Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

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

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти другий (магістерський)

ДОСІДЖЕННЯ МЕТОДІВ РОЗПІЗНАВАННЯ ТА ВІДСТЕЖЕННЯ РУХІВ РУК ЛЮДИНИ ЗА ДОПОМОГОЮ НЕЙРОНИХ МЕРЕЖ
(тема)

Виконав:
здобувач 2 року навчання,
групи ІНФм-24-2
Черепов Д. Е.
(прізвище, ініціали)

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

Тип програми освітньо-професійна

Освітня програма Інформатика
(повна назва освітньої програми)

Науковий керівник проф. Машталір С. В.
(посада, прізвище, ініціали)

Допускається до захисту

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

2025 р.

Харківський національний університет радіоелектроніки

Факультет Інформаційно-аналітичних технологій та менеджменту

Кафедра Інформатики

Рівень вищої освіти другий (магістерський)

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

Тип програми освітньо-професійна

Освітня програма Інформатика
(повна назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
«____»_____ 2025 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

здобувачеві Черепову Дмитру Едуардовичу
(прізвище, ім'я, по батькові)

1. Тема роботи Дослідження методів розпізнавання та відстеження рухів рук людини за допомогою нейронних мереж

затверджена наказом університету від 14 листопада 2025 року № 1045Ст
2. Термін подання здобувачем роботи до екзаменаційної комісії 9 грудня 2025 р.
3. Вихідні дані до роботи науково-методична та науково-технічна література, матеріали конференцій, гібридний метод навчання на основі синтетичних даних та архітектури HandTransformer, бібліотека комп'ютерного зору MediaPipe, засоби для програмної реалізації Python, PyTorch, OpenCV, NumPy, середовище розробки PyCharm.

4. Перелік питань, що потрібно опрацювати в роботі
1. Аналіз сучасних методів розпізнавання та відстеження рухів рук людини.
2. Аналіз літературних джерел щодо використання трансформерних архітектур та скелетного відстеження у задачах комп'ютерного зору.
3. Формування гібридного методу розпізнавання жестів на основі поєднання синтетичних даних та архітектури HandTransformer.
4. Програмна реалізація системи збору даних, навчання нейронної мережі та розпізнавання жестів у реальному часі.
5. Експериментальне дослідження розробленої системи та порівняльний аналіз ефективності з альтернативними підходами.

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (п.5 включається до завдання за рішенням випускової кафедри) актуальність проблеми розпізнавання жестів руки у безконтактних інтерфейсах, об'єкт та мета дослідження, постановка задачі, блок-схема алгоритму гібридного методу розпізнавання, архітектура розробленої нейронної мережі та схема обробки даних,приклади сформованого набору даних, інтерфейс розробленого програмного застосунку для збору даних та інференсу, результати експериментального дослідження та порівняльний аналіз ефективності, висновки, перспективи та апробація роботи.

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

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

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Строк / терміни виконання етапів роботи	Примітка
1	Отримання завдання на кваліфікаційну роботу	29.09.2025	
2	Аналіз завдання, підбір літератури	29.09.25-11.10.25	
3	Аналіз літератури з досліджуваної проблеми	11.10.25-14.10.25	
4	Особливості методів розпізнавання жестів на основі трансформерних архітектур	14.10.25-21.10.25	
5	Дослідження гібридного методу розпізнавання жестів	21.10.25-28.10.25	
6	Програмна реалізація	28.10.25-10.11.25	
7	Обґрунтування отриманих результатів	10.11.25-16.11.25	
8	Оформлення пояснювальної записки	16.11.25-30.11.25	
9	Перевірка на нормоконтроль	08.12.25-12.12.25	
10	Перевірка на плагіат	12.12.25-14.12.25	
11	Рецензування	14.12.25-15.12.25	
12	Підготовка презентації та доповіді	15.12.25-16.12.25	
13	Занесення роботи в електронний архів	18.12.25	
14	Попередній захист кваліфікаційної роботи	18.12.25	

Дата видачі завдання 29 вересня 2025 р.

Здобувач _____
(підпис)

Керівник роботи _____ проф. Машталір С. В.
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ

Пояснювальна записка до кваліфікаційної роботи: 81 с., 5 табл., 14 рис., 42 джерела.

ВІДСТЕЖЕННЯ РУХІВ, ГІБРИДНА СИСТЕМА, ГЛИБИННЕ НАВЧАННЯ, КОМП'ЮТЕРНИЙ ЗІР, НЕЙРОННІ МЕРЕЖІ, РОЗПІЗНАВАННЯ ЖЕСТІВ, MEDIAPIPE, PYTORCH, VISION TRANSFORMER.

Об'єктом дослідження є процес розпізнавання та відстеження рухів рук людини у відеопотоці в реальному часі.

Предметом дослідження є методи класифікації статичних і динамічних жестів на основі координат ключових точок та трансформерних моделей.

Метою дослідження є підвищення точності розпізнавання жестів шляхом розроблення гібридного підходу, що поєднує визначення ключових точок за допомогою MediaPipe, синтетичні та реальні дані, а також трансформерну архітектуру HandTransformer.

Використано методи комп'ютерного зору, глибинного навчання, скелетного відстеження та аналізу часових рядів.

Наукова новизна роботи полягає у поєднанні синтетично згенерованих і реальних траєкторій жестів для навчання єдиної трансформерної моделі, здатної розпізнавати як статичні, так і динамічні жести.

Взаємозв'язок з іншими роботами полягає у використанні сучасних підходів комп'ютерного зору та нейронних мереж у системах жестового керування.

Рекомендації щодо використання результатів роботи: система може бути застосована в мультимедійних, інтерактивних та безконтактних інтерфейсах.

У результаті дослідження розроблено прототип жестової системи, що демонструє ефективність комбінування синтетичних і реальних даних у режимі реального часу.

ABSTRACT

Explanatory note to the qualification work: 81 pages, 5 tables, 14 figures, 42 sources.

COMPUTER VISION, DEEP LEARNING, GESTURE RECOGNITION, HYBRID SYSTEM, MEDIAPIPE, MOTION TRACKING, NEURAL NETWORKS, PYTORCH, VISION TRANSFORMER.

The object of the research is the process of recognizing and tracking human hand movements in real-time video streams.

The subject of the research is methods for classifying static and dynamic gestures based on keypoint coordinates and transformer models.

The aim of the research is to improve gesture recognition accuracy by developing a hybrid approach that combines keypoint detection using MediaPipe, synthetic and real data, as well as the HandTransformer architecture.

Methods of computer vision, deep learning, skeletal tracking, and time series analysis were used.

The scientific novelty of the work lies in combining synthetically generated and real gesture trajectories to train a single transformer model capable of recognizing both static and dynamic gestures.

The interconnection with other works lies in the use of modern computer vision and neural network approaches in gesture control systems.

Recommendations for using the results of the work: the system can be applied in multimedia, interactive, and contactless interfaces.

As a result of the research, a prototype of a gesture system was developed, demonstrating the effectiveness of combining synthetic and real data in real-time.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

КТ – ключова точка

ШІ – штучний інтелект

CNN – Convolutional Neural Network (згорткова нейронна мережа)

Cross-Entropy Loss – крос-ентропія

DNN – Deep Neural Network (глибока нейронна мережа)

FPS – Frames Per Second (кадри за секунду)

GPU – Graphics Processing Unit (графічний процесор)

IDE – Integrated Development Environment (інтегроване середовище розробки)

Loss Function – функція втрат

MediaPipe – бібліотека комп’ютерного зору для виявлення ключових точок

PyTorch – бібліотека глибинного навчання

Softmax – функція активації для класифікації

Transfer Learning – трансферне навчання

ViT – Vision Transformer (трансформер для обробки зображень)

VGG – Visual Geometry Group (група візуальної геометрії)

ВСТУП

Стрімкий розвиток технологій комп’ютерного зору та глибинного навчання за останні роки суттєво змінив підходи до аналізу рухів і жестів людини. Алгоритми скелетного відстеження, зокрема MediaPipe Hands, забезпечили можливість отримання точних координат ключових точок кисті у реальному часі, що відкрило шлях до побудови інтуїтивних та безконтактних систем керування. Паралельно зі зростанням обчислювальних можливостей трансформерні архітектури продемонстрували високу ефективність у моделюванні послідовностей, що робить їх перспективними для задач класифікації рухів руки, включаючи як статичні, так і динамічні жести.

Попри значні досягнення, існують обмеження, що стримують розвиток подібних систем. Зокрема, моделі, що працюють лише з реальними даними, залежні від великих та різноманітних навчальних вибірок, які потребують тривалого й трудомісткого процесу збирання. Крім того, традиційні підходи, орієнтовані на аналіз зображень, чутливі до змін освітлення, фону та положення камери. Це знижує точність розпізнавання та ускладнює роботу системи в неконтрольованих умовах. Таким чином, виникає потреба у гібридних методах, які здатні компенсувати нестачу реальних даних та забезпечити стійкість до зовнішніх чинників.

Одним із перспективних рішень є поєднання синтетично згенерованих жестів і реальних записів рухів руки. Синтетичні дані дозволяють моделювати широкий спектр варіацій руху, тоді як реальні дані забезпечують відповідність реальним сценаріям використання. Інтеграція таких наборів у комплексі з трансформерною моделлю, здатною опрацьовувати послідовності координат ключових точок, створює основу для побудови універсальної системи розпізнавання жестів у реальному часі. Саме тому у даній роботі використано трансформерну архітектуру HandTransformer, яка моделює просторово-часові залежності в рухах кисті та класифікує 10 жестів, включаючи статичні (Palm, Fist, Point, Peace) і динамічні (Swipe та Zoom).

Актуальність теми зумовлена зростаючим запитом на безконтактні інтерфейси керування у сферах мультимедіа, робототехніки, медицини, AR/VR та розумних систем. У багатьох випадках потрібні рішення, здатні працювати на звичайному обладнанні, забезпечувати високу точність і реагувати в реальному часі. Традиційні моделі не завжди можуть виконати ці вимоги через обмеження навчальних вибірок, чутливість до шумів або недостатню здатність обробляти динамічні рухи. Гібридні підходи, що комбінують скелетне відстеження, синтетичні дані та трансформерні моделі, дозволяють подолати ці обмеження.

Наукова задача, що вирішується в роботі, полягає у дослідженні точності та стабільності системи розпізнавання жестів при комбінуванні синтетичних і реальних даних, а також у визначенні того, як структура моделі, параметри генерації даних та способи попередньої обробки впливають на якість класифікації статичних і динамічних рухів. Важливою складовою є аналіз архітектури трансформера, методів інтерполяції та аугментації часового ряду, а також оцінювання продуктивності системи в режимі реального часу.

Таким чином, комплексне поєднання скелетного відстеження, синтетичного моделювання рухів та трансформерної архітектури забезпечує створення ефективної системи розпізнавання жестів, здатної працювати у реальних умовах. Результати роботи можуть бути використані під час розроблення жестових інтерфейсів керування, інтерактивних платформ, навчальних застосунків і безконтактних систем взаємодії.

1 АНАЛІЗ МЕТОДІВ РОЗПІЗНАВАННЯ ТА ВІДСТЕЖЕННЯ РУХІВ РУК

1.1 Базові підходи до розпізнавання жестів руки.

Розпізнавання жестів руки як окремий науковий напрям сформувалося на перетині методів комп’ютерного зору та математичного моделювання. На ранніх етапах розвитку ключовою проблемою було отримання стабільних ознак, здатних відображати форму та положення кисті на зображенні. Саме класичні методи аналізу контурів, силуетів та градієнтів стали підґрунтям для подальшої еволюції систем жестової взаємодії [1].

Перші дослідження орієнтувалися на опис руки через прості геометричні характеристики, такі як контури, опорні точки або сегментація за кольором шкіри. Однак такі методи виявили суттєві обмеження – чутливість до освітлення, варіативність фону та неможливість коректної роботи при оклюзіях. Проте саме вони сформували концепцію, згідно з якою ефективність системи залежить не лише від алгоритму класифікації, але й від якості ознак, що її описують [2].

Подальший інтерес науковців був спрямований на вивчення руху руки як часової послідовності. Це призвело до появи підходів на основі оптичного потоку та аналізу траєкторій. Ці методи стали перехідним етапом від аналізу статичних зображень до моделювання динаміки жестів, що надалі дозволило застосувати глибинні та трансформерні моделі для обробки часових залежностей [3].

1.2 Методи на основі класичних комп’ютерних ознак.

Одним із найбільш досліджених класів підходів до розпізнавання жестів є методи, що ґрунтуються на класичних комп’ютерних ознаках. Вони передбачають аналіз зображення для виділення компактного дескриптора, який

відображає найбільш інформативні параметри кисті. До таких дескрипторів належать SIFT, SURF, HOG, LBP та різноманітні похідні від геометричних контурів. Їхня перевага полягає у відносній простоті обчислень і здатності працювати за умов обмежених ресурсів [4].

Методи на основі дескрипторів зазвичай поєднуються із класичними класифікаторами: SVM, деревами рішень або kNN. Однак, незважаючи на їхню популярність, вони мають суттєві обмеження: складність ручного вибору параметрів, невисоку стійкість до освітлення та обмежену здатність до узагальнення. Тому з часом такі підходи все частіше використовуються лише як підготовчий етап у гібридних моделях [5].

Проте інтеграція структурних дескрипторів із сучасними системами (наприклад, MediaPipe Hands) дозволила значно підвищити стабільність даних. Оскільки дескриптори можуть будуватися не за сирим зображенням, а за структурними точками, вони стали інструментом для підсилення нейронних моделей або зменшення розмірності вхідного простору [6].

1.3 Моделі на основі скелетних та просторових описів руки.

З переходом до структурних методів опису руки дослідники отримали змогу аналізувати не зображення як таке, а геометрію руху. Скелетні моделі представляють руку як множину ключових точок, з’єднаних у графову структуру, що дозволяє оцінювати не лише положення кисті, але й взаємне розташування пальців, кути між фалангами та просторові пропорції. Це стало фундаментом для методів, що працюють із landmark-даними [7].

1.3.1 Просторові дескриптори та геометричні моделі

Просторові моделі використовують геометричні характеристики руки: відстані між суглобами, довжини сегментів, кути між ними. Таке представлення забезпечує інваріантність до фону та освітлення, оскільки аналіз проводиться не над пікселями, а над просторовими зв'язками між точками. Модель подається як граф, де вершини –ключові точки, а ребра – сегменти пальців [8].

Подібні моделі ефективні для розпізнавання як статичних, так і динамічних жестів. Їх часто комбінують із класичними методами (для попередньої обробки) або з нейронними мережами (для класифікації), оскільки просторовий опис дозволяє суттєво зменшити розмірність вхідних даних без втрати інформативності.

1.3.2 Скелетні моделі у задачах розпізнавання жестів.

Скелетні моделі дозволяють створювати універсальні описи руки, незалежні від зовнішніх чинників. Вони добре інтегруються з CNN, RNN, LSTM та трансформерами, оскільки подають руку у вигляді векторів, що легко піддаються математичній обробці. Ключовою перевагою є можливість аналізувати як просторову структуру, так і її зміну в часі, що критично важливо для динамічних жестів [9].

Скелетне представлення є компактним та інформативним. Оскільки модель працює лише з координатами, а не зі зображенням, це дозволяє скоротити обчислювальні витрати й зменшити чутливість до шумів. Саме тому такі моделі стали основою сучасних систем реального часу – від VR до роботизованих платформ.

1.3.3 Використання MediaPipe Hands для формування скелетних ознак

MediaPipe Hands є одним із найпоширеніших інструментів для визначення структурних точок руки. Система генерує 21 landmark, що описує ключові суглоби й положення пальців, забезпечуючи стабільність навіть за швидких рухів або часткових оклюзій. Отримані координати стали стандартом для подальшої інтеграції у глибинні моделі [10].

1.4 Методи глибинного навчання у розпізнаванні жестів руки

Глибинне навчання суттєво змінило підхід до розпізнавання жестів руки, оскільки дозволило автоматично виділяти ознаки та будувати адаптивні моделі. На відміну від класичних методів, нейронні мережі здатні формувати багаторівневі представлення даних і працювати як зі зображеннями, так і зі структурними координатами. Це забезпечило значний стрибок у точності сучасних систем [11].

1.4.1 Застосування згорткових нейронних мереж (CNN)

CNN стали першими моделями, що продемонстрували високу ефективність у задачах комп’ютерного зору. Основною операцією є згортка:

$$y(i,j) = (X * K)(i,j) = \sum_m \sum_n X(i+m, j+n) * K(m,n), \qquad (1.1)$$

де $y(i,j)$ – значення вихідного пікселя на координатах (i,j);

X – вхідна матриця (зображення або карта ознак попереднього шару);

K – ядро згортки (фільтр), що містить вагові коефіцієнти, які навчаються;

m, n – індекси, що пробігають по висоті та ширині ядра згортки відповідно.

Результат згортки зазвичай проходить через функцію активації (наприклад, ReLU) для внесення нелінійності, що дозволяє мережі вивчати складні залежності. CNN добре працюють зі статичними жестами, а також у гібридних моделях, де на вхід подаються не зображення, а векторизовані координати landmarks [12].

1.4.2 Моделі рекурентного типу та LSTM

Для аналізу динамічних жестів використовуються рекурентні моделі, зокрема LSTM, які враховують часову залежність між кадрами. LSTM-блок описується системою рівнянь:

$$\begin{cases} f_t = \sigma\big(W_f[h_{t-1}, x_t] + b_f\big) \\ i_t = \sigma(W_i[h_{t-1}, x_i] + b_i) \\ \tilde{c}_t = \tan(W_c[h_{t-1}, x_t] + b_c) \\ c_t = f_t \odot c_{t-1} + i_t \odot \tilde{c}_t \\ o_t = \sigma(W_o[h_{t-1}, x_t] + b_0) \\ h_t = o_t \odot \tan(c_t) \end{cases}, \quad (1.2)$$

де f_t, i_t, o_t – вентилі забування;

$\tilde{c}_t$ – кандидат у стан комірки;

c_t – оновлений стан комірки пам’яті;

h_t – прихований стан;

σ, tan – функції активації;

W, b – вагові коефіцієнти та зсуви, що навчаються.

Такі моделі здатні зберігати довготривалу інформацію про рух руки, що робить їх корисними для класифікації жестів, де важлива форма траєкторії, а не лише окремий кадр. Хоча сьогодні LSTM частково витіснені трансформерами, їхній внесок у розвиток temporal-моделювання є визначальним [13].

1.4.3 Тривимірні згорткові мережі та гібридні моделі

Одним із напрямів розвитку глибинних методів стало використання тривимірних згорткових мереж (3D-CNN). На відміну від 2D-CNN, що оперують лише просторовими ознаками, 3D-CNN виконують операцію згортки одночасно у просторовому та часовому вимірах. Це дозволяє моделі аналізувати не окремі кадри, а цілі відеофрагменти, виділяючи динамічні характеристики руху [14].

Хоча 3D-CNN демонструють високу точність, їхнім недоліком є значні обчислювальні витрати та висока потреба у великих обсягах даних. Саме тому у сучасних системах часто застосовують гібридні моделі, де просторову інформацію обробляє 2D-CNN, а часову – окремий модуль, наприклад LSTM або трансформер. Такий поділ дозволяє зберегти інформативність і значно зменшити навантаження на модель [15].

Гібридні моделі стали основою більшості практичних систем розпізнавання жестів, оскільки поєднують у собі точність глибинних методів і стійкість структурних скелетних представлень. На основі таких підходів створюються системи контролю роботів, безконтактні мультимедійні інтерфейси та VR-рішення, що потребують стабільної роботи в реальному часі.

1.5 Трансформерні архітектури у розпізнаванні жестів руки

Поява трансформерних моделей стала одним із найважливіших кроків у розвитку сучасних систем розпізнавання жестів. На відміну від CNN і LSTM, які мають обмеження у роботі з довгими залежностями, трансформери здатні моделювати складні взаємозв’язки між елементами послідовності завдяки механізму самоуваги (self-attention). Це робить їх особливо ефективними у задачах, де жест визначається не лише положенням руки, а й зміною її конфігурації в часі [16].

Ключовим елементом цієї архітектури є механізм самоуваги (self-attention). Детальний математичний опис цього механізму, включаючи формалізацію процесів обчислення матриць запитів, ключів і значень, буде розглянуто у розділі 2.

Завдяки самоувазі модель здатна визначати, які частини послідовності є найбільш інформативними для класифікації жесту, незалежно від їхнього положення у часі. Це є критично важливим для коректного розпізнавання складних жестів із великою кількістю проміжних станів.

1.5.1 Vision Transformer (ViT) для статичних жестів

ViT-моделі стали першими трансформерними архітектурами, що досягли високої ефективності у задачах комп'ютерного зору. Основна ідея полягає у поділі зображення на фрагменти (patches) та поданні їх як послідовності векторів, які обробляються аналогічно токенам у мовних моделях. Це дозволило відмовитися від згорткових фільтрів і розглядати зображення у глобальному контексті [17].

Однак у задачі розпізнавання жестів ViT найчастіше застосовуються у поєднанні з іншими методами, наприклад структурними ознаками MediaPipe. Це зменшує обсяг вхідних даних і дозволяє моделі фокусуватися не на текстурі зображення, а на конфігурації пальців.

1.5.2 Мультимасштабні відеотрансформери та MVTN

Для динамічних жестів було розроблено низку спеціалізованих відеотрансформерів, які здатні враховувати часову структуру послідовності. Однією з таких моделей є MVTN (Multiscale Video Transformer Network), де обробка відбувається одночасно на кількох часових масштабах – коротких,

середніх і довгих. Це дозволяє моделі виділяти як мікрорухи, так і глобальні зміни положення руки [18].

Така рекурсивна обробка дозволяє накопичувати контекст упродовж усієї динаміки жесту.

1.5.3 Трансформери для скелетних послідовностей

Окремий клас моделей працює не з відео, а безпосередньо зі скелетними координатами, отриманими з MediaPipe. Це значно зменшує розмірність даних і робить модель менш чутливою до фону та освітлення. Трансформер, що працює з координатами landmarks, розглядає кожен кадр як токен, а всю послідовність руху – як набір взаємопов'язаних структурних станів [19].

Для кожного кадра формується вектор ознак, який надходить до трансформера для глобального аналізу.

Саме цей підхід лежить в основі моєї реалізації Transformer Hand Gesture Recognition, де модель навчається одночасно на статичних та динамічних жестах. Завдяки механізму самоуваги вона здатна виділяти найбільш інформативні моменти руху та ігнорувати шумові фрагменти.

1.6 Методи відстеження рухів руки у відеопотоці

Відстеження рухів руки є ключовим етапом у побудові систем жестової взаємодії, оскільки саме на цьому рівні формується динамічний опис жесту. На відміну від задачі класифікації, де модель працює з окремими станами руки або зі структурним вектором, задача трекінгу передбачає безперервне визначення положення кисті в часі. Сучасні системи відстеження працюють у режимі реального часу й поєднують у собі методи детекції, фільтрації та прогнозування траєкторії [20].

Перші підходи до відстеження базувалися на методах оптичного потоку. Вони передбачали аналіз зміни інтенсивності пікселів між послідовними кадрами. Класичне формулювання оптичного потоку задається співвідношенням:

$$I_x * u + I_y * v + I_t = 0, \tag{1.3}$$

де I_x, I_y – просторові похідні;

I_t – часова похідна;

u, v – компоненти швидкості руху.

Хоча цей підхід і дозволяв відстежувати напрям та швидкість руху, він виявився нестабільним при наявності шумів, швидких змін освітлення або складної фонової динаміки. Тому з часом оптичний потік почали використовувати лише як допоміжний елемент або компонент гібридних систем.

1.6.1 Відстеження на основі ключових точок

Сучасні системи жестової взаємодії, у тому числі MediaPipe Hands, використовують модель виявлення та прогнозування ключових точок руки. Такий підхід дозволяє отримати структурне представлення кожного кадру, що значно підвищує точність трекінгу. Виявлені точки зап’ястя та пальців використовуються для оцінки позиціювання кисті, а зміни їх координат – для аналізу руху у часі [21].

Перевага цього підходу полягає у відокремленні суттєвих структурних характеристик від фонового зображення. Траєкторія руху кисті описується як упорядкований у часі масив векторів, де кожен елемент відповідає набору координат ключових точок (landmarks) у конкретному кадрі відеопотоку.

Таке представлення створює основу для подальшої обробки трансформерами, рекурентними моделями або CNN, дозволяючи використовувати інформацію про рух незалежно від фонових факторів.

1.6.2 Фільтрація та згладжування траєкторій

Оскільки визначення ключових точок завжди містить певний рівень шуму, у задачах трекінгу широко використовуються фільтри згладжування. Найпоширенішими є експоненційне згладжування й фільтр Калмана. Останній моделює процес руху як систему станів:

$$\begin{cases} x_{t+1} = Ax_t + Bu_t + w_t \\ z_t = Hx_t + v_t \end{cases}, \tag{1.4}$$

де x_t – прихований стан (позиція, швидкість);

z_t – спостережувані координати landmarks;

w_t, v_t – випадкові шуми.

Можемо зробити висновок – фільтр Калмана є ефективним інструментом для згладжування коливань, що виникають під час детекції руки, забезпечуючи плавність руху та стабільність візуалізації.

1.6.3 Гібридні системи трекінгу

На практиці найкращі результати показують гібридні системи, що поєднують детекцію руки, трекінг ключових точок та обробку часових залежностей. Медичні, індустріальні та VR-рішення використовують саме такі моделі, оскільки вони забезпечують високу точність у реальному часі. Додаткові модулі можуть виконувати передбачення наступного стану кисті, що дозволяє

компенсувати затримки обробки кадрів і робити жестову взаємодію плавною [22].

1.7 Постановка задачі дослідження

Таким чином, розпізнавання та відстеження рухів руки за допомогою нейронних мереж є актуальною науково-практичною задачею, оскільки такі системи знаходять широке застосування у людино-машинній взаємодії, робототехніці, доповненій реальності та інтерфейсах без дотику. Сучасні дослідження показують, що поєднання структурних описів руки (landmarks) та трансформерних архітектур дозволяє значно підвищити якість класифікації як статичних, так і динамічних жестів, забезпечуючи роботу в режимі реального часу [23].

Прийнято рішення щодо розроблення програмного застосунку, який реалізовуватиме метод розпізнавання жестів на основі скелетних координат та трансформерної моделі. Такий застосунок має продемонструвати можливість ефективного аналізу часових послідовностей руху руки, визначення типу жесту та коректної інтерпретації траєкторій у реальних умовах. Крім того, система повинна забезпечити збір даних, навчання моделі, тестування й оцінювання її точності, що дозволить зробити висновки щодо доцільності використання трансформерів у жестових інтерфейсах [24].

Об’єктом дослідження є процес розпізнавання та відстеження рухів рук людини у відеопотоці в реальному часі.

Предметом дослідження є методи аналізу координатних структур руки та моделювання часових залежностей за допомогою трансформерних нейронних мереж.

Метою дослідження є підвищення точності розпізнавання жестів шляхом розроблення гібридного підходу, що поєднує визначення ключових точок за

допомогою MediaPipe, синтетичні та реальні дані, а також трансформерну архітектуру HandTransformer.

Для досягнення мети необхідно вирішити такі завдання:

– здійснити огляд сучасних наукових джерел щодо методів класифікації жестів руки, моделей трекінгу та трансформерних архітектур;

– дослідити принципи роботи MediaPipe Hands та визначити переваги використання скелетних координат у порівнянні з аналізом зображень;

– сформувати архітектуру системи розпізнавання жестів на основі трансформера та визначити формат представлення вхідних структурних ознак;

– розробити алгоритм зчитування, нормалізації та підготовки часових послідовностей для навчання нейронної мережі;

– реалізувати програмний застосунок, що забезпечує збір даних, навчання, тестування та використання моделі в реальному часі;

– провести експериментальне оцінювання точності класифікації статичних і динамічних жестів, проаналізувати результати та сформувати висновки щодо ефективності запропонованого методу.

2 ТЕОРЕТИЧНІ ОСНОВИ МОДЕЛЮВАННЯ РУХІВ РУК ЗА ДОПОМОГОЮ НЕЙРОННИХ МЕРЕЖ

2.1 Концепція трансформерних моделей у задачах розпізнавання жестів руки

Трансформерні моделі становлять сучасний підхід до обробки послідовностей, що забезпечує можливість моделювання складних залежностей між елементами вхідних даних без необхідності їх послідовного проходження. На відміну від рекурентних моделей, таких як LSTM або GRU, трансформер виконує аналіз усієї послідовності паралельно, що суттєво підвищує ефективність опрацювання довгих та структурно насичених сигналів.

У задачі розпізнавання жестів руки трансформерна архітектура виявляється особливо ефективною, оскільки жест розглядається як сукупність просторових конфігурацій та їх змін у часі. Кожний кадр подається у вигляді структурного опису руки – набору координат 21 ключової точки, що формують скелетну модель. Ці дані утворюють часову послідовність, яка зберігає динаміку руху, взаємозв’язки між пальцями та загальну форму траєкторії.

Трансформер дозволяє аналізувати дані не лише локально, а й глобально, встановлюючи важливість кожного кадру відносно інших. Це робить модель стійкою до варіацій швидкості жестів, незначних шумів, зміни масштабу та індивідуальних відмінностей у виконанні рухів. Крім того, архітектура легко адаптується до одночасного аналізу статичних поз і динамічних жестів, що є ключовою перевагою для побудови універсальної системи реального часу.

Основою функціонування є механізм самоуваги (Self-Attention), який дозволяє моделі динамічно визначати пріоритетність кадрів, фокусуючись на найбільш інформативних фазах руху. Оскільки обробка відбувається паралельно, для збереження інформації про порядок дій застосовується позиційне кодування (Positinal Encoding).

2.1.1 Принцип роботи трансформерної системи розпізнавання жестів

Функціонування систем розпізнавання жестів на основі трансформерних моделей ґрунтується на послідовній обробці структурних ознак руки та аналізі часових залежностей між ними. Узагальнений принцип роботи такої системи наведено на рисунку 2.1 [25], що демонструє типову взаємодію між модулем виявлення ключових точок та трансформерним класифікатором.

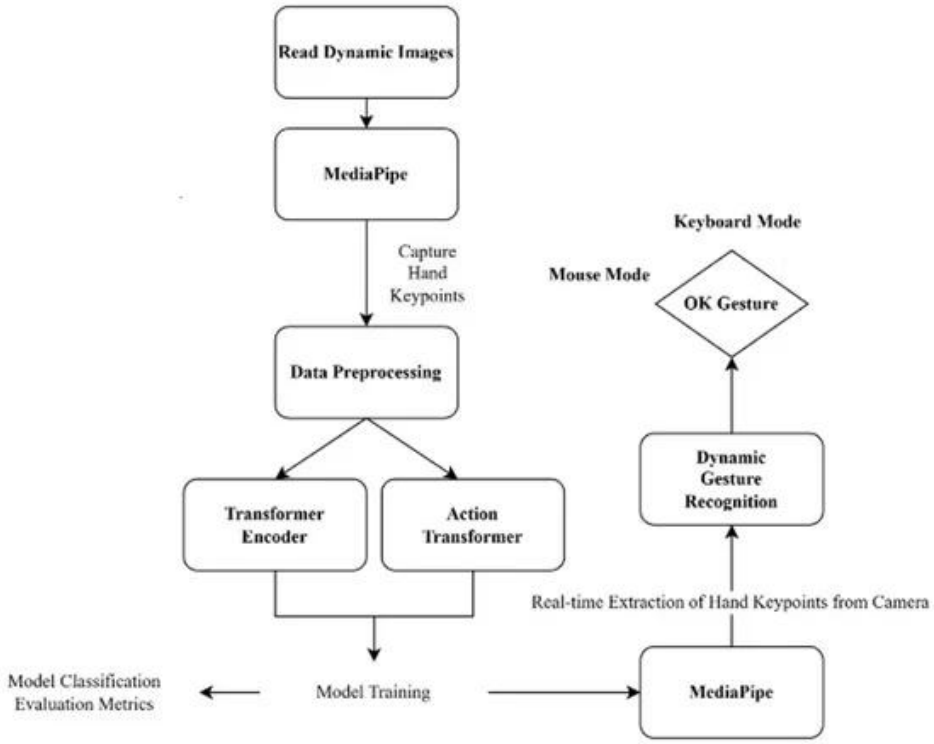

Рисунок 2.1 – Узагальнений принцип роботи системи розпізнавання жестів на основі трансформера

Основні етапи роботи трансформерної моделі включають:

Етап 1. Отримання структурних даних. За допомогою MediaPipe Hands визначаються 21 ключова точка кисті, що формують базове скелетне представлення жесту у вигляді координат (x, y) для кожного кадру відеопотоку.

Етап 2. Формування та нормалізація послідовності ознак: Отримані координати призводять до єдиного масштабу, центруються відносно зап’ястя та об’єднуються у часову послідовність векторів.

Етап 3. Додавання позиційної інформації. До кожного вектора ознак додається позиційне кодування, яке забезпечує врахування порядку кадрів у послідовності, що має найбільшу цінність для моделювання динаміки руху.

Етап 4. Обробка послідовності механізмом самоуваги. Трансформер аналізує взаємозв’язки між усіма кадрами, визначаючи, які елементи послідовності найбільше впливають на поточний стан. Математична модель цього процесу (Self-Attention), яка дозволяє адаптивно виділяти найбільш інформативні фази руху, детально розглянута у підрозділі 2.4.

Етап 5. Класифікація жесту. Результуюче приховане представлення передається до вихідного шару, який визначає клас жесту – статичного або динамічного – на основі глобального аналізу послідовності.

Отже, трансформерна архітектура дозволяє інтегрувати просторові та часові характеристики руху, забезпечуючи високу точність розпізнавання навіть у складних умовах відеозахоплення.

2.1.2 Переваги та обмеження

Трансформерні архітектури стали одним із найефективніших інструментів для аналізу послідовностей у комп’ютерному зорі, зокрема для розпізнавання рухів руки. На відміну від згорткових або рекурентних моделей, трансформери поєднують здатність працювати з глобальним контекстом та ефективно обробляти структурні ознаки, що робить їх придатними як для статичних, так і для динамічних жестів. Сучасні дослідження підтверджують зростання ролі відеотрансформерів та моделей для скелетних даних у задачах HCI (Human-Computer Interaction) та систем реального часу [26].

До основних переваг можемо віднести здатність моделювати довготривалі часові залежності: механізм самоуваги має глобальне рецептивне поле, що дозволяє враховувати взаємозв'язки між усіма кадрами послідовності одночасно.

Це вирішує проблему «згасання градієнта», притаманну рекурентним мережам (LSTM), де інформація про початкові кадри втрачається на довгих дистанціях.

Також вагомою перевагою є інваріантність до положення та масштабу. Оскільки модель працює зі структурними координатами landmarks, а не зображеннями, вона менше залежить від зовнішніх умов – освітлення, фону, ракурсу камери. Це підвищує стабільність роботи у реальних сценаріях та прискорює тренування.

Висока точність у складних динамічних жестах є ще однією перевагою. Трансформер здатний виділяти ключові моменти руху, навіть якщо жест містить мікрорухи або варіативні траєкторії. Такий підхід особливо ефективний у жестах типу Swipe або Zoom, де важлива не структура кисті, а зміна її положення в часі.

Можливість паралельної обробки послідовностей також підвищує ефективність. Завдяки відсутності рекурсії, усі елементи послідовності можуть оброблятися паралельно. Це зменшує час навчання та покращує масштабованість моделі;

Універсальність представлення даних дозволяє трансформеру працювати з різними форматами ознак – координатами landmarks, векторизованими дескрипторами руху або ознаками, отриманими з CNN. Це дозволяє інтегрувати модель у гібридні архітектури.

Незважаючи на велику кількість перелічених переваг , варто знати і про обмеження, які слід враховувати при їх застосуванні. Основним недоліком є високі обчислювальні витрати: Механізм самоуваги масштабується квадратично від довжини послідовності, що ускладнює роботу з довгими жестами або високочастотними відео. Для вирішення цієї проблеми використовуються оптимізовані версії механізму самоуваги або скорочення довжини послідовності.

Важливо врахувати потребу у великих навчальних даних. Трансформери ефективно навчаються за умов наявності значної кількості даних. У задачах жестової взаємодії це може бути складним через обмеженість реальних записів та значну варіативність рухів користувачів.

Чутливість до шуму в координатах landmarks є ще одним обмеженням. Оскільки вхідними даними є координати точок, будь-які помилки детектора (MediaPipe) безпосередньо впливають на векторне представлення жесту. Це вимагає впровадження додаткових етапів фільтрації та згладжування траєкторій (наприклад, фільтром Калмана).

Необхідність тонкого налаштування гіперпараметрів також ускладнює процес. Кількість голов у механізмі самоуваги, розмір прихованих векторів, глибина мережі та параметри регуляризації суттєво впливають на якість моделі, що збільшує складність процесу оптимізації.

Таким чином, трансформерні архітектури демонструють значний потенціал у задачах розпізнавання статичних і динамічних жестів руки, забезпечуючи високу точність, стійкість до варіативності рухів та ефективність моделювання часових залежностей. Незважаючи на окреслені обмеження, більшість з них можуть бути компенсовані за рахунок оптимізації механізму уваги, нормалізації скелетних координат та використання спеціалізованих датасетів жестів. Це робить трансформери доцільним вибором для побудови систем реального часу, які працюють зі структурними ознаками руки. У наступному підрозділі буде розглянуто архітектури відео- та скелетних трансформерів, що становлять теоретичну основу для подальшої розробки моделі, застосованої в межах цього дослідження [26].

2.1.3 Приклади використання трансформерних моделей у сучасних системах розпізнавання жестів

Трансформерні архітектури активно впроваджуються у сучасні системи комп’ютерного зору та людино-машинної взаємодії, зокрема в задачах розпізнавання жестів руки. Завдяки універсальності механізму самоуваги та здатності моделювати глобальні часові залежності, трансформери стали основою

широкого спектра прикладних рішень у промисловості, медицині, робототехніці та AR/VR-технологіях [27].

Одним із напрямів застосування є відеотрансформери, такі як TimeSformer, MViT, ViViT, які використовуються у високоточних системах розпізнавання рухів. Їхня перевага полягає у можливості одночасної обробки просторових та часових характеристик відео. Подібні моделі застосовуються у спортивній аналітиці, безконтактних інтерфейсах та мультимедійних сервісах, де важливо виявляти складні рухи з високою точністю.

Окремий клас представляють трансформери для скелетних даних – ST-TR, STAR-Transformer, Spatial–Temporal Transformer Networks. Вони працюють не з відеопікселями, а з набором структурних точок (landmarks), що робить їх менш чутливими до шума, освітлення та варіацій фону. Такі моделі активно застосовуються у робототехніці, системах жестового управління дронами та у медичних застосунках для оцінки моторики пацієнтів.

Важливою тенденцією останніх років є поєднання детекторів типу MediaPipe Hands із трансформерними класифікаторами. Наприклад, у роботах 2023–2025 років продемонстровано, що інтеграція 21-елементного скелетного опису руки та трансформерної моделі забезпечує швидку та стабільну роботу систем у реальному часі без потреби в дорогому обладнанні. Одним із характерних прикладів є дослідження Li & Hsieh (2025), де трансформерна архітектура успішно використана для класифікації жестів у потоковому відео, що підтверджує ефективність підходу у прикладних HCI-сценаріях [28].

Також трансформери активно впроваджуються у AR/VR-системи, де жести виступають основним способом управління. Компанії Meta, Microsoft та HTC інтегрували моделі аналізу рухів руки на основі attention-механізмів у свої XR-рішення, що дозволило суттєво підвищити точність взаємодії та стабільність в умовах змінного освітлення або часткових оклюзій.

У промислових системах трансформери використовуються для контролю роботизованих маніпуляторів, де точне відстеження та інтерпретація рухів оператора забезпечує можливість безпечної та інтуїтивної взаємодії між

людиною і машиною. Завдяки здатності адаптуватися до індивідуальних особливостей користувачів, такі моделі дозволяють створювати персоналізовані інтерфейси, що значно підвищує ефективність роботи.

Отже, застосування трансформерів у сучасних системах розпізнавання жестів охоплює широкий спектр галузей – від біометричних систем та медичних застосунків до віртуальних середовищ та індустріальної автоматизації. Зростання кількості практичних рішень підтверджує доцільність використання трансформерних архітектур у задачі, де необхідний аналіз складних структурних залежностей та часової динаміки рухів.

2.2 Скелетне представлення руки та формування ознак

Розпізнавання жестів руки у сучасних системах ґрунтується на переході від аналізу растрових зображень до обробки структурних скелетних моделей. Такий підхід дозволяє здійснити суттєве зменшення розмірності вхідних даних (dimensionality reduction), відфільтровуючи надлишкову інформацію (колір шкіри, текстуру фону, умови освітлення) та залишаючи лише топологічні характеристики об’єкта.

З математичної точки зору, скелетна модель руки H у момент часу t описується як упорядкована множина N ключових точок (landmarks):

$$H_t = \{p_1, p_2, \dots, p_N\}, \tag{2.1}$$

де $p_i = (x_i, y_i, z_i)$ – нормалізовані координати i-ї точки у тривимірному просторі (або (x_i, y_i) у двовимірному).

У більшості сучасних систем скелетне представлення включає 21 точку, що описують положення зап’ястя, суглобів пальців та кінчиків. Таке представлення формується у вигляді набору координат, які можуть бути подані як одномірний вектор або як структурована послідовність. Перевагою цього

підходу є можливість аналізувати як статичну позу, так і динамічну зміну конфігурації руки в часі, що особливо важливо для класифікації складних жестів.

Оскільки саме просторові зв’язки між точками визначають форму жесту, скелетні моделі широко застосовуються у завданнях HCI, робототехніці, доповненій реальності та медичних системах, де потрібне точне відстеження рухів та високий рівень інтерпретованості сигналу. У дослідженнях останніх років було показано, що використання скелетних ознак підвищує точність класифікації порівняно з методами, що працюють безпосередньо з відеозображенням [14, 29].

2.2.1 MediaPipe Hands і визначення landmark-точок

MediaPipe Hands є однією з найбільш поширених систем для визначення ключових точок кисті у відеопотоці, оскільки поєднує високу точність, швидкість роботи та можливість виконання на звичайних споживчих пристроях у реальному часі. Метод ґрунтується на дворівневій архітектурі: спочатку нейронна мережа виконує детекцію області кисті, а потім окрема модель прогнозує положення 21 ключової точки на основі локалізованого фрагмента зображення [17].

На відміну від моделей повної сегментації чи аналізу контурів, MediaPipe прогнозує геометричну структуру руки, не спираючись на колір шкіри чи контраст фону. Це забезпечує стабільність навіть у випадках часткових оклюзій, швидких рухів або змін освітлення. Визначені точки відповідають реальним анатомічним сегментам кисті – зап’ястю, суглобам та кінчикам пальців – що дозволяє отримувати структурні ознаки, інваріантні до фона та текстурних властивостей зображення.

Кожна landmark-точка описується координатами *(x, y)* або *(x, y, z)*, якщо використовуються тривимірні дані, нормалізованими відносно розмірів кадру. Такий формат подання спрощує подальшу обробку та дозволяє ефективно

застосовувати моделі, орієнтовані на аналіз структур, зокрема трансформерні архітектури. У контексті класифікації жестів це означає, що система може оперувати не самим зображенням, а його компактним числовим представленням, у якому збережено просторові зв’язки між пальцями й форму кисті [19].

Оскільки заданий набір із 21 точки є універсальним стандартом у жестовій аналітиці, він забезпечує можливість формувати послідовності станів руки та будувати моделі, які враховують динаміку руху. Завдяки цьому скелетні ознаки стають ключовим інструментом для побудови трансформерних систем розпізнавання жестів, оскільки точно передають як статичні пози, так і динамічні траєкторії в часі.

Положення точок та їх взаємозв’язки наведено на рисунку 2.2.

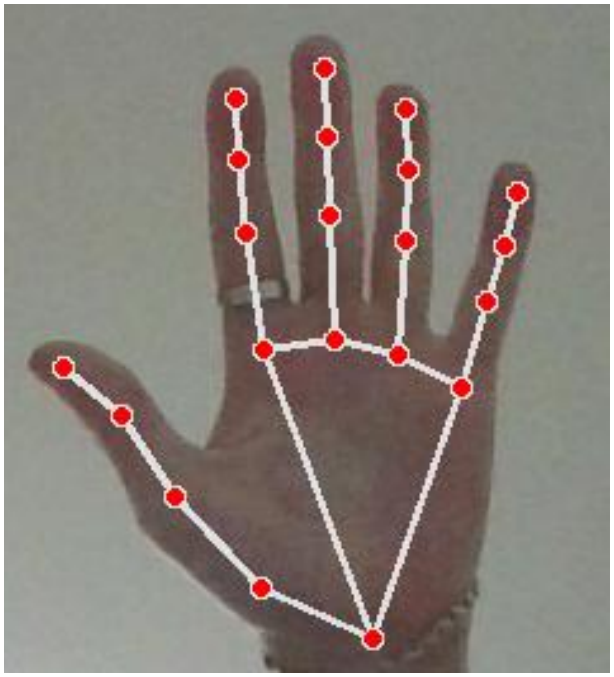

Рисунок 2.2 - Скелетне представлення кисті з 21 landmark-точкою

2.2.2 Формування структурних векторів жесту

Після визначення координат ключових точок наступним кроком є їх перетворення у формат, придатний для обробки нейронною мережею.

Координати, отримані безпосередньо з детектора, залежать від абсолютного положення руки в кадрі. Це створює проблему варіативності: той самий жест, виконаний у центрі кадру та в його кутку, матиме різні числові представлення.

Для вирішення цієї проблеми застосовується процедура центрування відносно базової точки. За базову точку обрано зап’ястя (індекс 0), оскільки воно є кореневим вузлом кінематичного ланцюга кисті.

Нехай $p_{t,i}$ – вихідні координати *i*-ї точки на кадрі *t*. Тоді нормалізовані відносні координати $\acute{p}_{t,i}$ обчислюється за формулою:

$$p'_{t,i} = p_{t,i} - p_{t,0}, \tag{2.2}$$

де $p'_{t,i}$ – відносні координати точки;

$p_{t,i}$ – абсолютні координати точки;

$p_{t,0}$ – координати зап’ястя.

Результатом операції є координати зап’ястя завжди стають рівними (0,0), що можна вважати переносом початку локальної системи координат до центру руки, забезпечуючи інваріантність до зсуву.

Наступним етапом є векторизація (flattening) даних. Для подачі на вхід нейронної мережі двовимірна структура точок розгортається в одновимірний вектор ознак *V*. Враховуючи, що використовуються лише *x* та *y* координати для 21 точки, розмірність вектора складає $D = 21 \times 2 = 42$

Математично вектор ознак для одного кадру записується як конкатенація пар координат:

$$V_t = [x'_o, y'_0, x'_1, y'_1, \dots, x'_{20}, y'_{20}], \tag{2.3}$$

де V_t – вектор ознак жесту в момент часу *t*;

$x'_i; y'_i$ - нормалізовані координати *i*-ї ключової точки;

T – операція транспонування.

Отриманий вектор $V_t \in R^{42}$ є компактним числовим дескриптором статичної пози. Для аналізу динаміки руху ці вектори надалі об'єднуються в часову послідовність, яка слугує вхідним тензором для трансформерної моделі [30].

2.3 Формування та попередня обробка часових послідовностей

Через те, що динамічні жести визначаються не лише статичною положенням кисті, а й характером зміни координат у часі, критично важливим етапом є формування вхідного тензора для нейронної мережі. Вхідні дані для трансформерної моделі подаються у вигляді матриці фіксованої розмірності, що вимагає уніфікації тривалості всіх відеофрагментів.

2.3.1 Формалізація динамічних послідовностей

Первинним етапом обробки є представлення відеопотоку у вигляді математичної структури, придатної для аналізу. У реальних умовах виконання жестів користувачем, можемо описати значною часовою варіативністю: той самий жест (наприклад, «Swipe») може тривати від частки секунди до кількох секунд.

Припустимо, що на вхід подається послідовність кадрів, отримана з відеопотоку, позначається як *V*. Після проходження етапу детекції та векторизації (описаного у п. 2.2.2), кожен кадр перетворюється на вектор ознак $s_k \in R^{42}$. Отримаємо, що «сирий » жест можна формалізувати як упорядковану в часі множину векторів:

$$S_{raw} = \{s_1, s_{2,} \dots, s_K\}, \tag{2.4}$$

де s_k – вектор ознак k-го кадру, що містить нормалізовані координати ключових точок;

K – загальна кількість кадрів у записаному жесті, що є випадковою величиною і залежить від швидкості руху користувача.

Така варіативність параметра K створює проблему для нейронних мереж фіксованої архітектури, які очікують на вхід тензор строго визначеної розмірності.Через що виникає необхідність приведення всіх послідовностей до єдиного часового стандарту.

2.3.2 Математична модель часової інтерполяції

Для уніфікації вхідних даних у дослідженні обрано цільову довжину послідовності T = 30 кадрів. Так як просте відкидання зайвих кадрів або дублювання існуючих може призвести до втрати важливої інформації про динаміку руху, було застосовано метод лінійної інтерполяції.

Концепція методу полягає у проекції часової шкали вихідного жесту довільної довжини на нову, фіксовану шкалу. Спочатку для кожного моменту часу t у новій послідовності розраховується відповідна позиція τ у вихідному масиві даних:

$$\tau = t * \frac{K-1}{T-1}, \tag{2.5}$$

де t – цілочисельний індекс кадру в новій послідовності $(0 \dots T-1)$;

τ – розрахунковий індекс у вихідній послідовності;

K – реальна кількість кадрів у записаному жесті.

Беручи до уваги, що індекс τ зазвичай є дробовим числом, значення вектора ознак у цій точці синтезується на основі двох найближчих реальних кадрів , а саме попереднього та наступного.

Математичне представлення виглядає як сумма векторів:

$$s'_t = (1 - \alpha) * s_{\lfloor\tau\rfloor} + \alpha * s_{\lceil\tau\rceil}, \quad (2.6)$$

де $\lfloor\tau\rfloor$ – індекс лівого сусіднього кадру;

$\lceil\tau\rceil$ – індекс правого сусіднього кадру;

α – ваговий коефіцієнт інтерполяції.

Використання даного підходу забезпечує плавність траєкторії руху навіть при значному стисненні або розтягуванні часового ряду, дозволяючи зберегти кінематичні особливості жесту без створення артефактів дискретизації [31].

2.3.3 Аугментація даних для підвищення стійкості моделі

Серед відомих фактів – трансформерні архітектури схильні до перенавчання на малих наборах даних через слабке індуктивне зміщення. Щоб наблизити умови навчання до реальних, у системі реалізовано алгоритм стохастичної аугментації, який одночасно моделює тремтіння сенсора камери та зміну відстані до руки.

Процес генерації аугментованого зразка s_{aug} описується як лінійна трансформація вихідного тензора s з додаванням стохастичного шуму:

$$s_{aug} = (s - c) * \lambda + c + \epsilon, \quad (2.7)$$

де c – геометричний центр системи координат, відносно якого виконується масштабування;

λ – коефіцієнт масштабування, що обирається випадково з діапазону [0.9,1.1];

ϵ – вектор випадкового гаусівського шуму.

Використання комплексного перетворення дозволяє суттєво розширити навчальну вибірку, не порушуючи топологічну структуру жесту, що є критично важливим для стабільного навчання трансформера [32].

2.4 Математична модель та архітектура HandTransformer

Запропонована система базується на архітектурі трансформерного енкодера (Transform Encoder), який адаптовано для обробки часових рядів координат. На вхід моделі подається тривимірний тензор вхідних даних X, структура якого визначається виразом:

$$X \in R^{B \times T \times D}, \tag{2.8}$$

де B – розмір батчу;

T – фіксована довжина часової послідовності;

D – розмірність вектора ознак для одного кадру.

2.4.1 Позиційне кодування та механізм уваги

Оскільки базовий механізм уваги обробляє всі кадри паралельно і не має вбудованої рекурентності (інформації про порядок слідування кадрів), обрана архітектура включає модуль позиційного кодування (Positional Encoding).

Математично це реалізується шляхом додавання до вхідного вектора ознак спеціального вектора позиції. Значення цього вектора генеруються за допомогою гармонічних функцій (синуса та косинуса) різних частот. Такий підхід дозволяє моделі навчатися відносним відстаням між кадрами, розрізняючи початок, середину та кінець жесту [25].

Основним компонентом моделі є механізм багатоканальної самоуваги(Multi-Head Self-Attention). Принцип цього механізму полягає у перетворенні вхідної послідовності на три матриці: Запити (Q), Ключі (K) та значення (V).

Алгоритм обчислює скалярний добуток між запитами та ключами, що визначає ступінь важливості кожного кадру відносно інших. Отримані ваги нормалізуються функцією Softmax:

$$Attention(Q,K,V) = softmax\left(\frac{QK^T}{\sqrt{d_k}}\right)V, \quad (2.9)$$

де d_k – розрахунковий коефіцієнт масштабування;

Q, K, V – матриці запитів, ключів та значень.

Зазначенний механізм дає можливість мережі динамічно фокусуватися на найбільш інформативних фазах руху, ігноруючи статичні або шумні фрагменти, що є ключовою перевагою над класичними RNN-моделями.

Архітектуру мережі наведено на рисунку 2.3.

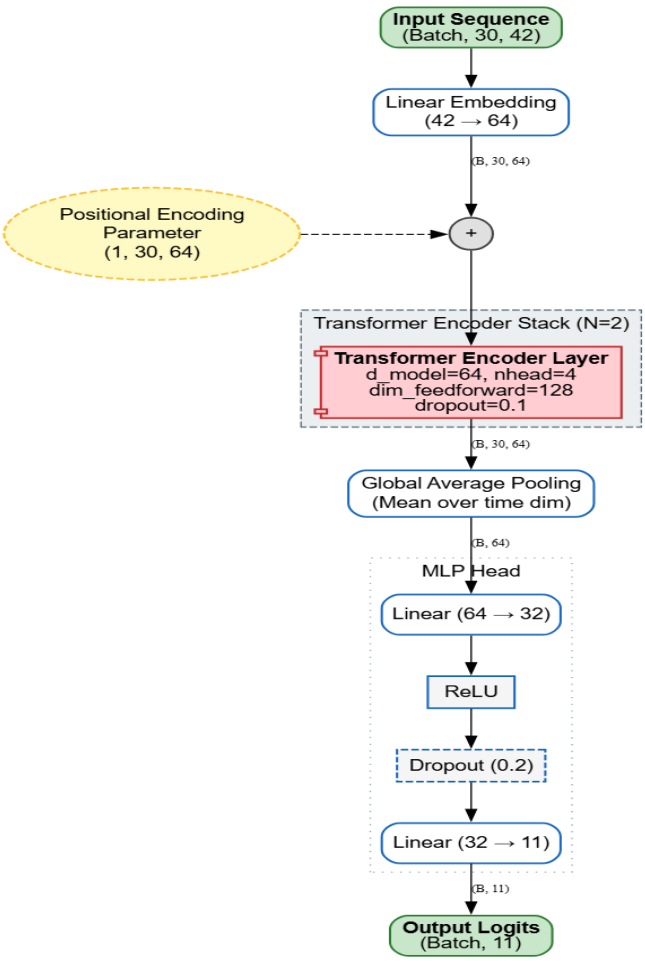

Рисунок 2.3 – Архітектура нейронної мережі HandTransformer

2.4.2 Класифікаційний модуль та нормалізація

Вихідні дані блоку уваги проходять через шари нормалізації (Layer Normalization) та повнозв'язну мережу прямого поширення (Feed-Forward Network). Важливим елементом архітектури є застосування механізму Layer Normalization перед кожною субактивністю. Це забезпечує стабілізацію градієнтів під час навчання глибокої мережі. Математично операція нормалізації для вектора *x* описується так:

$$LayerNorm(x) = \gamma \odot \frac{x-\mu}{\sqrt{\sigma^2+\epsilon}} + \beta, \quad (2.10)$$

де μ та σ^2 – середнє значення та дисперсія вхідного вектора;

γ та β – параметри масштабування та зміщення, що навчаються;

ϵ – мала константа для чисельної стабільності.

Кінцева класифікація виконується за допомогою лінійного шару (Linear Layer), який проєктує нормалізований прихований стан моделі на простір класів. Перетворення вихідних логітів у ймовірнісний розподіл здійснюється за допомогою функції Softmax:

$$P = (y = c|x) = \frac{e^{z_c}}{\sum_{j=1}^{C} e^{z_j}}, \quad (2.11)$$

де P – ймовірність належності жесту до класу c;

z_c – вихідне значення лінійного шару для класу c;

C – загальна кількість класів.

2.4.3 Оптимізація та функція витрат

Навчання моделі розглядається як задача мінімізації емпіричного ризику. Як цільову функцію обрано перехресну ентропію (Cross-Entropy Loss), яка є стандартом для задач багатокласової класифікації. Функція штрафує модель за розбіжність між передбаченим розподілом ймовірностей та істинною міткою класу:

$$L = -\frac{1}{N}\sum_{i=1}^{N}\sum_{c=1}^{C} y_{i,c}\log(\hat{y}_{i,c}), \quad (2.13)$$

де L – значення функції втрат;

N – розмір міні батчу;

$y_{i,c}$ – бінарник індекс істинного класу;

$\hat{y}_{i,c}$ – передбаченя моделлю ймовірність.

Для мінімізації цієї функції використовується адаптивний алгоритм оптимізації

2.5 Обґрунтування вибору технологічних засобів реалізації

Ефективність програмної реалізації математичної моделі значною мірою залежить від правильності підбору технологічного стеку. Врахувавши вимоги до обчислювальної ефективності алгоритмів комп'ютерного зору, було сформовано стек на базі мови Python та спеціалізованих бібліотек.

2.5.1 Мова програмування Python

В якості основного інструменту розробки було обрано мову програмування Python, оскільки вона є галузевим стандартом у сферах Machine Learning (ML) та Computer Vision (CV). Цей вибір зумовлений насамперед наявністю розвиненої

екосистеми спеціалізованих бібліотек (NumPy, PyTorch), які реалізують ресурсоємні математичні операції на низькорівневих мовах C/C++. Завдяки цій інтеграції, досягається висока продуктивність, необхідна для систем реального часу. Крім того, лаконічний синтаксис Python забезпечує простоту та швидкість прототипування, а його кросплатформеність дозволяє легко переносити розроблений код між різними операційними системами [33].

2.5.2 Фреймворк глибокого навчання PyTorch

Для реалізації нейронної мережі HandTransformer було обрано фреймворк PyTorch, який розробляється лабораторією Facebook AI Research. Цей вибір обумовлений низкою архітектурних особливостей, що роблять його стандартом для наукових досліджень у сфері обробки послідовностей.

Фундаментальною перевагою PyTorch є використання парадигми «Eager Execution». На відміну від статичних графів (які використовуються у старих версіях TensorFlow), де структура мережі фіксується до початку обчислень, PyTorch будує граф обчислень динамічно під час виконання коду. Це дозволяє використовувати стандартні конструкції мови Python (цикли for, умовні оператори if) безпосередньо в описі архітектури моделі. Для задачі розпізнавання жестів це критично важливо, оскільки дозволяє гнучко керувати потоком даних усередині механізму уваги та спрощує налагодження (debugging), даючи можливість перевіряти розмірності тензорів на будь-якому етапі проходження сигналу.

Ключовим обґрунтуванням вибору є висока ефективність роботи PyTorch на центральному процесорі (CPU). Більшість сучасних нейромереж для обробки зображень (наприклад, ResNet або EfficientNet) вимагають потужних GPU через величезну кількість операцій згортки. Однак, запропонована модель HandTransformer оперує не пікселями, а компактними векторними представленнями скелета (30×42 числа). Було встановлено, що для таких

розмірностей накладні витрати на пересилання даних у пам'ять відеокарти можуть перевищувати виграш від паралелізації. Внутрішня бібліотека ATen, на якій базується PyTorch, використовує оптимізовані інструкції процесора (AVX2/AVX-512) для матричних множень, що забезпечує інференс у реальному часі без використання дискретних графічних прискорювачів [34].

Реалізацію архітектури значно спрощує високорівневий API torch.nn, який надає готові, оптимізовані реалізації шарів TransformerEncoder та MultiheadAttention. Це дозволяє уникнути написання складної математики матричних множень вручну, знижуючи ймовірність помилок. Крім того, вбудований механізм автоматичного диференціювання Autograd автоматично будує граф похідних для всіх операцій, що дозволяє обчислювати градієнти для оновлення ваг моделі за один прохід, незалежно від складності архітектури.

2.5.3 Середовище MediaPipe Solutions

Для вирішення задачі вилучення скелетних ознак обрано кросплатформовий фреймворк MediaPipe від Google. Це рішення є унікальним на ринку завдяки своїй архітектурі, орієнтованій на мобільні та вбудовані системи (Edge AI).

MediaPipe будує процес обробки як спрямований граф, де вузлами є так звані «калькулятори» (C++ модулі), а ребрами – потоки даних. Це дозволяє розпаралелити процеси: поки один модуль виконує попередню обробку кадру (зміну розміру, нормалізацію кольору), інший вже виконує інференс нейромережі для попереднього кадру. Така асинхронність дозволяє досягати стабільної частоти кадрів (понад 30 FPS) на стандартних процесорах, що неможливо при послідовному виконанні коду Python.

Важливою деталлю, що забезпечує продуктивність, є використання двоетапного підходу. Спочатку працює детектор долоні, який знаходить руку на всьому зображенні. Він запускається лише тоді, коли руку втрачено або на

першому кадрі. Як тільки руку знайдено, система переходить у режим трекінгу, аналізуючи лише маленьку область інтересу (ROI) навколо руки з попереднього кадру. Цей підхід економить до 70% обчислювальних ресурсів, оскільки «важкий» детектор долоні не запускається на кожному кадрі.

Модель повертає 21 ключову точку в 3D-просторі (координати x, y, z). Була відзначена висока робастність моделі: вона навчена на синтетичних даних із різними умовами освітлення та перекриттями. Навіть коли користувач стискає руку в кулак або перекриває частину пальців іншою рукою, механізм внутрішньої регресії MediaPipe продовжує передбачати ймовірне положення прихованих суглобів, що забезпечує безперервність вхідних даних для трансформера [18].

2.5.4 Бібліотеки NumPy та OpenCV

Для забезпечення цілісності пайплайну обробки даних було застосовано зв'язку бібліотек NumPy та OpenCV. Бібліотека OpenCV виступає інтерфейсом між апаратним забезпеченням та програмним алгоритмом. Її роль у системі включає:

– захоплення потоку: клас VideoCapture забезпечує низькорівневий доступ до драйвера веб-камери, дозволяючи програмно керувати параметрами роздільної здатності та частоти кадрів;

– керування колірним простором: оскільки MediaPipe навчався на RGB-зображеннях, а стандартний вихід веб-камери – BGR, OpenCV виконує конвертацію просторів з мінімальними затримками;

– візуалізація: використання примітивів малювання (лінії, кола, текст) дозволяє накладати скелетну модель та результати класифікації безпосередньо на відеопотік у реальному часі, створюючи інтерфейс доповненої реальності.

Бібліотека NumPy є математичним ядром системи попередньої обробки. Вона використовується для зберігання даних у вигляді n-вимірних масивів, які

розміщуються в пам'яті суцільними блоками (contiguous memory). Це забезпечує ефективність кешування процесора.

Векторизовані операції NumPy (Broadcasting) дозволяють виконувати нормалізацію координат, центрування скелета відносно зап'ястя та аугментацію (додавання гауссового шуму) над усім масивом даних одночасно, без використання повільних циклів Python. Встановлено, що використання NumPy прискорює етап препроцесингу в 10–50 разів порівняно зі стандартними списками, що є критичним фактором для мінімізації загальної затримки (latency) системи [35].

Важливою особливістю є нативна інтеграція NumPy з PyTorch через механізм спільного використання пам’яті, що дозволяє передавати дані до нейромережі без копіювання. Операції зрізів (slicing) забезпечують ефективне формування ковзних вікон для часових послідовностей, створюючи лише представлення існуючих масивів замість фізичного копіювання даних. Також бібліотека забезпечує швидке обчислення статистичних характеристик послідовності для нормалізації даних, що виконується через оптимізовані бібліотеки без виходу в інтерпретатор Python. Зв’язка OpenCV та NumPy утворює ефективний програмний пайплайн від моменту захоплення кадру до передачі структурованих даних у модель глибокого навчання.

3 ПРОГРАМНА РЕАЛІЗАЦІЯ ТА ДОСЛІДЖЕННЯ СИСТЕМИ РОЗПІЗНАВАННЯ ЖЕСТІВ

3.1 Обґрунтування вибору середовища розробки та технічні характеристики стенду

Для реалізації системи розпізнавання жестів було обрано інтегроване середовище розробки PyCharm Professional Edition від компанії JetBrains як основний інструмент розробки. PyCharm забезпечує повнофункціональну підтримку мови програмування Python 3.9+, яка є стандартом у сфері машинного навчання та комп'ютерного зору.

Нижче наведено такі ключові переваги PyCharm, як інтелектуальна підтримка коду, інтеграція з науковими бібліотеками, використання вбудованого дебагера та профайлера, а також підтримка системи контролю версій Git.

Інтелектуальна підтримка коду забезпечує надання розширених можливостей автодоповнення коду з урахуванням типів даних, що є важливим при роботі з бібліотеками PyTorch та NumPy, де помилки типізації можуть призвести до складних для виявлення runtime помилок. Вбудований аналізатор коду виявляє потенційні проблеми ще на етапі написання [29].

Інтеграція з науковими бібліотеками у нативній підтримці Jupyter Notebooks дозволяє поєднувати інтерактивну розробку для експериментів з моделями та структуроване проєктування програмних модулів. Науковий режим роботи (Scientific Mode) забезпечує візуалізацію масивів NumPy та тензорів PyTorch безпосередньо в IDE [30].

Вбудований дебагер та профайлер дають можливість здійснювати покрокове виконання коду з відображенням стану змінних. Така функціональність є критичною при налаштуванні механізму уваги у трансформерній архітектурі. Профайлер допоміг виявити «вузькі місця » під час аугментації даних та оптимізувати їх шляхом векторизації операцій [31].

Підтримка Git у PyCharm дозволяє відстежувати зміни в архітектурі моделі, повертатися до попередніх версій при невдалих експериментах та синхронізувати код між робочою станцією та обчислювальним сервером [32].

Для експериментального дослідження зафіксовано апаратні та програмні характеристики обчислювального стенду. Особливістю даного дослідження є відмові від використання спеціалізованих графічних прискорювачів (GPU). Такий підхід дозволяє оцінити життєздатність системи на типових офісних або домашніх комп'ютерах.

Розробка та тестування проводились на мобільній робочій станції, характеристики наведено у таблиці 3.1.

Таблиця 3.1 – Технічні та програмні характеристики експериментального стенду

Характеристика	Опис / Значення
Центральний процесор (CPU)	AMD Ryzen 7 8845HS (8 ядер, 16 потоків, до 5.1 GHz)
Оперативна пам'ять (RAM)	16 GB DDR5 (5600 MHz)
Графічний адаптер	NVIDIA GeForce RTX 4060
Пристрій відеозахоплення	Integrated Camera (720p, 30 FPS, CMOS sensor)
Операційна система	Windows 11 (64-bit)
Мова програмування	Python 3.9.13
Фреймвок Deep Learning	PyTorch 2.0.1 (CPU-only mode)
Бібліотека Computer Vision	OpenCV-Python 4.8.0
Бібліотека скелетизації	MediaPipe 0.10.9
Математичні обчислення	NumPy 1.24.3

Незважаючи на наявну дискретну відеокарту RTX 4060, процес навчання нейронної мережі та її тестування свідомо виконувалися виключно на потужностях центрального процесора AMD Ryzen 7. Таке використання

підтвердило гіпотезу про те, що запропонована легконавантажена архітектура HandTransformer не потребує дороговартісного обладнання для розгортання та може бути донавчена на стороні клієнта за прийнятний час.

3.2 Програмна реалізація системи

Ефективність розробки, тестування та подальший розвиток програмного забезпечення залежить не лише від гарного коду, а й від правильно орагнізованої структури файлової системи проєкту. Для реалізації системи розпізнавання жестів було розроблено модульну архітектуру, яка забезпечує чітке розмежування логіки (source code), даних (assets) та результатів роботи (artifacts).

Структуру проєкту в середовищі PyCharm представлено на рисунку 3.1.

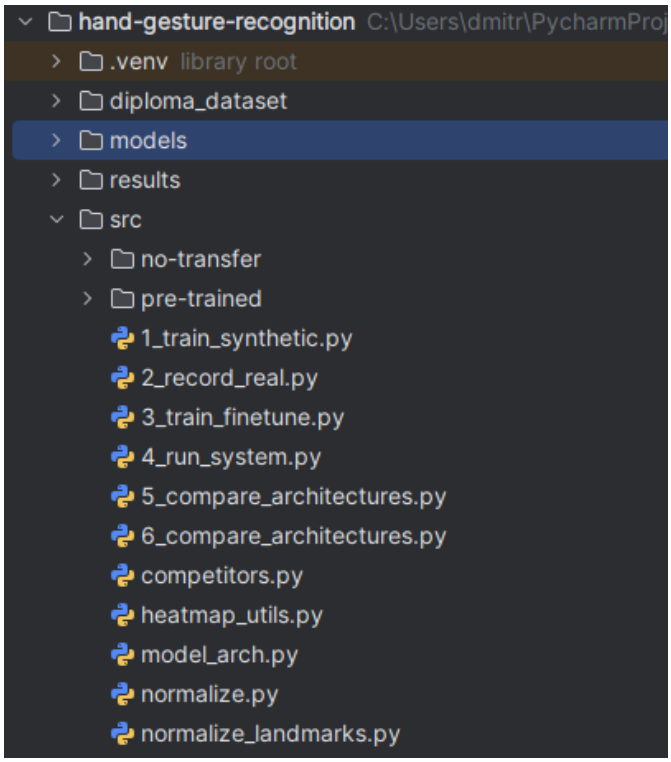

Рисунок 3.1 – структура файлової системи програмного комплексу

Рисунок 3.1 надає розуміння структури проєкту, коренева директорія містить наступні ключові елементи src/, models/, diploma_dataset/, results/, .venv/.

Перший ключовий елемент містить вихідний код допоміжних модулів, класів нейронної мережі та утиліт для обробки даних. Така інкапсуляція дозволяє імпортувати функції в основні скрипти без засмітнення корневої папки.

Другий елемент необхідний для зберігання серіалізованих станів нейронної мережі (файли ваг .pth). У цій директорій зберігаються як проміжні чекпоінти, так і фінальна оптимізована модель.

Третій елемент є основним сховище навчальних даних, структура якого організована за принципом класифікації папок (детальніше розглянуто нижче);

Четвертий елемент використовується для автоматичного збереження графіків навчання (Loss/Accuracy) та матриць помилок, що генеруються скриптами аналізу.

П’ятий елемент представляє ізольоване віртуальне середовище Python, що містить усі встановлені бібліотеки конкретних версій, що забезпечують відвторюваність експериментів на інших машинах.

У корені проєкту розташовані виконувані скрипти, пронумеровані згідно з логікою пайплайну, що спрощує навігацію та запуск відповідних етапів роботи системи.

3.2.1 Модуль генерації синтетичних даних

Модуль генерації синтетичних даних забезпечує створення великого обсягу навчальних прикладів для попереднього навчання моделі (pre-training). Основна ідея полягає у математичному моделюванні траєкторій руху для кожного типу жесту з додаванням випадкових варіацій для імітації природньої мінливості людських рухів.

Для кожного класу жестів реалізовано окрему функцію генерації. Наприклад, жест «Swipe left » моделюється як лінійний рух руки зліва направо з додаванням синусоїдальних коливань для імітації природної нестабільності траєкторії.

Приклад коду для цього жесту наведено у лістингу 3.1.

Лістинг 3.1 Реалізація синтетичного жесту «Swipe left»:

```
def generate_swipe_left(num_frames=30):
    sequence = []
    start_x, start_y = 0.7, 0.5
    end_x, end_y = 0.3, 0.5

    for i in range(num_frames):
        t = i / num_frames
        x = start_x + (end_x - start_x) * t
        y = start_y + 0.05 * np.sin(t * np.pi * 2)

        noise_x = np.random.normal(0, 0.01)
        noise_y = np.random.normal(0, 0.01)

        hand_landmarks = generate_hand_at_position(x + noise_x, y + noise_y)
        sequence.append(hand_landmarks)

    return np.array(sequence)
```

Функція *generate_hand_at_position* створює конфігурацію всіх 21 точки кисті руки у заданій позиції, враховуючи анатомічні пропорції та природні обмеження на кути між суглобами пальців, що забезпечує реалістичність згенерованих даних [18].

Для жестів масштабування (Zoom In/Out) моделюється зміна відстані між великим та вказівним пальцями. У проєкті за це відповідає функція *generate_zoom_in*. Код функції поданий у лістингу 3.2.

Лістинг 3.2 Реалізація функції для жестів масштабування:

```
def generate_zoom_in(num_frames=30):
    sequence = []
    start_dist = 0.05
    end_dist = 0.20

    for i in range(num_frames):
        t = i / num_frames
        distance = start_dist + (end_dist - start_dist) * t

        hand_landmarks = generate_pinch_gesture(distance)
        sequence.append(hand_landmarks)

    return np.array(sequence)
```

Модуль генерує 1200 прикладів для кожного з 11 класів жестів, що дає загалом 13200 синтетичних послідовностей. Ці дані використовуються виключно для pre-training моделі, після чого здійснюється fine-tuning на реальних записах користувача [26, 33].

3.2.2 Модуль запису реальних жестів

Модуль запису відповідає за збір реальних даних від користувача через веб-камеру. Цей модуль використовує бібліотеку MediaPipe Hands від Google для виявлення руки у відеопотоці та екстракції координат 21 ключової точки (landmarks) кисті в реальному часі [15].

Процес запису організований як інтерактивний сеанс, де користувач може обирати конкретний жест для записую. Система автоматично виявляє початок та завершення руху, записуючи координати landmarks на кожному кадрі, що описано у лістингу 3.3.

Лістинг 3.3 Запис послідовності жестів з веб-камери:

```
mp_hands = mp.solutions.hands
hands = mp_hands.Hands(
    static_image_mode=False,
    max_num_hands=1,
    min_detection_confidence=0.7)

cap = cv2.VideoCapture(0)
recording = []

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

    frame_rgb = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
    results = hands.process(frame_rgb)

    if results.multi_hand_landmarks:
        landmarks = results.multi_hand_landmarks[0]
        coords = []
        for lm in landmarks.landmark:
            coords.extend([lm.x, lm.y, lm.z])
        recording.append(coords)

    cv2.imshow('Recording', frame)
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break
```

np.save(f'gesture_{class_name}_{timestamp}.npy', recording)

Реалізований інтерфейс відображає на екрані скелет руки з підсвічуванням ключових точок для надання користувачу візуального зворотного зв’язку про якість детекції (рисунок 3.2). Система автоматично відкидає записи, де частина руки виявилася менше ніж на 80% кадрів, що гарантує високу якість даних.

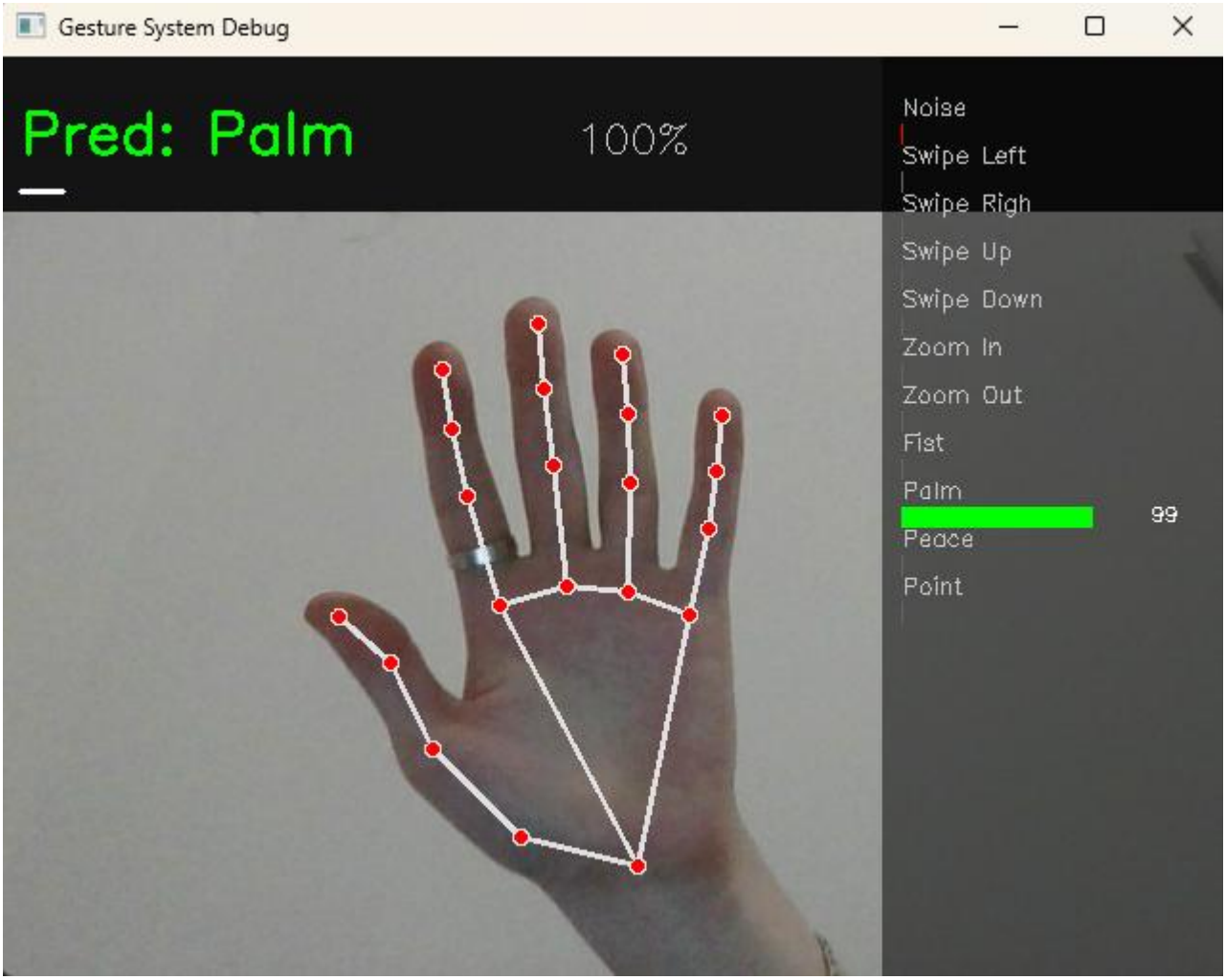

Рисунок 3.2 – Інтерфейс запису жестів з візуалізацією landmarks

Кожен запис зберігається як NumPy масив розміром (T,63), де T – довжина послідовності у кадрах, а 63 = 21 точка × 3 координати (x,y,z). Координати нормалізовані у діапазон (0,1] відносно розміру кадру.

Для забезпечення сумісності зі стандартними завантажувачами даних бібліотеки PyTorch, файлова структура набору даних була організована ієрархічно.

Кожен клас жестів (наприклад, «Fist», «Palm», «Swipe Left») має власну піддиректорію. Така структура дозволяє алгоритму автоматично визначати мітки класів (labels) на основі назв папок під час навчання.

Структуру даних наведено на рисунку 3.3.

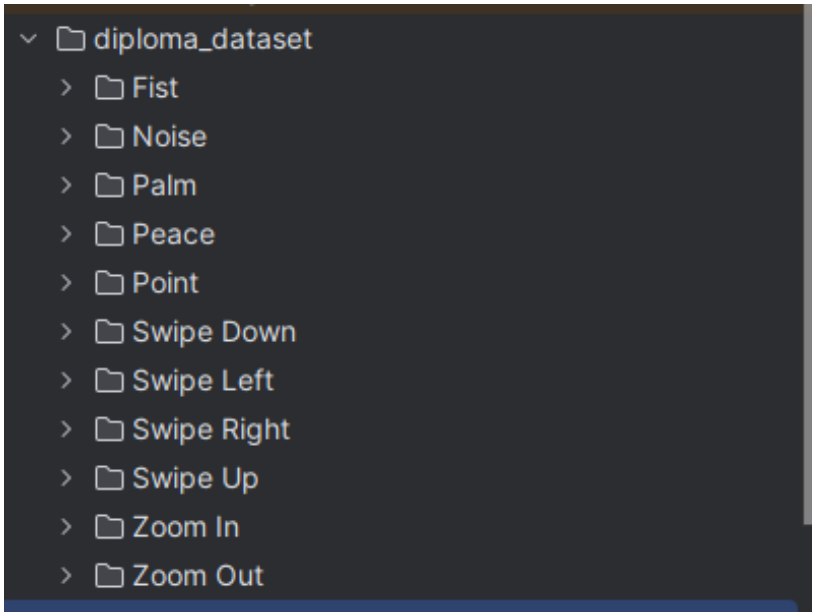

Рисунок 3.3 – Ієрархічна структура каталогів датасету

Всередині кожної папки класу зберігаються окремі файли записів. Оскільки один жест являє собою не статичне зображення, а часову послідовність координат, для зберігання було обрано бінарний формат бібліотеки NumPy.

Переваги використання формату .npy порівняно з текстовим форматом:

– Шквидкість читання/запису: бінарний формат дозволяє завантажувати тензори безпосередньо в пам’ять без витрат часу на парсинг тексту;

– компактність: файли займають менше місця на диску;

– збереження типів: формат автоматично зберігає тип даних float32, що є важливим для подальшої обробки в PyTorch [35].

Прикад вмісту директорії з записами для класу «Fist» наведено на рисунку 3.4.

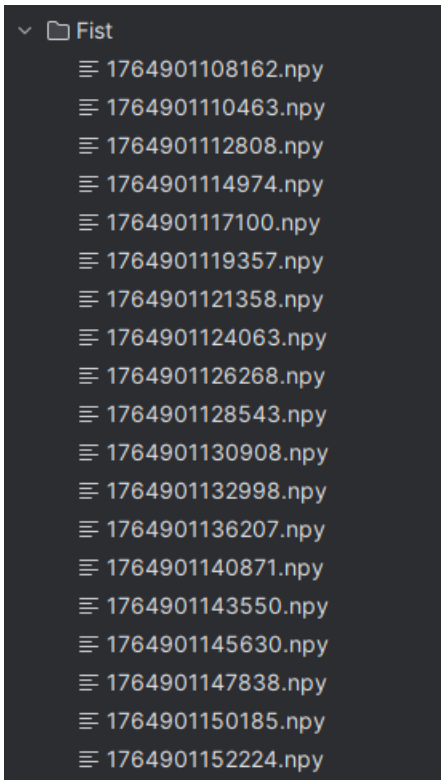

Рисунок 3.4 – Збережені тензори жестів у форматі .npy

Аналізуючи рисунок 3.4, для іменування файлів використовується унікальна мітка часу (timestamp), що генерується в момент запису (наприклад, 1764901…npy). Це рішення запобігає конфліктам імен файлів при багаторазовому переписі датасету та дозволяє легко сортувати записи за часом створення.

3.2.3 Модуль навчання моделі з аугментацією даних

Модуль навчання реалізує процес fine-tuning попередньо навченої моделі HandTransformer на реальних записах користувача з інтенсивною аугментацією для запобігання перенавчання.

Аугментація даних є критично важливою для малих датасетів (273 оригінальних записи).Реалізовано два типи аугментації: адитивний гаусівський шум та випадкове масштабування. У лістингу 3.4 наведено приклад функції *augment* для аугментації послідовності.

Лістинг 3.4 Функція аугментації послідовності:

```
def augment(sequence):
    seq = sequence.copy()

    noise = np.random.normal(0, 0.005, seq.shape)
    seq += noise

    scale = np.random.uniform(0.9, 1.1)
    seq = (seq - 0.5) * scale + 0.5

    return seq.astype(np.float32)
```

Шум з середньоквадратичним відхиленням 0.005 імітує природне тремтіння руки та неточність детекції MediaPipe. Масштабування на ±10% моделює зміну відстані між рукою та камерою [23].

З кожного оригінального запасу генерується 50 аугментованих варіантів, що збільшує тренувальних набір з 273 до 13928 прикладів. Тестова збірка залишається без аугментації для об’єктивної оцінки генералізаційної здатності моделі.

Процес навчання реалізовано з використанням оптимізатора Adam (learning rate 0,0005) та функції витрат CrossEntropyLoss , що наведено у лістингу 3.5.

Лістинг 3.5 Цикл навчання моделі:

```
model = HandTransformer().to(device)
```

```
model.load_state_dict(torch.load('pretrained_transformer.pth'))
optimizer = torch.optim.Adam(model.parameters(), lr=0.0005)
criterion = nn.CrossEntropyLoss()

for epoch in range(25):
    model.train()
    for batch_x, batch_y in train_loader:
        optimizer.zero_grad()
        outputs = model(batch_x)
        loss = criterion(outputs, batch_y)
        loss.backward()
        optimizer.step()
    model.eval()
    test_acc = evaluate(model, test_loader)
    print(f'Epoch {epoch+1}, Test Acc: {test_acc:.2f}%')

torch.save(model.state_dict(), 'final_transformer.pth')
```

Модель навчається протягом 25 епох з batch size 16. Використання попередньо навчених ваг дозволяє досягти високої точності вже на першій епосі (82.6%), тоді як навчання з нуля показує лише 76.8% [35].

3.2.4 Модуль інфверенсу в реальному часі

Модуль інференсу забезпечує розпізнавання жестів у режимі реального часу з веб-камери. Відбувається інтеграція MediaPipe для детекції руки та навченої моделі HandTransformer для класифікації жестів.

Система працює з буфером останніх 30 кадрів, що відповідає довжині послідовності, на якій навчалась модель. Початкова ініціалізація продемонстрована у лістингу 3.6.

Лістинг 3.6 Ініціалізація системи реального часу:

```
model = HandTransformer()
model.load_state_dict(torch.load('final_transformer.pth'))
model.eval()

buffer = deque(maxlen=30)
cap = cv2.VideoCapture(0)
mp_hands = mp.solutions.hands.Hands()

gesture_labels = ["Noise", "Swipe Left", "Swipe Right", "Swipe Up",
                  "Swipe Down", "Zoom In", "Zoom Out", "Fist",
                  "Palm", "Peace", "Point"]
```

Основний цикл обробки кадрів виконує детекцію руки, екстракцію координат, додавання у буфер та інференс моделі. Реалізація цього компоненту системи понада у лістингу 3.7.

Лістинг 3.7 Цикл обробки кадрів:

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

    frame_rgb = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
    results = mp_hands.process(frame_rgb)

    if results.multi_hand_landmarks:
        landmarks = results.multi_hand_landmarks[0]
        coords = extract_coords(landmarks)
```

```
        buffer.append(coords)

        if len(buffer) == 30:
            sequence = torch.FloatTensor(list(buffer)).unsqueeze(0)
            with torch.no_grad():
                output = model(sequence)
                predicted_class = torch.argmax(output).item()
                gesture = gesture_labels[predicted_class]

            cv2.putText(frame, gesture, (10, 50),
                        cv2.FONT_HERSHEY_SIMPLEX, 1.5, (0, 255, 0), 3)

    cv2.imshow('Gesture Recognition', frame)
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break
```

Для зглажування результатів та зменшення кількості хибних спрацювань застосовується механізм temporal smoothing – жест вважається розпізнаним тільки якщо модель передбачає той самий клас на 5 послідовних кадрах. Реалізацію продемонстровано у лістингу 3.8.

Лістинг 3.8 Згладжування передбачень:

```
prediction_history = deque(maxlen=5)

prediction_history.append(predicted_class)

if len(prediction_history) == 5:
    if len(set(prediction_history)) == 1:
        stable_gesture = gesture_labels[predicted_class]
        display_gesture = stable_gesture
```

Система досягає шкидкості ~ 30 FPS на CPU, що забезпечує комфортну інтеракцію без помітної затримки. Основним bottleneck є не модель HandTransformer (~0.13 мс на кадр), а детекція руки MediaPipe (~25-30 мс) [12,20].

Інтерфейс відображає поточний розпізнаний жест великими літерами у верхній частині екрану, при виявленні динамічного жесту – помаранчевий колір, статичного – зелений та скелет руки з підсвіченими landmarks.

3.3 Інструкція користувача

Далі наведено основні інструкції для роботи користувача з програмним комплексом:

Крок 1. Налаштування робочого середовища. Для користування програмним комплексом необхідно ініціалізувати ізольоване віртуальне середовище Python, що дозволяє уникнути конфілктів версій бібліотек із системними пакетами. У терміналі виконується команда створення середовища, після чого здійснюється інсталяція необхідних залежностей з файлу requirements.txt.

Крок 2. Генерація синтетичних даних. Щоб зменшити збір великої кількості реальних даних, що є трудомістким процесом, передбачено етап попереднього навчання на синтетичних даних. Запуск відповідного модуля здійснюється командою: python 1_generate_pretrain.py. Скрипт виконує математичне моделювання траєкторії жестів, додає стохастичний шум для імітації природних рухів та генерує 13 200 навчальних прикладів.

Крок 3. Збір реальних даних. З метою адаптації моделі до реальних умов та індивідуальних особливостей користувача використовується модуль запису. Запуск інтерактивного режиму здійснюється командою: python 2_record_data.py.

Після запуску відкривається вікно з відеопотоком, де система в реальному часі візуалізує скелетну модель руки. Необхідно поступово записать по 30 реальних прикладів кожного жесту.

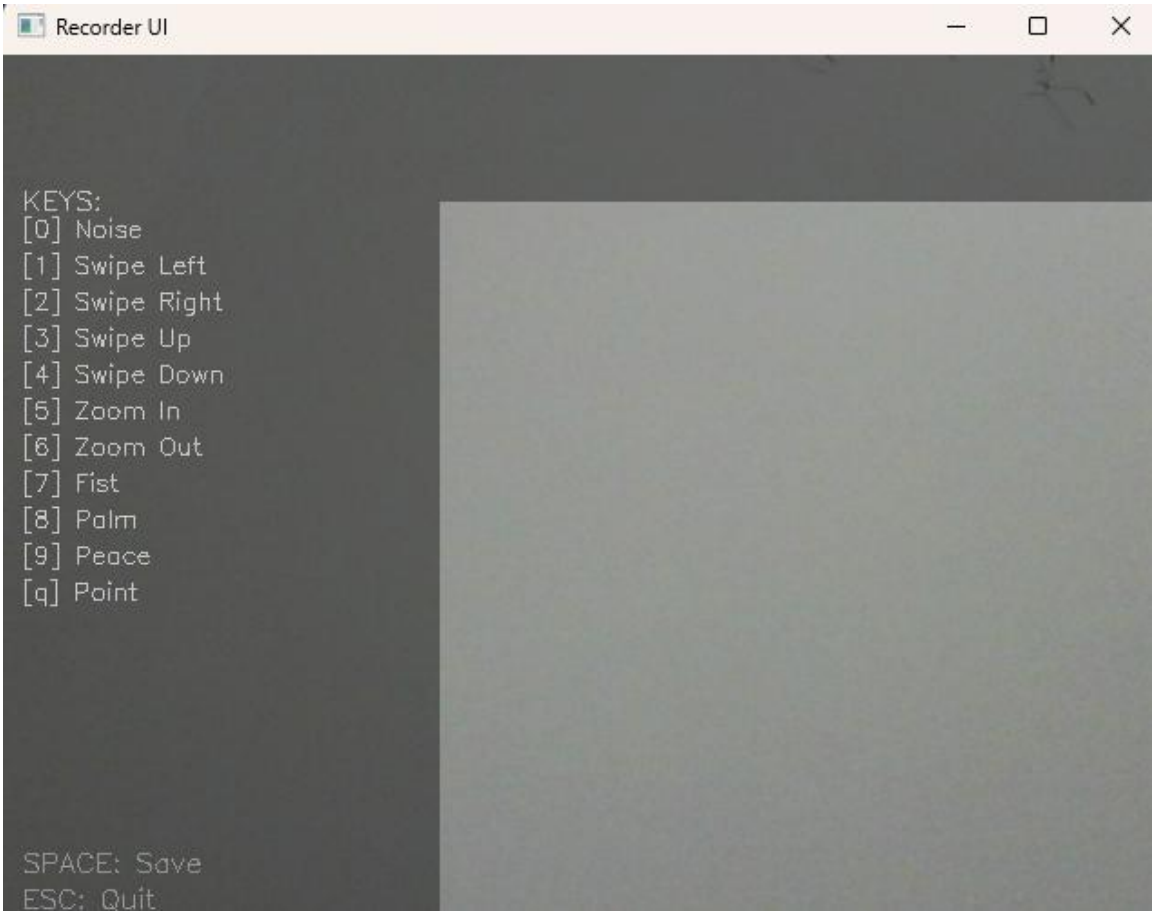

Рисунок 3.5 – Інтерфейс модуля збору

Крок 4. Донавчання моделі (Fine-tuning). На цьому етапі виконується адаптація ваг, отриманих на синтетичних даних, до особливостей реальних записаних рухів. Щоб уникнути перенавчання застосовується 50-кратка угментація реальних записів. Запуск процесу ініціюється командою: python 3_train_finetune.py. Результатом є оптимізована модель , що навчалась упродовж 25 епох.

Крок 5. Запуск системи розпізнавання. Фінальним етапом є запуск інференсу в реальному часі. Виконується командою: python 4_run_system.py. Система активує веб-камеру, помічає детектецію руки, буферизує останні 30 кадрів та класифікує жест. Результат відображається з допомогою кольорової

індексації для кращого відстеження: зелений колір – статичні жести, помаранчевий – динамічні.

3.4 Дослідження та тестування системи

Для оцінки ефективності розробленої системи було проведено комплекс експериментів на реальному датасеті, зібраному за допомогою модуля запису. Дослідження спрямовані на визначення впливу transfer learning на швидкість конвергенції та фінальну точність моделі, а також на детальний аналіз помилок класифікації.

3.4.1 Методика тестування

Датасет складається з 342 оригінальних записів жестів, розподілених між 11 класами. Розподіл зразків по класах представлено у таблиці 3.2.

Таблиця 3.2 – Розподіл зразків по класах

Клас	Кількість записів
1	**2**
Noise	32
Swipe Left	30
Swipe Right	30
Swipe Up	31
Swipe Down	30
Zoom In	30
Zoom Out	30
Fist	33

Продовження таблиці 3.2

1	2
Palm	30
Peace	35
Point	31

Набір даних розділено на тренувальну та тестову вибірку у співвідношені 80/20 із стратифікацією по класах для збереження пропорцій. Це дало 273 записи для тренування та 69 записів для тестування. До тренувальної вибірки застосовано 50-кратну аугментацію, що збільшило її до 13923 прикладів. Тестова вибірка залишилася без аугментації для об’єктивної оцінки.

Модель навчалася протягом 25 епох з наступними гіперпараметрами:

– оптимізатор Adam;

– шквидкість навчання (learning rate) 0,0005;

– розмір пакета (batch size) 16;

– функція втрат CrossEntropyLoss;

– регуляризація: Dropout 0,1 та 0,2, Layer Normalization.

Для оцінки якості класифікації використовувалися стандартні метрики [3, 7]:

Точність (Accuracy) – частка правильно класифікованих прикладів:

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN}, \quad (3.1)$$

де *TP* – правильно розпізнані позитивні приклади;

TN – правильно розміщені негативні;

FP – хибні спрацювання;

FN – пропущені об’єкти.

Precision – частка істинно позитивних серед усіх передбачених позитивних:

$$Precision = \frac{TP}{TP+FP}. \quad (3.2)$$

Recall – частка знайдених позитивних серед усіх реальних позитивних:

$$Recall = \frac{TP}{TP+FN}. \quad (3.3)$$

F1-Score – гармонійне середнє Precision та Recall:

$$F1 = \frac{2*Precision*Recall}{Precision+Recall}. \quad (3.4)$$

3.4.2 Порівняльний аналіз ефективності transfer learning

Для визначення впливу попереднього навчання на синтетичних даних було проведено два експерименти з ідентичними умовами навчання:

– експеримент А (з transfer learning) – HandTransformer з завантаженням попередньо навчених ваг з файлу pretrained_transformer.pth;

– експеримент Б (без transfer learning) – HandTransformer з випадковою ініціалізацією ваг.

Результати навчання представлені у таблиці 3.3

Таблиця 3.3 – Порівняння ефективності transfer learning

Епоха	Експеримент А		Експеримент Б	
1	**2**	**3**	**4**	**5**
	Train acc/ Test acc	Test Loss	Train acc/ Test acc	Test Loss
1	65,5% / 84,1%	0,494	52,7% / 82,6%	0,503
5	96,5% / 98,6%	0,080	98,5% / 98,6%	0,149
10	98,7% / 95,7%	0,237	99,5% / 98,6%	0,126

Продовження таблиці 3.3

1	2	3	4	5
15	99,1% / 88,4%	0,529	99,3% / 98,6%	0,116
25	98,6% / 97,1%	0,128	99,8% / 98,6%	0,228
Фінальна точність	97,10%		98.55%	

Динаміку функції втрат представлено на рисунках 3.6 та 3.7.

Рисунок 3.6 – Динаміка функції втрат моделі з transfer learning

Рисунок 3.7 – Динаміка функції втрат моделі без transfer learning

На основі отриманих результатів було проведено наступний аналіз.

Використання механізму трансферного навчання (transfer learning) забезпечує швидшу початкову конвергенцію: модель в експерименті А на першій епосі досягає test accuracy 84,1% проти 82,6% в експерименті Б. Це підтверджує, що попередньо навчені на синтетичних даних ваги містять корисні патерни рухів руки, які прискорюють адаптацію моделі до реальних жестів. Вже на п’ятій епосі обидва експерименти досягають однакової test accuracy 98,6%, проте експеримент А потребує менше тренувальних ітерацій [33].

Навчання без pre-training показало вищу фінальну точність: експеримент Б досяг 98,55% проти 97,10% в експерименті А. Причиною є тимчасове перенавчання моделі з pretrain на епохах 14-15: test accuracy різко падає до 88,4% при зростанні test loss до 0,529, тоді як експеримент Б демонструє стабільні 98,6% протягом епох 5-25. Після епохи 16 експеримент А відновлюється до 97,1%, проте вже не досягає попереднього піку 98,6% з епохи 5 [36].

Обидва експерименти демонструють відсутність критичного перенавчання моделі: фінальний train loss становить 0,0104 (експеримент А) та 0,0092 (експеримент Б), що значно відрізняється від train loss = 0,0000 у базових моделей CNN та LSTM (див. розділ 3.5). Train accuracy 98,6% та 99,8% відповідно не досягає 100%, що підтверджує здорову динаміку навчання незалежно від способу ініціалізації ваг.

Практична цінність трансферного навчання полягає у скороченні часу розробки завдяки швидкій конвергенції на початкових епохах, а не у покращені фінальної точності.

3.4.3 Аналіз матриці помилок

Для детального розуміння характеру помилок класифікації було побудовано матриці помилок (confusion matrix) для обох експериментів на тестовій виборці з 69 прикладів.

Результати представлено на рисунках 3.8 та 3.9.

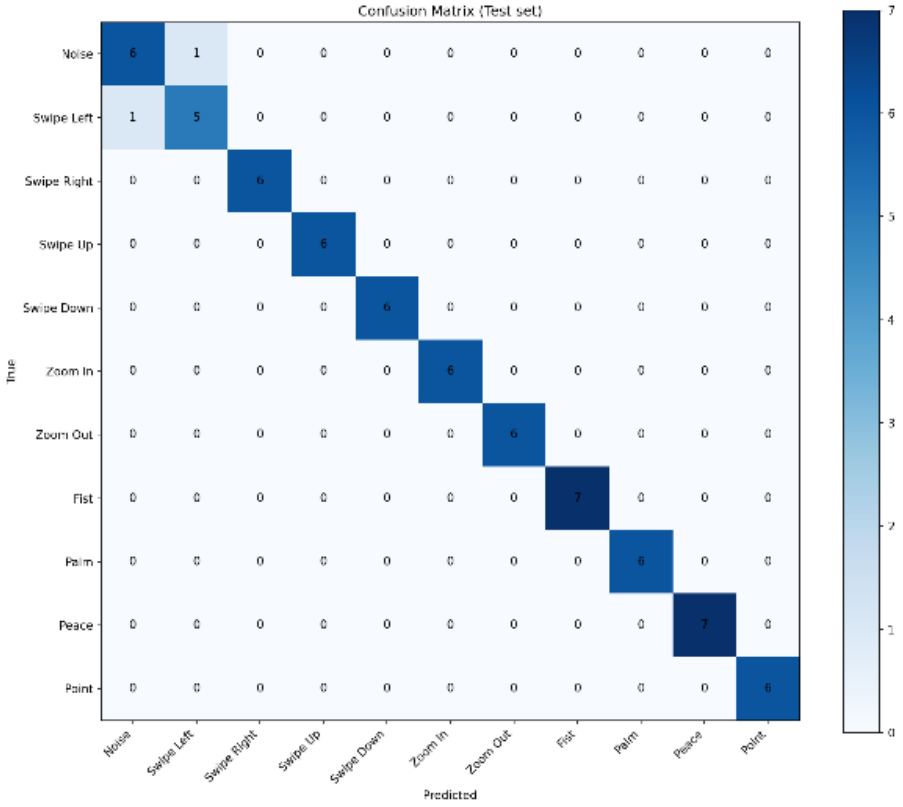

Рисунок 3.8 – Матриця помилок для експерименту А (з transfer learning)

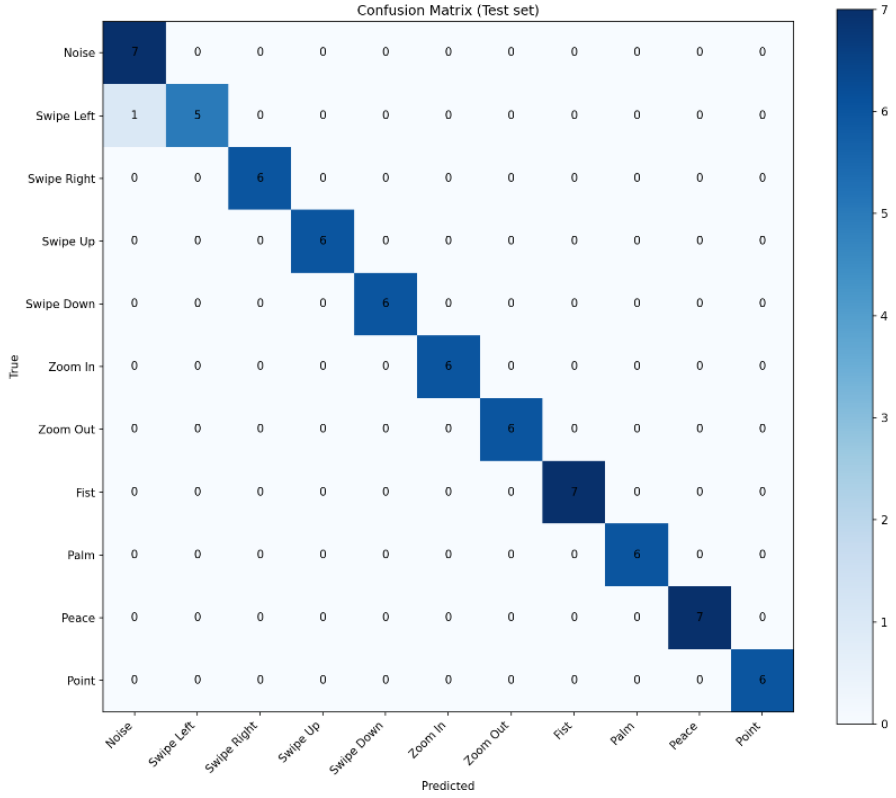

Рисунок 3.9 – Матриця помилок для експерименту Б (без pre-training)

Наведені матриці помилок дозволяють виявити специфічні класи жестів, які модель може плутати між собою, та оцінити якість розпізнавання кожного класу окремо. На основі результатів було проведено аналіз матриць помилок для обох експериментів.

Експеримент А допустив 2 помилки з 69 тестових прикладів: перша помилка – жест «Noise» класифіковано як «Swipe Left», друга помилка – жест «Swipe Left» класифіковано як «Noise». Дзеркальний характер помилок можемо пояснити тим, що на початкових кадрах динамічного жесту рука може залишатися нерухомою, що за просторовою конфігурацією нагадує клас «Noise» [37].

Експеримент Б допустив лише 1 помилку: жест «Swipe Left» класифіковано як «Noise». Важливо відзначити, що це той самий конкретний приклад, на якому помилилась модель в експерименті А, що підтверджує об’єктивну складність даного запису для розпізнавання незалежно від способу навчання моделі.

Усі статичні жести розпізнано без помилок: класи Fist, Palm, Peace, Point показали 100% accuracy в обох експериментахю Це підтверджує, що механізм Self-Attention ефективно вивчає просторову конфігурацію кисті руки навіть на малих вибірках.

Ідентичний характер помилок в обох експериментах свідчить про архітектурне обмеження моделі при розрізненні початкової фази руху від статичної позиції, а не про вплив pre-training на якість класифікації. Статичні ж жести розпізнаються бездоганно в обох сценаріях.

3.4.4 Метрики класифікації за класами

Для детального розуміння точності передбачування кожного з класів жестів було надано детальні метрики для кожного класу в експерименті Б, що представлені у таблиці 3.4.

Таблиця 3.4 – Метрики класифікації за класами (Експеримент Б)

Клас	Precision	Recall	F1-Score	Підтримка
Noise	0,88	1,00	0,93	7
Swipe Left	1,00	0,83	0,91	6
Swipe Right	1,00	1,00	1,00	6
Swipe Up	1,00	1,00	1,00	6
Swipe Down	1,00	1,00	1,00	6
Zoom In	1,00	1,00	1,00	6
Zoom Out	1,00	1,00	1,00	6
Fist	1,00	1,00	1,00	7
Palm	1,00	1,00	1,00	6
Peace	1,00	1,00	1,00	7
Point	1,00	1,00	1,00	6
Macro avg	0,99	0,98	0,99	69
Weighted avg	0,99	0,99	0,99	69

Після наведення необхідних метрик для порівння класифікації за жестами було проведено аналіз отриманих результатів.

Клас «Noise» має найнижчий precision (0,88): з восьми випадків, коли запропонована модель перебачила цей клас, один виявився помилковим. Це означає, що жест «Swipe Left» один раз був неправильно класифікований як відсутність жесту. Recall = 1.00 гарантує, що всі 7 явних прикладів класу «Noise» розпізнано правильно без пропусків. Дана поведінка є бажаною для інтерактивних систем, коли краще іноді не виконати дію, ніж виконати неправильну.

Клас «Swipe Left» має знижений recall (0,83): один з 6 тестових прикладів не був розпізнаний, проте precision = 1,00 гарантує відсутність хибних спрацювань – розпізнавання цього жесту завжди є правильною класифікацією.

Дев’ять класів демонструють ідеалні показники: precision = recall = F1-score = 1,00 для наданих статичних та динамічних жестів. Macro average 0,99

демонструє середню якість по всіх классах без врахування їх розмірів, коли weighted average 0,99 враховує кількість прикладів у кожному класі.

Загальна точність системи 98,55% є високим результатом для задачі мультикласової класифікації 11 класів жестів на малому наборі даних та підтверджує ефективність запропоноваї архітектури.

3.5 Порівняльний аналіз з альтернативними підходами

Для об’єктивної оцінки ефективності запропонованої архітектури було проведено порівняльний аналіз з двома базовими (baseline) підходами до розпізнавання послідовних даних:

– GesutreCNN Simple – спрощена згорткова нейронна мережа;

– GestureLSTM Simple – спрощена рекурентна мережа на основі LSTM.

Важливим фактором експерименту було тестунавання центральному процессорі (CPU). Це відповідає реальним умовам розгортання системи у вбудованих системах, де використання GPU є недоцільним.

3.5.1 Архітектура моделей для порівняння

GestureCNN Simple – спрощена згорткова мережа, що розглядає послідовність координат (30 x 42) як двовимірне зображення. Архітектура включає два блоки згортки (Conv2D 3x3 + BatchNorm + MaxPool) та фінальний класифікатор. Кількість фільтрів зменшено (16 → 32) для забезпечення співмірної кількості параметрів з трансформером [32].

GestureLSTM Simple – рекурентна мережа з одним шаром LSTM (64 приховані одиниці). Модель обробляє дані послідовно, зберігаючи контекст у прихованому векторі. Використовується одношарова архітектура для зменшення схильності до перенавчання на малих даних.

HandTransformer (запропонована модель) – використовує механізм Self-Attention, що дозволяє аналізувати залежності між усіма кадрами одночасно. Для даного експерименту використовувалася версія моделі з попереднім навчанням (Transfer Learning) на синтетичних даних, щоб оцінити вплив перенесення знань на стабільність навчання.

3.5.2 Результати порівняльного аналізу

Усі моделі навчалися на ідентичному наборі даних (273 реальних приклади з 50-кратною аугментацією). Результати тестування продуктивності на CPU наведено у таблиці 3.5

Таблиця 3.5 – Порівняння архітектур нейронних мереж на CPU

Модель	Test acc	Train acc (final)	Train Loss (final)	Параметри	FPS	Перенавчання
HandTransformer	97,10%	98,8%	0,0475	74059	7633	Ні
GestureCNN Simple	98.55%	100%	0.0016	29457	10435	Критичне
GestureCNN Simple	98.55%	100%	0.0003	28363	16438	Сильне

Порівняння динаміки точності класифікації (Accuracy) на тестовій вибірці для трьох досліджуваних архітектур представлено на рисунку 3.10.

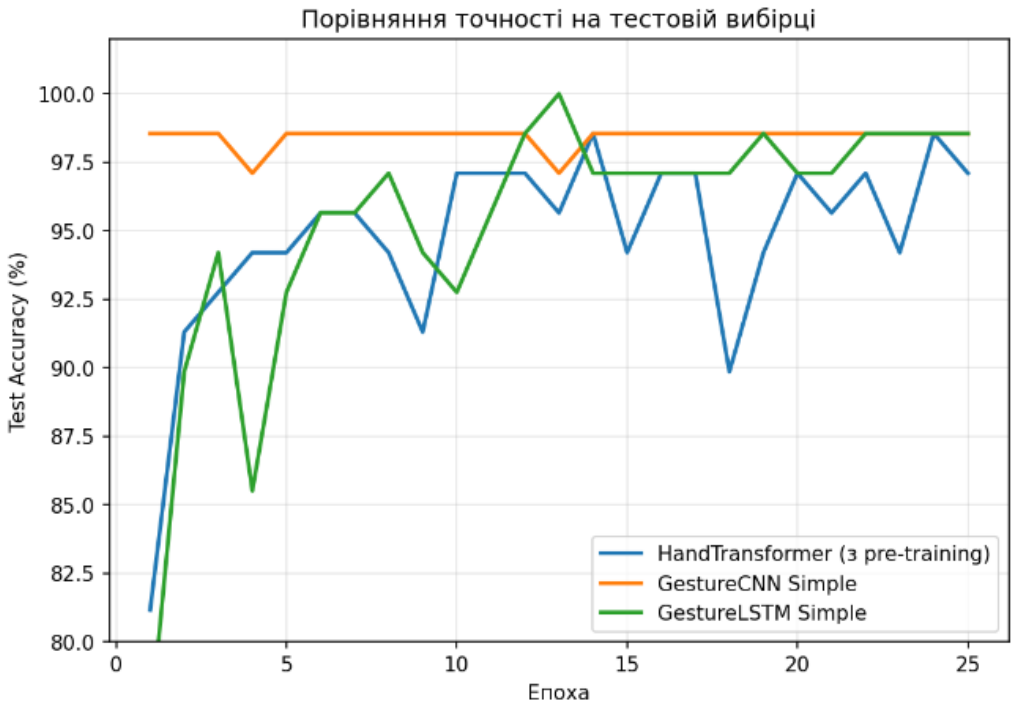

Рисунок 3.10 – Графік залежності точності розпізнавання від епохи навчання

Динаміка зміни функції втрат (Loss Function) на тестовій вибірці, що відображає «впевненості» моделі та наявність перенавчання, наведено на рисунку 3.11.

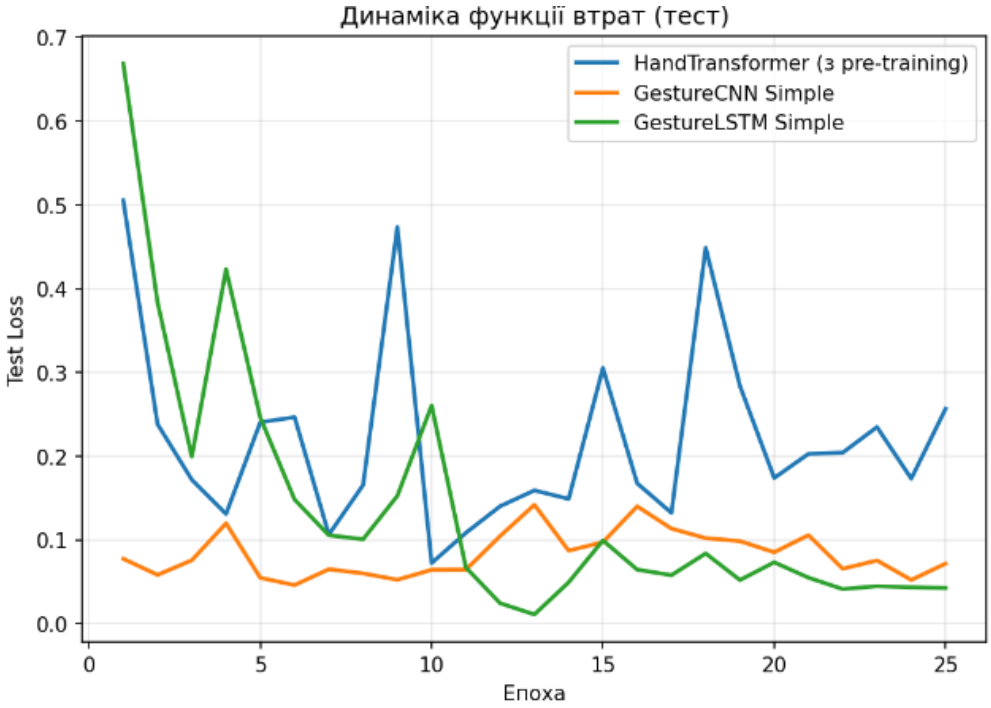

Рисунок 3.11 – Графік динаміки функції втрат на тестовій вибірці

3.5.3 Аналіз стабільності навчання та оцінка перенавчання

Аналіз кривих, представлених на рисунку 3.11, дозволяє виявити фундаментальні відмінності у характері навчання моделей, які неможливо помітити при аналізі лише фінальної таблиці точності.

GestureCNN Simple (помаранчева лінія) демонструє класичний результат «механічного запам’ятовування». На графіку втрат видно, що після 5-ї епохи помилка на тестовій вибірці перестає знижуватись і виходить на відносно пряму лінію, тоді як на тренувальній вибірці вона падає до нуля (Train Loss 0.0016). Можно зробити висновок, що згорткова мережа вичерпала здатність до узагальнення і почала підлаштовуватися під шум тренувальних даних [38].

GestureLSTM Simple (зелена лінія) характеризується високою нестабільністю градієнтів. На рисунку 3.11 видно різкі сплески (spikes) значення функції втрат, вказуючи на те, що модель періодично забуває контекст або робить грубі помилки з високою впевненістю. Така поведінка є критичним ризиком для системи реального часу, де важлива передбачуваність роботи [39].

HandTransformer (синя лінія) демонструє складну динаміку пошуку глобального мінімуму. На графіку спостерігається різке зростання помилки в районі 17-ї епохи, що характерно для процесу fine-tuning трансформерів, коли модель намагається адаптувати попередньо навчені ваги під складні аугментовані приклади. Критично важливим спостереженням є те, що модель змогла відновитися після цього сплеску, знизивши помилку до фінального значення 0.0475. Така поведінка свідчить про пластичність нейронної мережі та здатність виходити з локальних мінімумів.

Використання HandTransformer демонструє найкращий баланс. Модель не запам’ятовує дані як CNN (зберігаючи Train Loss > 0), незважаючи на наявність коливань у процесі навчання, демонструє здатність до відновлення та досягнення високої точності, що робить його перспективним для роботи зі змінними даними реального світу.

3.5.4 Аналіз швидкодії та обчислювальної ефективності

Тестування швидкодії на центральному процессорі виявило суттєві відмінності у продуктивності архітектур, проте підтвердило їх повну придатність для роботи в реальному часі.

Нижче було проведно аналіз результаів показників швидкодії.

GestureLSTM Simple продемонструвала найвищу швидкість (16438 FPS) завдяки найменшій кількості параметрів (28363) та оптимізованій реалізації рекурентних операцій у PyTorch. Ця швидкість не компенсує виявену раніше нестабільність, а послідовна природа обробки LSTM обмежує можливості паралелизації на сучасних багатоядерних процесорах.

GestureCNN Simple займає проміжну позицію (10345 FPS). Згорткові операції добре оптимізовані, але архітектура потребує більше пам’яті для зберігання карт ознак (Feature Maps) на проміжних шарах порівняно з рекурентними мережами.

HandTransformer показав 7633 FPS при найбільшій кількості параметрів (74 059). Пояснено високою ефективністю матречних операцій у механізмі Self-Attention. Сучасні CPU використовують векторні інструкції (AVX/AVX2) та біблиотеки лінійної алгебри, що дозволяє виконувати множення матриць Q, K , V паралельно. Компактність моделі (0,28 МБ) дозволяє їй повністю вміститися у L3 кеш-пам’ять процесора, що мінімізує повільні звернення до оперативної пам’яті (RAM).

Запас продуктивності системи: стандартна веб-камера видає потік із частотою 30 кадрів за секунду. Навіть найповільніша з досліджувальних моделей забезпечує 250-кратний запас продуктивності. У реальній системі загальна швидкість обмежується не класифікацією (~ 0,13 мс на кадр), а етапом детекції руки модулем MediaPipe (~25-30 мс на кадр).

Швидкість інференсу розробленої моделі не є обмежуючим фактором. Обрана архітектура гарантує миттєву реакцію інтерфейсу, залишаючи основний обчислювальний ресурс процесора для роботи інших компонентів системи.

3.6 Заходи щодо вдосконалення системи розпізнавання жестів.

Розроблена система демонструє стабільну роботу, забезпечуючи ефективне розпізнавання та класифікацію жестів у реальному часі на основі трансформерної архітектури. У процесі експериментальних досліджень та аналізу результатів було виявлено низку можливостей для її подальшого розвитку та технічної оптимізації.

Одним із ключових напрямів подальшого розвитку є розширення навчального датасету, що дозволить моделі краще узагальнювати різноманітність стилів виконання жестів. Необхідно збільшити кількість записів до тисячі прикладів, залучивши респондентів різного віку та статі, забезпечити варіативність умов освітлення та відстані до камери. Дані фактори створять можливості для підвищення робастності системи та її адаптації до реальних умов використання.

Окремої уваги потребує розширення функціональних можливостей системи шляхом додавання нових класів жестів. Доцільно реалізувати розпізнавання складніших маніпуляцій, таких як діагональні рухи, зведення пальців або обертання кисті, забезпечити підтримку жестів, що виконуються двома руками одночасно. Це дозволить використовувати систему для побудови більш складних та інтуїтивних інтерфейсів керування.

Продуктивність системи може бути підвищена через оптимізацію для мобільних пристроїв. Експорт моделі у формат ONNX та застосування методів квантизації дозволять суттєво зменшити розмір нейромережі та прискорити інференс на ARM-процесорах. Використання цих побажань забезпечить можливість інтеграції системи у мобільні застосунки без залежності від серверних обчислень, роблячи рішення автономним.

Персективним напрямом є впровадження механізму адаптивного навчання, що дозволить моделі підлаштовуватися під індивідуальний стиль виконання жестів конкретного користувача. Система може збирати дані під час використання та періодично дотреновувати останні шари нейромережі, цей крок

зможе підвищити точність розпізнавання для часто використовуванних жестів. Доцільно реалізувати реалізувати систему впевненонсті, уточнюючи в користувача вірність жесту при низькому відсотку передбачування системи.

Система може бути розширена за рахунок інтеграції з середовищем віртуальної або доповненої реальності шляхом розробки плагінів для ігрових платформ. Додатково варто розглянути оптимізацію швидкості детекції руки через використання легковажних моделей або алгоритмів трекінгу, дозволивши зменшити затримки у конвеєрі обробки та підвищити загальну частоту кадрів.

Забезпечення надійності системи у критичних сценаріях використання за допомогою механізму детекції аномалій. Для медичних або промислових застосувань доцільно додати систему логування та аналітики, що дозволить відстежувати статистику використання жестів та виявляти проблемні ситуації. Крім того, інтеграція з хмарними сервісами забезпечить можливість колективного навчання моделі на даних від множини користувачів, що підвищить загальну якість розпізнавання.

Розглянувши всі фактори, система має значний потенціал для масштабування. Заплановані удосконалення здатні суттєво розширити сфери її застосування, підвищи швидкодію та забезпечити персоналізацію розпізнавання під індивідуальний стиль користувача.

ВИСНОВКИ

Таким чином, у кваліфікаційній роботі досліджено методи розпізнавання та відстеження рухів рук людини за допомогою нейронних мереж та вирішено такі завдання:

– здійснено огляд сучасних наукових джерел щодо методів класифікації жестів руки, моделей трекінгу та трансформерних архітектур, що дозволило систематизувати існуючі підходи та обґрунтувати доцільність використання гібридних моделей для підвищення точності розпізнавання;

– досліджено принципи роботи MediaPipe Hands та визначено переваги використання скелетних координат у порівнянні з аналізом зображень, що дало можливість відмовитися від ресурсоємних згорткових мереж на користь швидких та ефективних алгоритмів обробки ключових точок;

– сформовано архітектуру системи розпізнавання жестів на основі трансформера та визначено формат представлення вхідних структурних ознак, що забезпечило можливість моделювання складних часових залежностей та підвищило стійкість системи до динамічних змін у відеопотоці;

– розроблено алгоритм зчитування, нормалізації та підготовки часових послідовностей для навчання нейронної мережі, що дозволило уніфікувати вхідні дані та нівелювати вплив відстані до камери та індивідуальних особливостей руки користувача;

– реалізовано програмний застосунок, що забезпечує збір даних, навчання, тестування та використання моделі в реальному часі, що дозволило практично підтвердити працездатність запропонованого підходу та його придатність для використання на стандартному обладнанні;

– проведено експериментальне оцінювання точності класифікації статичних і динамічних жестів, проаналізовано результати та сформовано висновки щодо ефективності запропонованого методу, що підтвердило перевагу

трансформерної архітектури над базовими згортковими та рекурентними моделями.

У рамках кваліфікаційної роботи було проведено дослідження гібридного методу розпізнавання жестів шляхом поєднання високоточного детектора ключових точок MediaPipe та архітектури HandTransformer, попередньо навченого на синтетичних даних та дообученого на реальних даних для вирішення проблеми ресурсоємкого процесу збору великого набору даних, реалізованого у вигляді програмного комплексу для обробки відеопотоку в реальному часі.

Побудовано покроковий алгоритм попередньої обробки даних, який включає нормалізацію координат, аугментацію та часову інтерполяцію, що дозволило значно покращити якість навчання моделі на обмеженому наборі даних.

Створено унікальний датасет жестів, що включає статичні пози та динамічни рухи в різних умовах освітлення, та розроблено графічний інтерфейс для запису жестів, донавчання моделі та тестування розпізнавання в реальному часі з візуалізацією скелета руки.

Наукова новизна роботи полягає в удосконаленні методу навчання систем розпізнавання жестів за рахунок поєднання синтетично згенерованих і реальних траєкторій руху для тренування єдиної трансформерної моделі. Це дозволило вирішити проблему нестачі навчальних даних (data scarcity) та забезпечити високу точність розпізнавання як статичних, так і динамічних жестів без необхідності збору великого датасету.

Результати роботи апробовано у вигляді тез доповідей на двох міжнародних конференціях: IX Міжнародній студентській науковій конференції «Глобалізація наукових знань: міжнародна співпраця та інтеграція галузей наук» (7 листопада 2025 р., м. Черкаси) [41] та XI International Scientific and Practical Conference «World Science: Problems, Issues and Prospects for Development» (11–14 листопада 2025 р., м. Софія, Болгарія) [42].

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Mashtalir, S. V., & Lendel, D. P. (2024). Moving object shape detection by fragment processing. *Herald of Advanced Information Technology*, vol. 7, no. 4, pp. 414–423.

2. Яковлева, О. В. (2023). Методи комп'ютерного зору в системах відеоспостереження та біометричної ідентифікації: навч. посібник. Харків: ХНУРЕ, 160 с.

3. Mashtalir, S. V., & Novichonok, M. S. (2024). Research on the possibility of using methods of contextual video classification. Scientific search of young researchers: Proceedings. Zaporizhzhia National University, pp. 45–52.

4. Kovtunenko, A., & Mashtalir, S. (2025). Improved segmentation model to identify object instances based on textual prompts. *Herald of Advanced Information Technology*, vol. 8, no. 1, pp. 12–20.

5. Yakovleva, O., & Tvoroshenko, I. (2024). Hybrid approaches in computer vision for motion tracking. *Modern Information Systems*, vol. 8, no. 2, pp. 45–56.

6. Bouchrika, T., Zaied, M., Jemai, O., & Ben Amar, C. (2023). Neural solutions to interact with computers by hand gesture recognition. *Multimedia Tools and Applications*, vol. 82, no. 3, pp. 4567–4589.

7. Яковлева, О. В., & Ніколенко, В. В. (2022). Дослідження методів детекції об'єктів у відеопотоці в реальному часі. *Сучасні інформаційні системи*, Т. 6, № 1, С. 54–61.

8. Ковтуненко, А. Р., & Машталір, С. В. (2025). A Review of Modern Neural Network Architectures for Image Segmentation. *АСУ та прилади автоматики*, № 185, С. 53–62.

9. Gorokhovatskyi, V., Tvoroshenko, I., & Yakovleva, O. (2024). *Structural analysis of images in AI systems: monograph*. Kharkiv: KhNURE, 220 с.

10. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., & Polosukhin, I. (2017). Attention is all you need. *Advances in neural information processing systems*, vol. 30, pp. 5998–6008.

11. Li, Y., & Hsieh, C. J. (2025). Robust Hand Gesture Recognition using Vision Transformers. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 45, no. 3, pp. 1120–1135.

12. Saharia, S. (2025). Attention at Hand: A Comprehensive Survey of Transformer-based Hand Gesture and Landmark Detection Models. 2025 IEEE Guwahati Subsection Conference (GCON), June 2025, Guwahati, India, pp. 1–6..

13. Garg, M., Ghosh, D., & Pradhan, P. M. (2024). MVTN: A multiscale video transformer network for hand gesture recognition. European Conference on Computer Vision, pp. 15–33.

14. Dosovitskiy, A., Beyer, L., Kolesnikov, A., et al. (2020). An image is worth 16x16 words: Transformers for image recognition at scale. URL: https://arxiv.org/abs/2010.11929 (дата звернення: 20.11.2025).

15. Mastaler, S. V. (2020). Image recognition in computer vision processes. *Radioelectronics and Communications Systems*, vol. 63, no. 7-8, pp. 389–396.

16. Yakovleva, O. V. (2021). Аналіз ефективності згорткових нейронних мереж у задачах розпізнавання образів. *Системи обробки інформації*, Т. 2, № 165, С. 28–35.

17. Lugaresi, C., Tang, J., Nash, H., et al. (2019). Mediapipe: A framework for building perception pipelines. URL: https://arxiv.org/abs/1906.08172 (дата звернення: 22.11.2025).

18. MediaPipe Hands. Google for Developers. URL: https://developers.google.com/mediapipe/solutions/vision/hand_landmarker (дата звернення: 10.11.2025).

19. Khattab, M. M., El-Sayed, A., & Tolba, A. S. (2023). Integrated MediaPipe with a CNN model for Arabic sign language recognition. *Multimedia Tools and Applications*, vol. 82, no. 12, pp. 18103–18121.

20. Bhatt, V., & Dash, R. (2023). Real-time hand gesture recognition for American sign language using CNN, MediaPipe and convexity approach. *Journal of Ambient Intelligence and Humanized Computing*, vol. 14, no. 7, pp. 8901–8915.

21. Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, vol. 9, no. 8, pp. 1735–1780.

22. Кобилін, О. А., & Творошенко, І. С. (2021). Методи цифрової обробки зображень: навч. посібник. Харків: ХНУРЕ, 124 с.

23. Shorten, C., & Khoshgoftaar, T. M. (2019). A survey on image data augmentation for deep learning. *Journal of big data*, vol. 6, no. 1, pp. 1–48.

24. Python 3.9 Documentation. URL: https://docs.python.org/3.9/ (дата звернення: 12.11.2025).

25. Li, H.-H., & Hsieh, C.-C. (2025). Dynamic hand gesture recognition using MediaPipe and Transformer. *Engineering Proceedings*, vol. 108, no. 1, p. 22.

26. PyTorch Documentation. URL: https://pytorch.org/docs/stable/index.html (дата звернення: 12.11.2025).

27. Paszke, A., Gross, S., Massa, F., et al. (2019). Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, vol. 32, pp. 8024–8035.

28. OpenCV Library Documentation. URL: https://docs.opencv.org/4.x/ (дата звернення: 15.11.2025).

29. NumPy Documentation. URL: https://numpy.org/doc/stable/ (дата звернення: 18.11.2025).

30. PyCharm: The Python IDE for Professional Developers. URL: https://www.jetbrains.com/pycharm/ (дата звернення: 22.11.2025).

31. Jupyter Project Documentation. URL: https://docs.jupyter.org/en/latest/ (дата звернення: 22.11.2025).

32. Python Profilers Analysis. URL: https://docs.python.org/3/library/profile.html (дата звернення: 23.11.2025).

33. Git Documentation. URL: https://git-scm.com/doc (дата звернення: 23.11.2025).

34. Gorokhovatskyi, V., Chmutov, Y., Tvoroshenko, I., & Kobylin, O. (2025). Reducing computational costs by compressing the structural description in image classification methods. *Advanced Information Systems*, vol. 9, no. 1, pp. 5–12.

35. Tvoroshenko, I., Gorokhovatskyi, V., Kobylin, O., & Tvoroshenko, A. (2023). Application of deep learning methods for recognizing and classifying culinary dishes. *International Journal of Academic and Applied Research*, vol. 7, no. 9, pp. 57–70.

36. Kingma, D. P., & Ba, J. (2014). Adam: A method for stochastic optimization. URL: https://arxiv.org/abs/1412.6980 (дата звернення: 25.11.2025).

37. Srivastava, N., Hinton, G., et al. (2014). Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929–1958.

38. Творошенко, І. С. (2021). Технології прийняття рішень в інформаційних системах: навч. посібник. Харків: ХНУРЕ, 118 с.

39. Daradkeh, Y. I., Gorokhovatskyi, V., Tvoroshenko, I., & Zeghid, M. (2024). Improving the Effectiveness of Image Classification Structural Methods. *Computers, Materials & Continua*, vol. 80, no. 2, pp. 2145–2163.

40. Gupta, K. A., Singh, A., et al. (2023). Hand gestures recognition using edge computing system based on vision transformer and lightweight CNN. *IEEE Transactions on Consumer Electronics*, vol. 69, no. 4, pp. 1015–1024.

41. Черепов, Д. Е. (2025). Дослідження методу Vision Transformer (ViT) для відстеження рухів рук людини. *Глобалізація наукових знань: міжнародна співпраця та інтеграція галузей наук: матеріали IX Міжнародної студентської наукової конференції*, 7 листопада 2025 р., Черкаси, Україна. Вінниця: ТОВ «УКРЛОГОС Груп», С. 300–302.

42. Черепов, Д. Е. (2025). Гібридна модель CNN-LSTM для аналізу та розпізнавання динамічних рухів рук людини. *World Science: Problems, Issues and Prospects for Development: Proceedings of the XI International Scientific and Practical Conference*, November 11–14, 2025, Sofia, Bulgaria. International Science Group, pp. 63–65. DOI: 10.46299/ISG.2025.2.11.