ОДЕСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ імені І.І.МЕЧНИКОВА
(повне найменування вищого навчального закладу)

Факультет математики, фізики та інформаційних технологій
(повне найменування інституту, назва факультету (відділення))

Кафедра комп'ютерних систем та технологій
(повна назва кафедри (предметної, циклової комісії))

Кваліфікаційна робота

на здобуття ступеня вищої освіти «бакалавр»

«Моделювання нейромережевої системи розпізнавання та перекладу мови жестів»

«Modeling a neural network system for sign language recognition and translation»

Виконав: здобувачка денної (очної) форми навчання

спеціальності 122 Комп'ютерні науки

Освітня програма Комп'ютерні науки
(назва освітньої програми)

Воінова Олександра Володимирівна
(прізвище, ім'я, по-батькові здобувача)

Керівник д.т.н., проф. Гунченко Ю. О.
(науковий ступінь, звання, прізвище та ініціали, підпис)

Рецензент к.т.н. доц.Камєнєва А.В.
(науковий ступінь, вчене звання, прізвище та ініціали)

Рекомендовано до захисту:
Протокол засідання кафедри
комп'ютерних систем і технологій

№ ___ від «___» _____ 2026 р

Завідувач кафедри
_____ Юрій ГУНЧЕНКО
(підпис)

Захищено на засіданні ЕК № ___
протокол № __ від «__» ___2026 р

Оцінка _____ / _____ / _____
(за національною шкалою/шкалою ECTS/ бали)

Голова ЕК
_____ Микола МАЛАКСІАНО
(підпис)

Одеса - 2026

АНОТАЦІЯ

Тема кваліфікаційної роботи «Моделювання нейромережевої системи розпізнавання та перекладу мови жестів».

Мета роботи — створення програмної системи розпізнавання літер дактильної абетки американської жестової мови (ASL) у реальному часі.

Кваліфікаційна робота присвячена розробці та дослідженню нейромережевої системи розпізнавання жестів американської жестової мови на основі методів комп'ютерного зору та глибокого навчання. Актуальність роботи обумовлена необхідністю створення доступних інструментів комунікації для людей з порушеннями слуху та розвитком технологій автоматизованого перекладу жестової мови.

У роботі використано набір даних ASL Alphabet Dataset, для якого виконано стратифіковане тричастинне розбиття на навчальну, валідаційну та тестову вибірки у співвідношенні 70 % / 15 % / 15 %. Проведено дослідження та порівняння декількох архітектур згорткових нейронних мереж, а також застосовано методи аугментації даних для підвищення стійкості моделей до змін умов зйомки та положення руки. За результатами експериментів було обґрунтовано вибір архітектури нейромережевого класифікатора жестів.

Запропонований підхід забезпечує прийнятну точність розпізнавання жестів при зменшених обчислювальних витратах та дозволяє використовувати легковажну модель на локальних персональних комп'ютерах без застосування графічного прискорювача. Практичним результатом роботи є програмна система, що підтримує розпізнавання жестів у режимі реального часу та може бути використана як основа для подальшого розвитку засобів автоматизованого перекладу жестової мови.

Ключові слова: жестова мова, американська жестова мова, ASL, комп'ютерний зір, розпізнавання жестів, згорткова нейронна мережа, нейронна мережа, класифікація зображень, машинне навчання, глибоке навчання, аугментація даних.

ABSTRACT

The topic of the qualification work is “Modeling a neural network system for recognizing and translating sign language”.

The purpose of the work is to create a software system for recognizing letters of the dactylic alphabet of American Sign Language (ASL) in real time.

The qualification work is devoted to the development and research of a neural network system for recognizing American Sign Language gestures based on computer vision and deep learning methods. The relevance of the work is due to the need to create accessible communication tools for people with hearing impairments and the development of automated sign language translation technologies.

The work uses the ASL Alphabet Dataset dataset, for which a stratified three-part division into training, validation and test samples was performed in the ratio of 70% / 15% / 15%. Several architectures of convolutional neural networks were studied and compared, and data augmentation methods were also applied to increase the stability of models to changes in shooting conditions and hand position. The experimental results justified the choice of the architecture of the neural network gesture classifier.

The proposed approach provides acceptable accuracy of gesture recognition with reduced computational costs and allows the use of a lightweight model on local personal computers without the use of a graphics accelerator. The practical result of the work is a software system that supports gesture recognition in real time and can be used as a basis for further development of automated sign language translation tools.

Keywords: sign language, American Sign Language, ASL, computer vision, gesture recognition, convolutional neural network, neural network, image classification, machine learning, deep learning, data augmentation.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ І СКОРОЧЕНЬ 6
ВСТУП 7
РОЗДІЛ 1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ІСНУЮЧИХ РІШЕНЬ 10
1.1 Жестові мови як засіб комунікації 10
1.2 Огляд методів розпізнавання жестів 12
1.2.1 Класичні методи комп'ютерного зору 12
1.2.2 Згорткові нейронні мережі 13
1.2.3 Підходи на основі ключових точок (landmark-based) 13
1.2.4 Гібридні рішення та сучасні тенденції 14
1.3 Огляд існуючих програмних рішень 15
1.4 Постановка задачі 16
Висновки до розділу 1 17
РОЗДІЛ 2 ТЕОРЕТИЧНІ ОСНОВИ ТА ОБРАНІ МЕТОДИ 18
2.1 Архітектура MediaPipe HandLandmarker 18
2.2 Багатошаровий перцептрон як класифікатор 19
2.3 Функція втрат і алгоритм оптимізації 21
2.4 Регуляризація і запобігання перенавчанню 22
2.5 Метрики оцінки якості класифікації 23
2.6 Обґрунтування feature engineering на основі геометричних відстаней . 24
Висновки до розділу 2 25
РОЗДІЛ 3 ПРОЄКТУВАННЯ ТА РЕАЛІЗАЦІЯ СИСТЕМИ 26
3.1 Архітектура системи 26
3.2 Вибір технологічного стеку 27
3.3 Опис тренувального датасету 27
3.4 Реалізація виділення ознак 29
3.5 Архітектура нейромережі 30
3.6 Стратифіковане 3-частинне розбиття вибірки 31
3.7 Реалізація режиму розпізнавання у реальному часі 32
Висновки до розділу 3 33
РОЗДІЛ 4 ЕКСПЕРИМЕНТАЛЬНІ ДОСЛІДЖЕННЯ 34
4.1 Методика проведення експериментів 34
4.2 Результати тренування моделей 35

4.3 Порівняльна оцінка моделей на незалежній тестовій вибірці 37
4.4 Аналіз помилок класифікації 39
4.5 Pareto-аналіз компромісів 41
4.6 Обмеження методу та шляхи подальшого розвитку 42
Висновки до розділу 4 43
ВИСНОВКИ 44
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 47
ДОДАТОК А 50
А.1 Модуль виділення ознак (1_extract_features.py) 50
А.2 Тренування запропонованої моделі (2a_train_distances.py) 51
А.3 Тренування альтернативної моделі (2b_train_raw.py) 52
А.4 Тренування альтернативної моделі (2c_train_cnn.py) 53
А.5 Повна оцінка натренованих моделей (3_evaluate.py) 54
А.6 Зведене порівняння трьох моделей (4_compare_models.py) 55
А.7 Розпізнавання жестів у реальному часі через веб-камеру (5_inference.py) 56
ДОДАТОК Б 58
ДОДАТОК В 60

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ І СКОРОЧЕНЬ

ASL – American Sign Language (американська жестова мова)

CNN – Convolutional Neural Network (згорткова нейронна мережа)

MLP – Multilayer Perceptron (багатошаровий перцептрон)

HOG – Histogram of Oriented Gradients (гістограма орієнтованих градієнтів)

ReLU – Rectified Linear Unit (функція активації лінійний випрямляч)

FPS – Frames Per Second (частота кадрів за секунду)

КР – кваліфікаційна робота

МЛ – машинне навчання

НМ – нейронна мережа

ПЗ – програмне забезпечення

ВСТУП

Жестові мови є природним засобом спілкування для людей з вадами слуху та мовлення. За оцінками Всесвітньої федерації глухих, у світі налічується понад 70 мільйонів людей, які користуються жестовими мовами як основним засобом комунікації [1]. Кожна національна жестова мова має власну граматику, лексику та абетку, причому американська жестова мова (англ. American Sign Language, далі — ASL) є однією з найпоширеніших у світі. Її дактильна (пальцьова) абетка слугує важливим допоміжним інструментом для побуквенного відтворення власних назв, термінів і слів, які не мають окремого жестового позначення [2].

Створення програмних засобів автоматичного розпізнавання жестових мов залишається актуальною науково-практичною задачею як в галузі комп’ютерного зору, так і в галузі асистивних технологій. Поширення відеокамер у мобільних пристроях, ноутбуках та інтернет-камерах, а також стрімкий розвиток методів машинного навчання створюють передумови для побудови доступних інтерактивних систем перекладу, які могли б полегшити повсякденне спілкування людей з порушеннями слуху з тими, хто жестовою мовою не володіє.

Існуючі підходи до розпізнавання дактильної абетки умовно поділяють на дві групи. Першу складають методи, що працюють безпосередньо з пікселями зображення, серед яких — згорткові нейронні мережі (англ. Convolutional Neural Networks, далі — CNN). Другу групу складають так звані landmark-based методи, у яких на першому етапі з кадру виділяються ключові точки скелета руки, після чого класифікація здійснюється за компактним вектором геометричних ознак. Обидві групи мають свої переваги та обмеження. CNN досягають високої точності на однорідному датасеті, але є вимогливими до обчислювальних ресурсів і чутливими до зовнішніх умов зйомки. Landmark-based підходи поступаються у деталізації представлення, але є інваріантними до фону, освітлення та масштабу руки, що робить їх

привабливими для застосувань у реальному часі на пристроях без графічного прискорювача [3].

Зважаючи на вищезазначене, актуальність обраної теми зумовлюється потребою у створенні легких, швидкодіючих і стійких до зовнішніх умов нейромережевих систем розпізнавання жестів дактильної абетки, придатних до використання у режимі реального часу на стандартному персональному комп'ютері без використання спеціалізованого обладнання.

Метою роботи є створення програмної системи розпізнавання літер дактильної абетки американської жестової мови (ASL) у реальному часі.

Для досягнення поставленої мети необхідно розв'язати наступні технічні задачі:

1) Проаналізувати предметну область, оглянути існуючі методи розпізнавання дактильної абетки та програмні рішення;
2) Обґрунтувати вибір двоступеневої архітектури системи з використанням бібліотеки MediaPipe HandLandmarker як засобу виділення ключових точок руки;
3) Запропонувати схему feature engineering на основі 24 нормалізованих геометричних відстаней, інваріантну до масштабу та орієнтації руки;
4) Розробити архітектуру багатошарового перцептрона та виконати його тренування на датасеті ASL Alphabet Dataset;
5) Для порівняльної оцінки натренувати дві альтернативні моделі — багатошаровий перцептрон на сирих координатах ключових точок та згорткову нейронну мережу на зображеннях;
6) Провести експериментальне дослідження точності, швидкодії та розміру моделей із застосуванням стратифікованого 3-частинного розбиття вибірки та confusion matrix;
7) Реалізувати програмний модуль розпізнавання у реальному часі, який виводить розпізнаний жест та значення впевненості моделі.

Для розв'язання поставлених задач використано методи комп'ютерного зору (виділення ключових точок руки за допомогою згорткової мережі

BlazePalm у складі MediaPipe HandLandmarker), методи машинного навчання (багатошаровий перцептрон, згорткові нейронні мережі, алгоритм оптимізації Adam, регуляризація Dropout), методи статистичної оцінки якості класифікаторів (стратифікована перехресна перевірка, метрики Precision, Recall, F1-score, аналіз матриці помилок), а також методи об’єктно-орієнтованого програмування мовою Python.

Кваліфікаційна робота складається зі вступу, чотирьох розділів, висновків, списку використаних джерел та додатків. У першому розділі проаналізовано предметну область, оглянуто методи розпізнавання дактильної абетки та сформульовано постановку задачі. У другому розділі описано теоретичні основи обраних методів. У третьому розділі викладено архітектурні рішення та програмну реалізацію системи. У четвертому розділі наведено результати експериментального дослідження. У висновках узагальнено отримані результати.

РОЗДІЛ 1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ІСНУЮЧИХ РІШЕНЬ

1.1 Жестові мови як засіб комунікації

Жестова мова — це повноцінна природна мова з власною граматикою, синтаксисом, морфологією та лексикою, що використовує для передачі значень рухи рук, форму долоні, орієнтацію пальців, а також міміку та позу тіла. Незважаючи на поширене у непрофесійному середовищі уявлення, жестові мови не є візуальними копіями розмовних мов і не зводяться до жестового алфавіту. Кожна країна або регіон має свою національну жестову мову — британську (BSL), українську (УЖМ), американську (ASL), німецьку (DGS) тощо, причому ці мови не є взаємно зрозумілими [1, 4].

Американська жестова мова, обрана предметом цієї роботи, є офіційно визнаною жестовою мовою глухої спільноти США та Канади. За даними Національного інституту глухоти США (NIDCD), ASL є четвертою за кількістю користувачів мовою у Сполучених Штатах. Граматика ASL суттєво відрізняється від англійської: вона має SVO-порядок слів, мінімально використовує службові слова, активно застосовує просторову локалізацію об’єктів і одночасну подачу декількох ознак за рахунок паралельного використання обох рук, виразу обличчя та руху корпусу [4].

У жестовій мові виокремлюють декілька рівнів організації знаків. Перший — лексикалізовані жести, що позначають окремі поняття чи слова цілком (так звані «sign-words»). Другий — дактилеми, тобто жести-літери дактильної (пальцьової) абетки, які використовуються для побуквенного відтворення тих слів, які не мають окремого знаку: власних назв, рідкісних термінів, неологізмів, назв брендів. Дактилеми ASL стандартизовані і кожна літера латинського алфавіту має власну сталу позу долоні і пальців (рис. 1.1), що відрізняє ASL-дактилологію від дворучних алфавітів (BSL) і дозволяє виконувати дактилування однією рукою.

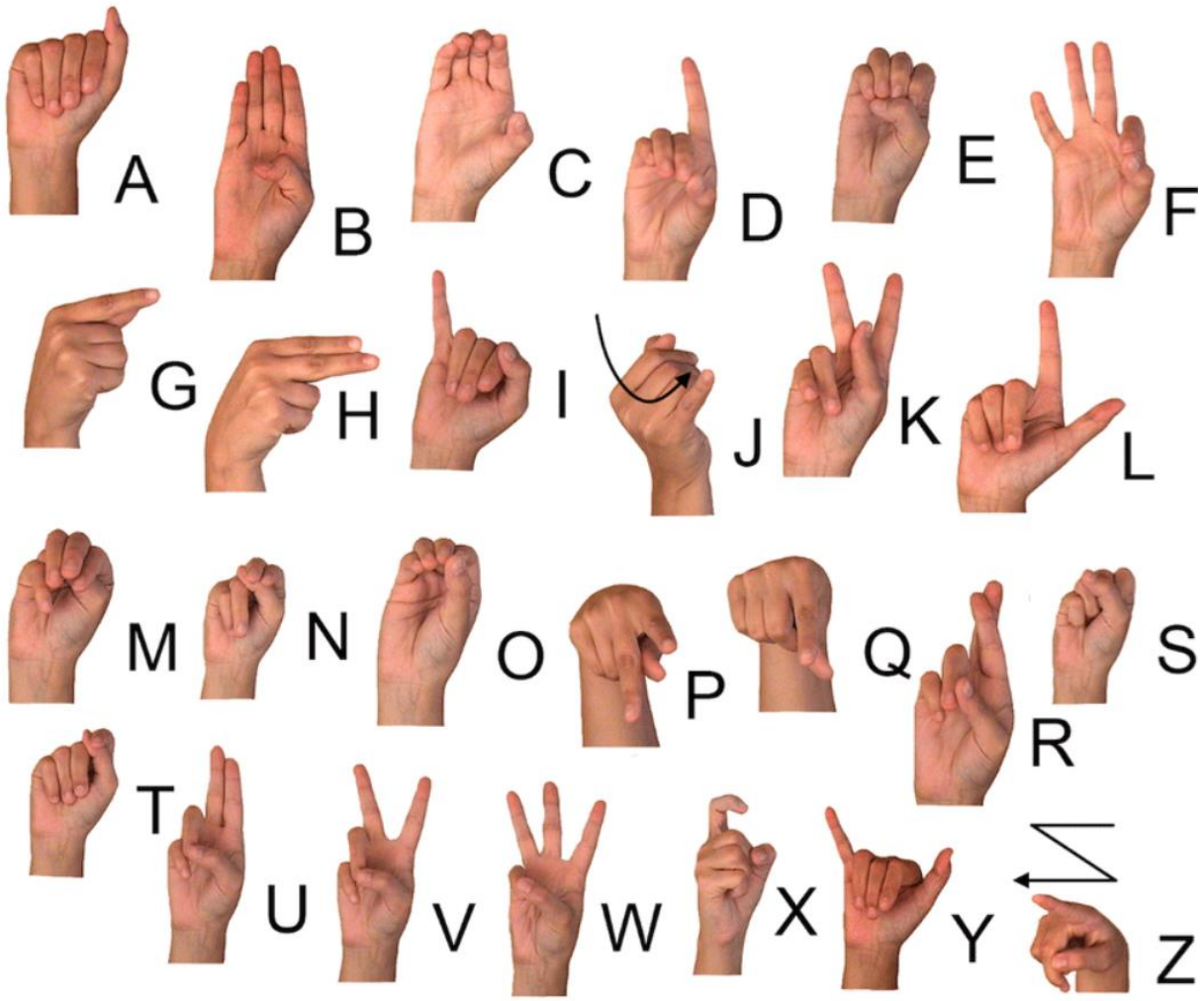


Рисунок 1.1 – Дактильний алфавіт ASL

Завдання автоматичного розпізнавання жестових мов прийнято поділяти на дві відмінні задачі. Перша — розпізнавання ізольованих жестів, у тому числі дактилем, коли жест статично утримується протягом певного інтервалу часу. Друга — розпізнавання неперервної жестової мови, коли необхідно сегментувати потік неперервних жестів і виділити з нього окремі знаки з урахуванням контексту і перехідних рухів. Друга задача суттєво складніша і досі вважається відкритою проблемою у машинному навчанні [3, 5].

У межах кваліфікаційної роботи розглядається саме перша задача — розпізнавання статичних дактилем, тобто фіксація літери, яку користувач показує перед камерою у вигляді усталеної пози долоні. Такий вибір зумовлений двома причинами. По-перше, для розпізнавання дактильної

абетки у вільному доступі існують великі датасети, а сама абетка має чітко визначену скінченну множину класів (26 літер латинського алфавіту), що дозволяє формалізувати задачу як класичну задачу багатокласової класифікації. По-друге, побудова робочого розпізнавача дактильної абетки є необхідним фундаментальним кроком, без якого неможлива побудова повноцінного перекладача жестових мов.

1.2 Огляд методів розпізнавання жестів

Історичний розвиток методів автоматичного розпізнавання жестів умовно поділяють на три покоління. Перше покоління спирається на ручне feature engineering з використанням класичних дескрипторів комп'ютерного зору. Друге — на end-to-end-навчання глибоких нейронних мереж безпосередньо на сирих зображеннях. Третє — на гібридні підходи, у яких глибока мережа використовується для виділення семантично змістовних ознак, після чого класифікація здійснюється легким класифікатором [3, 6].

1.2.1 Класичні методи комп'ютерного зору

До появи глибокого навчання (приблизно до 2012 року) розпізнавання жестів виконувалось на основі вручну сконструйованих ознак. Найпоширенішими дескрипторами були гістограма орієнтованих градієнтів (HOG), масштабно-інваріантна перетворююча ознака (SIFT), бінарні шаблонні ознаки (BRIEF, ORB), а також статистики кольору шкіри в HSV/YCbCr-просторах. Виділені ознаки подавались на класичні класифікатори: метод опорних векторів (SVM), метод k найближчих сусідів (k-NN), випадковий ліс (Random Forest) [6].

Перевагою таких підходів є їхня прозорість і обчислювальна простота. Однак вони мають суттєві обмеження: висока чутливість до зміни освітлення, фону, текстури шкіри руки, кута повороту і відстані до камери. Як показано в

[7], точність класичних класифікаторів дактильної абетки рідко перевищує 75–85 % навіть на однорідному датасеті, і різко падає при перевірці на відмінних умовах зйомки.

1.2.2 Згорткові нейронні мережі

Поява глибокого навчання та архітектур AlexNet [8], VGG [9], ResNet [10] суттєво змінила ландшафт задачі розпізнавання зображень. Згорткова нейронна мережа автоматично навчається ієрархічної ієрархії ознак: нижні шари виділяють краї та текстури, середні — частини об’єкта, вищі — семантичні концепції. Стосовно дактильної абетки CNN досягають точності 95–99 % на стандартних датасетах [11].

Незважаючи на високу точність, чисто-CNN-підходи мають низку недоліків. По-перше, вони вимогливі до обчислювальних ресурсів, а саме тренування CNN середнього розміру може зайняти години навіть на сучасній відеокарті. По-друге, CNN, навчена на однорідному датасеті, схильна до перенавчання на специфічних рисах фону та освітлення. Точність на out-of-distribution даних може катастрофічно падати. По-третє, розмір моделі (від десятків мегабайт до сотень) ускладнює розгортання на пристроях з обмеженими ресурсами [12].

1.2.3 Підходи на основі ключових точок (landmark-based)

Альтернативою чисто-CNN-підходам є двоступеневий конвеєр обробки. На першому етапі з зображення виділяється скелет руки — структуроване представлення у вигляді набору ключових точок (анатомічних маркерів). На другому — компактний класифікатор (MLP, SVM, Random Forest) приймає рішення на основі вектора ознак, обчислених із координат точок. Перевага схеми полягає в тому, що перший етап вже виконує неявну сегментацію руки

і відкидає інформацію про фон та освітлення, передаючи на другий етап тільки геометричну інформацію [3, 13].

Серед готових засобів для виділення скелета руки найвідомішим є MediaPipe Hands від Google (з 2020 року поширюється під назвою MediaPipe HandLandmarker) [13]. Це двоступенева система, де перша підмережа BlazePalm виявляє долоню на зображенні, а друга — регресує 21 анатомічну ключову точку (зап’ясток, вузли і кінчики пальців). MediaPipe HandLandmarker працює в реальному часі на CPU, а на мобільних пристроях — за рахунок апаратного прискорення.

Важливою перевагою landmark-based-підходів є зменшення розмірності вхідних даних. Якщо під час обробки зображення нейронна мережа аналізує десятки тисяч пікселів, то після виділення скелетної моделі руки подальша обробка виконується лише для невеликого набору координат ключових точок. Це дозволяє суттєво скоротити обчислювальні витрати, зменшити вимоги до апаратного забезпечення та підвищити швидкодію системи без істотної втрати якості розпізнавання. Саме тому landmark-based-підходи набули значного поширення у задачах реального часу та на пристроях з обмеженими обчислювальними ресурсами.

У поєднанні з легким класифікатором landmark-based підхід дозволяє побудувати real-time-розпізнавач дактильної абетки, який має на порядки менший розмір моделі та на порядки вищу швидкодію в порівнянні з end-to-end CNN, при незначній втраті точності. Саме цей підхід обрано основою кваліфікаційної роботи.

1.2.4 Гібридні рішення та сучасні тенденції

Окремо слід зазначити сучасну тенденцію до використання трансформерів та архітектури self-attention для розпізнавання жестів і жестових мов [14]. Такі моделі забезпечують результати на задачах розпізнавання неперервної жестової мови, проте їхня обчислювальна вартість

на порядки вища за CNN, а тренування потребує значно більших датасетів. Для ізольованої задачі дактильної абетки трансформери не дають суттєвої переваги і не виправдовують своєї складності.

У 2022–2023 роках з'являлись також підходи на основі графових нейронних мереж (Graph Neural Networks), які трактують скелет руки як граф із 21 вузла та фіксованими ребрами зв'язків між фалангами [15]. Такі моделі демонструють добру стійкість до варіацій орієнтації руки, проте поступаються MLP за швидкодією на CPU.

1.3 Огляд існуючих програмних рішень

У відкритому доступі представлено низку програмних рішень з розпізнавання жестів дактильної абетки ASL. Більшість із них реалізовано як демонстраційні відкриті проєкти на платформі GitHub та споріднених. Огляд найвідоміших з них наведено далі.

Перший представник — проєкт «Sign-Language-Recognition» [16], реалізований мовою Python з використанням бібліотеки TensorFlow і застосовує згорткову нейромережу архітектури VGG-16 на зображеннях. Розмір моделі сягає 500 МБ, точність на тестовому наборі — 96 %. Недоліки: висока обчислювальна вартість, відсутність real-time-демо.

Другий — проєкт «ASL-Alphabet-Recognition» [17] на основі бібліотеки PyTorch з власною компактною CNN. Розмір моделі — 8 МБ, точність — 92 %. Робочий real-time-демо реалізовано на бібліотеці OpenCV. Недолік — відсутність порівняння з landmark-based-підходами.

Третій — комерційний проєкт «GoSign» [18] від компанії SignAll Technologies, що використовує фірмову апаратну Kinect-камеру з виявленням глибини. Технічних деталей у відкритому доступі не наведено. Недолік — потреба у спеціалізованому обладнанні.

Аналіз існуючих рішень показав, що:

1) переважна більшість досліджень спирається на CNN, тоді як landmark-based-підходи представлені непропорційно слабко.
2) у відкритих джерелах немає опублікованих робіт із систематичним порівнянням MLP на геометричних відстанях, MLP на сирих координатах та CNN на одному датасеті в єдиній методології оцінки.
3) обмеження CNN-підходів щодо обчислювальної ефективності та чутливості до фону залишаються невирішеними.

1.4 Постановка задачі

На основі проведеного аналізу формалізовано постановку задачі кваліфікаційної роботи.

Дано: набір зображень рук, що демонструють жести літер дактильної абетки ASL, з відомою міткою класу для кожного зображення. Тренувальний набір становить датасет ASL Alphabet Dataset обсягом приблизно 87 000 зображень розміром 200×200 пікселів, розподілених за 29 класами (літери A–Z, плюс жести «space», «del», «nothing»). Зображення зняті у студійних умовах з рівномірним фоном.

Потрібно: побудувати програмну систему, яка за відеопотоком веб-камери (роздільність 640×480, частота 30 кадрів за секунду, кольорова модель RGB) у реальному часі визначає літеру дактильної абетки, що відповідає поточній позі руки користувача. Система повинна виводити на екран ім’я розпізнаного класу та значення впевненості моделі у відсотках, працювати на стандартному персональному комп’ютері без використання графічного прискорювача та забезпечувати швидкодію не нижче 25 кадрів за секунду.

Критерії успіху: точність розпізнавання на незалежному тестовому наборі не нижче 90 %; розмір моделі — не більше 1 МБ; час обробки одного кадру — не більше 40 мілісекунд на CPU.

Обмеження: розглядається лише статичне розпізнавання поз окремих літер, без урахування динамічних компонентів жестів («J», «Z», які виконуються рухом). Розпізнавання слів та речень неперервної жестової мови, а також інших дактильних абеток (BSL, УЖМ) виходить за межі цієї роботи. Не розглядаються апаратні розв'язання з RGB-D-камерою або сенсорами руху.

Допоміжна задача — порівняльне дослідження. Окрім запропонованого підходу (MLP на 24 геометричних відстанях), для порівняння натреновано та оцінено:

a) MLP на 63 нормалізованих координатах ключових точок.
b) згорткову нейромережу на зображеннях.

Усі моделі тренуються та тестуються на однаковому стратифікованому 3-частинному розбитті датасету (70 % / 15 % / 15 %), що забезпечує чесну порівняльну оцінку.

Висновки до розділу 1

У першому розділі проаналізовано предметну область кваліфікаційної роботи. Подано загальну характеристику жестових мов та особливостей дактильної абетки американської жестової мови. Розглянуто три покоління методів автоматичного розпізнавання жестів: класичні методи комп'ютерного зору, чисто-CNN-підходи та гібридні landmark-based-підходи. Виявлено, що landmark-based-підхід забезпечує найкращий компроміс між точністю, швидкодією та розміром моделі для задач реального часу на CPU.

Проведено огляд існуючих програмних рішень і виявлено брак досліджень із систематичним порівнянням landmark-based- та image-based-підходів на єдиній методологічній основі. Сформульовано постановку задачі кваліфікаційної роботи, обумовлено критерії успіху та допоміжну задачу порівняльного експериментального дослідження.

РОЗДІЛ 2 ТЕОРЕТИЧНІ ОСНОВИ ТА ОБРАНІ МЕТОДИ

2.1 Архітектура MediaPipe HandLandmarker

MediaPipe HandLandmarker — це модуль зі складу відкритої бібліотеки MediaPipe, розробленої компанією Google для побудови мультимодальних конвеєрів обробки даних у реальному часі. Модуль призначений для виявлення рук та виділення на зображенні 21 анатомічної ключової точки скелета руки. Архітектура модуля складається з двох послідовних згорткових нейронних мереж: BlazePalm — для локалізації обмежувальних рамок долонь, та HandLandmark — для регресії координат ключових точок у виділеній області [13].

Перша мережа BlazePalm побудована за принципами архітектури Single Shot MultiBox Detector (SSD), оптимізованої для низької затримки. Вона приймає на вхід зображення розміром 256×256 пікселів та виводить набір прямокутних регіонів, кожен з яких потенційно містить долоню. Особливістю BlazePalm є фокус на детекції долоні (а не всієї руки), оскільки долоня має жорсткішу геометрію, ніж пальці, і легше детектується на різних позах. Друга мережа HandLandmark отримує на вхід регіон долоні, повертає його у канонічну орієнтацію та регресує координати 21 ключової точки. Розташування ключових точок наведено на рис. 2.1.

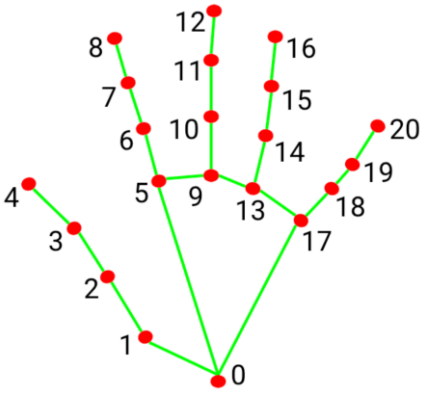


Рисунок 2.1 - Схема розташування 21 ключової точки руки

Кожна ключова точка має тривимірну координату (x, y, z), де x та y — нормалізовані координати у площині зображення (значення від 0 до 1), а z — глибина у системі координат, центрованій на зап’ястку, із масштабом, нормалізованим відносно розміру долоні. Семантично точки розташовані у фіксованих анатомічних позиціях: точка 0 — зап’ясток, точки 1–4 — фаланги великого пальця, точки 5–8 — вказівного, 9–12 — середнього, 13–16 — безіменного, 17–20 — мізинця. Така стандартизована схема дозволяє будувати на її основі різноманітні класифікатори жестів.

У документації [13] зазначено, що модель MediaPipe HandLandmarker натренована на власному датасеті Google обсягом понад 30 000 зображень рук, анотованих експертами. Точність регресії на тестовому наборі — медіанна похибка приблизно 2 % від розміру долоні, що є достатнім для більшості практичних задач. Модель доступна у форматі TensorFlow Lite і має розмір приблизно 8 МБ для CPU-варіанта.

2.2 Багатошаровий перцептрон як класифікатор

Багатошаровий перцептрон (англ. Multilayer Perceptron, MLP) — це клас прямоспрямованих штучних нейронних мереж, що складається з вхідного шару, одного або декількох прихованих шарів, та вихідного шару. Кожен шар містить набір нейронів-перцептронів, кожен з яких обчислює зважену суму своїх входів та застосовує до неї нелінійну функцію активації. Зв’язки між шарами повнозв’язні: вихід кожного нейрона попереднього шару подається на вхід кожному нейрону наступного шару [19]. Загальну структуру багатошарового перцептрона наведено на рис. 2.1.

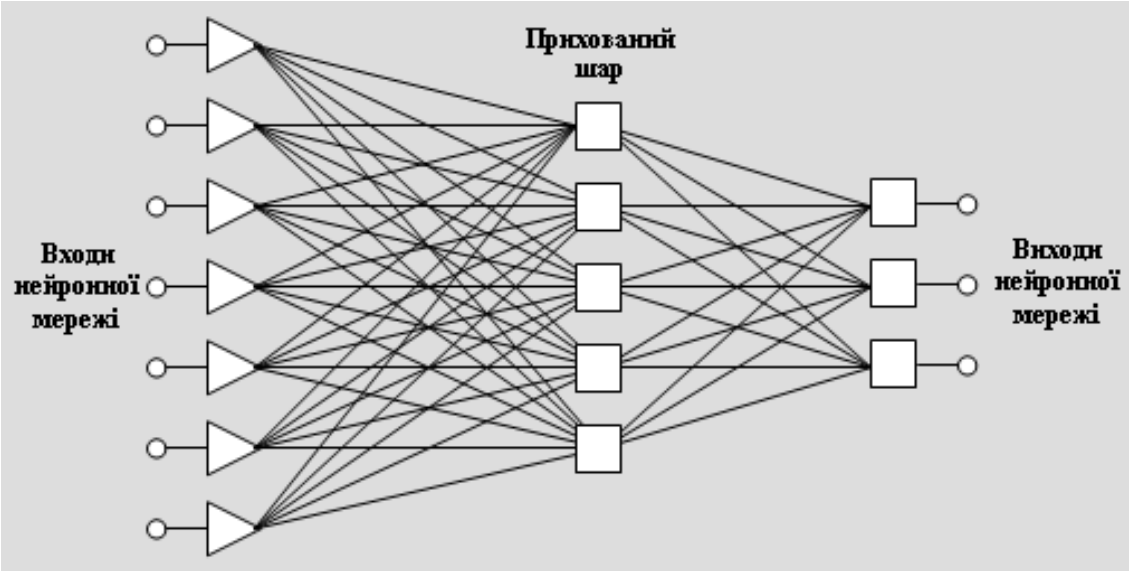


Рисунок 2.1 - Багатошаровий перцептрон

Математично прямий прохід через шар MLP описується рівнянням (2.1):

$$h = f(W * x + b), \tag{2.1}$$

де:

- x — вектор входу розміру n;
- W — матриця вагових коефіцієнтів розміру m×n;
- b — вектор зсувів розміру m;
- f — нелінійна функція активації, що застосовується поелементно;
- h — вектор виходу розміру m.

Найпоширенішою функцією активації прихованих шарів є ReLU (Rectified Linear Unit), що визначається рівнянням (2.2):

$$ReLU(x) = \max(0, x)\ , \tag{2.2}$$

Перевагами ReLU над класичним сигмоїдом і гіперболічним тангенсом є простота обчислення, відсутність насичення в позитивній області, що сприяє стабільному поширенню градієнта при тренуванні глибоких мереж [20].

У задачах багатокласової класифікації останнім шаром мережі є softmax-шар, що нормалізує вихід так, щоб сума ймовірностей класів дорівнювала одиниці. Функція softmax визначається рівнянням (2.3):

$$softmax(z)_i = \frac{\exp(z_i)}{\sum_j \exp(z_j)}, \tag{2.3}$$

де:

- z — вхідний вектор передостаннього шару (логіти);
- i — індекс класу;
- знаменник — нормалізуюча сума за всіма класами.

2.3 Функція втрат і алгоритм оптимізації

Стандартним вибором функції втрат для задачі багатокласової класифікації є категоріальна крос-ентропія (cross-entropy loss). У формулюванні sparse_categorical_crossentropy, що використовується у цій роботі, мітки класів подаються у вигляді цілих чисел, а функція втрат для одного зразка обчислюється за формулою (2.4):

$$L(y, \hat{y}) = -\log(\hat{y}_y), \tag{2.4}$$

де:

- y — справжній індекс класу;
- $\hat{y}_y$ — передбачена моделлю ймовірність класу y.

Тренувальна цільова функція є середньою величиною L по всім зразкам тренувального набору.

Для оптимізації параметрів мережі застосовано алгоритм Adam (Adaptive Moment Estimation) [21]. Adam — це стохастичний градієнтний метод, що використовує адаптивні швидкості навчання для кожного параметра окремо. Метод поєднує переваги двох інших розширень класичного стохастичного градієнтного спуску: AdaGrad (адаптація швидкості за історією

градієнтів) та RMSProp (експоненційне згладжування історії). Оновлення параметрів описується системою рівнянь (2.5):

$$\begin{cases} m_t = \beta_1 * m_{\{t-1\}} + (1-\beta_1) * g_t, \\ v_t = \beta_2 * v_{\{t-1\}} + (1-\beta_2) g_{t^2}, \\ \theta_t = \theta_{\{t-1\}} - \alpha * \frac{\widehat{m}_t}{\sqrt{\widehat{v}_t + \varepsilon}}, \end{cases} \quad (2.5)$$

де:

- g_t — градієнт цільової функції за параметрами на ітерації t;
- m_t, v_t — експоненційно згладжені оцінки першого та другого моментів градієнта;
- $\beta_1 = 0.9$, $\beta_2 = 0.999$ — параметри згладжування;
- $\widehat{m}_t$, $\hat{v}_t$ — їх скориговані за зміщенням оцінки;
- α — глобальна швидкість навчання (за замовчуванням 0.001);
- $\varepsilon = 10^{-8}$ — мала константа для уникнення поділу на нуль;
- θ_t — параметри моделі.

Adam широко застосовується у практиці глибокого навчання завдяки своїй стійкості до вибору гіперпараметрів та збіжності на широкому класі задач без додаткового підлаштування.

2.4 Регуляризація і запобігання перенавчанню

Перенавчання — це явище, за якого модель добре виконує класифікацію на тренувальному наборі, але демонструє суттєво нижчу точність на даних, які вона не бачила. Стандартною технікою боротьби з перенавчанням у нейронних мережах є dropout-регуляризація [22]. Її ідея полягає у тому, що під час тренування на кожному кроці випадковим чином «вимикається» (множиться на нуль) певна частка нейронів прихованого шару. Це примушує мережу не покладатись на жоден окремий нейрон і навчитись робити прогнози за допомогою колективної інформації, розподіленої між багатьма нейронами.

Формально dropout-шар з параметром p (ймовірність вимкнення нейрона) під час тренування виконує покомпонентне множення на маску Бернуллі: h' = h ⊙ Bernoulli(1-p) / (1-p). Поділ на (1-p) забезпечує те, що математичне сподівання активації після dropout залишається таким самим, як і без нього. Під час інференсу dropout вимкнено, всі нейрони активні. У запропонованому MLP застосовано dropout з p=0.2 після кожного прихованого шару.

2.5 Метрики оцінки якості класифікації

Для всебічної оцінки якості багатокласового класифікатора використано стандартний набір метрик: точність (accuracy), точність-влучність (precision), повнота (recall), F-міра, а також матриця помилок (confusion matrix) [23].

Загальна точність визначається як відношення кількості правильно класифікованих зразків до загальної кількості зразків (формула 2.6):

$$Accuracy = \frac{(TP+TN)}{(TP+FP+TN+FN)}, \tag{2.6}$$

де:

- TP — істинно позитивні;
- FP — хибно позитивні;
- TN — істинно негативні;
- FN — хибно негативні класифікації.

У задачі багатокласової класифікації accuracy обчислюється як частка зразків, для яких передбачений клас збігається зі справжнім.

Метрики precision і recall визначаються для кожного класу окремо за формулами (2.7) і (2.8):

$$Precision = \frac{TP}{TP+FP}, \tag{2.7}$$

$$Recall = \frac{TP}{TP+FN}, \tag{2.8}$$

Precision відповідає на питання «яка частка зразків, які модель віднесла до класу C, дійсно належить до класу C?», тоді як recall — «яку частку дійсних зразків класу C модель змогла розпізнати?». Для збалансованої оцінки використовується гармонічне середнє цих метрик — F1-міра (формула 2.9):

$$F1 = \frac{2*Precision*Recall}{Precision+Recall}, \quad (2.9)$$

Матриця помилок (confusion matrix) — це двовимірна таблиця, у якій рядки відповідають справжнім класам, а стовпці — передбаченим. Значення в комірці (i, j) дорівнює кількості зразків справжнього класу i, які модель віднесла до класу j. Діагональні елементи відповідають правильним класифікаціям; недіагональні — помилкам. Аналіз confusion matrix дозволяє виявити систематичні помилки моделі — пари класів, що часто плутаються між собою.

2.6 Обґрунтування feature engineering на основі геометричних відстаней

Запропонований у роботі підхід полягає у поданні скелета руки не у вигляді 63-вимірного вектора (21 точка × 3 координати), а у вигляді 24-вимірного вектора нормалізованих геометричних відстаней. Структура цього вектора така: 20 відстаней від зап’ястка (точки 0) до решти 20 точок, що описують загальну геометрію розпростертості пальців; 4 відстані між кінчиками сусідніх пальців (точки 4–8, 8–12, 12–16, 16–20), що описують ступінь розведення пальців. Усі 24 значення нормуються діленням на максимальну з них, що забезпечує інваріантність до масштабу руки та відстані до камери.

Теоретичне обґрунтування такого вибору полягає у наступних спостереженнях. По-перше, абсолютні координати ключових точок є надлишковими: на дактильні літери впливає не положення руки в просторі, а

лише її конфігурація — взаємні відстані між суглобами. Перехід до представлення у відстанях природним чином відкидає інформацію про абсолютну позицію. По-друге, ключові інформаційні відстані для розпізнавання дактильних літер — це:

a) Розпростертість пальців, що визначається відстанню від зап'ястка до кінчика.
b) Ступінь зведення сусідніх пальців.
c) Положення великого пальця. Ці три характеристики у сумі покриваються запропонованим набором з 24 ознак.

Очікувані переваги даного представлення такі: компактніший простір ознак (24 проти 63) призводить до меншої моделі та зниження ризику перенавчання; геометрична інтерпретованість ознак полегшує діагностику помилок; інваріантність до масштабу і зсуву зменшує потребу в аугментації даних під час тренування. Очікувана плата — можлива незначна втрата точності, оскільки 24-вимірне представлення є проєкцією 63-вимірного на меншу підпросторову розмірність. Кількісне порівняння обох представлень наведено у розділі 4.

Висновки до розділу 2

У другому розділі описано теоретичні основи методів, обраних для реалізації системи розпізнавання дактильної абетки. Розглянуто архітектуру MediaPipe HandLandmarker як засобу виділення скелета руки, наведено принципи роботи багатошарового перцептрона, обґрунтовано вибір функцій активації ReLU та softmax, функції втрат cross-entropy та алгоритму оптимізації Adam. Розглянуто dropout-регуляризацію як засіб запобігання перенавчанню. Описано стандартні метрики оцінки якості класифікаторів — accuracy, precision, recall, F1, confusion matrix. Окремо обґрунтовано запропоноване подання скелета руки у вигляді 24 нормалізованих геометричних відстаней.

РОЗДІЛ 3 ПРОЄКТУВАННЯ ТА РЕАЛІЗАЦІЯ СИСТЕМИ

3.1 Архітектура системи

Розроблена система побудована за принципом конвеєрної обробки даних. Вхідні дані — кадри відеопотоку з веб-камери у кольоровому просторі BGR. Конвеєр складається з чотирьох послідовних етапів (рис. 3.1):

1) попередня обробка кадра — вертикальне дзеркальне відображення для природної взаємодії та перетворення в RGB.
2) виявлення скелета руки за допомогою MediaPipe HandLandmarker.
3) обчислення вектора з 24 нормалізованих геометричних відстаней.
4) класифікація вектора натренованою нейромережею MLP.

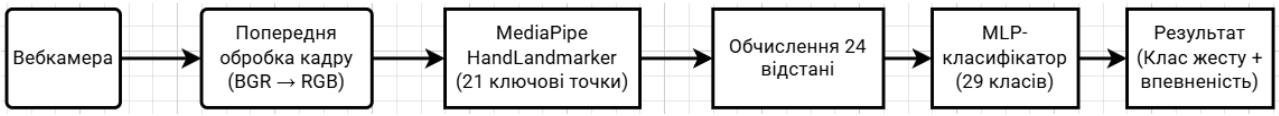


Рисунок 3.1 – Архітектура системи розпізнавання жестів

Результатом роботи системи є мітка класу та значення впевненості, що накладаються на відеокадр і виводяться користувачу у вікні OpenCV.

Окремо від основного режиму інференсу реалізовано допоміжні режими, що використовуються для підготовки моделі:

a) збір датасету у вигляді CSV-файлу векторів ознак.
b) тренування трьох порівнюваних моделей.
c) оцінка моделей на незалежному тестовому наборі.
d) формування зведеної порівняльної таблиці.

Усі режими реалізовано як окремі скрипти, що дозволяє запускати їх послідовно та незалежно.

3.2 Вибір технологічного стеку

Реалізація системи виконана мовою Python версії 3.12. Вибір мови зумовлений зрілою екосистемою бібліотек машинного навчання та комп'ютерного зору, а також можливістю швидкого прототипування. Зокрема, використано такі бібліотеки.

TensorFlow / Keras (версія 2.15) — фреймворк глибокого навчання для побудови та тренування нейронних мереж. Keras-API забезпечує лаконічне декларативне описання архітектури моделі. Збереження натренованої моделі здійснюється у форматі .keras.

MediaPipe (версія 0.10) — бібліотека Google для побудови мультимодальних конвеєрів обробки даних у реальному часі. У роботі використано модуль HandLandmarker з попередньо натренованою моделлю hand_landmarker.task для виділення скелета руки.

OpenCV (версія 4.8) — бібліотека комп'ютерного зору, що використовується для захоплення кадрів з веб-камери, базової обробки зображень та виведення результату на екран.

NumPy (версія 1.24), pandas, scikit-learn (версія 1.3) — допоміжні бібліотеки для числових обчислень, обробки табличних даних та класичних методів машинного навчання (зокрема, train_test_split для стратифікованого розбиття та classification_report для формування звітів).

matplotlib (версія 3.7) і seaborn (версія 0.12) — бібліотеки для побудови графіків, що використовуються для візуалізації кривих навчання та матриць помилок.

3.3 Опис тренувального датасету

Для тренування моделей використано датасет ASL Alphabet Dataset, опублікований у відкритому доступі на платформі Kaggle [24]. Датасет містить приблизно 87 000 кольорових зображень розміру 200×200 пікселів,

розподілених за 29 класами: 26 літер латинського алфавіту (A–Z) та три службових жести: «space» (пробіл), «del» (видалення), «nothing» (порожній жест). Кожен клас містить приблизно 3 000 зображень, отриманих у студійних умовах при природному освітленні. Зображення зняті з декількох ракурсів та з обома руками різних людей. Приклади зображень із датасету наведено на рис. 3.2.

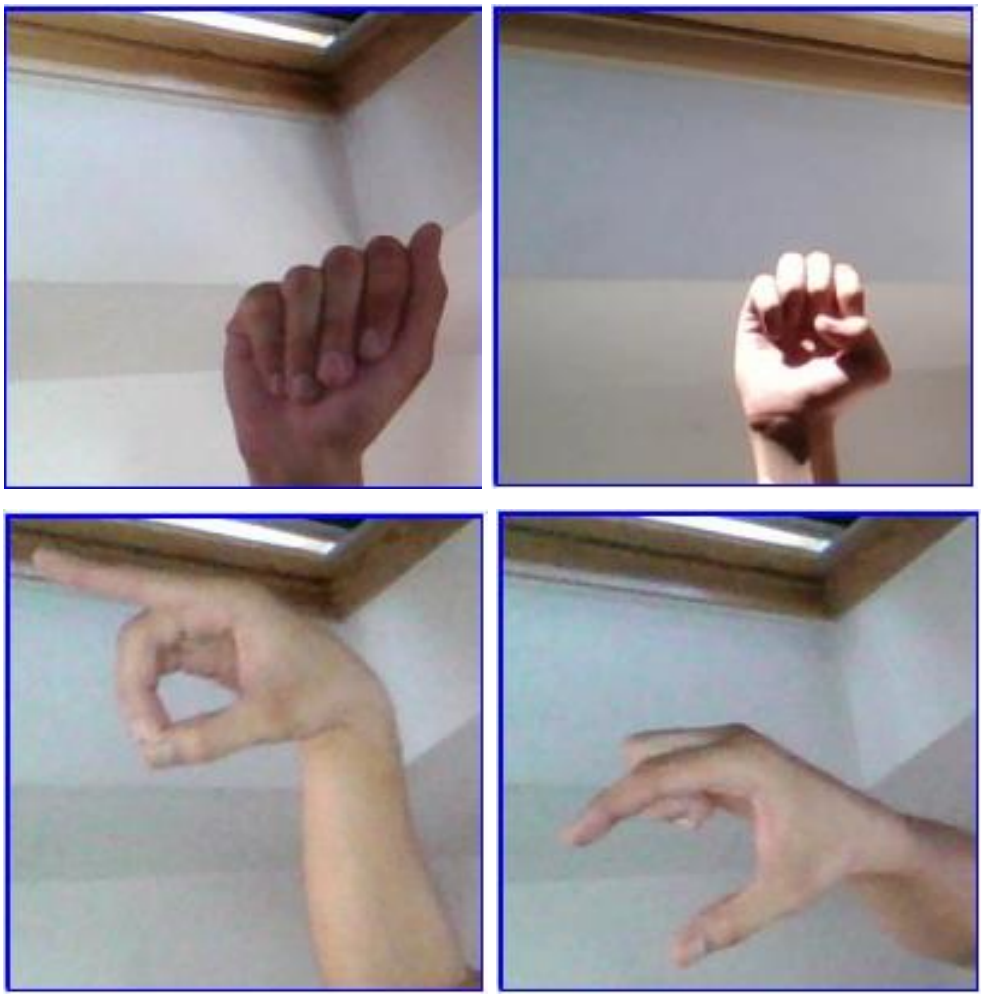


Рисунок 3.2 – Приклади зображень із датасету ASL Alphabet Dataset

У межах кваліфікаційної роботи з кожного класу обрано перших 400 зображень, що дає тренувальний пул обсягом 11 600 зображень. Це дозволило прискорити процес тренування при збереженні достатньої статистичної надійності оцінок. Тестовий набір asl_alphabet_test, що поставляється разом з

датасетом, містить лише 28 зображень (по одному на клас) і використовується лише як додатковий out-of-distribution тест.

3.4 Реалізація виділення ознак

Виділення вектора ознак з зображення руки реалізовано у модулі 1_extract_features.py. Ключова функція landmarks_to_distances приймає на вхід список з 21 ключової точки, отриманий від MediaPipe HandLandmarker, та повертає 24-вимірний нормалізований вектор. Текст функції наведено в лістингу 3.1. Повний текст програмної реалізації наведено в додатку А.1.

Лістинг 3.1 — Обчислення 24 нормалізованих геометричних відстаней

```
def landmarks_to_distances(landmarks):
    distances = []
    wrist = landmarks[0]
    for i in range(1, 21):
        lm = landmarks[i]
        d = math.sqrt((lm.x - wrist.x)**2 + (lm.y - wrist.y)**2)
        distances.append(d)
    fingertips = [(4, 8), (8, 12), (12, 16), (16, 20)]
    for p1, p2 in fingertips:
        a, b = landmarks[p1], landmarks[p2]
        d = math.sqrt((a.x - b.x)**2 + (a.y - b.y)**2)
        distances.append(d)
    max_dist = max(distances) or 1.0
    return [x / max_dist for x in distances]
```

Перші 20 значень вектора відповідають відстаням від зап’ястка (точка 0) до решти 20 точок. Наступні 4 значення — відстані між кінчиками сусідніх пальців. Усі 24 значення нормуються діленням на максимум, що забезпечує інваріантність до масштабу руки. Схему формування 24-вимірного вектора геометричних ознак наведено на рис. 3.3.

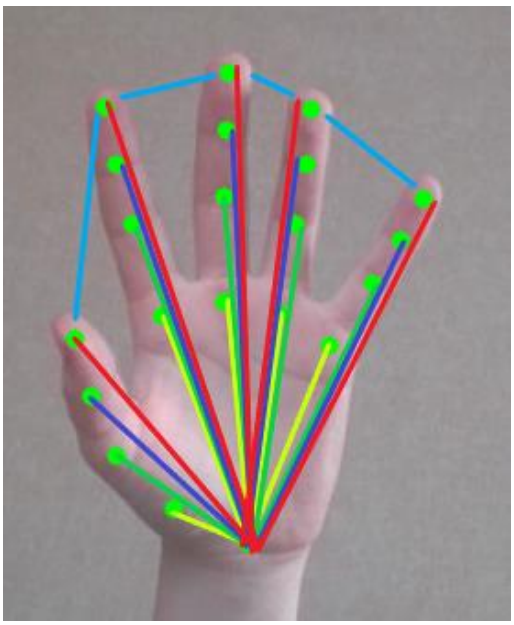


Рисунок 3.3 – Формування 24-вимірного вектора геометричних ознак

Друга функція, landmarks_to_raw, реалізована за аналогічним принципом, але повертає 63-вимірний вектор сирих координат, нормалізованих відносно зап’ястка.

3.5 Архітектура нейромережі

Запропонована нейромережа є багатошаровим перцептроном з двома прихованими шарами. Вхідний шар має розмірність 24 (за кількістю ознак). Перший прихований шар — повнозв’язний з 128 нейронами і функцією активації ReLU, після нього застосовано dropout з ймовірністю 0.2. Другий прихований шар — повнозв’язний з 64 нейронами, ReLU, dropout 0.2. Вихідний шар — повнозв’язний з 29 нейронами (за кількістю класів) і softmax-активацією. Загальна кількість параметрів моделі — приблизно 12 000.

Опис архітектури мовою Keras наведено в лістингу 3.2, а її структурну схему — на рис. 3.4. Фрагменти реалізації тренування наведено в додатках А.2–А.4

Лістинг 3.2 — Архітектура запропонованого MLP-класифікатора

```
model = tf.keras.Sequential([
    tf.keras.layers.Input(shape=(24,)),
    tf.keras.layers.Dense(128, activation="relu"),
    tf.keras.layers.Dropout(0.2),
    tf.keras.layers.Dense(64, activation="relu"),
    tf.keras.layers.Dropout(0.2),
    tf.keras.layers.Dense(29, activation="softmax"),
])
model.compile(
    optimizer="adam",
    loss="sparse_categorical_crossentropy",
    metrics=["accuracy"],
)
```

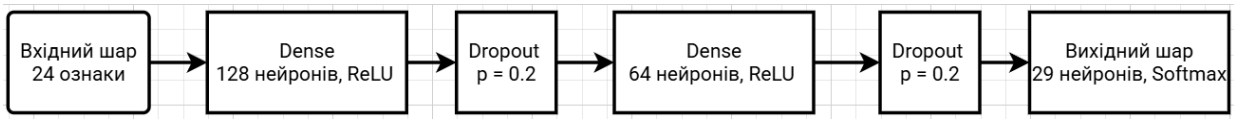


Рисунок 3.4 – Архітектура запропонованого MLP-класифікатора

Гіперпараметри тренування: оптимізатор Adam зі стандартними значеннями (швидкість навчання 0.001, $\beta_1 = 0.9$, $\beta_2 = 0.999$); функція втрат sparse_categorical_crossentropy; метрика моніторингу accuracy. Тренування виконується 50 епох з розміром міні-батчу 32. Цей вибір обґрунтовано результатами попередніх експериментальних запусків, які показали стабілізацію точності на валідаційному наборі вже на 30–40-й епосі.

3.6 Стратифіковане 3-частинне розбиття вибірки

Для забезпечення коректної експериментальної оцінки якості моделей застосовано стратифіковане 3-частинне розбиття, а саме тренувальна підвибірка (70 %), валідаційна (15 %), тестова (15 %):

- тренувальна підвибірка використовується для оптимізації параметрів моделі (backpropagation);
- валідаційна для моніторингу якості під час тренування та діагностики перенавчання;

- тестова для остаточної оцінки моделі і не торкається до завершення тренування. Стратифікація гарантує, що пропорції класів зберігаються в кожній з трьох підвибірок.

Для усіх трьох порівнюваних моделей використано однаковий генератор псевдовипадкових чисел зі значенням SEED = 42. Це забезпечує те, що тестова підвибірка складається з тих самих зразків в усіх трьох експериментах, що робить порівняння моделей чесним.

3.7 Реалізація режиму розпізнавання у реальному часі

Демонстраційний модуль реалізовано у файлі 5_inference.py. На кожному кадрі відеопотоку виконується наступна послідовність дій: захоплення кадру з камери; вертикальне дзеркальне відображення для природної взаємодії; конвертація BGR → RGB для подачі в MediaPipe; виявлення скелета руки; за наявності виявлення — обчислення 24-вимірного вектора відстаней та інференс моделі; накладення на кадр візуальних позначок ключових точок та тексту з міткою класу і значенням впевненості; виведення кадру у вікні OpenCV. Цикл триває до натискання клавіші q. Повний текст модуля інференсу наведено в додатку А.7

Колір тексту обирається залежно від рівня впевненості: зелений при впевненості понад 80 %, помаранчевий — при нижчій. Це дає користувачу візуальний зворотний зв’язок щодо надійності розпізнавання конкретного жесту. Розмір шрифту тексту встановлено достатньо великим, щоб бути зчитуваним з відстані 1–2 метри від екрана.

Експериментальні запуски на стандартному ноутбуці з процесором Intel Core i5 показали стабільну роботу демонстраційного модуля з частотою 28–32 кадри на секунду, що задовольняє критерію real-time, сформульованому в постановці задачі. Приклад роботи розробленої системи у режимі реального часу наведено на рис. 3.4.

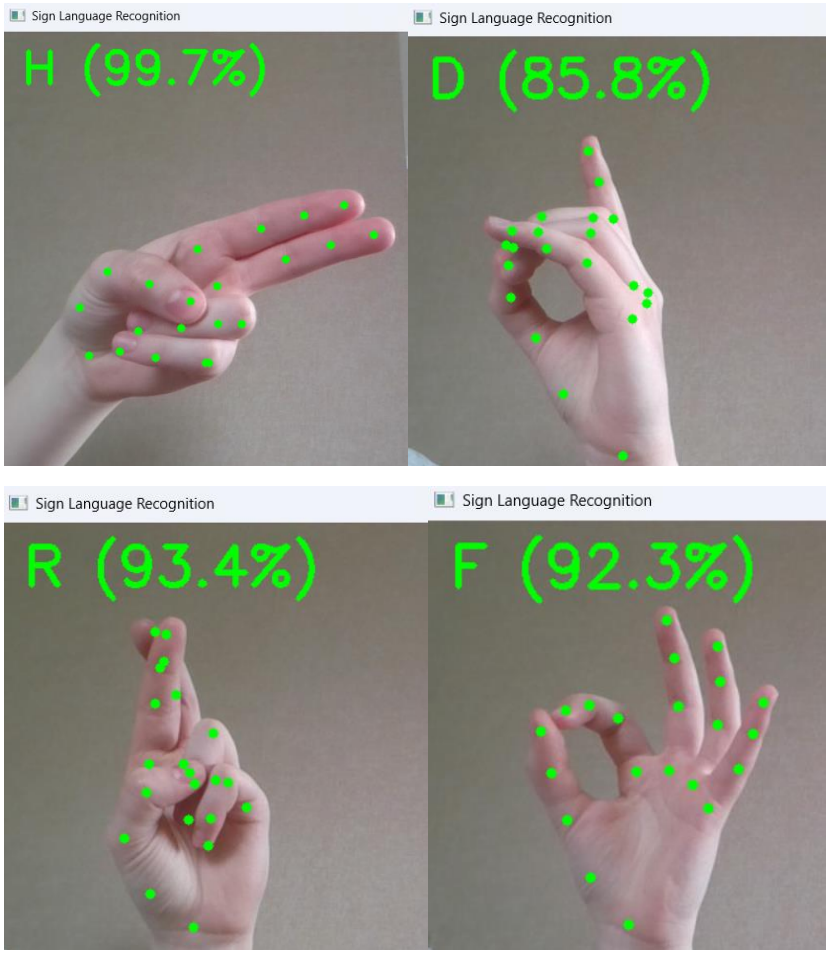


Рисунок 3.4 – Робота системи в режимі реального часу

Висновки до розділу 3

У третьому розділі представлено результати програмної реалізації запропонованої системи. Описано архітектуру системи у вигляді конвеєра з чотирьох етапів. Обґрунтовано вибір технологічного стеку(Python, TensorFlow, MediaPipe, OpenCV). Подано опис тренувального датасету ASL Alphabet Dataset, наведено реалізацію виділення вектора ознак та архітектуру запропонованого MLP-класифікатора. Окрему увагу приділено експериментально коректній методології — стратифікованому 3-частинному розбиттю вибірки з фіксованим SEED для всіх порівнюваних моделей. Описано реалізацію демонстраційного модуля, що працює у режимі реального часу.

РОЗДІЛ 4 ЕКСПЕРИМЕНТАЛЬНІ ДОСЛІДЖЕННЯ

4.1 Методика проведення експериментів

Експериментальне дослідження проведено з метою порівняльної оцінки трьох розглянутих нейромережевих підходів за критеріями точності, швидкодії та розміру моделі. Експерименти побудовані за єдиною методологічною основою: усі три моделі тренуються та тестуються на однаковому стратифікованому 3-частинному розбитті датасету ASL Alphabet Dataset з фіксованим SEED = 42.

Розбиття вибірки виконано за допомогою функції train_test_split з бібліотеки scikit-learn у двоетапному режимі: спочатку від загального обсягу зразків відрізаються 15 % під тестовий набір; після цього з решти 85 % відрізаються 15/85 ≈ 17.65 % під валідаційний набір. Таким чином, тренувальна підвибірка становить 70 %, валідаційна — 15 %, тестова — 15 % від загального обсягу. Для MLP-моделей розміри підвибірок: train ≈ 7 280, val ≈ 1 561, test ≈ 1 561 зразків. Для CNN-моделі: train ≈ 8 120, val ≈ 1 740, test ≈ 1 740 зразків. Різниця у кількостях для CNN зумовлена тим, що для неї використано всі 11 600 зображень, тоді як MLP-моделі натреновано лише на тих зразках, де MediaPipe HandLandmarker зміг успішно виявити скелет руки (приблизно 89 % від загальної кількості).

Усі моделі тренувалися протягом 50 епох (для MLP) або 20 епох (для CNN — з огляду на більший обсяг та повільнішу збіжність) із розміром міні-батчу 32 (MLP) або 64 (CNN). Як обчислювальне середовище використано стандартний персональний комп’ютер з процесором Intel Core i5 без використання графічного прискорювача. Усі цифри, наведені у цьому розділі, отримані саме на цій конфігурації.

4.2 Результати тренування моделей

Зведені результати тренування трьох моделей наведено в таблиці 4.1.

Таблиця 4.1 — Результати тренування трьох моделей

Модель	**Точність тренування**	**Val точність**	**Час навч., с**
MLP на 24 відстанях	91.38 %	92.26 %	18.8
MLP на 63 координатах	97.20 %	97.89 %	18.7
CNN	96.45 %	92.67 %	111.2

Аналіз кривих навчання, побудованих в процесі тренування, показав певні особливості.

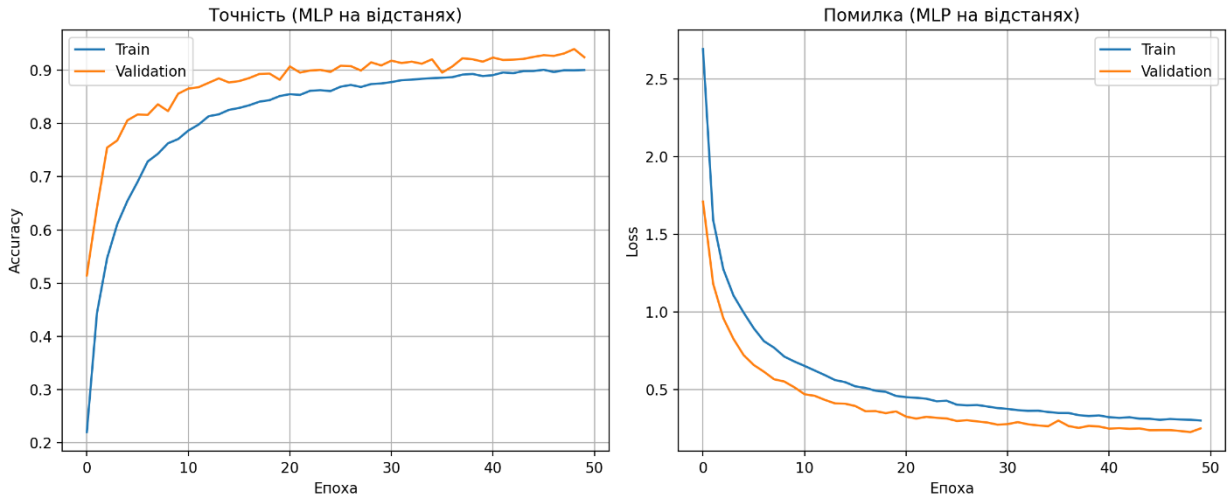


Рисунок 4.1 — Криві навчання MLP на 24 нормалізованих відстанях

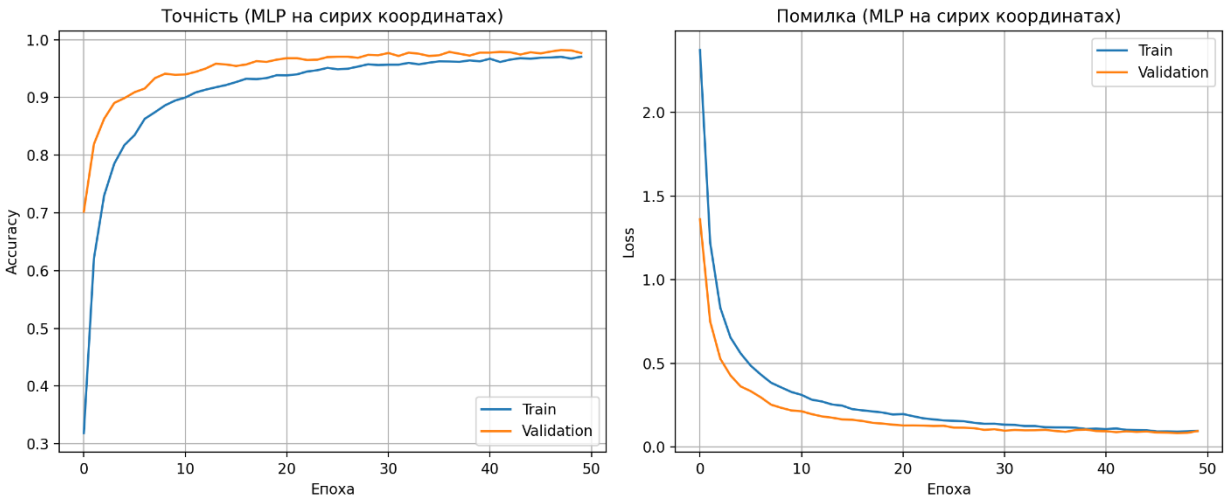


Рисунок 4.2 — Криві навчання MLP на 63 сирих координатах

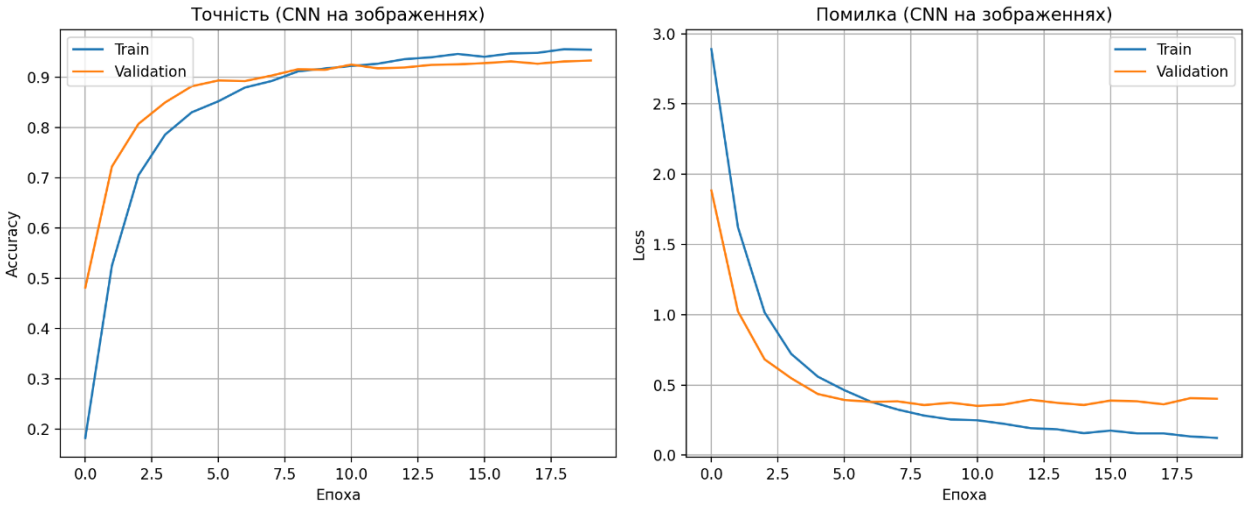


Рисунок 4.3 — Криві навчання CNN на зображеннях 64×64

Криві обох MLP-моделей демонструють стабільну монотонну збіжність з виходом на плато до 30-ї епохи. Різниця між тренувальною та валідаційною точністю не перевищує 1 % для обох MLP, що свідчить про відсутність помітного перенавчання.

Для CNN-моделі спостерігається інша картина: тренувальна точність досягає 96.45 %, тоді як валідаційна стабілізується на рівні 92.67 %. Різниця у 3.8 % свідчить про помірне перенавчання CNN навіть на стратифіковано

розділеному датасеті, де train і val набори мають однаковий розподіл. Це підтверджує загальну тенденцію CNN-архітектур до перенавчання при обмеженому обсязі тренувальних даних. Детальні матриці помилок для альтернативних моделей наведено в додатку Б (рис. Б.1–Б.2).

Час тренування MLP-моделей на CPU склав приблизно 19 с кожне, тоді як CNN-моделі — 111 с, що в 6 разів довше. Це різниця особливо суттєва для багаторазових експериментів та підлаштування гіперпараметрів.

4.3 Порівняльна оцінка моделей на незалежній тестовій вибірці

Оцінка моделей на повністю незалежному тестовому наборі (15 % від загального обсягу) є основним результатом експериментального дослідження. Результати порівняння наведено в таблиці 4.2.

Таблиця 4.2 — Зведена порівняльна оцінка трьох моделей на незалежному тестовому наборі

Модель	**Точність**	**FPS**	**Розмір**	**Тест-зразків**
MLP на 24 нормалізованих відстанях (запропонований)	92.25 %	3 661	185 кб	1561
MLP на 63 сирих координатах	98.08 %	10 236	243 кб	1561
CNN на зображеннях 64×64	92.87 %	629	8 097 кб	1740

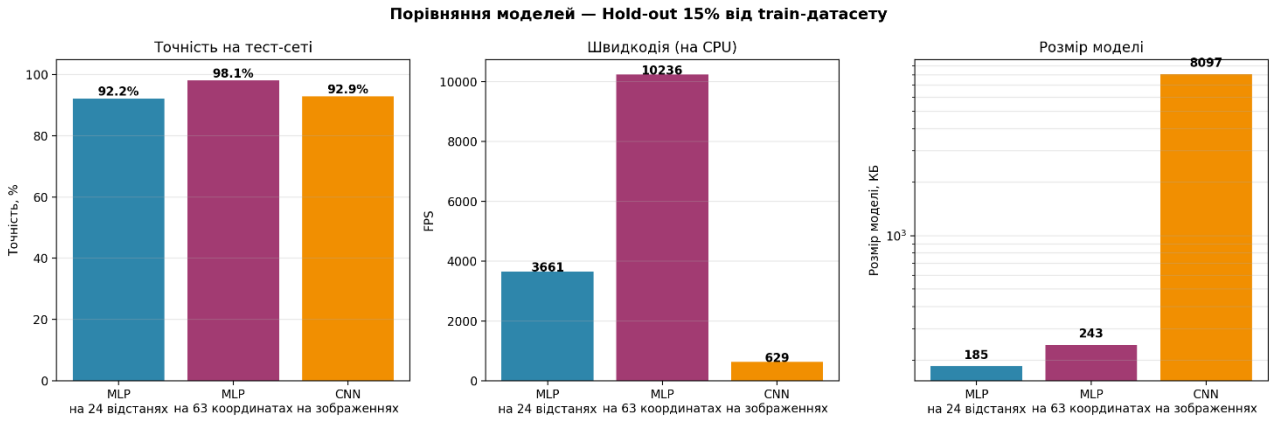


Рисунок 4.4 — Порівняння трьох моделей за точністю, швидкодією та розміром моделі

Аналіз отриманих результатів дозволяє зробити декілька важливих спостережень.

По-перше, обидві MLP-моделі демонструють точність, конкурентоспроможну з CNN. Запропонована модель (MLP на 24 відстанях) досягає 92.25 % проти 92.87 % у CNN — різниця у 0.62% є статистично малозначущою на тестовому наборі обсягом 1561–1740 зразків. Це повністю підтверджує гіпотезу про принципову достатність 21 ключової точки руки для розпізнавання дактильної абетки.

По-друге, MLP на 63 сирих координатах показала найвищу точність — 98.08 %. Перевага в 5.83 % відсоткових пунктів над запропонованою моделлю пояснюється тим, що 63-вимірний вектор містить повну інформацію про геометрію скелета (включаючи третю координату z, що не використовується в запропонованому 24-вимірному поданні), тоді як 24 відстані є проєкцією геометрії на меншу підпросторову розмірність.

По-третє, за швидкодією та розміром моделі переваги мають саме MLP-підходи. Запропонована модель забезпечує пропускну здатність 3 661 кадр на секунду в режимі батчевого інференсу — у 5.8 разів більше за CNN-модель. Її розмір — 185 кб — у 44 рази менший за CNN (8 097 кб) і в 1.3 рази менший за MLP на сирих координатах. У режимі real-time інференсу одиничного кадру (з

урахуванням накладних витрат на виклик моделі) запропонована модель забезпечує стабільну роботу на 28–32 кадри на секунду на CPU, що повністю задовольняє вимогам real-time системи з постановки задачі.

4.4 Аналіз помилок класифікації

Глибший аналіз помилок класифікації запропонованої моделі проведено за матрицею помилок (confusion matrix), наведеною на рисунку 4.5, та повним classification report. У таблиці 4.3 наведено п’ять класів з найнижчими значеннями F1-міри, що відповідають літерам, на яких запропонована модель помиляється найчастіше.

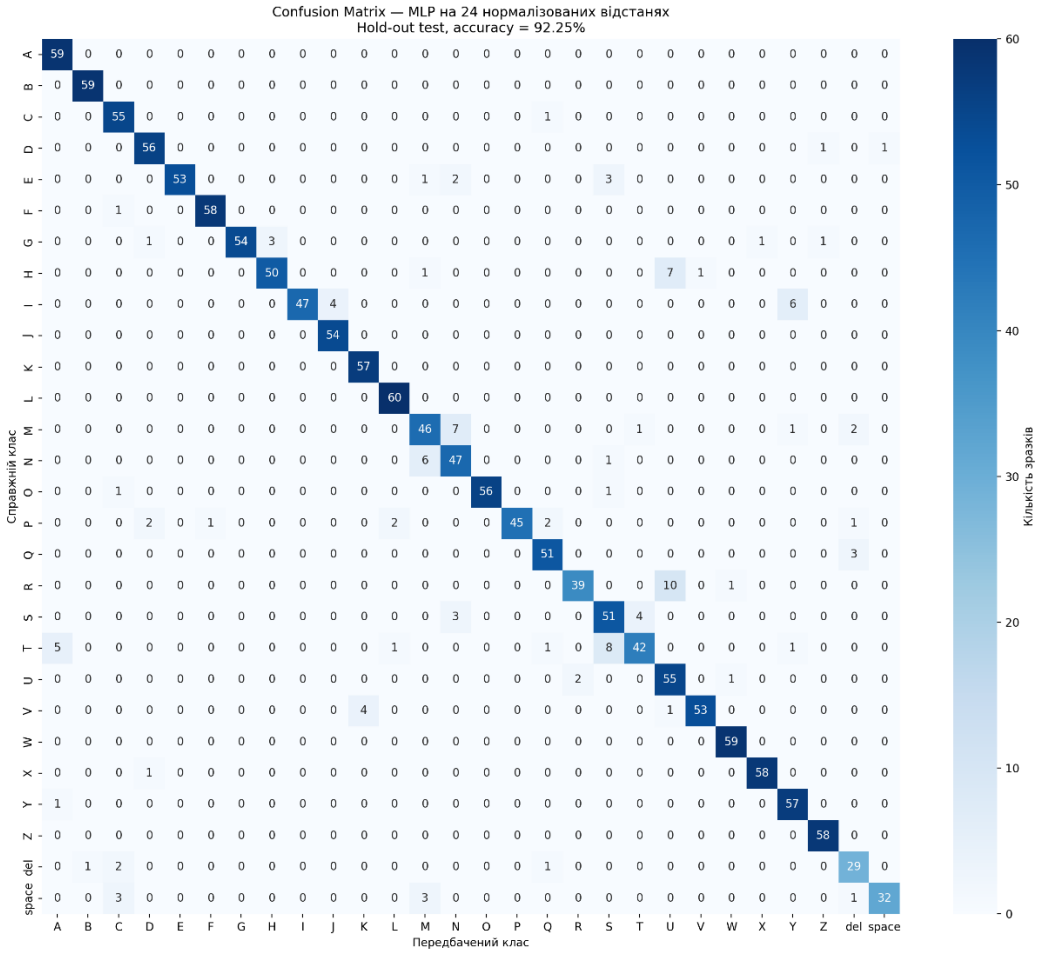


Рисунок 4.5 — Матриця помилок запропонованої моделі (MLP на 24 нормалізованих відстанях)

Таблиця 4.3 — Класи з найнижчою якістю розпізнавання (запропонована модель)

Літера	Точність	Recall	F1-міра
T	0.89	0.72	0.80
M	0.81	0.81	0.81
N	0.80	0.87	0.83
S	0.80	0.88	0.84
U	0.75	0.95	0.84

Спостерігається систематична картина: усі п’ять найгірших за якістю літер — T, M, N, S, U — є дактилемами із позою «закритого кулака», де відрізнити одну літеру від іншої можна лише за тонким положенням великого пальця. Зокрема, літери M, N, S виконуються майже ідентично — кулаком з великим пальцем, прикритим відповідно одним, двома або всіма пальцями зверху. Літери T та S відрізняються тільки тим, чи знаходиться великий палець між вказівним і середнім пальцями, чи поверх кулака.

Цей висновок має дві важливі практичні наслідки. По-перше, спостережені помилки не свідчать про дефект моделі — вони відповідають об’єктивно складним для розрізнення випадкам, де навіть досвідчений людський спостерігач часто потребує додаткових візуальних підказок (повертання руки, зменшення швидкості тощо). По-друге, ця інформація може бути використана для подальшого вдосконалення системи: цілеспрямоване збільшення обсягу тренувальних даних саме для класів M, N, S, T, або введення додаткових ознак, що описують положення великого пальця окремо.

Для порівняння, MLP-модель на 63 сирих координатах справляється з цими ж літерами помітно краще: F1-міри для M, N, S, T коливаються в діапазоні 0.92–0.95. Це додатково підтверджує, що інформація про z-координату ключових точок (третій вимір, що описує глибину руки) є істотно

важливою для розпізнавання саме цих складних літер. У запропонованому 24-вимірному поданні z-координата не використовується (відстані обчислюються лише в площині XY), що можна трактувати як одне з обмежень обраного підходу та напрям його подальшого розвитку.

4.5 Pareto-аналіз компромісів

Жоден з трьох розглянутих підходів не виграє за всіма критеріями одночасно. Представлення результатів у двовимірному просторі «точність — обчислювальна вартість» (Pareto-діаграма, наведена на рисунку 4.6) виявляє нетривіальну структуру компромісів.

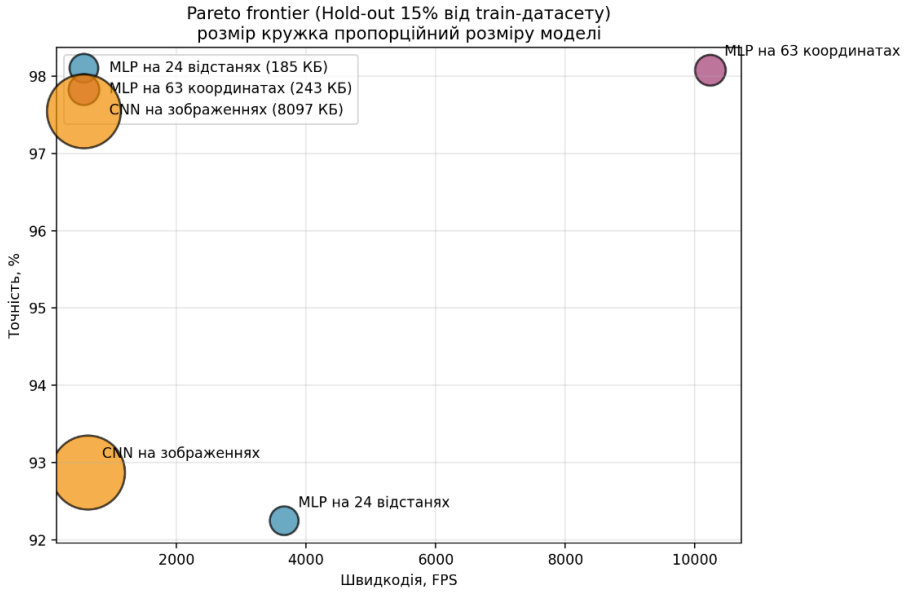


Рисунок 4.6 — Pareto-діаграма «точність — швидкодія» (розмір кружка пропорційний розміру моделі)

Якщо абсолютним пріоритетом є точність — обирають MLP на 63 координатах. Якщо пріоритетом є мінімальний розмір моделі при достатній точності — обирають запропонований MLP на 24 відстанях. CNN-підхід при поточних результатах не є оптимальним за жодним з критеріїв: він

поступається MLP як за точністю, так і за швидкодією, а за розміром моделі — у 44 рази. Окрему перевагу CNN могла б демонструвати на out-of-distribution даних з абсолютно іншим типом фону або освітлення, проте обмежений обсяг тестового набору не дозволяє зробити надійний висновок з цього питання.

У постановці задачі сформульовано вимоги real-time-системи на стандартному комп'ютері без графічного прискорювача. Серед розглянутих підходів обом критеріям real-time (≥25 FPS) відповідають усі три моделі, проте за критерієм мінімізації розміру моделі (≤1 МБ) — лише MLP-моделі. Запропонований MLP на 24 відстанях задовольняє всі вимоги постановки задачі та обирається як остаточне рішення для подальшої інтеграції в демонстраційний модуль.

4.6 Обмеження методу та шляхи подальшого розвитку

У ході експериментального дослідження виявлено такі обмеження запропонованого підходу. По-перше, точність розпізнавання залежить від точності виділення скелета руки бібліотекою MediaPipe HandLandmarker. У тих рідкісних випадках, коли MediaPipe не виявляє руку (близько 11 % зразків датасету), система не може зробити прогноз. По-друге, поточна реалізація використовує лише статичну позу руки і не враховує літери J та Z, які в дактильній абетці ASL виконуються рухом. По-третє, систематичні помилки на закрито-кулакових літерах (M, N, S, T) можуть бути зменшені за рахунок включення z-координати в простір ознак.

Як перспективні напрями подальшого розвитку можуть бути запропоновані: розширення feature engineering включенням третьої координати; додавання тимчасової інформації (рекурентна мережа на послідовності кадрів) для розпізнавання динамічних літер J та Z; розширення системи до розпізнавання цілих слів і фраз через побудову модулів сегментації потоку та темпоральної моделі.

Висновки до розділу 4

У четвертому розділі представлено результати експериментального дослідження. Описано методику стратифікованого 3-частинного розбиття вибірки. Натреновано три порівнюваних моделі (MLP на 24 відстанях, MLP на 63 координатах, CNN на зображеннях). Наведено зведені порівняльні таблиці точності, швидкодії та розміру моделей. Запропонована модель (MLP на 24 нормалізованих геометричних відстанях) досягає точності 92.25 % при розмірі моделі 185 кб та швидкодії, що задовольняє вимогам реального часу. Проведено детальний аналіз помилок класифікації та виявлено систематичні складні випадки на закрито-кулакових літерах. Pareto-аналіз показав, що запропонований підхід забезпечує оптимальний компроміс «точність — обчислювальна вартість» для real-time системи на CPU.

ВИСНОВКИ

У ході виконання кваліфікаційної роботи побудовано програмну систему розпізнавання літер дактильної абетки американської жестової мови у режимі реального часу. Мету роботи досягнуто, всі поставлені завдання виконано в повному обсязі. Основні результати, отримані в роботі, такі:

1. Проаналізовано предметну область автоматичного розпізнавання жестових мов; виокремлено задачу розпізнавання дактильних літер як фундаментальний етап побудови ширших систем перекладу жестових мов. Розглянуто три покоління методів розпізнавання — класичні методи комп'ютерного зору з ручним feature engineering, чисто-CNN-підходи та гібридні landmark-based-підходи. Виявлено, що для задач реального часу на пристроях без графічного прискорювача саме landmark-based-підхід забезпечує найкращий компроміс між точністю, швидкодією та компактністю моделі.

2. Обґрунтовано та реалізовано двоступеневу архітектуру системи: на першому етапі бібліотека MediaPipe HandLandmarker виділяє з кадру 21 анатомічну ключову точку скелета руки; на другому етапі натренована нейромережа класифікує жест за компактним вектором геометричних ознак. Запропоновано та реалізовано схему feature engineering на основі 24 нормалізованих геометричних відстаней (20 відстаней від зап'ястка до решти точок та 4 відстані між кінчиками сусідніх пальців), інваріантну до масштабу руки та відстані до камери.

3. Розроблено архітектуру багатошарового перцептрона з двома прихованими шарами (128 і 64 нейрони з функцією активації ReLU та регуляризацією Dropout 0.2) і вихідним softmax-шаром на 29 класів. Виконано тренування моделі на датасеті ASL Alphabet Dataset з використанням алгоритму оптимізації Adam і функції втрат категоріальної крос-ентропії. Загальна кількість параметрів моделі — приблизно 12 тисяч, розмір збереженої моделі — 185 кб.

4. Для коректної порівняльної оцінки додатково натреновано дві альтернативні моделі: багатошаровий перцептрон на 63 сирих нормалізованих координатах ключових точок та згорткову нейронну мережу на зображеннях розміру 64×64 пікселі. Усі три моделі натреновано та протестовано на однаковому стратифікованому 3-частинному розбитті вибірки (70 % / 15 % / 15 %) з фіксованим seed = 42.

5. Проведено експериментальне дослідження, у ході якого встановлено, що запропонована модель досягає точності 92.25 % на незалежному тестовому наборі обсягом 1561 зразків. Альтернативний MLP на 63 координатах показав вищу точність — 98.08 %, але більший розмір моделі. Згорткова нейромережа дала точність 92.87 %, проте має у 44 рази більший розмір моделі та у 6 разів повільніший інференс. Незважаючи на нижчу точність порівняно з MLP на сирих координатах, запропонована модель використовує менший простір ознак, характеризується компактністю та забезпечує достатню якість розпізнавання для практичного використання в системах реального часу. Для аналізу результатів використано матриці помилок (confusion matrix), які дозволили виявити найбільш проблемні для класифікації класи жестів.

6. Детальний аналіз матриць помилок показав, що систематичні помилки концентруються на закрито-кулакових літерах M, N, S, T, U (значення міри від 0.80 до 0.84). Обґрунтовано, що ці помилки не свідчать про дефект моделі, а відповідають об’єктивно складним для розрізнення випадкам, що виконуються майже ідентичною позою долоні з різницею у тонкому положенні великого пальця.

7. Реалізовано демонстраційний модуль розпізнавання у режимі реального часу мовою Python з використанням бібліотек OpenCV, MediaPipe і TensorFlow. Експериментальні запуски показали стабільну роботу модуля з частотою 28–32 кадри на секунду на стандартному ноутбуці з процесором Intel Core i5 без використання графічного прискорювача, що повністю задовольняє вимогам real-time-системи з постановки задачі.

Виконання поставлених завдань дозволило досягти мети кваліфікаційної роботи та підтвердити ефективність запропонованого підходу до розпізнавання дактильної абетки ASL у режимі реального часу.

У ході роботи було досліджено та експериментально оцінено три нейромережеві підходи до розпізнавання жестів дактильної абетки на єдиній методологічній основі — стратифікованому 3-частинному розбитті датасету. Показано, що запропонований підхід забезпечує раціональний компроміс між точністю розпізнавання, обчислювальними витратами та розміром моделі серед розглянутих альтернатив для задач реального часу на CPU.

Перспективні напрями подальшого розвитку запропонованої системи такі:

a) розширення простору ознак включенням третьої координати ключових точок для покращення розпізнавання закрито-кулакових літер.
b) додавання тимчасової складової через рекурентну нейронну мережу для розпізнавання динамічних літер J та Z.
c) розширення системи від рівня окремих літер до рівня цілих слів і фраз через побудову модулів сегментації жестового потоку та темпоральної моделі мови.

Результати кваліфікаційної роботи апробовано на всеукраїнській науково-практичній конференції. За тематикою роботи опубліковано тези доповіді [26] (див. Додаток В).

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. World Federation of the Deaf. Sign Languages [Електронний ресурс]. — Режим доступу: https://wfdeaf.org/our-work/human-rights-of-the-deaf/sign-language/

2. Padden C., Humphries T. Inside Deaf Culture. — Cambridge: Harvard University Press, 2005. — 224 p.

3. Rastgoo R., Kiani K., Escalera S. Sign Language Recognition: A Deep Survey // Expert Systems with Applications. — 2021. — Vol. 164. — P. 113794.

4. Stokoe W.C. Sign Language Structure: An Outline of the Visual Communication Systems of the American Deaf // Studies in Linguistics, Occasional Papers. — Buffalo: University of Buffalo, 1960. — № 8. — 78 p.

5. Koller O., Forster J., Ney H. Continuous sign language recognition: Towards large vocabulary statistical recognition systems handling multiple signers // Computer Vision and Image Understanding. — 2015. — Vol. 141. — P. 108–125.

6. Bilal S., Akmeliawati R., El Salami M. Vision-based hand posture detection and recognition for Sign Language // Proc. of the IEEE International Conference on Mechatronics. — 2011. — P. 1–6.

7. Pugeault N., Bowden R. Spelling It Out: Real-Time ASL Fingerspelling Recognition // IEEE International Conference on Computer Vision Workshops (ICCV-W). — 2011. — P. 1114–1119.

8. Krizhevsky A., Sutskever I., Hinton G.E. ImageNet Classification with Deep Convolutional Neural Networks // Advances in Neural Information Processing Systems. — 2012. — Vol. 25. — P. 1097–1105.

9. Simonyan K., Zisserman A. Very Deep Convolutional Networks for Large-Scale Image Recognition // arXiv preprint. — 2014. — arXiv:1409.1556. — 14 p.

10. He K., Zhang X., Ren S., Sun J. Deep Residual Learning for Image Recognition // Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). — 2016. — P. 770–778.

11. Garcia B., Viesca S.A. Real-time American Sign Language Recognition with Convolutional Neural Networks // Stanford CS231n Project Report. — 2016. — 8 p.

12. Tan M., Le Q. EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks // International Conference on Machine Learning (ICML). — 2019. — P. 6105–6114.

13. Zhang F., Bazarevsky V., Vakunov A. et al. MediaPipe Hands: On-device Real-time Hand Tracking // arXiv preprint. — 2020. — arXiv:2006.10214. — 5 p.

14. Vaswani A., Shazeer N., Parmar N. et al. Attention Is All You Need // Advances in Neural Information Processing Systems. — 2017. — Vol. 30. — P. 5998–6008.

15. Yan S., Xiong Y., Lin D. Spatial Temporal Graph Convolutional Networks for Skeleton-Based Action Recognition // Proc. of the AAAI Conference on Artificial Intelligence. — 2018. — P. 7444–7452.

16. Sign-Language-Recognition: ASL recognition using VGG-16 [Електронний ресурс]. — Режим доступу: https://github.com/ardamavi/Sign-Language-Digits-Dataset

17. ASL-Alphabet-Recognition: Real-time ASL alphabet recognition with PyTorch [Електронний ресурс]. — Режим доступу: https://github.com/loicmarie/sign-language-alphabet-recognizer

18. SignAll Technologies [Електронний ресурс]. — Режим доступу: https://www.signall.us/

19. Goodfellow I., Bengio Y., Courville A. Deep Learning. — Cambridge: MIT Press, 2016. — 800 p.

20. Nair V., Hinton G.E. Rectified Linear Units Improve Restricted Boltzmann Machines // Proc. of the 27th International Conference on Machine Learning (ICML). — 2010. — P. 807–814.

21. Kingma D.P., Ba J. Adam: A Method for Stochastic Optimization // arXiv preprint. — 2014. — arXiv:1412.6980. — 15 p.

22. Srivastava N., Hinton G., Krizhevsky A. et al. Dropout: A Simple Way to Prevent Neural Networks from Overfitting // Journal of Machine Learning Research. — 2014. — Vol. 15. — P. 1929–1958.

23. Sokolova M., Lapalme G. A systematic analysis of performance measures for classification tasks // Information Processing & Management. — 2009. — Vol. 45, № 4. — P. 427–437.

24. Akash. ASL Alphabet Dataset [Електронний ресурс] // Kaggle. — 2018. — Режим доступу: https://www.kaggle.com/datasets/grassknoted/asl-alphabet

25. Abadi M., Agarwal A., Barham P. et al. TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems [Електронний ресурс]. — 2015. — Режим доступу: https://www.tensorflow.org

26. Воінова О.В. Моделювання нейромережевої системи розпізнавання та перекладу мови жестів [Електронний ресурс] // Матеріали всеукраїнської науково-практичної конференції молодих вчених, аспірантів і студентів «Інтелектуальні інформаційні системи». — Миколаїв, 2025. — С. 63. — Режим доступу: https://surl.li/wxigdu

ДОДАТОК А

А.1 Модуль виділення ознак (1_extract_features.py)

Лістинг А.1 — Виділення вектора ознак зі скелета руки

```
from __future__ import annotations
import csv
import os
import tim
import cv2
import mediapipe as mp
from utils import create_hand_detector, landmarks_to_distances,
landmarks_to_raw

DATA_PATH = "dataset/ASL_Alphabet_Dataset/asl_alphabet_train"
IMAGES_PER_CLASS = 400
CSV_DIST = "landmarks_dist.csv"
CSV_RAW = "landmarks_raw.csv"
EXTENSIONS = (".jpg", ".jpeg", ".png")

def main() -> None:
    detector = create_hand_detector()
    classes = sorted(
        d for d in os.listdir(DATA_PATH) if
os.path.isdir(os.path.join(DATA_PATH, d))
    )
    print(f"Знайдено класів: {len(classes)}")
    print(f"По {IMAGES_PER_CLASS} зображень на клас.\n")

    start = time.time()
    with open(CSV_DIST, "w", newline="", encoding="utf-8") as
f_dist, open(
        CSV_RAW, "w", newline="", encoding="utf-8"
    ) as f_raw:
        w_dist = csv.writer(f_dist)
        w_raw = csv.writer(f_raw)

        for class_name in classes:
            class_dir = os.path.join(DATA_PATH, class_name)
            images = sorted(
                f for f in os.listdir(class_dir) if
```

Лістинг А.1 , аркуш 1

```
f.lower().endswith(EXTENSIONS))[:IMAGES_PER_CLASS]

          success = 0
          for img_name in images:
                img = cv2.imread(os.path.join(class_dir,
img_name))
                if img is None:
                    continue
                rgb = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
                mp_image =
mp.Image(image_format=mp.ImageFormat.SRGB, data=rgb)
                result = detector.detect(mp_image)
                if not result.hand_landmarks:
                    continue
                lm = result.hand_landmarks[0]
                w_dist.writerow([class_name,
*landmarks_to_distances(lm)])
                w_raw.writerow([class_name,
*landmarks_to_raw(lm)])
                success += 1
            print(f"  {class_name:>8}:
{success:4d}/{len(images)} зразків")

    elapsed = time.time() - start
    print(f"\nЗавершено за {elapsed/60:.1f} хв.")
    print(f"Створено: {CSV_DIST}, {CSV_RAW}")
```

Лістинг А.1, аркуш 2

А.2 Тренування запропонованої моделі (2a_train_distances.py)

Лістинг А.2 — Тренування MLP на 24 нормалізованих відстанях

```
    X_trainval, X_test, y_trainval, y_test = train_test_split(
        X, y, test_size=0.15, random_state=SEED, stratify=y
    )
    X_train, X_val, y_train, y_val = train_test_split(
        X_trainval, y_trainval, test_size=0.1765,
        random_state=SEED, stratify=y_trainval
    )
    print(f"  Train: {len(X_train)}  Val: {len(X_val)}  Test:
```

Лістинг А.2 , аркуш 1

```
{len(X_test)}")

    model = tf.keras.Sequential(
        [
            tf.keras.layers.Input(shape=(X.shape[1],)),
            tf.keras.layers.Dense(128, activation="relu"),
            tf.keras.layers.Dropout(0.2),           tf.keras.la
yers.Dense(64, activation="relu"),
            tf.keras.layers.Dropout(0.2),
            tf.keras.layers.Dense(len(class_names),
activation="softmax"),
        ],
        name="MLP_Distances",
    )
    model.compile(
        optimizer="adam",
        loss="sparse_categorical_crossentropy",
        metrics=["accuracy"],
    )
    model.summary()
    history = model.fit(
        X_train, y_train,
        epochs=EPOCHS, batch_size=BATCH_SIZE,
        validation_data=(X_val, y_val), verbose=2,
    )
    train_time = time.time() - t0
```

Лістинг А.2 , аркуш 2

А.3 Тренування альтернативної моделі (2b_train_raw.py)

Лістинг А.3 — Тренування MLP на 63 сирих координатах ключових точок

```
    model = tf.keras.Sequential([
            tf.keras.layers.Input(shape=(X.shape[1],)),
            tf.keras.layers.Dense(128, activation="relu"),
            tf.keras.layers.Dropout(0.2),
            tf.keras.layers.Dense(64, activation="relu"),
            tf.keras.layers.Dropout(0.2),
            tf.keras.layers.Dense(len(class_names),
activation="softmax"),],name="MLP_Raw",)
    model.compile( optimizer="adam",
        loss="sparse_categorical_crossentropy",
        metrics=["accuracy"],)
    model.summary()
```

А.4 Тренування альтернативної моделі (2c_train_cnn.py)

Лістинг А.4 — Тренування CNN на зображеннях розміром 64×64 пікселі

```
def load_dataset() -> tuple[np.ndarray, np.ndarray, list[str]]:
    classes = sorted(
        d for d in os.listdir(DATA_PATH) if
os.path.isdir(os.path.join(DATA_PATH, d)))
    images: list[np.ndarray] = []
    labels: list[int] = []
    for class_id, class_name in enumerate(classes):
        class_dir = os.path.join(DATA_PATH, class_name)
        files = sorted(os.listdir(class_dir))[:IMAGES_PER_CLASS]
        for fname in files:
            img = tf.keras.preprocessing.image.load_img(
                os.path.join(class_dir, fname),
                target_size=(IMAGE_SIZE, IMAGE_SIZE),)
            arr = tf.keras.preprocessing.image.img_to_array(img)
/ 255.0
            images.append(arr)
            labels.append(class_id)
        print(f"  {class_name:>8}: {len(files)} зображень")

    X = np.array(images, dtype=np.float32)
    y = np.array(labels, dtype=np.int32)
    return X, y, classes
    model = tf.keras.Sequential(
        [ tf.keras.layers.Input(shape=(IMAGE_SIZE, IMAGE_SIZE,
3)),
            tf.keras.layers.Conv2D(32, 3, activation="relu"),
            tf.keras.layers.MaxPooling2D(2),
            tf.keras.layers.Conv2D(64, 3, activation="relu"),
            tf.keras.layers.MaxPooling2D(2),
            tf.keras.layers.Conv2D(128, 3, activation="relu"),
            tf.keras.layers.MaxPooling2D(2),
            tf.keras.layers.Flatten(),
            tf.keras.layers.Dense(128, activation="relu"),
            tf.keras.layers.Dropout(0.3),
            tf.keras.layers.Dense(len(class_names),
activation="softmax"),
        ],
        name="CNN_Images",)
    model.compile( optimizer="adam",
loss="sparse_categorical_crossentropy", metrics=["accuracy"],)
    model.summary()
```

А.5 Повна оцінка натренованих моделей (3_evaluate.py)

Лістинг А.5 — Повна оцінка натренованих моделей

```
    print("Інференс...")
    t0 = time.time()
    preds = model.predict(X_test, verbose=0)
    inference_time = (time.time() - t0) / len(X_test) * 1000  #
мс на 1 кадр
    y_pred = np.argmax(preds, axis=1)

    # --- Метрики ---
    acc = accuracy_score(y_test, y_pred)
    report = classification_report(
        y_test, y_pred,
        labels=list(range(len(class_names))),
        target_names=class_names,
        zero_division=0,
        output_dict=True,
    )
    cm = confusion_matrix(y_test, y_pred,
labels=list(range(len(class_names))))
    size_kb = os.path.getsize(cfg["model_file"]) / 1024
    print(f"\n--- Результати ---")
    print(f"Загальна точність:     {acc:.4f}  ({acc*100:.2f}%)")
    print(f"Час інференсу 1 кадру: {inference_time:.2f}
мс  (~{1000/inference_time:.0f} FPS)")
    print(f"Розмір моделі:         {size_kb:.1f} КБ")
    print(f"\nClassification report:")
    print(classification_report(
        y_test, y_pred,
        labels=list(range(len(class_names))),
        target_names=class_names,
        zero_division=0,
    ))
```

А.6 Зведене порівняння трьох моделей (4_compare_models.py)

Лістинг А.6 — Порівняння моделей з побудовою діаграм та Pareto-графіку

```
    fig, axes = plt.subplots(1, 3, figsize=(15, 5))
    names = [LABELS[r["key"]] for r in rows]
    colors = ["#2E86AB", "#A23B72", "#F18F01"][: len(rows)]
    axes[0].bar(names, [r["accuracy"] * 100 for r in rows],
color=colors)
    axes[0].set_ylabel("Точність, %");
axes[0].set_title("Точність на тест-сеті")
    axes[0].set_ylim(0, 105); axes[0].grid(axis="y", alpha=0.3)
    for i, r in enumerate(rows):
        axes[0].text(i, r["accuracy"] * 100 + 1,
f"{r['accuracy']*100:.1f}%",
                     ha="center", fontweight="bold")

    axes[1].bar(names, [r["fps"] for r in rows], color=colors)
    axes[1].set_ylabel("FPS"); axes[1].set_title("Швидкодія (на
CPU)")
    axes[1].grid(axis="y", alpha=0.3)
    for i, r in enumerate(rows):
        axes[1].text(i, r["fps"] + 1, f"{r['fps']:.0f}",
ha="center", fontweight="bold")
    axes[2].bar(names, [r["size_kb"] for r in rows],
color=colors)
    axes[2].set_ylabel("Розмір моделі, КБ");
axes[2].set_title("Розмір моделі")
    axes[2].set_yscale("log")
    axes[2].grid(axis="y", alpha=0.3, which="both")
    for i, r in enumerate(rows):
        axes[2].text(i, r["size_kb"] * 1.1,
f"{r['size_kb']:.0f}",
                     ha="center", fontweight="bold")
    fig.suptitle(f"Порівняння моделей — {title_suffix}",
fontsize=13, fontweight="bold")
    plt.tight_layout()
    bar_path = os.path.join(source_dir, "comparison_bars.png")
    plt.savefig(bar_path, dpi=200)
    plt.close()
    print(f"Збережено: {bar_path}")
    plt.figure(figsize=(9, 6))
```

Лістинг А.6 , аркуш 1

```
for r, c in zip(rows, colors):
        plt.scatter(
            r["fps"], r["accuracy"] * 100,
            s=max(80, np.sqrt(r["size_kb"]) * 30),
            c=c, alpha=0.7, edgecolors="black", linewidths=1.5,
            label=f"{LABELS[r['key']].replace(chr(10), ' ')}
({r['size_kb']:.0f} КБ)",
        )
        plt.annotate(
            LABELS[r["key"]].replace("\n", " "),
            (r["fps"], r["accuracy"] * 100),
            xytext=(10, 10), textcoords="offset points",
fontsize=10,
        )
    plt.xlabel("Швидкодія, FPS")
    plt.ylabel("Точність, %")
    plt.title(f"Pareto frontier ({title_suffix})\nрозмір кружка
пропорційний розміру моделі")
    plt.grid(True, alpha=0.3)
    plt.legend(loc="best")
    plt.tight_layout()
    pareto_path = os.path.join(source_dir, "pareto.png")
    plt.savefig(pareto_path, dpi=200)
    plt.close()
```

Лістинг А.6 , аркуш 2

А.7 Розпізнавання жестів у реальному часі через веб-камеру (5_inference.py)

Лістинг А.7 — Розпізнавання жестів у режимі реального часу

```
    while True:
        ret, frame = cap.read()
        if not ret:
            break

        frame = cv2.flip(frame, 1)
        rgb_frame = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
        mp_image = mp.Image(image_format=mp.ImageFormat.SRGB,
data=rgb_frame)
```

Лістинг А.7 , аркуш 1

```
        result = detector.detect(mp_image)

        label_display = "Searching..."
        confidence = 0.0

        if result.hand_landmarks:
            for hand_landmarks in result.hand_landmarks:
                h, w, _ = frame.shape
                for lm in hand_landmarks:
                    cv2.circle(frame, (int(lm.x * w), int(lm.y *
h)), 4, (0, 255, 0), -1)
features = np.array([landmarks_to_distances(hand_landmarks)],
dtype=np.float32)
                preds = model(features, training=False).numpy()
                idx = int(np.argmax(preds[0]))
                confidence = float(np.max(preds[0]))
                label_display = class_names[idx]
        color = (0, 255, 0) if confidence > 0.8 else (0, 165,
255)
        cv2.putText(
            frame, f"{label_display} ({confidence*100:.1f}%)",
            (20, 50), cv2.FONT_HERSHEY_SIMPLEX, 1.5, color, 3,
        )
        curr_time = time.time()
        fps = 1 / (curr_time - prev_time)
        prev_time = curr_time

        cv2.imshow("Sign Language Recognition", frame)
        if cv2.waitKey(1) & 0xFF == ord("q"):
            break
    cap.release()
    detector.close()
    cv2.destroyAllWindows()
```

Лістинг А.7 , аркуш 2

ДОДАТОК Б

Графічні матеріали — криві навчання та матриці помилок

Б.1 Матриця помилок MLP на 63 координатах — internal_test/raw/confusion_matrix.png.

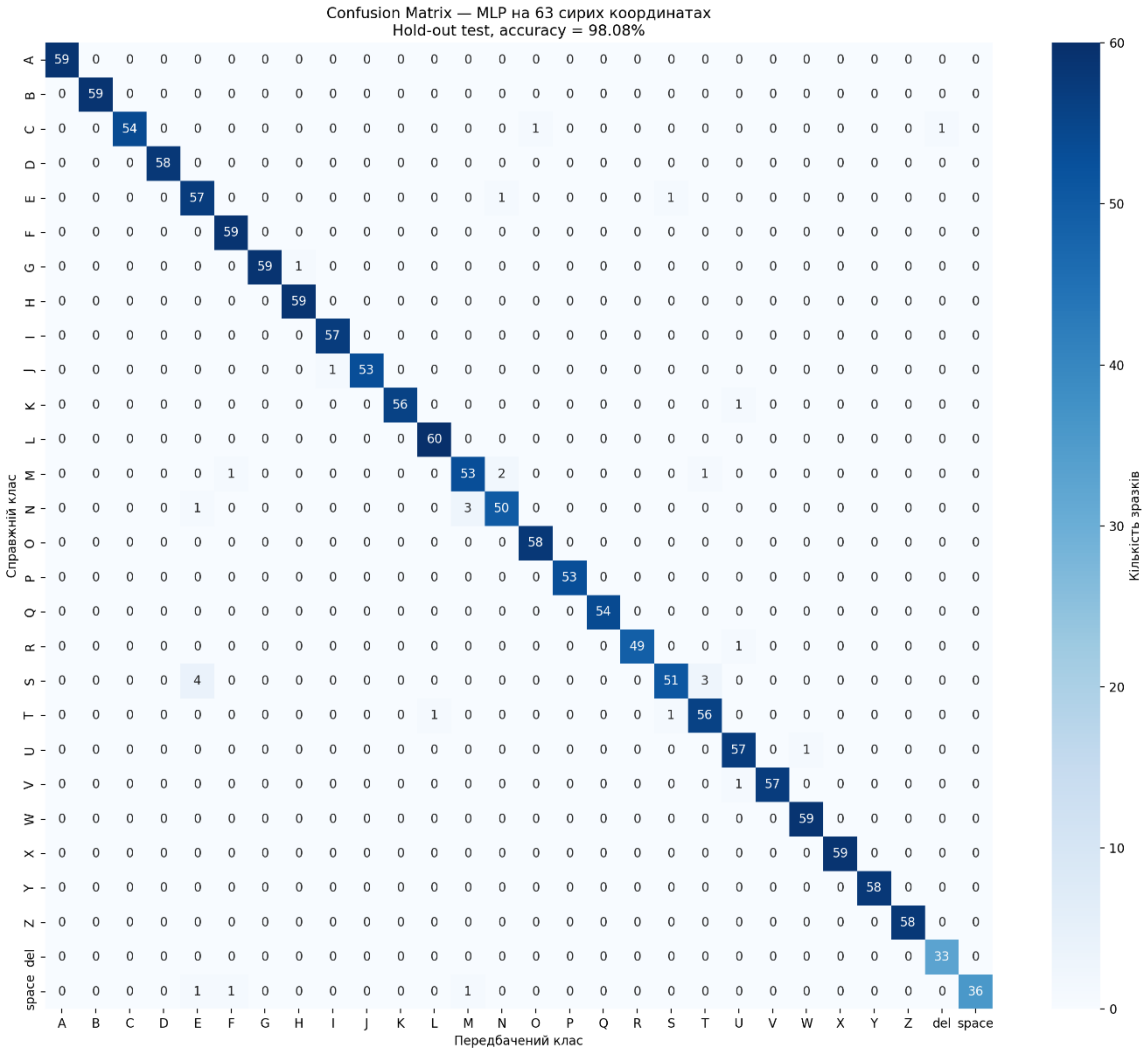


Рисунок Б.1 — Матриця помилок альтернативної моделі (MLP на 63 сирих координатах)

Б.2 Матриця помилок CNN — internal_test/cnn/confusion_matrix.png.

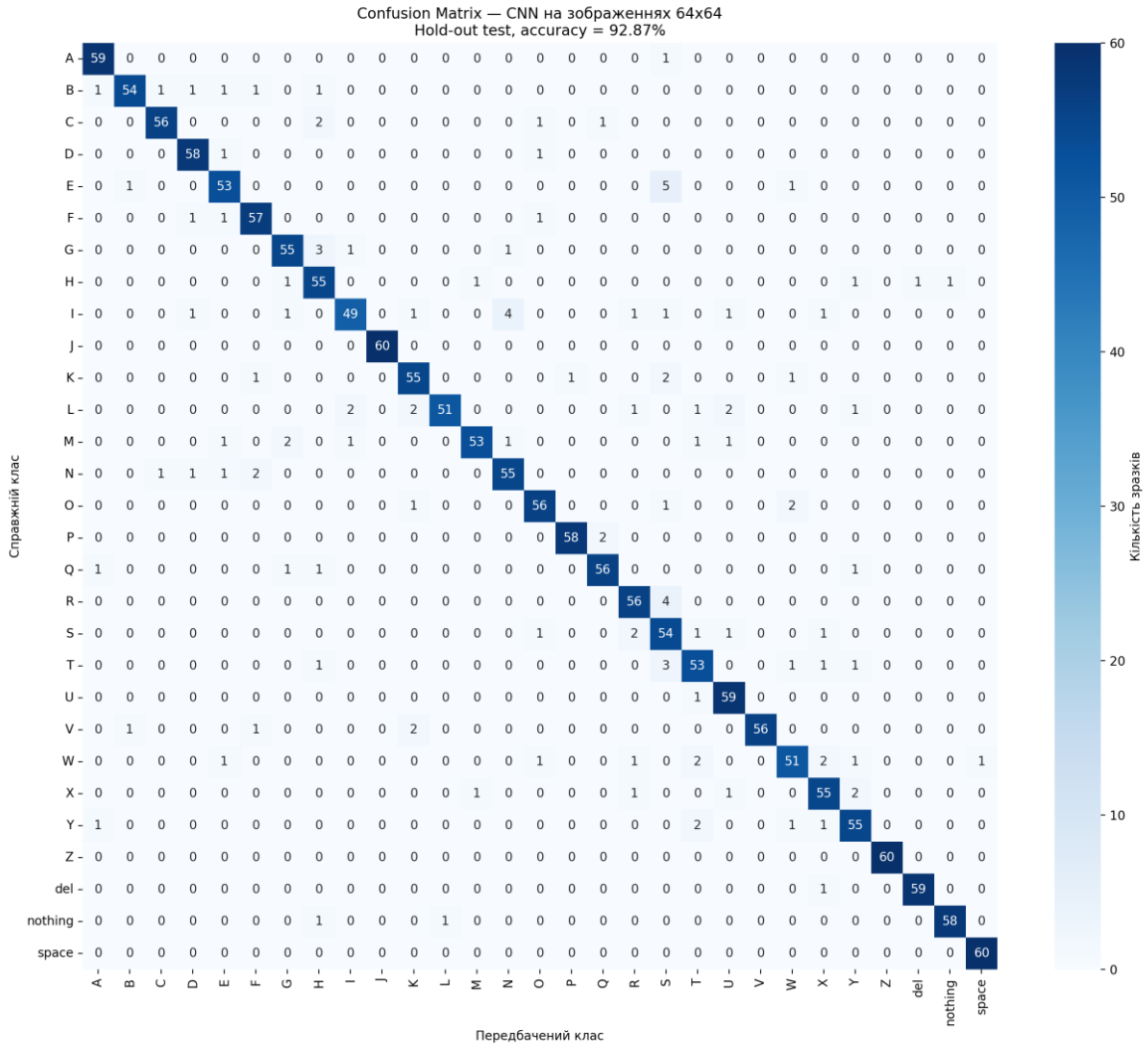


Рисунок Б.2 — Матриця помилок альтернативної моделі (CNN на зображеннях 64×64)

ДОДАТОК В

Матеріали апробації результатів дослідження

Рисунок В.1 — Титульна сторінка збірника матеріалів конференції «Інтелектуальні інформаційні системи», Миколаїв, 2025



Міністерство освіти і науки України
Чорноморський національний університет ім. Петра Могили
Факультет комп'ютерних наук
Кафедра інтелектуальних інформаційних систем

Інтелектуальні інформаційні системи

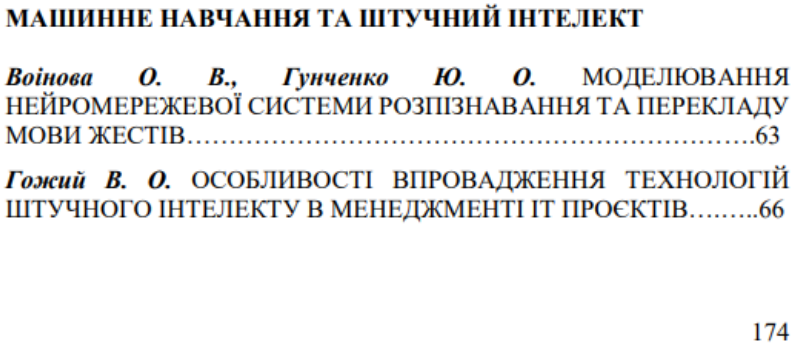
Матеріали всеукраїнської науково-практичної конференції молодих вчених, аспірантів і студентів

ТЕЗИ

4 – 5 грудня 2025 року

Миколаїв

Рисунок В.2 — Сторінка змісту збірника з назвою тез доповіді



МАШИННЕ НАВЧАННЯ ТА ШТУЧНИЙ ІНТЕЛЕКТ

Воінова О. В., Гунченко Ю. О. МОДЕЛЮВАННЯ НЕЙРОМЕРЕЖЕВОЇ СИСТЕМИ РОЗПІЗНАВАННЯ ТА ПЕРЕКЛАДУ МОВИ ЖЕСТІВ……63

Гожий В. О. ОСОБЛИВОСТІ ВПРОВАДЖЕННЯ ТЕХНОЛОГІЙ ШТУЧНОГО ІНТЕЛЕКТУ В МЕНЕДЖМЕНТІ ІТ ПРОЄКТІВ……..66

174

Рисунок В.3 — Тези доповіді «Моделювання нейромережевої системи розпізнавання та перекладу мови жестів»

МАШИННЕ НАВЧАННЯ ТА ШТУЧНИЙ ІНТЕЛЕКТ

УДК 004.85; 004.89

Воїнова О.В., Гунченко Ю.О.
Одеський національний університет імені І.І.Мечникова, м. Одеса, Україна

МОДЕЛЮВАННЯ НЕЙРОМЕРЕЖЕВОЇ СИСТЕМИ РОЗПІЗНАВАННЯ ТА ПЕРЕКЛАДУ МОВИ ЖЕСТІВ

Мова жестів є повноцінним засобом комунікації для мільйонів людей з порушеннями слуху, однак її застосування в повсякденному житті часто обмежується недостатньою кількістю перекладачів та відсутністю технологій, здатних працювати у реальному часі. Це створює бар'єри у спілкуванні, доступі до освіти, отриманні медичних чи державних послуг.

Мова жестів використовує певні рухи рук, міміку та вираз обличчя для передавання алфавіту, цифр, слів та емоцій. Кожна країна має свою систему жестів з різними рухами рук, наприклад американська мова жестів (ASL) та індіанська мова жестів (ISL). Найпоширеніша мова жестів у світі — ASL.

З ростом використання комп'ютерів і смартфонів стало зручніше застосовувати їх для спілкування з глухими людьми. За даними Всесвітньої організації охорони здоров'я, глухі люди становлять близько 6% населення світу, тому необхідно впроваджувати технології для інтеграції людей з інвалідністю в спільноти. [1]. Потрібні переклади з жестової мови на мовлення та навпаки, щоб полегшити це спілкування.

З останніх досліджень досліджень перекладу жестів можна виділити наступні:

1. У роботі «Алгоритм лінійного дискримінантного аналізу (LDA)» індійську жестову мову розпізнавали в MATLAB, використовуючи LDA-класифікатор. Набір даних складався з десяти зображень для кожного з 26 жестів індійського алфавіту, а результати перетворювали у текст і голос [2].

2. «Згорткова нейронна мережа» — застосування CNN для класифікації жестів у відеопотоці (30–40 кадрів/с). Попередню обробку

Рисунок В.4 — Тези доповіді «Моделювання нейромережевої системи розпізнавання та перекладу мови жестів»

виконували за допомогою OpenCV, а розпізнані жести озвучували через Google Audio. Набір даних включав жести літер, цифр і статичних слів [3].

3. У дослідженні «Глибоке навчання» (1000 відео індійських жестів) запропонували модель реального часу, де MobileNet використовували для класифікації 1000 дій [4].

4. «Мультимодальний з урахуванням скелета» — представлена систему, яка працює з трьома наборами: турецьким, китайським та американським [5].

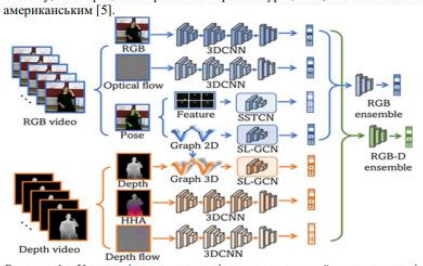


Рисунок 1 – Концепція системи розпізнавання жестової мови на основі скелету з глобальною ансамблевою моделлю (SAM-SLR-v2)

Оскільки представлені дослідження суттєво відрізняються за методами, обсягами даних та результатами, важливо порівняти їх між собою, щоб визначити сильні сторони кожного підходу та загальні тенденції розвитку систем розпізнавання жестової мови, що представлено у таблиці 1 [2].

Сучасні підходи до розпізнавання й перекладу жестової мови засновані на нейромережах і забезпечують підвищену точність та швидкість. Найперспективніші CNN та мультимодальні моделі працюють добре навіть за складних умов, але часто обмежені даними або обладнанням.

В роботі проведено аналіз існуючих систем перекладу жестової мови, показано, що подальший розвиток таких систем має бути спрямований на створення універсальних моделей, здатних працювати

Рисунок В.5 — Тези доповіді «Моделювання нейромережевої системи розпізнавання та перекладу мови жестів»

з різними мовами жестів, підтримувати динамічні послідовності та забезпечувати роботу в режимі реального часу. Це дозволить значно розширити можливості комунікації людей із порушеннями слуху та сприятиме їхній інтеграції у цифрове середовище та суспільство загалом.

Таблиця 1 – Порівняння методів розпізнавання жестів

Підхід	Використаний набір даних	Мова жестів	Показники ефективності
Метод лінійного дискримінантного аналізу (LDA)	10 зображень для кожного з 26 жестів індійського алфавіту	Індійська	Заявлена висока точність
Згорткова нейронна мережа	Комплекс зображень літер, цифр та статичних жестів	Американська	Точність становила 86%
Глибоке навчання з використанням MobileNet	1000 дій, по 30 відео (30 кадрів) на кожен жест	Індійська	Точність — 92,1%
Мультимодальна скелетна система	Турецький, китайський та американський набори даних	Турецька, китайська, американська	ТОП-1 точність: 99,76%, 99,95%, 84,94%

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Y. Shovkovyi . Automatic sign language translation system using neural network technologies and 3D animation. *Innovative technologies and scientific solutions for industries.* 2023. № 4(26). С. 108–121. URL: https://doi.org/10.30837/itssi.2023.26.108 (дата звернення: 25.11.2025).

2. Najib F. M. Sign language interpretation using machine learning and artificial intelligence. *Neural computing and applications.* 2024. URL: https://doi.org/10.1007/s00521-024-10395-9 (дата звернення: 25.11.2025).

3. Shubham Kr Mishra. Recognition of hand gestures and conversion of voice for betterment of deaf and mute people. *CrossRef listing of deleted dois.* 2019. Vol. 1. URL: https://doi.org/10.1007/978-3-8349-6966-8_5 (дата звернення: 25.11.2025).

4. Amrithaa I. S. Sandhiya A. Translating Indian sign language to text using deep learning. *International journal of innovative science and research technology.* 2022. Т. 7, № 5. С. 7154–7161.

Рисунок В.6 — Відомості про видавництво збірника

В авторській редакції
Комп'ютерна верстка *В. Шевченко*
Друк *С. Волинець*. Фальцювально-палітурні роботи *О. Мішалкіна.*

Підп. до друку 30.12.2025.
Формат 60x84 $^{1}/_{16}$. Папір офсет.
Гарнітура «Times New Roman». Друк ризограф.
Ум. друк. арк. 10,5. Обл.-вид. арк. 8,2.
Тираж 10 пр. Зам. № 7156.

Видавець і виготовлювач: ЧНУ ім. Петра Могили.
54003, м. Миколаїв, вул. 68 Десантників, 10.
Тел.: 8 (0512) 50–03–32, 8 (0512) 76–55–81,
e-mail: rector@chmnu.edu.ua.
Свідоцтво суб'єкта видавничої справи ДК № 6124 від 05.04.2018.