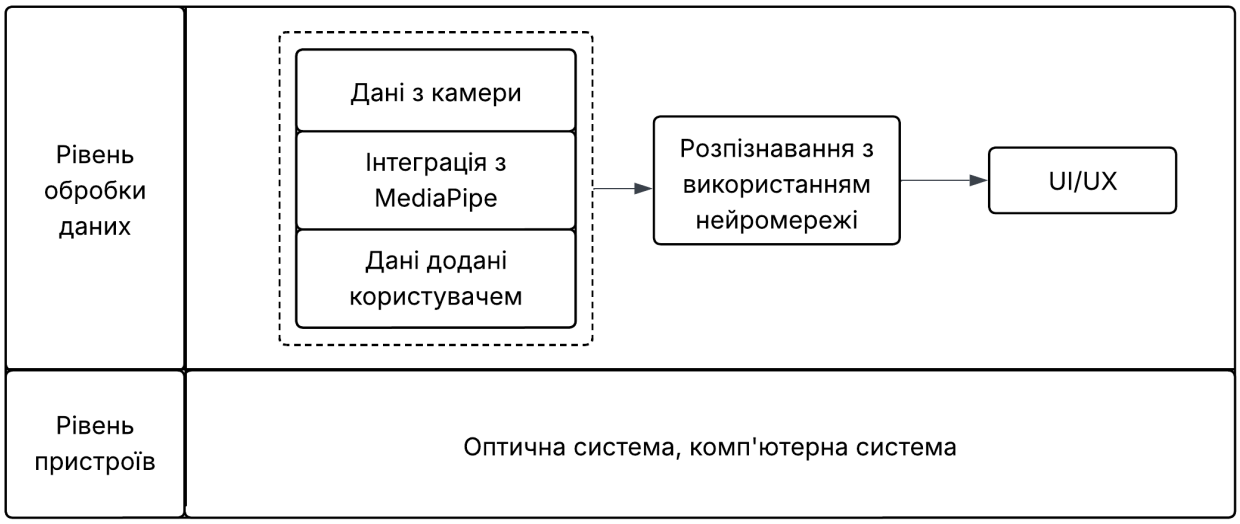

Рисунок В.1 — Архітектура комп’ютерної системи розпізнавання мови жестів

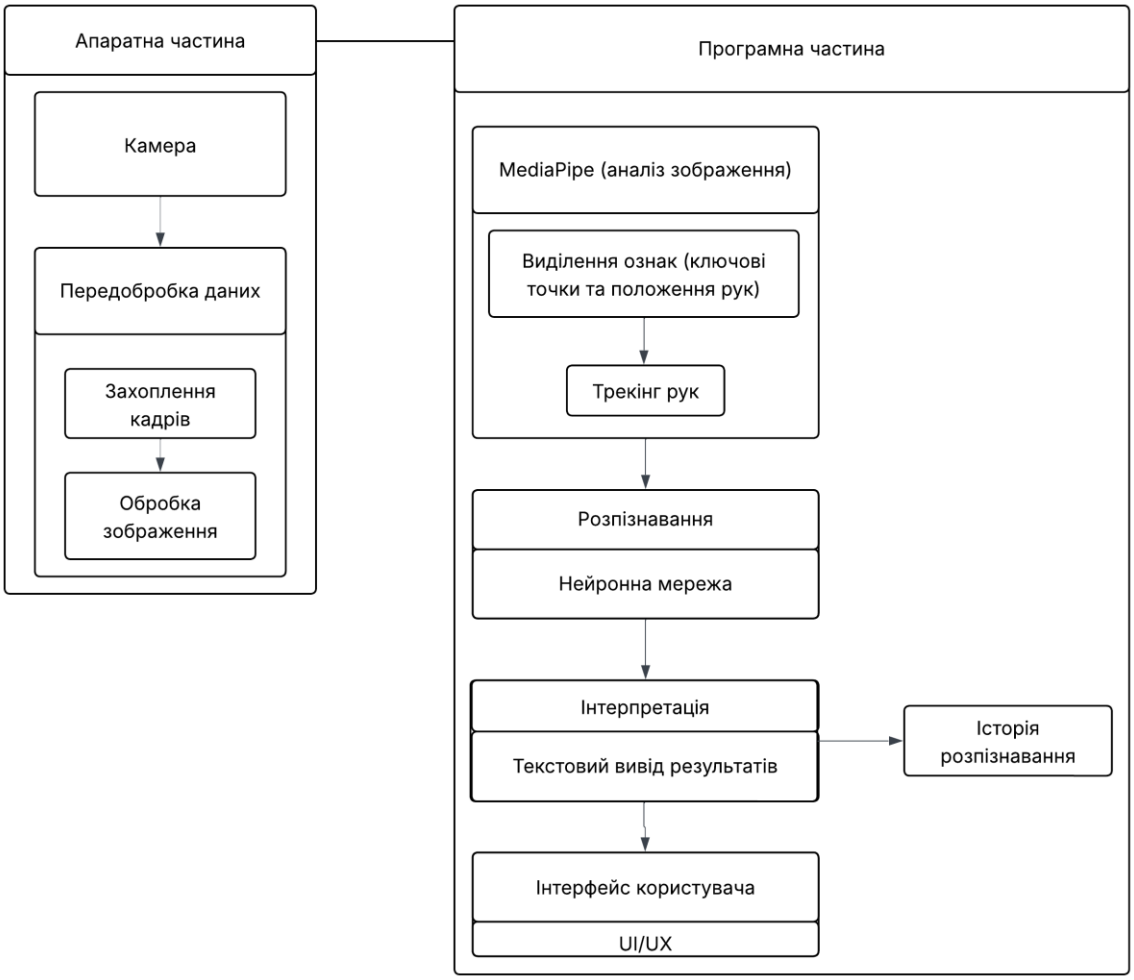

Рисунок В.2 — Структурна схема комп’ютерної системи розпізнавання мови жестів

ДОДАТОК Г

Ілюстративна частина

КОМП’ЮТЕРНА СИСТЕМА РОЗПІЗНАВАННЯ МОВИ ЖЕСТІВ З ВИКОРИСТАННЯМ ШТУЧНОГО ІНТЕЛЕКТУ

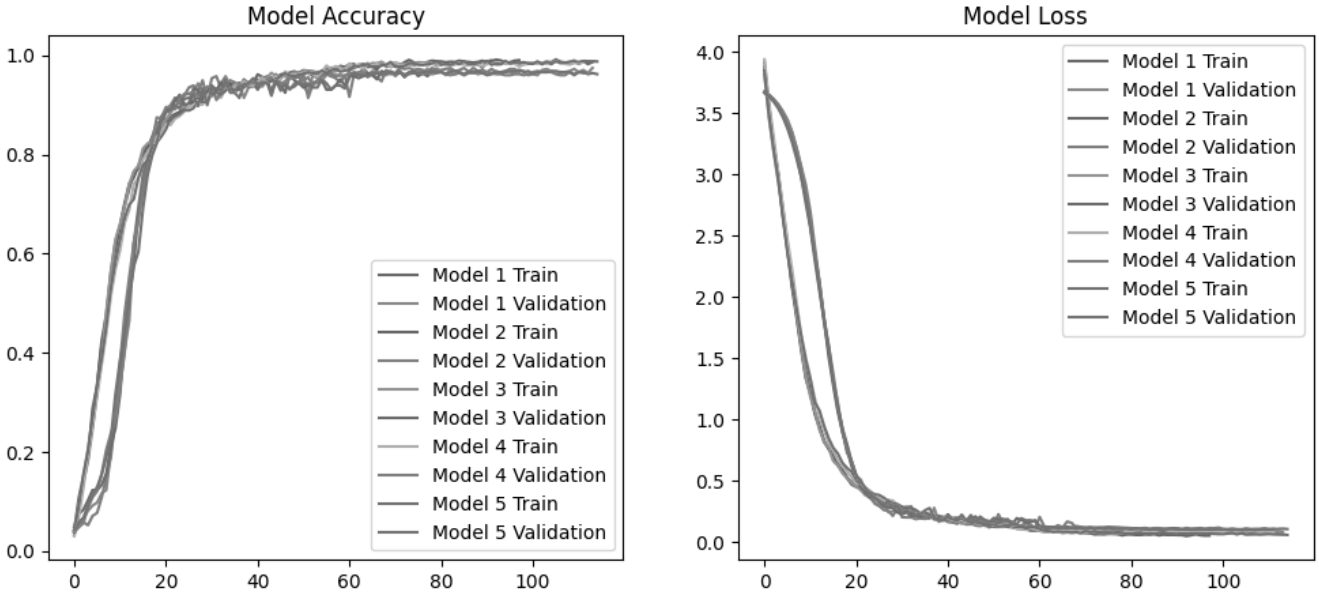

Рисунок Г.1 — Графіки точності та функції втрат моделей ансамблю

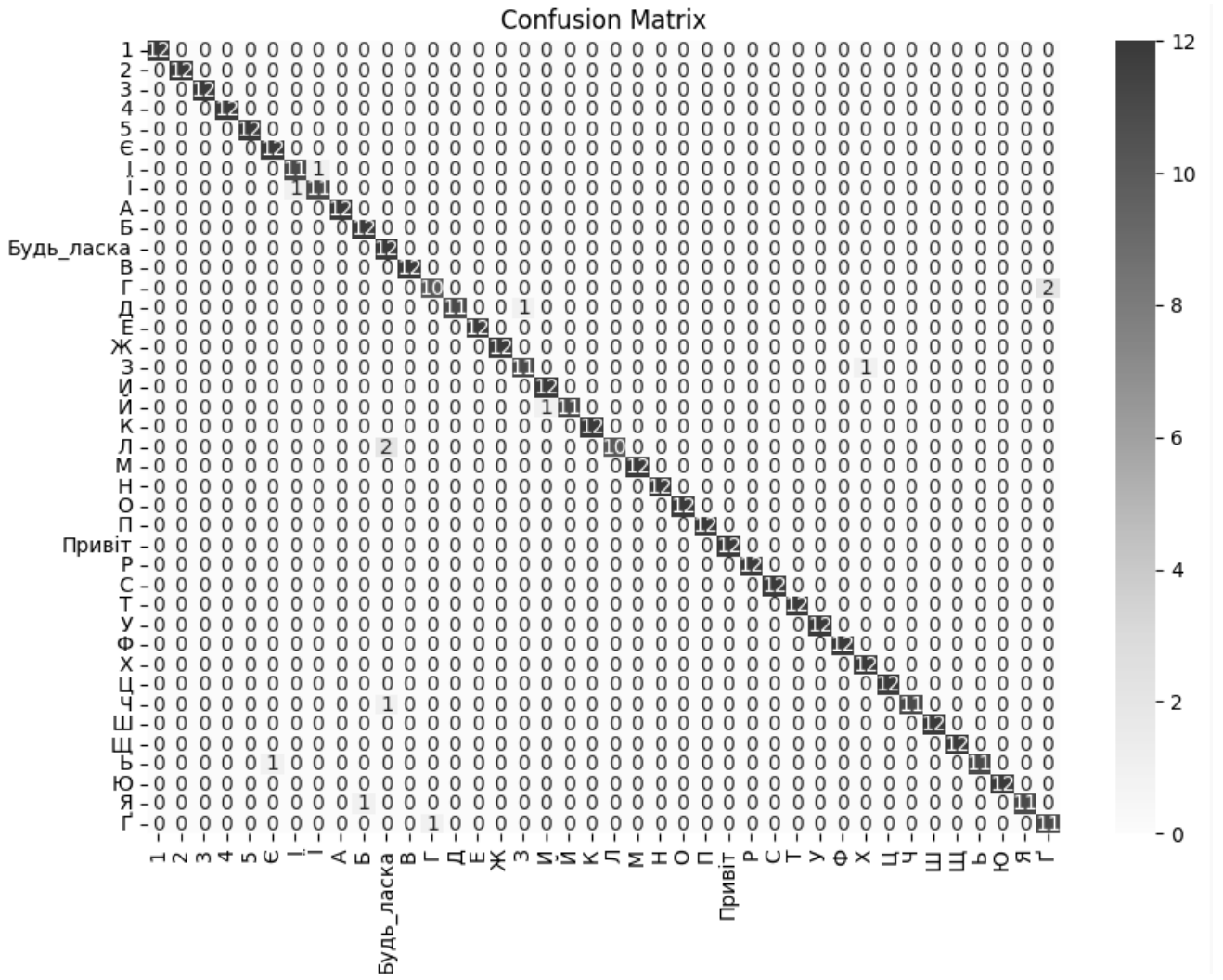

Рисунок Г.2 — Матриця плутанини

ДОДАТОК Ґ

Лістинги програмного забезпечення

Лістинг Ґ.1 — Обробка координат та навчання моделі

```
@app.route('/process_dataset', methods=['POST'])
def process_dataset():
    def run_process():
        try:
            training_progress['progress'] = 0
            training_progress['status'] = 'Обробка датасету...'
            training_progress['log'] = 'Запуск process_dataset.py\n'
            training_progress['completed'] = False

            process = subprocess.Popen(

                ['python', '../training/process_dataset.py'],

                stdout=subprocess.PIPE,

                stderr=subprocess.STDOUT,

                universal_newlines=True)

            for line in process.stdout:

                training_progress['log'] += line

                if 'Processing videos' in line and '%' in line:

                    percent = int(line.split('%')[0].split()[-1])

                    training_progress['progress'] = percent / 2

            process.wait()

            training_progress['status'] = 'Початок навчання моделі...'

            training_progress['log'] += '\nЗапуск train_model.py\n'

            train_process = subprocess.Popen(

                ['python', '../training/train_model.py'],

                stdout=subprocess.PIPE,

                stderr=subprocess.STDOUT,

                universal_newlines=True)

            for line in train_process.stdout:

                training_progress['log'] += line
```

```
                if 'Epoch' in line and '/' in line:
                    epoch_info = line.split('Epoch ')[1].split('/')[0]
                    epoch = int(epoch_info)
                    training_progress['progress'] = 50 + (epoch / 400 * 50)
            train_process.wait()
            training_progress['status'] = 'Навчання завершено!'
            training_progress['progress'] = 100
            training_progress['completed'] = True
        except Exception as e:
            training_progress['status'] = f'Помилка: {str(e)}'
            training_progress['log'] += f'\nПомилка: {str(e)}\n'
    thread = threading.Thread(target=run_process)
    thread.start()
    return jsonify({'status': 'started'})
@app.route('/training_progress')
def training_progress_stream():
    def generate():
        last_sent = 0
        while True:
            if training_progress['progress'] > last_sent or training_progress['completed']:
                data = {
                    'progress': training_progress['progress'],
                    'status': training_progress['status'],
                    'log': training_progress['log'],
                    'completed': training_progress['completed']}
                yield f"data: {json.dumps(data)}\n\n"
                last_sent = training_progress['progress']
                if training_progress['completed']:
                    break
            time.sleep(1)
```

```
    return Response(generate(), mimetype='text/event-stream')
@app.route('/training_results')
def training_results():
    try:
        return send_from_directory('../result', 'training_report.html')
    except Exception as e:
        return jsonify({'status': 'error', 'message': str(e)}), 500
```

Лістинг Ґ.2 — Алгоритм обробки кадрів, формування послідовності та передбачення жесту нейронною мережею

```
def process_frames():
    while True:
        try:
            data = frame_queue.get()
            if data is None:
                break
            user_id, img_data = data
            with lock:
                if user_id not in user_sessions:
                    user_sessions[user_id] = {
                        "sequence": [],
                        "current_gesture": {"gesture": None, "confidence": 0.0,
"processing_time": 0}}
                user_data = user_sessions[user_id]
            nparr = np.frombuffer(base64.b64decode(img_data.split(',')[1]), np.uint8)
            frame = cv2.imdecode(nparr, cv2.IMREAD_COLOR)
            keypoints = extract_keypoints(frame)
            if keypoints is not None:
                with lock:
                    user_data["sequence"].append(keypoints)
                    user_data["sequence"] = user_data["sequence"][-30:]
                    if len(user_data["sequence"]) == 30:
                        prediction_start = time.time()
                        prediction = model.predict(np.expand_dims(user_data["sequence"],
axis=0), verbose=0)[0]
                        processing_time = int((time.time() - prediction_start) * 1000)
                        class_idx = np.argmax(prediction)
                        confidence = float(prediction[class_idx])
                        if confidence > 0.6:
```

```
                user_data["current_gesture"] = {
                    "gesture": inverse_labels.get(class_idx, "Unknown"),
                    "confidence": confidence,
                    "processing_time": processing_time}
            else:
                user_data["current_gesture"] = {
                    "gesture": None,
                    "confidence": confidence,
                    "processing_time": processing_time}
except Exception as e:
    logger.error(f"Помилка в потоці обробки: {e}")
```