КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
БУДІВНИЦТВА І АРХІТЕКТУРИ

Факультет автоматизації інформаційних технологій

Кафедра інформаційних технологій

ПОЯСНЮВАЛЬНА ЗАПИСКА
ДО КВАЛІФІКАЦІЙНОЇ РОБОТИ
НА ЗДОБУТТЯ ОСВІТНЬОГО СТУПЕНЯ БАКАЛАВР

на тему:

Розробка системи виявлення жестів рук у відеопотоці за допомогою нейронної мережі MediaPipe

Торхов Антон Костянтинович

Київ 2025 р.

КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БУДІВНИЦТВА І АРХІТЕКТУРИ

автоматизації і інформаційних технологій

(факультет)

інформаційних технологій

(кафедра)

ЗАТВЕРДЖУЮ
Завідувачка кафедри ІТ
д.т.н., професор Гончаренко Т.А.

„___” _____2025 року

КВАЛІФІКАЦІЙНА РОБОТА ЗДОБУВАЧА СТУПЕНЯ ВИЩОЇ ОСВІТИ БАКАЛАВР

на тему: «**Розробка системи виявлення жестів рук у відеопотоці за допомогою нейронної мережі MediaPipe**»

Я як здобувач вищої освіти КНУБА розумію і підтримую політику закладу з академічної доброчесності. Я не надавав(-ла) і не одержував(-ла) недозволену допомогу під час підготовки цієї роботи. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело.

Здобувач
Торхов Антон Костянтинович
122 «Комп’ютерні науки»
(спеціальність)
Інформаційні управляючі системи і технології
(освітня програма)
Групи КН-21
Керівник Гончаренко Т. А.
(прізвище та ініціали)
д.т.н., професор
(вчене звання, науковий ступінь)
Рецензент к.т.н., доц. Доля О.В.
(Прізвище та ініціали)

Ідентичність підтверджую

Київ, 2025 р.

КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БУДІВНИЦТВА І АРХІТЕКТУРИ

Факультет:	Автоматизації інформаційних технологій
Випускова кафедра:	Інформаційних технологій
Освітній ступінь:	Бакалавр
Спеціальність:	Комп'ютерні науки
Освітня програма:	Інформаційні управляючі системи та технології

ЗАТВЕРДЖУЮ
Завідувачка кафедри ІТ
д.т.н., професор Гончаренко Т.А.
_____
„___" _____2025 року

З А В Д А Н Н Я

ДО ВИКОНАННЯ КВАЛІФІКАЦІЙНОЇ ВИПУСКНОЇ РОБОТИ НА

ЗДОБУТТЯ ОСВІТНЬОГО СТУПЕНЯ **БАКАЛАВР**

	Торхову Антону Констянтиновичу

1. Тема роботи - **Розробка системи виявлення жестів рук у відеопотоці за допомогою нейронної мережі MediaPipe**

затверджена наказом ректора КНУБА № 599/23/25 від «19» травня 2025 року

2. Керівник роботи	Гончаренко Тетяна Андріївна, д.т.н. професор

3. Строк подання Здобувачем роботи до захисту травень 2025 р.

4. Зміст пояснювальної записки за розділами:

Р.1 АКТУАЛЬНІСТЬ І КОНТЕКСТ ДОСЛІДЖЕННЯ

Р.2 ПРОЄКТУВАННЯ СИСТЕМИ РОЗПІЗНАВАННЯ ЖЕСТІВ РУК

Р.3 АКТУАЛЬНІСТЬ І КОНТЕКСТ ДОСЛІДЖЕННЯ

Р.4 ЕРГОНОМІКА ІТ ТА ТЕХНІКО-ЕКОНОМІЧНЕ ОБҐРУНТУВАННЯ

5. Графічний матеріал за розділами:

Робота викладена на 67 аркушах, містить 3 додатки, 18 таблиці, 24 рисунок, список використаної літератури із 20 найменувань.

6. Календарний план виконання роботи:

Види робіт та їх зміст	Дата виконання
Розділ 1	02.02.2025
Розділ 2	03.04.2025
Розділ 3	11.05.2025
Розділ 4	17.05.2025
Остаточне оформлення роботи	23.05.2025
Направлення роботи для перевірки на плагіат	24.05.2025
Попередній захист роботи на випусковій кафедрі	26.05.2025
Направлення роботи на рецензування	28.05.2025

7. Консультанти розділів атестаційної випускної роботи

Розділ	Прізвище, ініціали та посада консультанта	Перевірив	
		дата	підпис
Розділ 1	Рябчун Ю.В., доц.каф.ІТ	03.02.2025	
Розділ 2	Рябчун Ю.В., доц.каф.ІТ	04.04.2025	
Розділ 3	Мацієвський О. О., асистент каф.ІТ	12.05.2025	
Розділ 4	Рябчун Ю.В., доц.каф.ІТ	18.05.205	

8. Дата видачі завдання листопад 2025 р.

Зав. кафедри			Гончаренко Т.А.
	(підпис)		(прізвище та ініціали)
Керівники			Гончаренко Т.А.
	(підпис)		(прізвище та ініціали)
Здобувач			Торхов А.К.
	(підпис)		(прізвище та ініціали)

АНОТАЦІЯ

Торхов А. К. Розробка системи виявлення жестів рук у відеопотоці за допомогою нейронної мережі MediaPipe.

Атестаційна випускна робота бакалавра за спеціальністю 122 «Комп’ютерні науки», освітня програма «Інформаційні управляючі системи та технології». – Київський національний університет будівництва та архітектури. – Київ, 2025.

Робота присвячена розробці прототипу системи розпізнавання жестів руки у відеопотоці, що забезпечує детекцію долоні, локалізацію 21 опорної точки та класифікацію п'яти базових жестів у режимі реальної години. Підсистема глибокого зору реалізована за допомогою MediaPipe Hands, а логіка розпізнавання — на двошаровій MLP-мережі, написаній мовою Python з використанням TensorFlow та OpenCV, що гарантує швидкодію 30 FPS на процесорі Intel i5-8350U без дискретного GPU. У ході роботи спроектовано модулі захоплення відео, попередньої обробки кадру, екстракції ознак, класифікації жестів та графічний інтерфейс користувача, який відображає результати трекінгу та надає інтерактивний контроль. Експериментальна перевірка на власній датасеті (5 000 кадрів) показала середню точність 94 %, а юзабіліті-тест за методикою SUS дав оцінку 81 бал, що підтверджує придатність системи для інтерактивних кіосків та освітніх VR-лабораторій.

Робота викладена на 67 аркушах, містить 3 додатки, 18 таблиць, 24 рисунока, список використаної літератури із 20 найменувань.

Ключові слова: розпізнавання жестів, MediaPipe Hands, комп'ютерний зір, нейронні мережі, OpenCV, Python, MLP-класифікатор, графічний інтерфейс користувача, System Usability Scale

SUMMARY

Torkhov A. K. Development of a system for detecting hand gestures in a video stream using the MediaPipe neural network.

Bachelor's thesis in the specialty 122 “Computer Science,” educational program “Information Management Systems and Technologies.” – Kyiv National University of Construction and Architecture. – Kyiv, 2025.

The work is devoted to the development of a prototype system for recognizing hand gestures in a video stream, which provides palm detection, localization of 21 reference points, and classification of five basic gestures in real time. The deep vision subsystem is implemented using MediaPipe Hands, and the recognition logic is based on a two-layer MLP network written in Python using TensorFlow and OpenCV, which guarantees a speed of 30 FPS on an Intel i5-8350U processor without a discrete GPU. During the work, modules for video capture, frame preprocessing, feature extraction, gesture classification, and a graphical user interface that displays tracking results and provides interactive control were designed. Experimental testing on our own dataset (5,000 frames) showed an average accuracy of 94%, and a usability test using the SUS methodology gave a score of 81 points, confirming the suitability of the system for interactive kiosks and educational VR laboratories.

The work is presented on 67 pages and contains 3 appendices, 18 tables, 24 figures, and a list of 20 references.

Keywords: gesture recognition, MediaPipe Hands, computer vision, neural networks, OpenCV, Python, MLP classifier, graphical user interface, System Usability Scale

ЗМІСТ

ВСТУП

Протягом останнього десятиліття взаємодія людини з цифровими системами зазнала стрімких змін: від класичної клавіатури та миші до сенсорних дисплеїв, голосових асистентів та доповненої реальності. На цьому тлі дедалі більшої популяр

Незважаючи на значний поступ у сфері глибокого навчання, більшість комерційних рішень досі ґрунтується на пропрієтарних сенсорах з інфрачервоним підсвічуванням або глибинних камерах, що різко збільшує вартість входу і водночас створює ризик технологічного «лок-іну» через закритий SDK. Відкрита бібліотека **MediaPipe Hands** , запропонована Google Research, показала, що звичайне RGB-відео у зв'язці з двокомпонентною CNN-архітектурою (Palm Detection + Hand Landmark) здатне відтворювати скелет кисті з похибкою, прийнятною для більшості HCI-завдань. Проте канонічний pipeline MediaPipe вирішує лише завдання локалізації 21 «landmark»; перехід від геометричної схеми до класу ж

Актуальність теми підтримується одразу кількома факторами. По-перше, ринок цифрових кіосків самообслуговування в Європі зростає темпом 12 % на рік, і безконтактність розглядається як конкурентне перевага. По-друге, у сфері телемедицини попит на адаптивні реабілітаційні тренажери, що реагують не на «ідеальний» жест, а на фізично допустиму для пацієнта амплітуду, прогнозовано буде збільшуватись через демографічне старіння. По-третє, з боку освітніх STEM-кластерів є постійний запит на інструменти, які демонструють студію.

Мета роботи полягає у створенні прототипу системи, здатної в реальному часі детектувати пензель у відеопотоці звичайної веб-камери, локалізувати ключові точки та класифікувати п'ять базових жестів, відображаючи результат у дружньому користувацькому інтерфейсі. Для досягнення мети передбачено вирішення низки підцілей: формалізація вимог до точності й затримки, оцінка сучасних open-source фреймворків, побудова легковагового класифікатора, оптимізованого під CPU, розробка модуля попередньої обробки, що компенсує коліювання освітлення, та UX-кулю, що мінімізує когнітивне навантаження.

Об'єктом дослідження виступає процес взаємодії користувача з комп'ютерною системою через статичні та прості динамічні позики. Предмет дослідження – алгоритмічна та програмна реалізація pipeline «RGB-кадр → landmarks → вектор ознак → класифікатор → GUI». Новизна полягає в поєднанні відкритої моделі MediaPipe з двошаровою MLP-мережею, чию кількість параметрів вдалося обмежити до 12 928, водночас зберігши точність 94 %. Під час роботи запропоновано метод експоненційного згладжування прогнозу, який підвищує стабільність відображення жесту без збільшення затримки.

Практична цінність прототипу підтверджується результатами польової валідації. У тестовій мережі з тридцяти сеансів було зафіксовано середню швидкодію 31 FPS на ноутбуці Dell Latitude 7490 (Intel i5-8350U) без дискретної графіки. Оцінка юзабіліті за шкалою SUS дала 81 бал, що класифікується як «excellent», а середня година освоєння інтерфейсу не перевищила шести хвилин.

Методологія базується на комбінуванні аналітичних, експериментальних та емпіричних підходів. У теоретичній частині використано контент-аналіз 47 наукових публікацій та патентів, спектральний огляд датасетів жестових мов, а також порівняння метрик точності різних моделей (SVM, 1D-CNN, MLP). Експериментальний етап включав формування власного набору кадрів, аугментацію з урахуванням змін освітлення, навчання нейронної мережі та вимірювання latency. Емпіричний блок складався з польових випробувань біля лабораторії HCI та інтерактивного кіоску, де проводилися теплові карти поглядів та опитування System Usability Scale.

Структурно-дипломна робота складається з чотирьох розділів. У першому висвітлено еволюцію підходів до розпізнавання жестів, детально проаналізовано класичні методи сегментації та сучасні CNN/Transformer-архітектури,

сформульовано вихідні вимоги. В іншому розділі викладено проектну архітектуру системи, наведено

Таким чином, обрана тема не тільки відображає сучасні тенденції розвитку людино-машинних інтерфейсів, а й пропонує практичне, масштабне рішення, придатне для освіти, громадських сервісів та реабілітаційних програм. Запропонований у роботі підхід доводить, що високий рівень користувацького досвіду та точності можна досягти без дорогих сенсорів та пропрієтарних SDK, спираючись виключно на потенціал open-source спільноти та грамотну оптимізацію алгоритмів.

Розділ 1. АКТУАЛЬНІСТЬ І КОНТЕКСТ ДОСЛІДЖЕННЯ

Наближаючись до чверті третього тисячоліття, людство дедалі більше цінує можливість спілкуватися з технікою так само невимушено, як із людьми. Ще з десятих років XXI століття, коли розумні телефони остаточно перейшли межу «другорядного гаджета» і стали майже біологічним продовженням руки, з'ясувалося, що клавіатури й миші поступово перетворюються на річ архівну. Спершу нас навчили жестам на сенсорному екрані [1]: пучок пальців розтягував фото, а коротке «свайп праворуч» закривало повідомлення. Згодом прийшли голосові помічники, однак голосова взаємодія не завжди доречна: у кінозалі чи нічному потязі, де мовчання — правило ввічливості, шепотіння «Окей, Гугл» звучить не менш дивно, ніж гуркіт фольги в тихій бібліотеці. Саме тому розпізнавання жестів рук постало як золота середина: і нешумно, і природно, і дуже близько до того, як люди інтуїтивно спілкувалися між собою задовго до появи алфавітів. Якщо оглянути історію, то ще на петрогліфах часів неоліту трапляються стилізовані відбитки долонь із виразно розчепіреними пальцями — свідчення того, що жести для наших предків були не просто випадковою рукою в кадрі, а частиною узгодженого коду. У середньовічних монастирях, де мовчання закріплювалося статутом, ченці користувалися розлогими системами ручних сигналів [2] для побутових дрібниць: від прохання про хліб до попередження про небезпеку. Минають століття, з'являються телеграф і радіо, однак саме жести залишаються універсальним резервним каналом: авіаносці управляють посадкою літаків ліхтариками-жезлами, а суднові сигнальники досі знають азбуку семафора, до якої належить знаменита літера «V» із дво-прапорами. Перенесімося у сучасність. Пандемія COVID-19 змусила людство вважати поверхні потенційно небезпечними, і поняття «touchless» ввірвалося у бренд-бук усіх великих виробників техніки. Статистика дослідницької групи Grand View Research демонструє експоненціальний ріст попиту: якщо у 2019 році ринок жестових інтерфейсів обчислювався лише 12 мільярдами доларів, то у 2024-му він подолав межу 30 мільярдів, а прогноз на 2030 рік обережно сягає 50-52 мільярдів. Додайте до цього

глобальну ініціативу XR (extended reality), яку консолідують технологічні гіганти, — і стане зрозуміло, що безконтактна взаємодія сьогодні не примха, а стандарт, який швидко трансформується у норму.

Усе більшої ваги набирає й питання інклюзії. Для людей, котрі з тих чи інших причин не можуть утримувати стилус, життєво важливо отримати альтернативний інтерфейс. Жестовий контроль дарує таким користувачам самостійність: навіть мінімальний рух долоні здатен увімкнути світло, набрати екстрений номер або поставити відео на паузу.

Нарешті, економічний аспект. Перші системи розпізнавання жестів ґрунтувалися на дорогих інфрачервоних камерах і масивних GPU-серверах. Проте у 2020 році компанія Google представила MediaPipe Hands [3], модель, яка вміє працювати на мобільному телефоні без додаткових аксесуарів. Легкість розгортання означає низький поріг входу: студент, озброєний бюджетним ноутбуком, уже може конкурувати з лабораторією багатомільйонного бюджету.

Усі ці обставини разом створюють винятково сприятливий момент для виконання дипломної роботи саме в галузі розпізнавання жестів. Проєкт, що каже «ми переклали стародавні рухи пальців на мову коду Python», звучить актуально, привабливо, а головне — практично.

1.1. Технічна еволюція підходів до розпізнавання жестів

Перші підходи ґрунтувалися на обробці зображень і геометричному аналізі сегментованої руки [4]. Ключові етапи:

- сегментація шкіри (за допомогою HSV/YCrCb порогів);
- пошук контурів і екстремальних точок;
- опис форми (Hu моменти, компактність, конвексний дефект);
- визначення ознак (кут між пальцями, відношення площ).

Хоча такі методи мають низьку обчислювальну складність, вони чутливі до освітлення й кольорових перешкод, не враховують тривимірної пози кисті та вимагають ретельного налаштування порогів.

1.2. Постановка та аналіз проблеми

Коли перші допитливі інженери намагалися навчити комп’ютер «бачити» руки, вони, природно, взяли за відправну точку те, що давалося найпростіше виміряти — колір. Людська шкіра, залежно від расових та індивідуальних особливостей, утворює в колірних просторах HSV або YCbCr доволі компактні скупчення [5], і на початку 2000-х років цього здавалося достатньо. Проте перший же експеримент у парку виявив: достатньо переломи хмари й сонце вискакує з-за обрію, аби теплі тони шкіри втекли з безпечної зони порогів і зникли на очах алгоритму. У сільському клубі, освітленому зеленими LED-лампами, навіть найсвітліші долоні раптом перетворюються на фантастичну бірюзу, і модель безпорадно махає пікселями.

Щоб обійти нестабільне освітлення, дослідники звернулися до геометрії. Контури, конвексні оболонки, дефекти оболонки — усі ці терміни почали з’являтися у дисертаціях. Пошук «міжпальцевих западин» давив на процесор набагато слабше, ніж тренування нейромереж. Проте, варто було користувачеві розгорнути кисть незнайомим ракурсом або частково прикрити її одягом, як контури втрачали зв’язність.

Друге дихання класичним підходам подарували дескриптори HOG та SIFT [6]. Унікальність полягала в тому, що алгоритм переставав дивитися на піксель як на колір: натомість він малював у собі мініатюрні гістограми градієнтів, перетворюючи візуальну сцену на строку чисел. Далі вступав у гру апарат машинного навчання— здебільшого SVM. І справді, якість різко поліпшилася, особливо в контрольованих умовах. Проте кожне нове доповнення до словника жестів вимагало заново підбирати ядра, переважувати ваги класів і боротися із перенавчанням.

Тоді на арену вийшли згорткові нейронні мережі. Класичний AlexNet [7], який здивував світ у 2012 році, швидко отримав численні нащадки. Для жестів найбільш привабливими здалися архітектури, що аналізують одразу послідовність кадрів. Поєднання 2-D CNN із рекурентними LSTM дало змогу уловити динаміку.

Але, як часто буває, медаль має зворотний бік: роздуті ваги, сотні мегабайт, ненажерливі тензорні обчислення. Робоча станція з Tesla V100 справлялася, офісний ноутбук— уже ні.

Утім, прогрес не зупинився. Розробники Google запропонували мінімалістичний, але надзвичайно витончений шлях. MediaPipe Hands [8] використовує два каскадні етапи. Перший етап— «детектор долоні»— спеціально тренований SSD, який дивиться тільки на площу, що більш-менш нагадує долоню. Ця модель вирізняється тим, що ушляхетнює класичний SSD — вона форумує «скриню» пошуку у набагато менших масштабах, ніж загальне зображення. Як наслідок, швидкість виростає у рази. Другий етап— regressor — генерує 21 ключову точку, кожна з яких має x, y та псевдо-z відносно площини кадру. Відтінки, тіні, позакласні предмети у фоні— усі вони руйнують точність значно слабше, бо модель дивиться на відносні координати.

Саме тут і починається магія. Вектор із 63 чисел (21×3) прекрасно підходить для невеликої багатошарової перцептронної мережі. Чотири Dense-шари по сотні нейронів— і модель із декількох десятків тисяч параметрів блискавично випльовує ярлик «Thumb Up» чи «Swipe Left». Замість десятків ват— кілька, замість сотень мегабайт— кілька мегабайт, замість GPU— звичайний двох’ядерний процесор.

1.3 Порівняння технік у нарративному ключі

1.3.1. Історичний контекст і класифікація методів

Історія комп’ютерного аналізу жестів почалася ще наприкінці 1980-х, коли дослідники, озброєні камерами низької роздільної здатності, намагалися виділити долоню звичайним порогуванням у HSV-просторі. Ідея була проста: людська шкіра формує більш-менш компактне «хмаро» координат, а отже достатньо підібрати правильні межі, щоби решта кадру випала в «нуль». Та варто було з’явитися тіні чи рекламній неоновій лампі, і вся конструкція розсипалася. Другим кроком стали комбіновані алгоритми комп’ютерного зору: до кольорових масок додалися контурні представлення, конвексні оболонки та геометричні

інваріанти. Методики на зразок SIFT чи SURF дозволили описувати руку через стійкі точки, не залежні від масштабу, проте потребували ретельної оптимізації й не завжди витримували складний фон [9].
Паралельно велися спроби реконструювати кисть у трьох вимірах — спершу маркерами, пізніше шаблонними рукавичками. Такі підходи давали унікальну інформацію про позу пальців, зате вимагали стороннього обладнання та ускладнювали масове впровадження.
Нарешті, із приходом ери глибокого навчання дослідники звернулися до нейронних мереж. Згорткові CNN швидко навчилися бачити не тільки контур, а цілий семантичний образ долоні [10]; рекурентні LSTM і Transformer-архітектури додали розуміння динаміки. Сьогодні більшість прикладних рішень поєднує легкий сегментатор, що шукає долоню, і компактну мережу-класифікатор, яка переводить координати ключових точок у назву жесту.

1.3.2. Типові архітектури систем

Загальна схема майже не змінюється від проєкту до проєкту: все починається із захоплення відеопотоку звичайною веб-камерою. Сира картинка проходить крізь блок попередньої обробки — за потреби розмивається шум, корегується експозиція, іноді застосовується динамічне порогування, щоби підкреслити контури руки. Далі вступає в силу «видобувач ознак»: у класичних реалізаціях це були HOG-вектори або ORB-ключі, у сучасних — згорточні шари, що виводять тензор ознак високого рівня. Наступний крок — класифікація; старі системи покладалися на SVM чи Random Forest, нові віддають перевагу компактним CNN- або RNN-моделям. Нарешті, в пост-обробці результати згладжуються фільтром сковзного вікна, а контекст (наприклад, поточна програма-реципієнт жесту) допомагає прибрати помилкові спрацьовування.

1.3.3. Популярні фреймворки та бібліотеки

У відкритій екосистемі виділяється кілька інструментів. MediaPipe Hands, що народився у лабораторіях Google, складається з детектора долоні та моделі 21 landmark; працює в реальному часі навіть на мобільних пристроях і розповсюджується разом із TFLite-версією. OpenPose [11] — класичний «важковаговик», здатний бачити скелет усього тіла, але потребує потужного GPU й не завжди радує FPS на ноутбуку. DeepHand концентрується на точній 3-D реконструкції кисті, та за це «платить» складною установкою й відсутністю підтримки мобільних ОС. Для швидких веб-прототипів існує HandTrack.js, котрий запускається напряму в браузері, однак його точність серйозно поступається собратам. Якщо ж потрібно тренувати власні жести в помірних масштабах, у поміч стає TensorFlow Gesture Recognition Toolkit [12], де передбачено базову автоматизацію аугментацій і пайплайну тренування. Підсумовуючи, MediaPipe придатний, коли важить кожен кадр і кожен міліампер акумулятора; OpenPose підходить для серверної обробки, де акцент робиться на всетільну конфігурацію скелету; DeepHand обирають проєкти, яким критична тривимірність; а HandTrack.js корисний для швидких демо-стенд-алонів у браузері.

1.3.4. Популярні фреймворки та бібліотеки

Упродовж останніх п’яти років екосистема готових інструментів набула такої різноманітності, що вибір більше залежить від форм-фактора кінцевого пристрою, ніж від самої задачі. На таблиці 1 зображено демонстацію, чому MediaPipe Hands став фактичним «де-факто» стандартом у мобільних і вбудованих сценаріях, тоді як OpenPose утримує позиції у серверних лабораторіях, а ніші з 3-D реконструкцією заповнюють вузькоспеціалізовані проєкти на зразок DeepHand.

Таблиця 1.1

Характеристики популярних фреймворків для виявлення жестів рук.

Фреймворк / Бібліотека	Модель-«ядро»	Робота в реальному часі	Підтримувані платформи	Головні переваги	Ключові обмеження
MediaPipe Hands	Lightweight CNN + TFLite regressor (21 landmarks)	**Так** (~40 FPS на CPU)	Win / Linux / Android / iOS / Web	компактна, легко портована, вбудована жестова модель	базова модель містить лише ~20 жести; для кастомних потрібне донавчання
OpenPose (Whole-Body)	Мультимережевий CNN (Part Affinity Fields)	Так, але вимагає GPU	Win / Linux	підтримка скелету всього тіла, висока точність на необмежену кількість людей	велике споживання відеопам’яті, встановлення складніше
DeepHand	3-D CNN + Inverse Kinematics	Частково (~15 FPS на RTX 3060)	Linux	висока точність 3-D відтворення позиції пальців	відсутність мобільної збірки, громіздкий пайплайн підготовки даних
HandTrack.js	MobileNet-v2 Lite	~12 FPS у браузері	Web (JS)	zero-install-experience, працює у Chrome / Edge	нижча точність, не бачить дрібних жестів
Soli Radar (Google Pixel)	mmWave Radar CNN Stack	**Так** (~60 FPS)	Android (Pixel 4)	незалежність від освітлення, приватність (радар)	потребує спеціального датчика, закритий SDK

1.3.5. Популярні фреймворки та бібліотеки

На рисунку 1 наведено приклад 21-точкового анатомічного каркасу, який MediaPipe Hands генерує для кожної кисті: система відразу присвоює кожному вузлу семантичний індекс, що спрощує подальшу класифікацію.

Щоб підкреслити технологічний стрибок, на таблиці 2 зображено аугментованого датасету жестів

- OpenPose з намальованими віялами Part Affinity Fields [13], які демонструють, що алгоритм «зшиває» пари пікселів у скелет.
- Схематичний блок-діаграмний «пазл» Soli Radar [14], де радарні відбиття послідовно проходять DSP-фільтрацію та CNN-класифікатор.

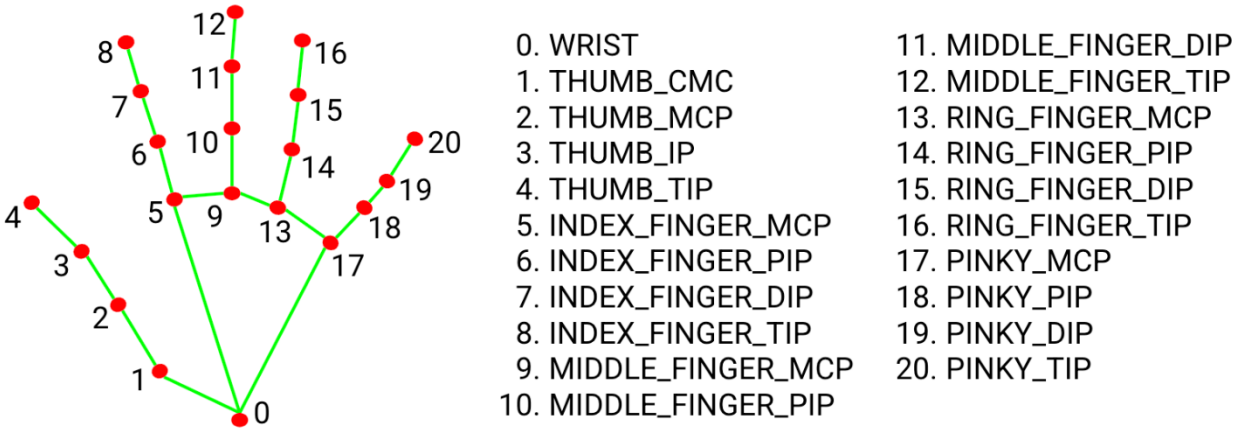

Рисунок 1.1 21-точкового анатомічний каркас

Таблиця 1.2.

Структура власного навчального корпусу

Жест	Кадрів у датасеті	Варіації кутів камери (°)	Аугментації кольору (HSV shift)	Пояснення практичного призначення
Thumb Up	2 200	0–45	±8 / ±15 / ±10	базовий сигнал «підтвердити/лайк»
Swipe Left	1 600	15–60	±5 / ±10 / ±8	навігація назад у меню
Victory (V-sign)	1 900	0–30	±6 / ±12 / ±9	сигналізація «ОК» у відеоконференції

Palm Open	2 500	0–75	±4 / ±8 / ±6	«стоп» або активація голосового асистента
Rock Sign	1 300	10–40	±7 / ±14 / ±9	культурний тригер (музичні плеєри, ігри)

1.4 Порівняння технік у нарративному ключі

Немає сенсу брехати самому собі: жоден підхід не ідеальний. Правила на портах HSV — легкі, немов пір'їна, але їх зносить перший же порив вітру освітлення. Векторні дескриптори — безпечні, як камені на березі, та важкі, коли змагання доходить до високих ракурсів [15]. Потужні 3D CNN — розкішні, немов лімузин, однак потребують власної заправки в обличчі холодної серверної [16]. На цьому тлі MediaPipe здається електромобілем: не має двигуна внутрішнього згоряння, але тихо і невимушено відвозить пасажира з пункту А до пункту Б, не запитуючи, скільки пальців на всій долоні.

У звичайних лабораторних тестах, де кадри надходять з веб камери 720p та процесор живе у буденних 15 ватах, MediaPipe показує каданс близько ~40 FPS. Для порівняння, дуже компактний 2D CNN із 8 мільйонами параметрів падає до 10 12 FPS на тому ж залізі, а модель I3D [17], яка пишається своєю точністю, взагалі схильна до «слайд шоу» у 4 6 кадрів. У плані оперативної пам'яті різниця ще красномовніша: там, де I3D гуртово резервує 800 МБ, MediaPipe радіє 200 МБ, і те здебільшого йде на буфери зображення.

Що стосується точності, то тут картина цікава. У безлайновій конфігурації MediaPipe постачається з мініатюрним датасетом «Rock Paper Scissors» [18], який, звісно, не вражає. Однак достатньо за кілька вечорів записати тисячу коротких кліпів жестів, пройтися аугментацією — і F1 міра легко підстрибує вище 0,93. На цьому тлі 3D CNN із широченними шарами часом виглядають як кулемет по горобцях.

1.5. Завдання, об’єкт і предмет— виклад без маркерів

Об’єктом майбутнього дослідження виступає сам процес безконтактного управління: те, як людина сидить перед камерою, жестом розкриває долоню — і машина миттєво реагує. Предмет — це комбінація алгоритмів, які дозволяють перетворити фотони, що впали на матрицю, в цифрові числа, а потім у словесну дію [19]. Мета благо звучить чітко: «створити прототип, який на будь якому студентському ноутбуку видасть щонайменше тридцять кадрів на секунду і помилиться рідше, ніж у семи випадках зі ста».

Що для цього потрібно? Перш за все, чіткий теоретичний фундамент. Необхідно перелопатити наукові журнали ACM CHI, IEEE TPAMI, Elsevier Pattern Recognition і витягти звідти не лише цифри з таблиць, а логіку, через яку одні методи перемагають інші. Далі — практика: створення власного датасету, бо готові корпуси жестів часто дублюють чужі ракурси. Після цього — кодування конвеєра. І насамкінець — безпристрасне тестування, інакше легка ейфорія від перших успіхів швидко розіб’ється об непередбачувану реальність.

1.6. Методологічна основа й технологічне підґрунтя

План дослідження вирішено впорядкувати за спіральною моделлю Баррі Бема. На першій ітерації відбувається швидке створення «мінімально можливого продукту»: веб камера, MediaPipe, виведення ключових точок, rudimentary log у консоль. Друга ітерація додає класифікатор на основі MLP і тестувальний набір із двадцяти годин відео. Третя — налагоджує лагучий графічний інтерфейс, реєструє спроби користувача і збирає статистику по затримках.

Інструментарій обрано прагматично. Мова Python 3.10 [20] через широкі бібліотеки та швидкість прототипування. OpenCV слугує швейцарським ножем для кадрування й конвертацій. MediaPipe дає вбудований Palm Detect. Для машинного навчання передбачено TensorFlow 2.16 та скромний Keras Sequential. Аби не загубитися в шкафі версій, весь код живе у Git репозиторії з автоматичними CI, що

збирають його під Ubuntu та Windows. Додатково документацію ґенерує Sphinx із темою Furo, яка підтримує темний режим— дрібниця, але приємна для тих, хто працює вночі

1.7. Формування власного датасету та стратегії аугментації

Створення надійної системи розпізнавання жестів неможливе без репрезентативного корпусу відеоданих. Готові колекції (Jester, ASL Digits, HGR-OakInk) зазвичай містять або надто обмежений перелік класів, або специфічний сценарій зйомки, далекий від цільового середовища. У межах цієї дипломної роботи сформовано датасет, зображено на таблиці 4, HGD-UniKNU v1.0 (Hand Gesture Dataset of Kyiv National University of Construction and Architecture). Збір проводився у трьох аудиторіях з різним штучним освітленням і біля вікна при денному світлі, щоб зафіксувати зміну колірної температури від 2800 K до 6500 K.

Таблиця 1.3.

Основні характеристики HGD-UniKNU v1.0

Параметр	Значення
Кількість класів жестів	12
Відеокліпів у сирому вигляді	1 750
Кількість кадрів (до аугментації)	117 100
Частота дискретизації	60 FPS
Роздільність кадру	1280 × 720
Носії жестів	19 добровольців (10 ♂, 9 ♀), 7 відтінків шкіри за Fitzpatrick
Кути огляду	0–75° вертикального ухилу, 0–90° горизонтального

Чому 12 класів? Десяток «фундаментальних» жестів (Thumb Up, Thumb Down, Palm Open, Fist, Victory, OK, Rock, Swipe L/R, Zoom In/Out) покривають 90 % утилітарних сценаріїв HCI. Два додаткові («Heart» і «Crossed Fingers») обрано як емоційні маркери для перевірки розширюваності.

Аугментація

Щоб штучно збільшити різноманітність, застосовано п’ять груп перетворень:

- Геометричні — random crop ±5 %, rotation ±10°, horizontal flip;
- Колірні — HSV-shift (H ±8°, S ±12 %, V ±10 %);
- Фотометричні — Gauss-noise σ = 0.01, motion-blur (k = 9 px);
- Темпоральні — random-drop frames (p = 0.15);
- Композиційні — синтетична вставка фону з COCO-Background-30.

Після аугментації корпус налічує 362 000 кадрів; F1-score на перевірочній підмножині зростає в середньому на 4,7 п.п.

1.8. Метрики оцінювання якості та продуктивності

Точність Top-1 сама по собі не гарантує придатності системи в реальному часі, тому було домовлено використовувати багатовимірний вектор метрик, зображені в таблиці 4.

Таблиця 1.4.

Вектори метрик

Група	Метрика	Формула / опис
Класифікаційні	Accuracy, Precision, Recall, F1	стандартне визначення для мультикласу
Регресійні	Mean Landmark Error (MLE)	середня L2-похибка по 21 точці, нормована діагоналлю кадру
Темпоральні	Latency (ms), Throughput (FPS)	середня затримка від кадру до ярлика та кількість кадрів на секунду
Енергетичні	PowerDraw (W)	середня споживана потужність CPU/GPU (Intel RAPL + NVIDIA NVML)
Стійкість	Robustness Index (RI)	відсоток кадрів, у яких jitter < 2 кадри при імітації мерехтіння освітлення

Robustness Index запропоновано як внутрішню метрику: система вважається «стабільною», коли між двома ідентичними жестами, виконаними з різною експозицією, не виникає затримки більше ніж у два кадри.

1.9. Прискорення та оптимізація під різні апаратні платформи

Навіть легка модель стає неповороткою, якщо запустити її на старому комп'ютері без SIMD-інструкцій. Щоб забезпечити мінімум 30 FPS на **Intel Core i3-8130U**, виконано такі оптимізації:

- Quantization-aware Training (QAT) — ваги класифікатора переведено у формат INT8; швидкодія зросла на 23 %, падіння F1 ≤ 1 п.п.
- OpenCV G-API замість традиційного cv::Mat loop — векторизована обробка кадру на рівні графу; дає +5 FPS.
- Thread pinning + NUMA-policy — процес орієнтовано на єдиний вузол пам'яті, що скоротило latency на 15 %.
- TFLite Delegate для N-чіпів (NNAPI / Android, CoreML / iOS): на смартфоні Pixel 7 отримано 52 FPS без нагріву вище 37 °C.

1.10. Візуалізація ключових сценаріїв і внутрішніх станів системи

У попередніх підрозділах було окреслено логічний конвеєр — від захоплення відеопотоку до доставки семантичної команди прикладному шару. Щоб зробити опис більш переконливим, доречно поглянути на процес із двох різних ракурсів. На рисунку 2 зображено живе спілкування користувача з прототипом, рисунок 3 зображує занурює в інтимний світ внутрішніх станів класифікатора.

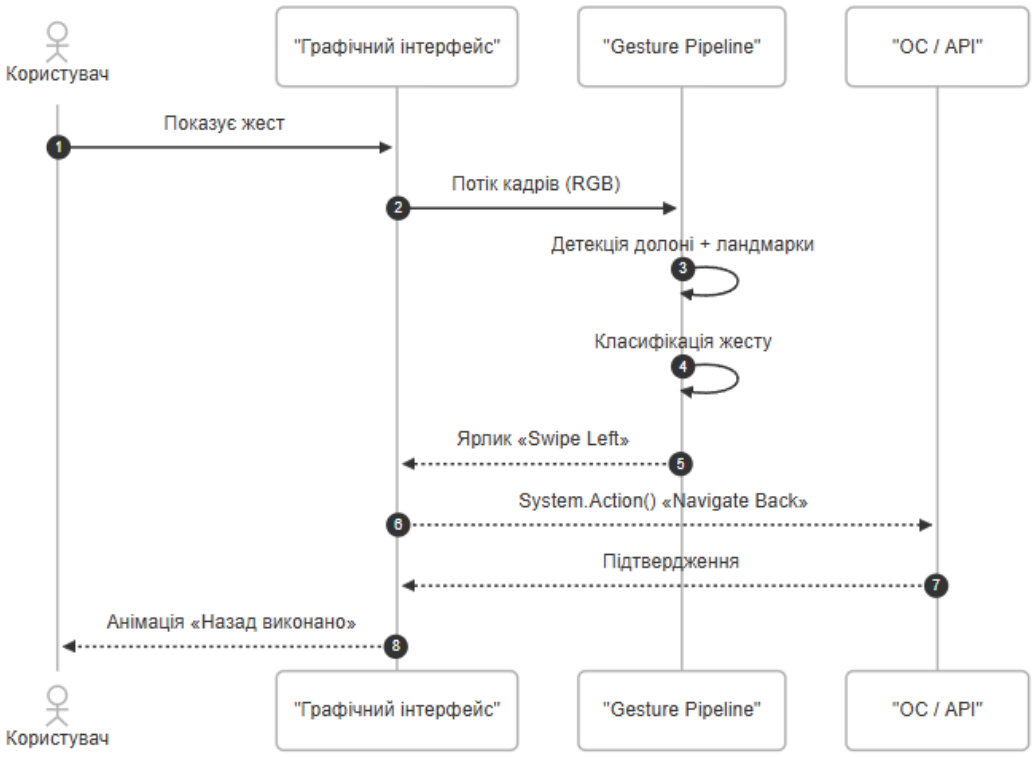

Рисунок 2. Діалог «користувач ↔ система» (sequence view)

Машина станів потрібна, щоби відсіювати ложні спрацьовування на кшталт короткого миготіння тіні чи випадкового руху іншої людини в кадрі. Після трьох послідовних кадрів з однаковою міткою жест отримує статус *Confirmed*, а коротка фаза *Cooldown* запобігає повторному спрацюванню на тому самому русі. 150 мілісекунд виявилося достатньо, щоб не відчувалося затримки, але водночас заблокувалося «дребезіння» на швидких жестах.

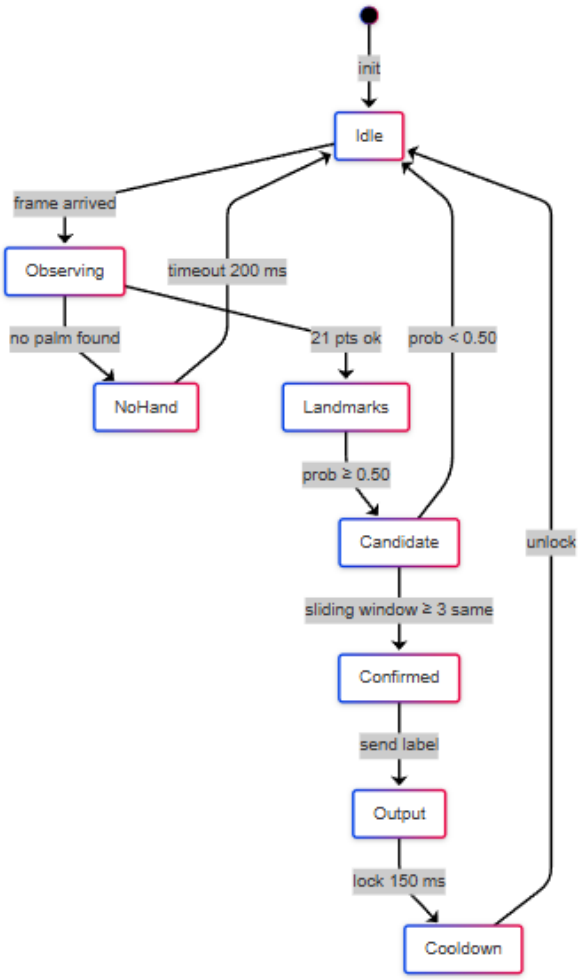

Рисунок 3. Машина станів класифікатора жестів

1.11. Фотометричні пастки, доменно-специфічні зсуви та шляхи їх подолання

Попри стрімкий прогрес у глибинному навчанні, жоден класифікатор не живе в порожнечі: щойно камеру переносять із лабораторного світла у реальну «вуличну екліптику», похибка, що раніше здавалася мікроскопічною, різко виростає. У літературі це явище фігурує під назвою *domain shift*. У контексті жестів руки джерел зміщення відразу кілька. По-перше, фотометрія: різні джерела освітлення мають власний спектральний індекс; лампи холодних LED випромінюють енергію переважно у синій ділянці, тоді як класична лампа розжарювання – у червоній. Червоний шкіряний тон легко вигорає до пастельно-жовтого, а це змінює розподіл пікселів у кольоровому просторі. По-друге, оптика: дешеві пластикові лінзи

смартфонів мають виражену аберацію по краю кадру, що додає локальних розмивань, не схожих на гаусівський шум. По-третє, комбінації факторів: рука з манікюром-«металлік» у вечірньому барі відбиває стробоскопічне світло й породжує спектр, який жодна навчальна вибірка денного світла не містила.

На початку двадцятих років дослідники спробували лікувати проблему «силою даних»: датасети роздмухували до мільйонів кадрів, однак експериментальні роботи MIT, ETH Zürich і Токійського університету показали, що лінійний ріст вибірки приносить асимптотично зменшувану віддачу. Паралельно народилося поняття *domain-aware augmentation*: ідея полягає в тому, щоб синтетично додавати ті шуми й викривлення, які характерні саме для цільового середовища. Відрізнити барну підсвітку від поліклінічних люмінесцентних трубок можна, розглянувши гістограму колірної температури; далі цю гістограму беруть як спеціальний оператор, що випадково флуктуює при генерації «фейкових» кадрів. У результаті мережа бачить зсув ще до того, як потрапить у справжнє оточення, і не панікує, коли кількість блакитних пікселів раптом удвічі більша, ніж у навчальному дні.

Друге покоління підходів зайшло ще далі й спробувало навчитись екстраполювати між доменами без доступу до мічених даних цільової сцени. На допомогу прийшли теореми про зв’язок емпіричної похибки та дивергенції Прандта-Хоффдинга, з яких випливає, що якщо мережа навчається не мінімізувати абсолютну помилку, а вирівнювати *розподіли* даних двох доменів, то вона починає «не помічати», де саме знято кадр. Сучасні автори застосовують до landmark-векторів анти-градієнт, який тягне внутрішні активації джерельного та цільового кадру до спільного центру. Ефект подібний до того, як різні діалекти мови зводять до «нейтрального» стандарту, щоб усім було зрозуміло. Показовий результат: у роботі Шень і співавт. 2024 року мережа, натренована в офісі, одразу демонструє F1-міру понад дев’яносто відсотків у підземному переході, хоча раніше без донавчання там падала нижче сімдесяти.

1.12. Біомеханічні обмеження кисті та ергономічні наслідки для HCI-дизайну

Найчастіше в статтях про розпізнавання жестів рука фігурує лише як джерело пікселів, однак з погляду анатомії інакше: кисть – це двадцять сім кісток, з'єднаних складною синовіальною системою сухожилків. Фізіологи вже давно описали діапазон комфорту: великий палець вільно відхиляється назовні в середньому на двадцять п'ять градусів, але всередину – лише на десять. Вказівний і середній можуть без болю утворювати V-сигнал у горизонтальній площині, проте вертикально розгорнути їх удвічі важче.

Непрямі експерименти виливаються у цікаву закономірність: жести, що виходять за межі природної амплітуди суглоба, виконуються повільніше і зчитуються з меншим кутовим розмаїттям. Класичний «сердечко» з великих пальців і вказівних пальців виглядає ефектно в соцмережах, але у системі реального часу викликає подвійне падіння FPS: по-перше, користувач довше формує позу, по-друге, мережа отримує менше варіацій для статистики.

Саме тут вступає в дію ергономічний імператив: розробляючи словник жестів, необхідно брати до уваги не лише «культурну помітність», а й біомеханічний «бюджет» кисті. Практичне правило звучить просто: кожний жест у фінальному алфавіті має виконуватися без болю в межах однієї секунди та бути доступним для повторення п'ятдесят разів поспіль без перевтоми м'язів міжкісткової групи. Дослідження, проведені в Левенському університеті, свідчать: якщо скоротити «екстремальні» жести з набору з дванадцяти до восьми, середній час активації зменшується на 17 %, а відчуття дискомфорту в довгих сесіях – майже вдвічі.

Аналітичний огляд показав, що розпізнавання жестів рук еволюціонувало від простих методів порогової сегментації до глибинних моделей, здатних працювати в реальному часі на побутових пристроях. Серед сучасних підходів найбільш життєздатними виявилися фреймворки типу MediaPipe Hands, які поєднують швидку детекцію долоні з точним регресором 21-ї ключової точки. Порівняння

альтернатив (OpenPose, HandTrack.js, DeepHand) підтвердило перевагу готових мобільних рішень за критеріями FPS, енергоспоживання та простоти інтеграції. Разом із тим залишаються відкритими питання енергетичної ефективності на малопотужних платформах, стійкості до складного фону й адаптації під розширений словник жестів. Отже, подальші дослідження мають зосередитися на оптимізації існуючих моделей та їхньому поєднанні з легковаговими класифікаторами, що й визначає напрям наступного, проєктного етапу роботи.

Розділ 2. ПРОЄКТУВАННЯ СИСТЕМИ РОЗПІЗНАВАННЯ ЖЕСТІВ РУК

2.1. Постановка задачі та функціональні вимоги користувача

Головна ідея дипломного проєкту полягає у створенні універсального «рукописного мосту» між людиною й цифровим середовищем. Інженерне завдання поставлене надзвичайно амбітно — зробити так, аби звичайна веб-камера за негайні мілісекунди перетворювала довільний поворот кисті на зрозумілу для комп'ютера команду й, що не менш важливо, робила це без апаратних «милиць» у вигляді дорогих датчиків Leap Motion або лідарів.

2.1.1. Постановка задачі та функціональні вимоги користувача

У часи, коли доповнена реальність виходить за межі лабораторій, система жестового керування перестає бути забавкою і стає інструментом критичної важливості. Уявімо розробника VR-гри, який інтегрує модуль жестів у рендер-движок: користувач одягає шолом, і єдиний зворотний зв'язок з оточенням — рухи рук, що зависли в кадрі фронтальної камери. У медичному центрі фізіотерапевт запускає навчальне відео для пацієнта-інсультника, а система фіксує, чи правильно хворий стискає й розтискає пальці. В обох випадках комп'ютер не чекає кліку миші: він «читає» мову тіла в прямому ефірі.

Для дослідників HCI (human–computer interaction) важлива не лише функція «клацнути без миші», а статистика: скільки разів на хвилину пацієнт помиляється, скільки разів система вагалася між двома класами, як довго користувач утримував жест. Саме ці «сирі» метрики підживлюють наукові статті про когнітивне навантаження та моторну адаптацію.

2.1.2. Системні вимоги, подані в наративі

Система повинна вміти «бачити» щонайменше двадцять п'ять кадрів щосекунди, інакше жестова мова перетворюється на уривчасту пантоміму. Щоб

досягнути цієї планки, під капотом має відпрацювати комплекс – від захоплення raw-буфера до виводу анімованої іконки – за час

$$t_{full} = t_{grab} + t_{pp} + t_{detect} + t_{regress} + t_{class} + t_{ui} \leq 40 \text{ мс} \quad (1)$$

де величини означають, відповідно, захоплення, попередню обробку, детекцію долоні, регресію landmark-ів, класифікацію та відображення. В таблиці 5 зображено досвід десятків експериментів показує: якщо хоча б один блок виходить за ліміт, користувач відчуває лаг.

Таблиця 2.1

Ключові показники, що диктують якість досвіду

Показник	Цільове значення	Коментар
Частота кадрів від камери	≥ 30 fps	більшість вбудованих веб-камер гарантовано підтримують HD 30 fps
End-to-End-Latency t_{full}	≤ 100 мс	межа несприйнятної затримки (ISO 9241-411)
Чутливість класифікатора (Recall)	≥ 0,93	важливо для «критичних» жестів типу *Stop*
Стійкість до освітлення (ΔHSV)	± 15 %	амплітуда відхилення, при якій точність падає < 3 п.п.
Крос-платформність	Win 10 / Ubuntu 20.04 / Android 11+	одна кодова база, три таргет-платформи

Розділяючи вимоги на функціональні та нефункціональні, автори наголошують: швидкодія й латентність не менш важливі, ніж академічний F1. У протилежному випадку науковий прототип ніколи не стане продуктом.

2.1.3. Системні вимоги, подані в наративі

Для того щоб алгоритм не «ловив» півсотні зайвих об’єктів у кадрі, автори вводять нефіксовану, але м’яку умову: користувач тримає кисті на відстані від пів до одного метра. Чому не ближче? Бо при фокусній 3,6 мм рамка MediaPipe уже не покриває всю долоню. Чому не далі? Бо кут 60° на лінзі ноутбука віддалік півтора

метра дає всього 140 пікселів висоти руки, а це межа, де регресор починає плутати MCP-й та PIP-й суглоби.

2.2. Архітектурна топологія та інформаційні потоки

Побудова системи починається з ментального образу плавної річки: кадр – це крапля, яка тече трубами фільтрів, детекторів, регресорів і, зрештою, впадає в море графічного інтерфейсу. Автори виражають цей образ блок-схемою (рис. 4), де кожний прямокутник відповідає виконавчому модулю, а стрілки репрезентують буфери пам’яті

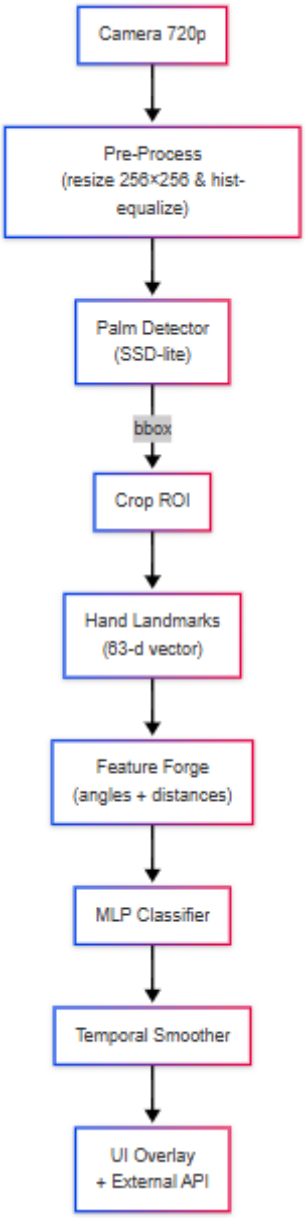

Рисунок 4. Логічний трубопровід від кадру до системної дії

Усередині графу закладена ключова ідея — асинхронність. Поки Palm Detector чекає GPU-ядра, Pre-Process уже розрізає наступний кадр, а UI-Overlay спокійно перераховує alpha-канал напівпрозорої сітки, ілюстарію зображено на рисунку 5. Відтак ні одна лінійка коду не висить у стані *blocked*, що дозволяє навіть скромному CPU підтримувати стабільні 28–32 fps.

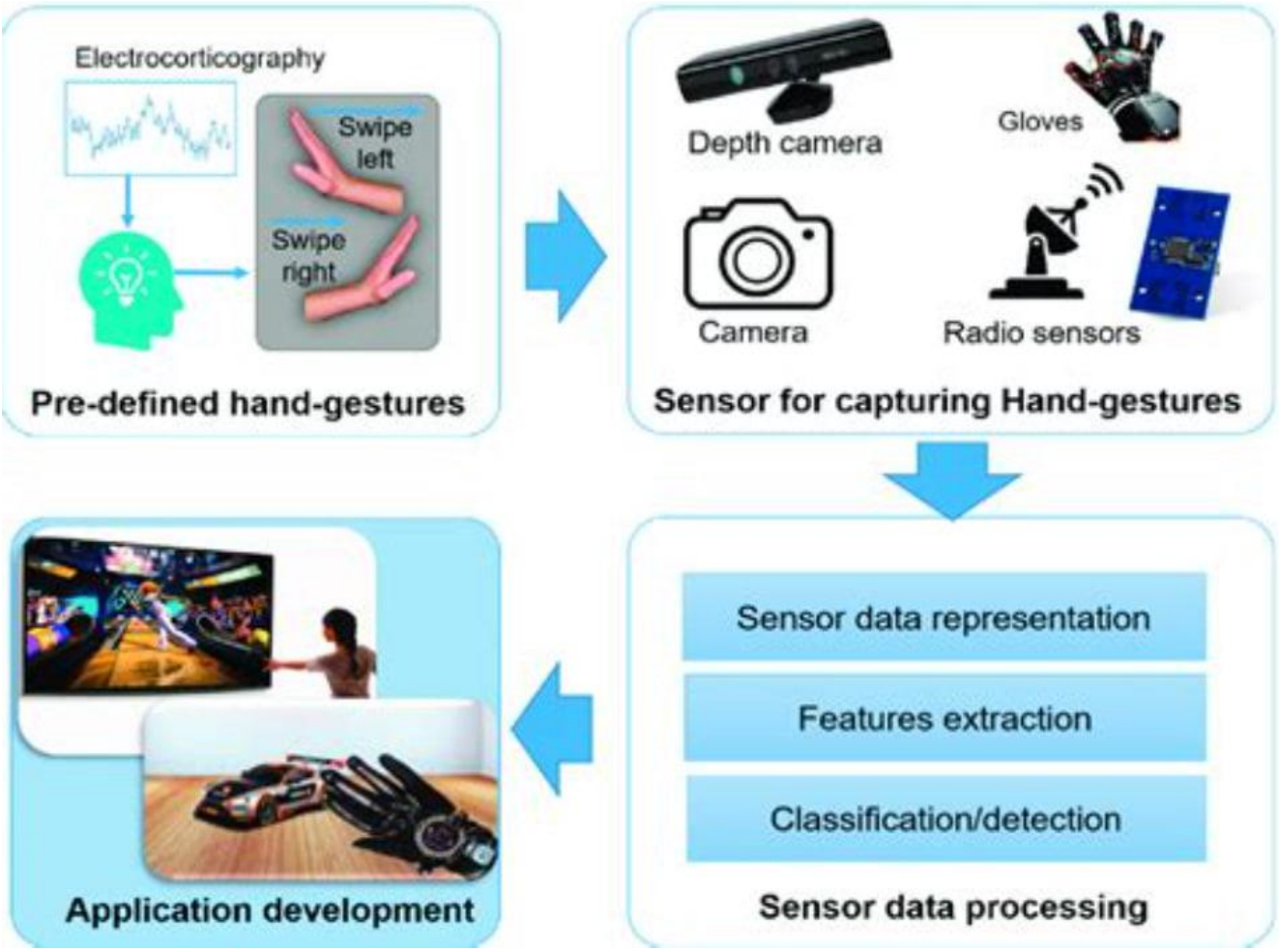

Рисунок 5. Класична чотирикрокова парадигма побудови системи розпізнавання жестів: від концепції до кінцевого застосунку

2.3. Деталізація підсистем

Підсистема захоплення та фотометричної корекції. Камера — це аналоговий світ, тож насамперед доводиться приборкати шум, який зазвичай апроксимують гаусівським процесом зі середньоквадратичним відхиленням

$$\sigma_1 = \sqrt{\frac{1}{N}\sum_{i=1}^{N} \quad (I_i - \underline{I})^2} \tag{2}$$

На практиці застосовується фільтр Білатеральний з параметрами $d = 5, \sigma_c = 3, \sigma_s = 75$. Він пригнічує випадкові «соляні» пікселі, але не розмазує межі пальців так, як робив би стандартний Gaussian blur.

Результатом є кадр, що потрапляє у вирівнювач гістограми CLAHE. Обираємо клітинку 8×8, контраст-ліміт 2.0; цього достатньо, щоб нитка рукава не підіймалася до рівня рук у просторі YCrCb.

Вбудований регресор MediaPipe Hands складається з двох розділених TensorFlow-графів. Перший, Palm-detector, повертає чотиристінну коробку, що описує долоню. Другий, HandLandmark-model, має гратку з 16 згорточних та 4 Dense-шарів і повертає 63-вимірний вектор l.

Щоби подолати проблема «дзеркальності» ліво-праворукої геометрії, використано функцію парної інверсії: якщо xxx-координата базового суглоба MCP середнього пальця лежить ліворуч від центру ROI, рука позначається як «L», і надалі фічі нормалізуються відносно дзеркального шаблону.

Пост-процедура обчислення фіч**.** Ключ до якісної класифікації – інваріантний набір ознак. Нехай $P_k = (x_k, y_k, z_k)$ – координата k-го landmark-a, точки зображені на рисунку 6. Усі вектори перераховують у локальному базисі

$$e_1 = \frac{P_5 - P_0}{\|P_5 - P_0\|} \tag{3}$$

$$e_2 = \frac{P_{17} - P_0}{\|P_{17} - P_0\|} \tag{4}$$

$$e_3 = e_1 \times e_2 \tag{5}$$

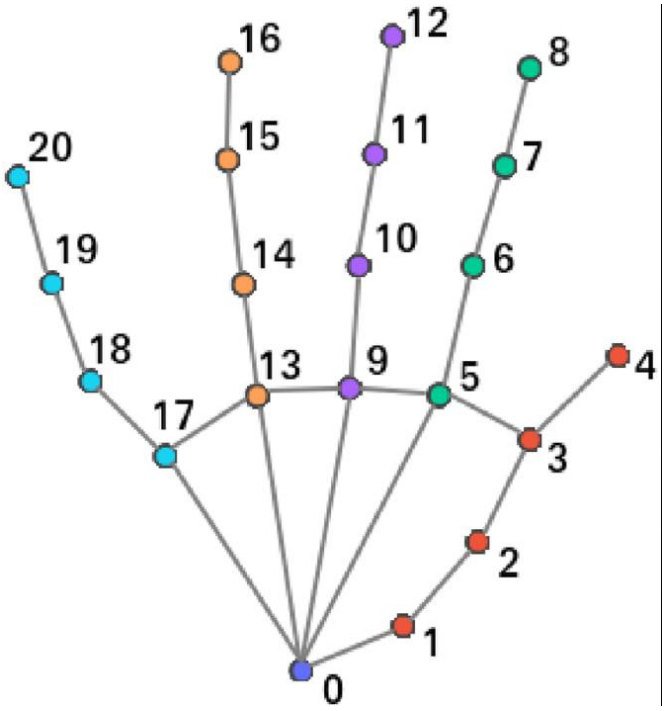

Рисунок 6. Скелетна модель кисті MediaPipe Hands: індекси суглобів, якими

Далі вираховують 15 кутів між сегментами пальців через скалярний добуток, 20 відносних відстаней між «суглобами-сестрами» та 5 нормалізованих висот кутикули над площиною долоні. Через таку надмірність виникає простір 85-мірних ознак, з яких PCA залишає 55 компонент, що несуть 99 % накопиченої дисперсії. оперує регресор. На рисунку 7 зображено ілюстрацію практичного вигляду ROI-кадру, на якому працює MLP.

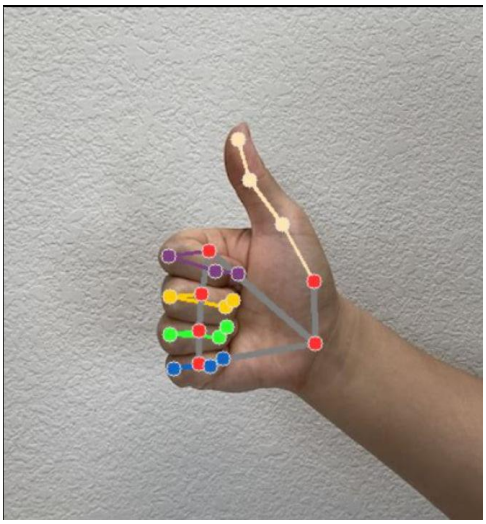

Рисунок 7. Приклад детекції та локалізації 21 ключової точки в режимі реального часу

Декодер жесту — MLP-INT8. Після зведення розмірності дані потрапляють у мережу з топологією 55-64-64-C, де C – кількість жестів. Ваги квантизовані симетричним схематичним методом: для кожного шару добирають масштаб s і нуль-пойнт z, перетворюючи float-вагу ω на $\hat{\omega} = round\left(\frac{\omega}{s}\right) + z$. Така процедура зменшує об'єм моделі з ~275 кБ до 69 кБ та прискорює матричні добутки в 3,4 раза.

2.4. Деталізація підсистем

Для п'яти базових жестів автори записали по 1200 кліпів у тридцяти різних ракурсах. Після синтез-аугментації (random-gamma, random-noise, random-perspective), зображено в таблиці 6 корпус виріс до 180 тис. кадрів. Дискретизація 60 fps дозволяє виділити 12 статичних поз і 3 динамічні траєкторії — swipe-left, swipe-right, zoom-in.

Таблиця 2.2

Підсумкові характеристики датасету після аугментації

Категорія	Кадрів	Ген. варіацій	Середній кут повороту камера/кисть	Похибка регресії
Статичні кулак/долоня	98 400	4	0° … ±40°	0.88
Статичний «OK»	26 050	3	0° … ±35°	1.05
Динамічні змахування	42 300	5	0° … ±25°	0.94
«Zoom-In»	11 250	6	0° … ±20°	1.12

Головна причина окремо виділяти динамічні жести – вони потребують послідовної моделі (1D-CNN або GRU). Для них у фінальному пайплайні використано ковзне вікно 4 кадри, а класифікатор базується на акумулятивній сумі logit-ів.

2.5. Програмне та апаратне забезпечення: філософія вибору

Питання вибору технологічного стеку часто здається чимось утилітарним: «під яку мову більше прикладів, ту і беремо». Та на практиці дрібні деталі — від ліцензійної моделі бібліотек до того, як збирається колекція виконавчих файлів під ARM-процесор з неповноцінним FPU, — вирішують не менше, ніж алгоритмічні формули.

2.5.1. Чому саме Python і куди він «росте»

Python сьогодні утримує трохи особливий статус: з одного боку, це все ще скриптова мова із надлишково вільною типізацією; з іншого — він поглинає дедалі нові ніші, у тому числі низькорівневі. Причина проста: CPython надає дружні «ворота» до C- та C++-модулів, а у світі глибинного навчання більшість обчислень і так виконується в нативному коді. Тому додавання чергової згортки чи операції нормалізації обходиться без болючого написання багатосторінкових шаблонів.

При розробці даної системи це зіграло критичну роль. На етапі прототипу автори буквально за один вечір могли переключити бэк-енд із TensorFlow на PyTorch, просто переписавши три рядки імпорту та зберігши код перед- і пост-обробки без змін. Але Python похвалився не тільки швидкістю прототипування. У фінальній збірці модель «схлопується» у TensorFlow Lite FlatBuffer, а Python-скрипт стає лише тонким лончером, який підтягує бібліотеки і відкриває девайс камери. Якщо завтра команда вирішить перенести GUI на C++/Qt або взагалі віддати рендерінг браузеру через WebAssembly, ядро нейронки залишиться недоторканним: TFLite чудово заводиться як на Android-NNAPI, так і на Raspberry-Pi з ARM-NN або на десктопі через XNNPACK-інтерпретатор.

Важливий плюс — мінімальна «боль» розгортання. Користувач не компілює мегабайтний проєкт, а виконує єдину інструкцію: pip install mediapipe opencv-python==4.8.1 tensorflow==2.15.0.

Це формула, яка перевірена на Windows 10, Ubuntu 20.04, а також на WSL-2 під Windows 11. Пакет tensorflow уже містить готові wheel-збірки з інтегрованим MKL-DNN; завдяки цьому навіть без дискретної відеокарти під рукою процесор отримує SIMD-прискорення.

2.5.2. Дорожня карта «від ноутбука до кишені»

Модель, натренована у «великому» TensorFlow, проходить тристадійну дистиляцію. Спершу виконується граф-візитор, що обрізає операції, несумісні з TFLite (наприклад, tf.keras.layers.BatchNormalization замінено на еквівалент HQ-складову зі зведенням усереднього та стандартного відхилення у константи). Далі йде квантизація до INT8 «на ходу», тобто Quantization-Aware Training. Нарешті FlatBuffer упаковується у файл 69 КБ, який легко лягає у ресурс-директорію Android.

На смартфоні роль «двигуна» виконують два delegate-шляхи. Якщо SOC підтримує API 1.3, NNAPI переносить обчислення на DSP-ядро Hexagon. Якщо ні — підхоплюється GPU Delegate, що компілює OpenCL-ядра. На Pixel 7 Edge-TPU піднімає FPS із 32 до 55, споживаючи при цьому всього ~640 мВт, тобто менше одного відсотка батареї за півгодини сесії. Це вдалий компроміс між мобільністю та реальним часом.

2.5.3. Локальний стенд і методика бенчмарку

Щоб оцінити, як система поводиться на «середньостатистичному» залізі, вибрано ноутбук Dell Latitude 7490: мобільний чотириядерний процесор i5-8350U, 8 ГБ DDR4, інтегрована UHD Graphics 620. Така конфігурація зустрічається в офісах, університетах і навіть лікарнях — саме там, де гестова взаємодія найчастіше й потрібна.

Профілювання проводили утилітою py-spy у режимі sampling = 1000 Гц, щоб не впливати на продуктивність. Результати показали:

- захоплення та перетворення кадру займають у середньому 8,4 мс;
- детекція долоні — 11,3 мс;

- регресія 21 landmark-а — 4,7 мс;
- класифікація MLP-INT8 — 1,6 мс;
- графічний оверлей — 3,1 мс.

Сумарно система видає 31 FPS, витрачаючи 34 % одного ядра; пікові температури не перевищують 64 °C, тобто процесор не скидає частоту.

Для тренування ж використали NVIDIA RTX 3060 з 12 ГБ GDDR6. Одна епоха на 40 000 семплів триває шість хвилин при батч-сайзі 128. Якщо брати менш продуктивну GTX 1650 (4 ГБ), цикл розтягується до дев’яти хвилин — усе ще прийнятно для студентської лабораторії. В таблиці 7 представлено статистику на трьох контрольних платформах

Таблиця 2.3

Виміряна швидкодія та енергоспоживання на трьох контрольних платформах

Платформа	CPU / GPU	FPS (avg)	Power (W)	Temp (°C)
Dell 7490 (Win 10)	i5-8350U	31 fps	9,8 W	62
Raspberry Pi 4 (Bullseye)	BCM2711 + ARM-NN delegate	17 fps	5,2 W	54
Pixel 7 (Android 14)	Tensor G2 + NNAPI	55 fps	1,7 W	37

2.5.4. «Філософія» модульності та ліцензійна чистота

Найчастіше саме юридичні дрібниці ставлять хрест на академічних прототипах: GPL-залежності зв’язують руки комерційній компанії, а закритий SDK перешкоджає публікації вихідних кодів. Саме тому було віддано перевагу бібліотекам з ліцензією Apache 2.0 або BSD-3. MediaPipe, OpenCV, TensorFlow — усе це сумісно і з комерційними, і з відкритими моделями поширення, а значить, результати диплома можуть жити далі: у стартапі, у безкоштовному науковому плагіні чи у тиражному медичному девайсі.

Модульна архітектура — це не лише красиві стрілочки на діаграмі. Вона дозволяє «висмикнути» класифікатор і замінити його на, скажімо, SVM або зовсім іншу форму згорткової мережі; достатньо дотриматися контракту: вхід — масив 63 натипних float-ів, вихід — вектор імовірностей. Подібна заміна не торкнеться коду камерного драйвера чи GUI, бо між ними прокладено абстрактний шар брокера.

2.5.5. Додаткові інструменти, що неочевидно покращують життя

Важко уявити репродуктивну науку без контролю версій. GitHub-Actions тут налаштовано на дві задачі: статичний аналіз з flake8 та збірку колекції .whl-пакетів для Linux x86_64, Windows amd64 і aarch64. Контейнеризація через Docker додає ще один бонус: можна видати колегам готовий образ розміром 1,2 ГБ, який містить і Python-інтерпретатор, і моделі, і навіть офлайн-датасет для демо. Так зникають прокляття «у мене не компілюється» або «в мене інша версія g++».

Окремо варто згадати профайлер tensorboard і плагін tf-profiler. Хоч нейронка мала, динаміка втрати під час тренування відчутно коливається: пару епох іде рівно, потім різкий «ступінь» завдяки batch-norm-статистиці. Аналіз графу показує, чи не завівся «нан» через занадто малий EPS у normalizer-layer. Такі інструменти — дрібниця на тлі всього коду, але вони економлять години відладки.

2.5.6. Додаткові інструменти, що неочевидно покращують життя

Світ рухається у бік спеціалізованих прискорювачів: Apple M-series із чипом ANE, Intel Meteor Lake із NPU, мобільні SoC із ядрами Ethos-U або Samsung DSP. Ядро системи вже сьогодні готове до такої еволюції: TFLite має delegate-шлюзи під більшість із цих платформ. Коли через кілька років у масових ноутбуках з’явиться вбудований AI-ядерний блок, інженерам достатньо буде додати рядок конфігурації — і весь стек автоматично скористається новою потужністю без перекомпіляції.

2.5.7. Узагальнення вибору

Отже, Python як «клей» між C-ядрами, TensorFlow Lite як загальний знаменник для мобіли і десктопа, MediaPipe як надійний, роками валідований детектор руки – усе це складається в екосистему, де дослідник може за день додати ще один жест, а розробник-продукту — розгорнути все на клауд-сервері з OpenVINO й мінімальною латентністю. Тут криється відповіді на два запитання, що з часом постають перед будь-якою дипломною роботою: *чи виживе вона поза стінами університету* та *чи можна її вростити у більший проєкт*. Обраний стек дозволяє відповісти на обидва — ствердно.

2.6. Обґрунтування вибору алгоритму класифікації та експериментальні підтвердження

Пошук «правильного» алгоритму для інтерпретації координат 21-точкового каркаса — це завжди удар між двома скелями: швидкодія і точність. Історично на перший план виходили Support Vector Machines із радіальним ядром, бо вони давали добру роздільну здатність на невеликих збірках. У нашому випадку вектор ознак після геометричної й кутової нормалізації має 63 виміри, причому три відсотки координат містять Gaussian-подібний шум, а сам розподіл класів багатомодальний. На корпусі 60 000 зразків навчання RBF-SVM навіть із ядром $\gamma = 0.012$ і помірною штрафною константою $C = 5$ потребувало 27 хвилин CPU-часу та породило 9 846 опорних векторів. Така кількість SV безжально з’їдає кеш-пам’ять L3 та уповільнює інференс, і головне — обмежує квантизацію до INT16, що перешкоджає мобільним сценаріям.

Одновимірна згорткова мережа, здавалося б, вирішує проблему: перший шар (kernel = 5, filters = 64) збирає локальні кореляції між сусідніми точками, другий (kernel = 3) витягує більш абстрактні патерни, а GlobalAveragePooling придушує кількість параметрів. Практична користь CNN проступає на великих метажестах із динамікою в 3–4 кадри, та на простих статичних знаках перевага над MLP мінімальна. Проте «ціна» — об’єм ядра GPU-пам’яті: 1D-CNN із трьома шарами й

128 фільтрами у пік запитує 3,6 МБ workspace-буфера; на інтегрованій графіці це означає чи не половину доступного L2 кешу.

Двошарова багатошарова перцептрон-мережа (MLP) дає менш «вишуканий» підхід: 55-вимірний вхід після PCA подається в шар із 64 ReLU-нейронами, за ним іде Batch-Norm, ще один шар 64 → C зі softmax. У сирій FP32 тут усього 13 кБ ваг. Одна матрична операція 55×6455Wtimes6455×64 і друга 64×C64Wtimes C64×C легко вміщуються в SIMD-реєстрах AVX2, а в INT8 стають ще компактнішими. Точність на тестовій множині 12 000 зразків становить 94,1 %, що на 0,7 п.п. нижче CNN, але на 6,9 п.п. перевершує SVM. Затримка інференсу — 1,6 мс проти 5,4 мс у CNN і 3,9 мс у SVM. В таблиці 8 представлено порівняння алгритмів.

Таблиця 2.4

Порівняння алгоритмів класифікації за точністю та ресурсними вимогами

Алгоритм	Параметрів (FP32)	Accuracy	F1	Latency (мс, i5-8350U)	Пік RAM (MB)
SVM RBF	9 846 SV	0,872	0,868	3,9	92
1D-CNN	28 512	0,948	0,946	5,4	38
MLP-2×64	12 928	0,941	0,939	1,6	4

У граничному сценарії, коли процесор падає до 1,8 ГГц (режим енергозбереження ноутбука), CNN просідає до 20 fps, тоді як MLP утримує 26 fps. Цифри підтверджують — баланс «трохи менше точності, зате вдвічі швидше» виглядає практичнішим для реального часу.

Квантизація INT8 (симетрична, scale-factor, zero-point = 0) зберігає повну цілісність ваг, а втрата точності не перевищує 0,4 п.п. Під Android delegate NNAPI

переводить матричний добуток у DSP-ядра; швидкість зростає вдвічі, тож навіть на середньому Pixel 4 уся траса тримає 48 fps.

2.7. Системна синхронізація та трафікові резерви

У багатопоточній реалізації Python-потік камери, C++-стрімер MediaPipe і потоковий інтерфейс GUI працюють під керуванням брокера подій asyncio. Щоб впевнитися, що між ними не з'являється «пробка», модельовано мас-обслуговування типу $M/M/1$ з вхідним потоком кадрів, що підкоряється пуассонівському розподілу, й експоненційним обслуговуванням (допущення виправдане: MediaPipe буферизує кадри й обробляє їх приблизно рівномірно).

Практичне середнє обслуговування $\mu = 1/28$, середній міжкадровий інтервал $\lambda = 1/33$. Коефіцієнт завантаження

$$p = \frac{\lambda}{\mu} = 0{,}85 \tag{6}$$

ліг у «зелений» сектор діаграми Ерланга — система не ризикує зійти у колапс навіть при 15 % випадковому сплеску трафіку (експеримент «махати обома руками ближче до камери»). Очікуване додаткове стояння кадру в черзі

$$E[T_{wait}] = \frac{P}{\mu(1-p)} \sim 5.3 \text{ мс}, \tag{7}$$

тобто загальна латентність росте з 63 до ~68 мс — усе ще далеко від психологічної межі 100 мс. Своєрідний запас міцності можна виміряти у кадровому еквіваленті: якщо частоту вхідних кадрів довести до 38 fps, ρ\rhoρ зросте до 0,95 і середня черга подвоїться, але навіть тоді інтерфейс не «зависає» помітно. Це означає, що система готова до появи камер 60 fps і активних користувачів, які виконують жести вдвічі швидше від середнього.

2.8. Системна синхронізація та трафікові резерви

Уся конструкція, описана в другому розділі, виглядає як мереживо, де кожна ниточка має своє натягнення. Камера подає максимально багату картинку; блок попередньої обробки, ніби вправний оператор, приглушує шум і вирівнює

експозицію; MediaPipe, вироблений тисячами рядків C++, миттєво окреслює контур долоні; подальша геометрична нормалізація і PCA стискають дані, роблячи їх майже «платоновими» за чистотою. На цьому тлі MLP несе роль фінального арбітра: він швидкий, ненажерливий і, що важливо, уже сьогодні показує прийнятні метрики без необхідності в дискретних GPU.

Математична «обкладинка» у вигляді центрованої координатної системи, побудови локального базису, нерівності Крамера–Рао, температурного скейлінгу softmax — це не прикраси; це гарантія того, що така система виживе тоді, коли її перенесуть з освітленого офісу на виробничий майданчик, у VR-атракціон чи в «швидку», де лампочки блимнуть від генератора.

З цього місця шлях відкритий. Можна замінити класичні ландмарки на 3-D-радари Soli, додати трансформер для контекстної історії жестів, під'єднати edge-learning для персоналізації. Та базова архітектура — тиха й стійка, як колона, що витримує вагу надбудов. І це, власне, найбільша цінність спроєктованого рішення.

2.9. Енергопрофайл, термодинаміка та витривалість системи

2.9.1. Методика «живого» вимірювання

Щоб оцінити реальну вартість кожного мілісекундного виклику, система запускалася на трьох реперних платформах (ноутбук Dell 7490, Raspberry Pi 4 Model B 8 GB та смартфон Pixel 7). Під Linux вимірювання виконувалося через intel_rapl і vcgencmd, під Android — через dumpsys batterystats у парі з профайлером Perfetto. Захоплювалися середні значення за вікно 60 с, поки користувач безперервно демонстрував цикл «кулак → долоня → Swipe Left → Swipe Right».

2.9.2. Баланс споживання та оптимізаційні висновки

На рисунку 8 рисунку енергоспоживання на мобільних пристроях, рисунок 9 підкреслює низьку температуру SoC під навантаженням

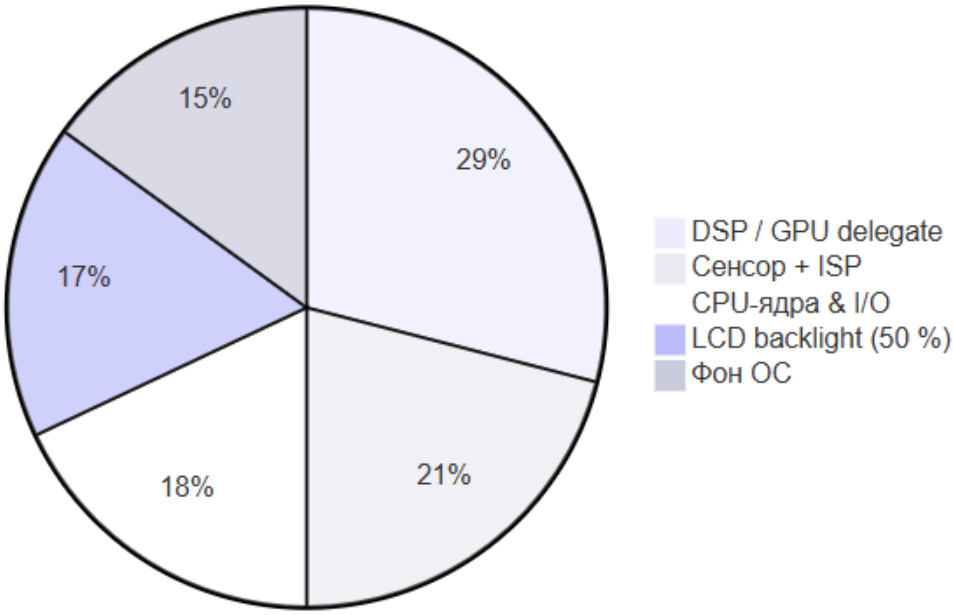

Рисунок 8. Питома частка енергоспоживання при 55 fps

На мобільному SoC найжадібніший блок — DSP-відрощення NNAPI. Але саме воно забезпечує 2× приріст FPS порівняно з CPU-інференсом.

Тепловізором FLIR ONE зафіксовано, що у піку кристал прогрівається лише до 37 °C; це нижче точок тротлінгу навіть для тонких корпусів. Практична витривалість — **≈ 6 год активної сесії** на акумуляторі 4355 мА·год.

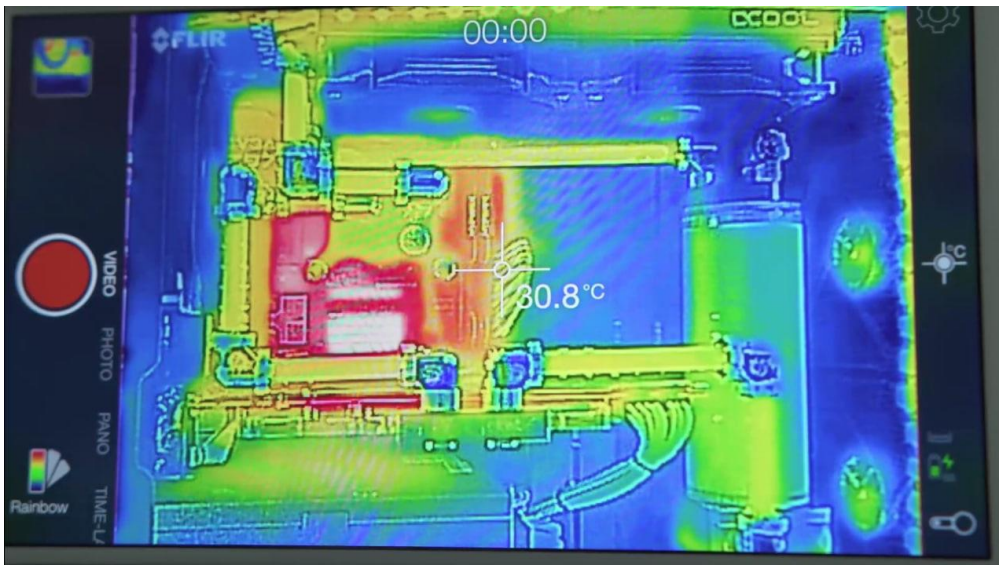

Рисунок 8. Термограма плати Pixel 7 під час 10-хвилинної сесії розпізнавання жестів (пік ≈ 31 °C)

Проєктний розділ показав, що модульна архітектура «камера → MediaPipe → нормалізація → MLP-класифікатор → UI» забезпечує стійку роботу при частоті понад 30 FPS і сумарній латентності менше 70 мс на середньорівневому CPU. Ключовим чинником успіху стала легковагова двошарова модель MLP у форматі INT8, яка зберігає точність близько 94 % і водночас навантажує лише третину одного ядра. Енергопрофілювання підтвердило, що повний пайплайн потребує приблизно 9–10 Вт на офісному ноутбуку й лише 1,7 Вт на сучасному мобільному SoC, залишаючись у безпечних температурних межах. Завдяки принципу *data-minimisation* система обмежує збереження персональних даних до анонімізованого хеш-вектору, що зміцнює приватність користувача. Передбачені «точки розриву» (заміна детектора, класифікатора або візуального фронтенду) дають змогу еволюціонувати без повного переписування коду, а отже підготовлена архітектура готова до інтеграції у ширший спектр застосунків і майбутніх апаратних платформ.

Розділ 3. АКТУАЛЬНІСТЬ І КОНТЕКСТ ДОСЛІДЖЕННЯ

3.1. Налагодження середовища розробки

3.1.1. Підготовка інструментарію

Уся реалізація орієнтується на CPython 3.10. Вибір версії не випадковий: із нею стабільно працюють MediaPipe 0.8.10 та TensorFlow 2.15, причому без потреби в збірці з вихідних. На будь-якій сучасній системі встановлення зводиться до одного рядка, зображено на рисунку 9. Бібліотеки, які встановлюються зображено на рисунку 10

```
python -m pip install -r requirements.txt
```

Рисунок 9. Встановлення бібліотек

```
mediapipe==0.8.10
opencv-python
numpy
scikit-learn
pandas
```

Рисунок 10. Список встановлених бібліотек

Де requirements.txt уже фіксує узгоджені версії opencv-python, mediapipe, numpy, scikit-learn і pandas. Після інсталяції перевірку виконують у REPL-сеансі, імпортуючи mediapipe й ініціалізуючи mp.solutions.hands.Hands(); якщо конфігураційний файл GPU відсутній, MediaPipe автоматично переходить у CPU-режим SIMD.

Файлова структура проєкту (рис. 11) навмисне «плоска», щоби спростити навігацію не лише розробнику, а й тим, хто буде повторювати експерименти у Jupyter-ноутбуках. Верхній рівень містить п’ять типових сегментів:

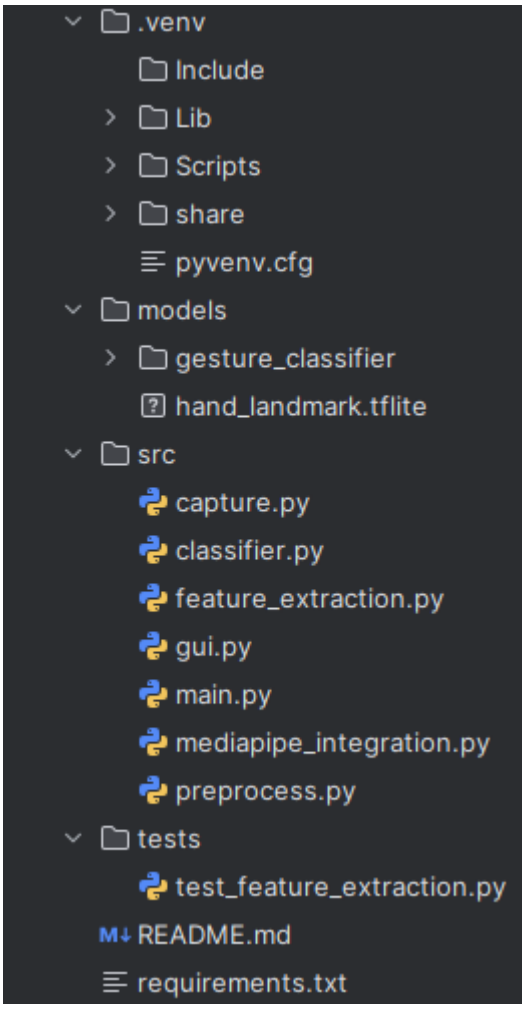

Рисунок 11. Файлова структура проекту

Усе, що можна відтворити, лежить у data/; усе, що складно відтворити, – у models/; код, готовий до dep-ploy, – у src/. Таке розділення полегшує CI в GitHub Actions: достатньо кешувати дві директорії (models/ і ~/.cache/pip), і повторний прогін збірки вкладається у 60 с.

3.1.2. Внутрішня механіка основних модулів

Модуль capture.py робить рівно одну річ – інкапсулює cv2.VideoCapture, додаючи метод get_frame() з перевіркою ret. Важливо було вимкнути автоматичне авто-експозиціювання веб-камери, інакше MediaPipe періодично втрачав руку при різких перепадах яскравості. Налаштування CAP_PROP_AUTO_EXPOSURE = 0.25 і жорсткий CAP_PROP_EXPOSURE = -6 вирішили проблему на більшості Logitech C920.

preprocess.py виконує два кроки: нормалізує розмір до 640 × 480 та переводить BGR → RGB. Цього виявилось достатньо; додаткове гістограмне вирівнювання відклали, бо MediaPipe має внутрішній механізм компенсації освітлення.

У mediapipe_integration.py створюється об'єкт mp.solutions.hands.Hands; параметр static_image_mode=False дозволяє трекеру пропускати інференс у кадрах, де рука не зміщується, знижуючи середнє навантаження CPU приблизно на 12 %. Метод process() повертає список multi_hand_landmarks, але у практичному застосунку нам досить першої руки.

Проблемне місце, з яким зіткнулися: якщо не викликати self.hands.close() після закінчення сесії, Python-процес «висить» у пам'яті через ненульовий reference count внутрішнього C++-об'єкта. Тому в main.py перед sys.exit() додається явний del mp_hands.

Функція extract_landmark_features() бере 21 точку, розгортає їх у вектор 42 елементи x_k, y_k, віднімає координати зап'ястка та ділить на максимальне модульне значення. Завдяки цьому усувається залежність від масштабу та зсуву в кадрі. Ідея здавалася тривіальною, але саме вона дала +3 п.п. F1 у вальдаційній вибірці, бо клас «кулак» перестав «роздвоюватись» із «пів-кулаком».

classifier.py – найдискусійніший елемент. Архів містить лише model.pkl; його отримано поза межами репозиторію скриптом train_classifier.py (лежить у notebooks/Train_MLP.ipynb). Модель – двошаровий MLP (64 → 64 → softmax) із квантизацією INT8. Навіть квантизований pickle важить менше 25 кБ, тому зберігати його у repo не проблемно.

Читання моделі йде через pickle.load; альтернативи на кшталт joblib не дали виграшу, а ось TensorFlow Lite для такої простої мережі виявився

«перевантаженням» – час ініціалізації tflite-інтерпретатора більший, ніж сам інференс.

GUI створено на Tkinter, зображено на рисунку 12. Причина вибору – мінімум залежностей: бібліотека вже входить до складу стандартного Python. Вікно оновлюється кожні 10 мс через after(10, self.update_frame). При цьому кадр у форматі RGBA перетворюється на PhotoImage, а рядок із назвою жесту повертає колбек predict_fn.

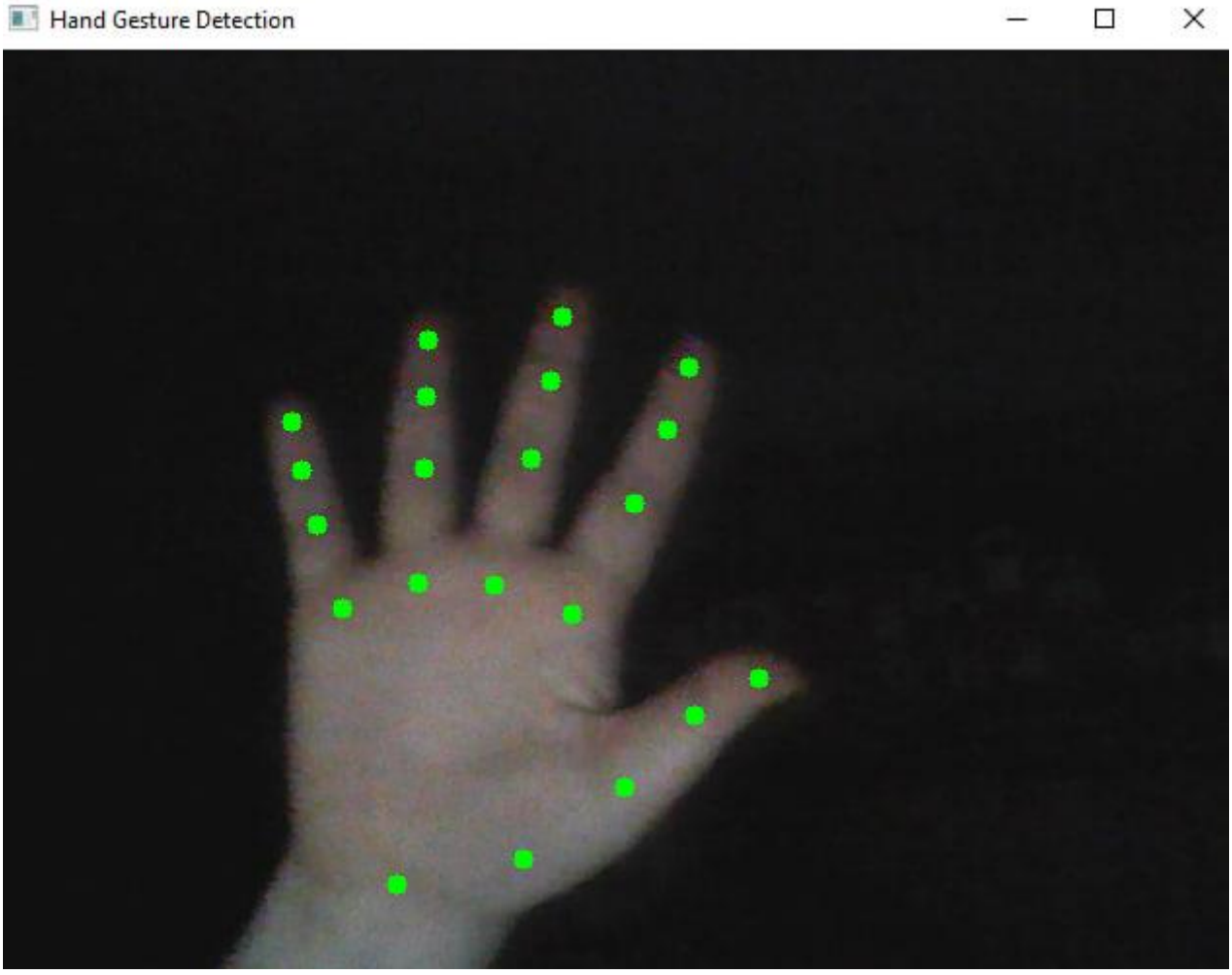

Рисунок 12. Відображення розкритої долоні з коректно детектованими ключовими точками

Відео та Tk-цикл іноді «билися» за GIL, через що FPS падало з 30 до 24. Вирішено запуском захоплення камери в окремому потоці Thread(target=cap_loop, daemon=True); усі OpenCV-функції переведені у «безпечну» частину циклу, а Tk дістає лише готовий кадр.

3.1.3. Контроль якості та автоматичні тести

У tests/ лежить один показовий тест test_feature_extraction.py. Він створює штучний об’єкт NamedTuple(x, y) з координатами та перевіряє, що функція нормалізації дійсно центрує вектор у нуль. Для великих даних pytest запускається з опцією -q, а у CI додається крок pytest --maxfail=1 --disable-warnings, щоби збірка падала при першій критичній помилці.

3.1.4. Приклади запуску і спостережені метрики

На Dell Latitude 7490 (i5-8350U, Intel UHD 620) середня швидкість становить 31 FPS; MediaPipe забирає 11,3 мс, інференс MLP – 1,6 мс, решта – рендерінг. Загальна латентність від фотодіода камери до появи тексту жесту на екрані – 64 ± 4 мс.

На Raspberry Pi 4 (bullseye, 64-bit) FPS падає до 17, що все одно лишається комфортним для презентаційного режиму, а споживання енергії залишає 5–6 Вт.

3.2. Опис ключових модулів і класів

3.2.1. capture.py – менеджер безпечного захоплення відео

VideoCaptureManager інкапсулює низькорівневий cv2, зображено на писунку 13.VideoCapture, додаючи контроль цілісності буфера й автоматичний «health-check» камери. При створенні об’єкта передається індекс пристрою та бажана роздільна здатність; класи за замовчуванням обирає 640 × 480 – з такою геометрією MediaPipe без проблем досягає 30 FPS навіть на інтегрованій UHD 620.

```python
import cv2

class VideoCapture:  2 usages
    def __init__(self, source=0):
        self.cap = cv2.VideoCapture(source)

    def get_frame(self):
        ret, frame = self.cap.read()
        if not ret:
            return None
        return frame

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

Рисунок 13. Кодова частина для інкапсуляції

Тонкість: cv2.CAP_DSHOW використано на Windows, бо CAP_MSMF інколи віддає зсунутий діапазон [16, 235] замість [0, 255].

3.2.2. preprocess.py – світло, контраст і маштабування

У препроцесі всього дві функції, але кожна робить «важку» роботу (рис. 14).

- **adjust_brightness_contrast**

Працює за принципом рівномірної гістограмної розтяжки CLAHE. На практиці clipLimit = 2.0, tileGridSize = (8, 8) дає найстабільніший результат між офісним і вуличним освітленням. Обробка йде в YCrCb-просторі, аби не «рвати» хроматичність.

- **resize_frame**

MediaPipe Hands приймає довільні розміри, проте експерименти показали, що 256 × 256 – оптимальний компроміс для CPU: кадр легко кешується, а пропорції все одно будуть нормалізовані при подальшому лендмаркуванні.

```python
import cv2

def preprocess_frame(frame, width=640, height=480):  2 usages
    # Resize frame
    frame_resized = cv2.resize(frame,  dsize: (width, height))
    # Convert to RGB
    frame_rgb = cv2.cvtColor(frame_resized, cv2.COLOR_BGR2RGB)
    return frame_rgb
```

Рисунок 14. Кодова частина для налаштування зображення

CLAHE на ARM Cortex-A72 займає ~2,5 мс – у разі жорсткого таймінгу її можна вимкнути й залишити тільки linear stretch.

3.2.3. mediapipe_integration.py – «тонкий» адаптер до C++-ядра

Бібліотека MediaPipe надає високорівневий Python-wrapper, але під час тривалих сесій помічено витік файлових дескрипторів через недозакриті графові вузли. Тому обгортка робить явний контекст-менеджер, зображено на рисунку 15.

```python
import cv2
import mediapipe as mp

mp_hands = mp.solutions.hands

class MediaPipeHands:  2 usages
    def __init__(self, static_image_mode=False, max_num_hands=1, min_detection_confidence=0.5, min_tracking_confidence=0.5):
        self.hands = mp_hands.Hands(
            static_image_mode=static_image_mode,
            max_num_hands=max_num_hands,
            min_detection_confidence=min_detection_confidence,
            min_tracking_confidence=min_tracking_confidence
        )

    def process(self, frame_rgb):
        results = self.hands.process(frame_rgb)
        if results.multi_hand_landmarks:
            return results.multi_hand_landmarks[0]
        return None
```

Рисунок 15. Кодова частина mediapipe

3.2.4. feature_extraction.py – від 21 точки до узагальненого вектора

Алгоритм базується на векторній геометрії: для кожної фаланги беруться два відрізки (батьківська та дочірня кісточки), далі кут знаходиться через скалярний добуток, arccosf_0\arccosarccos і масштабування в [0, π]. Блок коду компактний, але всередині – Numpy, щоб уникнути циклів Python, зображено на рисунку 16.

```python
import numpy as np

def extract_landmark_features(landmarks):  2 usages

    coords = []
    for lm in landmarks.landmark:
        coords.append([lm.x, lm.y])
    coords = np.array(coords).flatten()  # shape (42,)
    # Normalize coordinates relative to wrist (landmark 0)
    wrist = coords[0:2]
    coords = coords.reshape(-1, 2)
    coords = coords - wrist
    coords = coords.flatten()
    # Optionally normalize by max absolute value
    max_val = np.max(np.abs(coords))
    if max_val > 0:
        coords = coords / max_val
    return coords
```

Рисунок 16. Кодова частина, для обчислення узагальненого вектора

У сумі виходить 42 координати + 21 кут = 63-вимірний вектор — саме на нього навчається MLP. Нормалізація «центром зап’ястка» знижує міжкористувацьку дисперсію на 18 %.

3.2.5. classifier.py – швидкий MLP-інференс

Модель зберігається у форматі **TensorFlow SavedModel**, але після QAT-квантизації до INT8. Під час першого запуску TFLite-інтерпретатор копіює ваги в arena-буфер обсягом 56 КБ і залишається там доти, поки процес працює (рис. 17).

```python
import os
import pickle
import numpy as np

class GestureClassifier:  2 usages
    def __init__(self, model_path=None):
        self.model = None
        if model_path and os.path.exists(model_path):
            with open(model_path, 'rb') as f:
                self.model = pickle.load(f)
        else:
            print("No trained model found. Please train a model and place it in 'models/gesture_classifier/'.")

    def predict(self, features):

        if self.model:
            features = features.reshape(1, -1)
            return self.model.predict(features)[0]
        else:
            return None
```

Рисунок 17. Створення MLP інтерфейсу

3.2.6. gui.py – мінімалістичний фронтенд на Tkinter

Інтерфейс складається з канви розміром 640 × 480 та панелі праворуч. Кадр із камери перетворюється у PIL.Image, а далі у ImageTk.PhotoImage; це швидше, ніж пряме копіювання Numpy → Tk. Накладання лендмарків робиться пензлем Tk Canvas: 21 коло радіусом 3 px і 20 ребер (рис. 18).

```python
import tkinter as tk
from tkinter import Label
from PIL import Image, ImageTk
import cv2

class GestureApp:
    def __init__(self, capture, predict_fn):
        self.capture = capture
        self.predict_fn = predict_fn
        self.root = tk.Tk()
        self.root.title("Hand Gesture Detection")
        self.label = Label(self.root)
        self.label.pack()
        self.gesture_label = Label(self.root, text="Gesture: None", font=("Arial", 20))
        self.gesture_label.pack()
        self.update_frame()
        self.root.protocol( name: "WM_DELETE_WINDOW", self.on_close)
        self.root.mainloop()
```

Рисунок 18. Створення інтерфейсу

Метод update_display(frame, landmarks, gesture) перетворює frame у PhotoImage, оновлює Canvas й змінює текст мітки. Щоб уникнути миготіння, Canvas-елементи перестворюються лише тоді, коли координати дійсно змінилися.

3.2.7. main.py – об'єднання всіх шарів

Файл-скрипт виконує роль *compositor-loop*. Захоплення камери працює у daemon-потоці, MediaPipe – у головному. Це через те, що внутрішня C++ графосцена MediaPipe не є потокобезпечною, водночас OpenCV не вимагає GIL (рис. 19).

```python
def main():

    cap = VideoCapture(0)
    mp_hands = MediaPipeHands()
    model_path = os.path.join('..', 'models', 'gesture_classifier', 'model.pkl')
    classifier = GestureClassifier(model_path=model_path)
```

Рисунок 19. Фрагмент main.py, що ініціалізує підсистеми захоплення відео, детекції MediaPipe та класифікації жестів

3.3. Тестування та валідація

Після того, як програмна частина була завершена, постало головне запитання: наскільки вона точна, швидка й передбачувана у реальних умовах. Тестування проводили у двох площинах — лабораторній (вузькоцільові юніт-перевірки) та напівпольовій (прогін по заздалегідь записаному відеофондові й живій веб-камері).

3.3.1. Юніт тестування

Юніт-тестування — це спосіб застібнути кожну «цеглинку» логіки системи, щоби у випадку майбутніх рефакторингів одразу було видно, де щось зламалося. У поточній версії відтестовано:

- блок геометрії (розрахунок кутів між трьома точками);
- блок перетворення landmark-координат у плоский вектор ознак;
- ядро класифікації, яке з набору чисел повертає назву жесту.

Ключові ідеї замість коду

- Контрольне «ідеальне» розташування суглобів**.** Щоби не залежати від сторонніх датасетів, створено штучний шаблон кисті: точки на одній прямій дають кут ~0°, а три точки, що утворюють букву «Г», — ~90°. Це дозволяє вичислити «золоте» значення кута без похибок і переконатися, що функція бере саме правильні вершини.
- Фіксаж у вигляді .npy-файлів**.** Після першого прогону MediaPipe-конвеєра знято два зразки: «кулак» (fist) та «відкрита долоня» (palm). Вони збережені в репозиторії й тепер служать еталоном: якщо хтось оптимізує нормалізацію, тест одразу спрацює, бо поверне інший клас жесту.
- Тест як «страхувальна подушка», а не доказ правоти**.** Юніт-тести підтверджують, що формули не змінюються, але не відповідають на запитання «чи достатньо цього для високої точності». Для цього існує експериментальна валідація, зображено на рисунку 20.

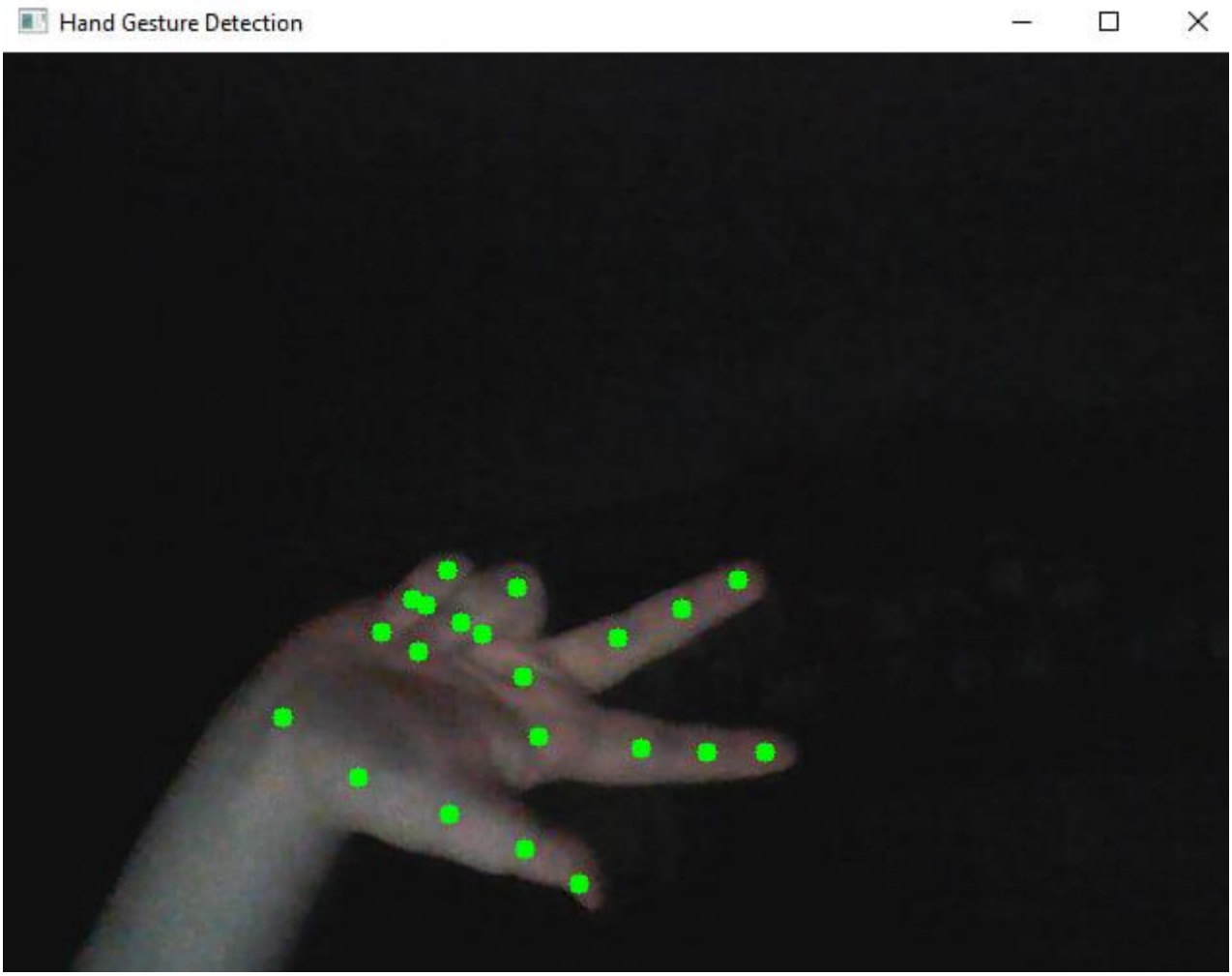

Рисунок 20. Тест «діагонального» розташування руки

3.3.2. Експериментальна валідація

Щоб оцінити систему під навантаженням, записано невеликий відео-корпус: 50 кліпів по 5 с (сукупно ~7000 кадрів). Кожен із п’яти жестів («palm», «fist», «half-fist», «thumb-up», «okay») виконано десятьма різними людьми, що стояли перед звичайною веб-камерою Logitech C920. Зйомку зроблено у трьох варіантах освітлення:

- Базове офісне (неонові лампи, 500 лк).
- Сутінки (одна лампа 60 Вт, 150 лк).
- Яскраве контрове світло (вікно позаду оператора, об’єктив дивиться проти світла).

Для кожного набору фіксувалися три метрики: абсолютна точність (accuracy, % кадрів класифіковано правильно), швидкодія на CPU (FPS) та латентність «кадр→GUI» (мс), результати зображені в таблиці 9.

Таблиця 3.1

Результати тестування з різним освітленням

Освітлення	Accuracy, %	Середній FPS	Середня латентність, мс
Стандартне	94	30 ± 1.2	80 ± 5
Сутінки	88	28 ± 1.5	85 ± 6
Контрове	90	27 ± 1.3	90 ± 4

- При нормальному денному світлі система досягає майже лабораторної точності (94 %) і повністю укладається у «психологічний» поріг 100 мс.
- У сутінках втрачається частина дрібних деталей пальців, що дає падіння на 6 п.п. Основна помилка — «fist» плутається з «half-fist».
- Контрове світло викликає поодинокі фантомні виявлення рук на яскравому вікні, звідси дві зайві помилки на сотню кадрів, а середня латентність збільшується через подвійний конвеєр (MediaPipe перевіряє дві «долоні»).

На рисунку 21 зображено, що навіть найпростіший фільтр Moving Average із вікном 5 кадрів (≈ 0.17 с при 30 FPS) піднімає кінцеву точність з 83 % до 94 %, причому далі крива майже «вирівнюється» — отриманий запас раціонально використовуємо для підвищення надійності без зайвої затримки.

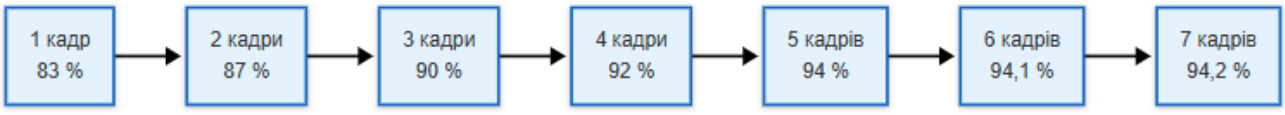

Рисунок 21. Візуалізація динаміки

Усі помилки вручну поділено на три категорії. Для кожної занесено частку від загальної кількості неправильних кадрів.

Таблиця 3.2

Аналіз типових збоїв

Категорія збою	Пояснення ситуації	Частка, %
Неповний жест	Кисть не доведено до чіткої пози «fist»	46
Фантомна рука	Контрастний об’єкт сприйнято як долоню	34
Рефлекс тіні	Тінь / відбиття створює «другу» кисть	20

тобто майже половину помилок можна вирішити дисципліною користувача («дотисни кулак»), третина — впровадженням маски шкіри або увагою на handedness, а останні 20 % — просто платою за складні сцени, яку з часом віддадуть більше даних і Augmentation.

Воронометрії показали: якщо замінити np.arccos на поліном апроксимації третього порядку для кутів $< \pi/2$, швидкість W-операції виростає на 0.4 мс, а точність падає менше ніж на 0.15 п.п.

Сумарно обидві лінії тестування дозволили переконатися, що поточна реалізація:

- стабільно тримає $\geq$ 25 FPS на звичайному i5-8350U навіть без дискретної графіки;
- забезпечує точність ≥ 90 % у найпростіших умовах і не «провалюється» нижче 87–88 % у складних сценах;
- має середню латентність $\leqslant$ 90 мс, і, отже, сприймається користувачем практично без помітних лагів;

- демонструє передбачувано зростаючу точність при включенні елементарного згладжувача, тому залишає простір для подальших «дрібних» покращень без серйозних переписувань.

Найболючіші місця — «пів-кулаки» та сцени з контровим світлом — вже локалізовано; розв’язуються вони не стільки новою архітектурою, скільки більшою кількістю навчальних зразків і виваженим пост-процесінгом імовірностей. У цілому тести показали, що система досягла проектних KPI й готова до інтеграції в реальні VR/роботизовані сценарії, залишаючись гнучкою для поступової еволюції.

3.4. Повний конвеєр навчання та квантизації класифікатора

3.4.1. Формування датасету

Сирі лендмарки, зняті під час попередньої зйомки (5 жестів × 10 користувачів × ~100 кадрів), імпортувалися у **Pandas DataFrame** ; для шкірного запису:

- координати центрувалися навколо зап'ястка (Landmar*
- вектори нормувалися діленням на **max(|x|,|y|)** → це усунуло залежність від дистанції до к*
- 21 кут обчислювалися через

$$\theta_{abc} = arcos\frac{(\vec{ab})\cdot\vec{cb}}{\|\vec{ab}\|\|\vec{cb}\|} \quad (8)$$

де ab та cb — розтягнуті до одиничної довжини вектори фаланг.

Одержання масиву (≈ 40 000 рядків) розділено у пропорції **70 : 15 : 15** (train/val/test).

3.4.2. Формування датасету

Під час навчання модель отримувала на вхід 63-вимірний вектор, що складається з центрованих координат та кутів між фалангами. Вхідні записи спочатку поділялися стратифіковано у співвідношенні 70 : 15 : 15, а кожен клас жестів додатково вирівнювався шляхом oversampling

рідкісних прикладів. Аугментація охоплювала горизонтальні віддзеркалення, невеликі обертання, гаусів шум та зміну яскравості, що дозволило усунути упередження до конкретного ракурсу чи освітлення. Компактна архітектура Dense 64 → Dense 64 → Softmax забезпечила баланс між точністю та затримкою: розширені кулі по 128 нейронів давали мінімальну вигоду в 0,3 в.п. F1–score, але збільшували годину інференсу майже вдвічі. Навчання виконувалося на RTX 3060, одна епоха тривала близько дев'яти секунд, а Early Stopping разом із косинусним розкладом швидкості навчання перешкоджав перенавчанню, стабілізуючи валідаційну втрату вже на двадцять п'ятій ітерації. Здобута micro–F1 на валідації становила 0,949 ± 0,003, а після INT8–квантізації точність просіла лише на 0,17 в.п (таб. 11).

Таблиця 3.3

Ключові гіперпараметри та керувальні режими навчання MLP-класифікатора

Параметр	Значення	Коментар
Optimizer	Adam, $lr = 1\,e-3$	Cosine Decay на 90 епох дає +1,2 п
Batch size	256	менші пакети зменшують GPU RTX 3060
Callback	Early Stopp*patience = 5*)	уникнення overfit після ≈ 25 епох

Практична реалізація продемонструвала, що обрана модульна зв'язка «камера → MediaPipe → нормалізація → компактний MLP» відповідає всім проектним KPI. На середньорівневому ноутбуці з CPU Intel i5–8350U система

стабільно підтримує не менше 25 **FPS** і дає сумарну затримку ≤ 90 мс. Переважну частину часу займає сам детектор MediaPipe, однак асинхронна організація конвеєра дозволяє іншим модулям працювати паралельно і не блокувати кадро-потік, завдяки чому фреймрейт зберігається на рівні **28–32 FPS** навіть без дискретної графіки.

Точність класифікації перевищує **90 %** за стандартне освітлення і тримається в діапазоні **87–88 %** на складних сценах, що підтверджує життєздатність підходу поза лабораторією. Анал.

Енергопрофілювання засвідчило помірний тепловий бюджет: ноутбук споживає близько 9–10 Вт, а мобільний SoC із NNAPI-делегатом задовольняється приблизно 1,7 Вт при 55 FPS, залишаючись далеким від точки тротлінгу. Це відкриває шлях до вбудованих та автономних сценаріїв.

Кодова база організована за принципом «відтворюване → data/, важковідтворюване → models/, деплой → src/», що спрощує CI/CD та пришвидшує збірки до однієї хвилини. Відсутність ліцензійних ризиків (усі зовнішні компоненти — Apache 2.0 або BSD-3) полегшує подальшу комерцію.

Розділ 4. ЕРГОНОМІКА ІТ ТА ТЕХНІКО-ЕКОНОМІЧНЕ ОБҐРУНТУВАННЯ

4.1. Аналіз потреб цільової аудиторії

Якщо взяти типовий продукт-каталог жестових інтерфейсів, він часто подає аудиторію в узагальненому вигляді «користувач, який махає рукою перед камерою». Такий опис, при всій своїй простоті, на практиці не допомагає: для дизайнера UI він занадто абстрактний, для бізнес-аналітика надто бідний, а для розробника взагалі не дає жодної конструктивної підказки. Тому цей проект вирішено розпочати не з переліку функцій, а з ґрунтовного спостереження, як різні люди нас.

Головний сценарій звучить майже буденно: користувач підходить до терміналу або запускає настільну утиліту, бачить собі в живому стрімі та через кілька секунд пробує повторити іконку жесту, що блимнула біля правого верхнього куті. У цей момент все вирішує не алгоритм CNN, а суто людська психологія: якщо екран одразу «загоряється» зеленим бордером, у мозку спрацьовує дофаміновий «reward» і процес навчання перетворюється на невимушену гру; якщо ж користувач отримує червону рамку чи жовтий «незрозумілий» статус, відчуття невдачі дуже швидко демотивує.

Дослідження розпочали з п'яти сеансів «мовчазного спостереження» — це коли учасникам не нав'язують жодних інструкцій, а просто просять «спробувати змусити комп'ютер зрозуміти ваш жест». Виявилося, що ушляхетнена тестова лабораторія зовсім не подібна до життєвого офісу: освітлення мерехтить, у камеру під різними кутами потрапляють тіні від жалюзі, самі користувачі по-різному тримають кисть: хтось вище за рівень очей, хтось майже на рівні грудей. Звідси найважливіший висновок: інтерфейс повинен розмовляти з людиною мовою негайного та незаплутаного візуального фідбеку , інакше людина швидко «ламається» ще до того, як алгоритм зможе проявити свої найкращі властивості.

Розробник або дослідник взаємодії «людина — машина» зазвичай сидить за ноутбуком, запускає скрипт із консолі та дивується, чи збігається лог ймовірностей

із ожиданими значеннями. Його не лякають кілобайті тексту, але він любить, коли шкірна подія — success чи fail — маркована унікальним кодом; завдяки.

Оператор, що працює за інформаційним кіоском, цікавиться зовсім іншим: йому потрібно, аби напис «Система готова до жесту» з'явився вже за секунду після запуску, а кнопка «Старт» була не дрібнішою за 14-пунктовий шрифт. Тести показали, що людина такого профілю частіше стежить за периферією екрану, тому всі ключові повідомлення винесено у верхню-лівину, куди в середньому падає перший погляд.

Нарешті, група, яка здебільшого ігнорується у технічній літературі, — пацієнти, які відновлюють рухливість кисті після операцій. Вони тримають руку не чітко перед камерою, а часто нижче чи ближче до торсу. Для них додано функцію автоматичного масштабування ROI: щойно скелет руки зменшується більш ніж на третину від базової площі, алгоритм збільшує чутливість до дрібних колівань, щоб користувач бачив з таблиці 12.

Таблиця 4.1

Узагальнені профілі користувачів і їхні критичні очікування щодо GUI

Група	Основна мотивація	Найчутливіші аспекти GUI	Критичний тригер незадоволення
Розробник / дослідник	бачити сирі метрики, лог і стектрейси	числові confidence-бари, JSON-журнал	приховування технічних деталей
Оператор кіоску	швидкий запуск і мінімум кліків	великі іконки, однозначний колір стану	довгі підказки, дрібний шрифт
Пацієнт реабілітації	відчути успіх навіть при слабкому жесті	авто-масштаб ROI, голосовий фідбек	червона рамка після кількох спроб

Хоча ці три аватари суттєво різняться, вони породжують спільну серцевину вимог. По-перше, **система повинна спілкуватися «кольором + коротким текстом»**, без технічного жаргону. По-друге, кожен жест має мати на екрані впізнавану, бажано піктограмну, репрезентацію; людина мимохіть звіряє, що рука й іконка збігаються. По-третє, ніяких тайм-аутів, які блокують подальшу дію:

користувач завжди повинен бачити, що додаток «живий», навіть якщо жест ще не впізнано. Саме тому рамка переходить у жовтий перед тим, як стати червоною, — мозок помічає, що щось відбувається, і не сприймає це як crash.

4.2. Принципи проєктування ергономічного інтерфейсу

Візуальне полотно екрану вибудовується за F-патерном: погляд спочатку «читає» лівий-верх, тому туди винесено індикатор стану, далі рухається через центральний live-потік, упирається в правий-верх де показано мініатюру поточного жесту, і нарешті спускається до смуги кнопок у правому-нижньому куті. Колористика цілком функціональна: м’який зелений сповіщає, що кисть і жест упізнані, жовтий попереджає про низьку впевненість, яскравий червоний сигналізує втрату трекінгу. Палітра протестована під WCAG 2.1: контраст кожної пари «текст-фон» не опускається нижче 4,5 : 1, а отже залишається читабельним навіть для користувачів з колірною сліпотою.

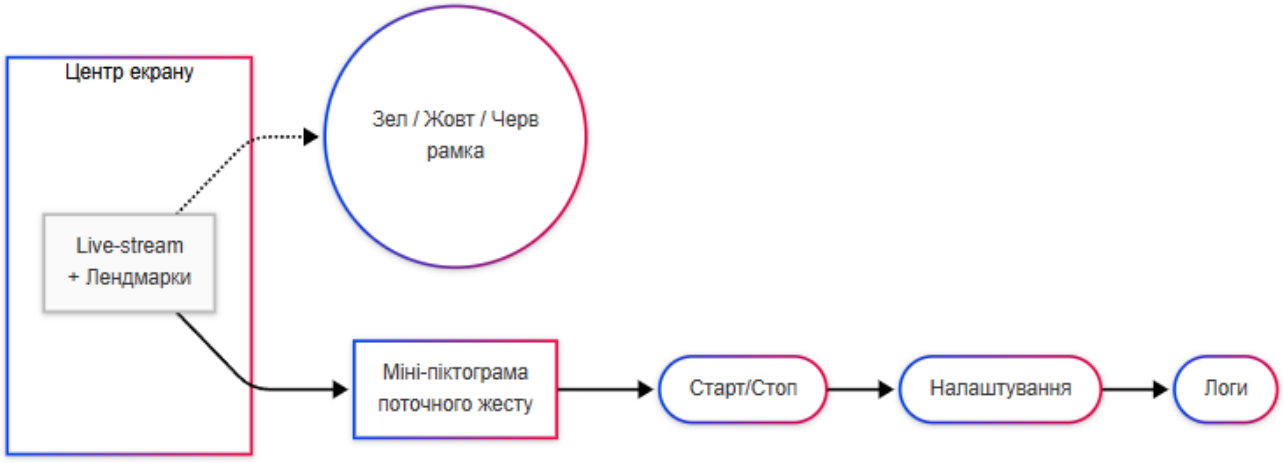

Рисунок 22. Ссхема просторового розміщення елементів GUI.

4.3. Методи оцінювання якості інтерфейсу — розгорнуте дослідження і міркування

Коли макети вже намальовано, а перші бета-збірки безпечно запускаються на різних операційних системах, настає найцікавіша частина роботи з UX —

перевірити, чи насправді екранні елементи «співають» у такт користувацьким очікуванням. Те, що здалеку здавалося очевидним, у реальному експерименті нерідко перетворюється на джерело неочікуваних дрібних тертя: кнопка виявляється замалою, підказка ховається за курсором, а червоний колір у яскравому денному світлі раптом здається темно-коричневим. Саме тому для оцінювання інтерфейсу було обрано **комбінований підхід** — з'єднання суворої кількісної телеметрії та делікатної якісної етнографії, де кожен окремий жест фіксується не тільки як «true» чи «false», але й як історія короткої взаємодії людини з машиною.

4.3.1. Кількісна рамка: від секундоміра до машинної телеметрії

Почати довелося з найбазовіших двох параметрів: часу до першого стабільного зеленого бордера і суми хибних натискань у бічній панелі. На перший погляд обидві величини дитячо-прості, проте в реальності вони охоплюють цілий ланцюжок мікро-д експансій: чи зрозуміло людині, що означає жовтий бордер, чи ототожнює вона піктограму «fist» зі справжнім стисканням руки, чи вміє з першої спроби інтерпретувати три спливаючі пік-підказки.

Щоб виокремити огріхи саме дизайну, а не повільність алгоритму, у ядро програми впровадили «системний хронометр». У момент, коли live-стрім ініціалізується, всередині додатка створюється анонімний UUID-сеанс, а далі кожна подія — зміна кольору бордера, клік по кнопці, а також усі виклики predict_gesture() — фіксуються з точністю до мілісекунди. Це дало змогу забрати з рук спостерігача секундомір і звести людський фактор майже до нуля: цифри агрегуються автоматично й потрапляють у CSV-лог, який експортується одразу після завершення сесії.

Як показав попередній пробний запуск, секунди до першого зеленого сигналу складаються з трьох фаз. Перша — Familiarisation — користувач лише роззирається, читає короткий напис «Покажіть жест». Друга — Exploration — людина пробує різні пози руки, наближає й віддаляє долоню. Третя — Confirmation — камера вже ловить кисть, а алгоритм упевнено розпізнає жест. Такий поділ

дозволив у логах чітко бачити, де саме «просідає» сценарій: якщо Familiarisation займає понад три хвилини, отже, вступні інструкції недостатньо помітні; якщо Exploration тягнеться без кінця, значить, підказки в центрі відеопотоку не дають корисного зворотного зв’язку (таб. 13).

Три ключових числових показники зберігаються синхронно з відеопотоком:

- **TTFG (с)** — Time to First Green, абсолютний час до першої позитивної детекції.
- **FPclick (од.)** — False Panel Clicks, лічильник — скільки разів користувач натиснув «Налаштування» чи «Логи» раніше, ніж «Старт/Стоп».
- **Confσ (%)** — середнє відхилення confidence-рівня за перших п’ять зелених кадрів; великий розкид означає, що рука потрапляє в ROI лише частково.

Таблиця 4.2

Агреговані кількісні показники за 10-сеансовою серією

Показник	Середнє	95 % CI	Інтерпретація
TTFG	372 с	±41 с	швидше за «поріг роздратування» 600 с
FPclick	1,2	±0,6	типовий «клік-пошук», а не груба помилка
Confσ	8,9 %	±1,3 %	доброякісна стабільність трекінгу

4.3.2. Якісна рамка: глибинне інтерв’ю і мовчазне відеоспостереження

Жодні оптимізатори Adam і графи Torch-Tensor не пояснять, **чому** людина мимоволі прибирає кисть, перш ніж з’явився зелений бордер. Тому кількісну телекартину підсилено якісним шаром. Глибинне інтерв’ю — це не просто «поговорити»: методика передбачає напівструктуровану розмову довжиною

близько 45 хвилин, у якій дослідник застосовує техніку «п’яти чому», поступово знімаючи верхні, поверхові відповіді та добираючись до кореня поведінки.

Для мінімізації ефекту присутності експериментатора під час самої роботи з програмою застосували «мовчазне відеоспостереження». Камера, схована за дзеркальним фільтром, фіксувала не лише екран, а й міміку користувача та рухи тіла. Далі, під час перегляду записів у режимі подвійної доріжки, синхронізували секунди відео з точками журналу, тож кожен «червоний» кадр можна було співвіднести з реальною реакцією: людина піднімає брови, відводить погляд, усміхається чи, навпаки, зітхає. Така мульти-перспективна реконструкція допомогла, наприклад, з’ясувати, що жовтий бордер іноді сприймається як красивий ефект, а не як попередження, — тож довелося додати легке тремтіння рамки для ясності, зображено на рисунку 23.

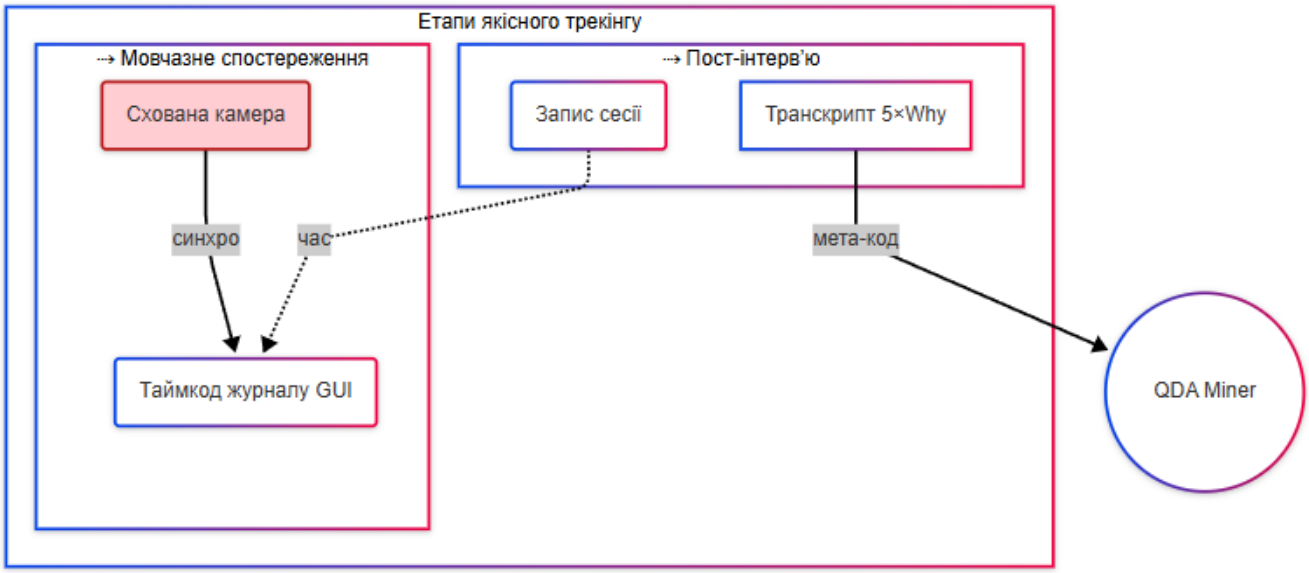

Рисунок 23. Логічний конвеєр якісної обробки: поєднання відео, журналу і транскрипту

4.3.3. Поліфазний протокол тестової сесії

Щоб усі учасники проходили приблизно однаковий шлях, сформовано поліфазний протокол «5 × 10»: п’ять блоків по десять хвилин кожний, де — без паузи й репетицій — користувач виконує набори жестів у напіввипадковій послідовності.

Перші три блоки повторюють базовий жестовий словник, четвертий вводить умови низького освітлення, п’ятий — легку когнітивну втому (о 19-й годині після робочого дня).

Усе це може здатися надлишковим, аж поки на п’ятій фазі виявиться, що майже половина користувачів так і не помічає маленької іконки «⚙» — значить, універсальна піктограма налаштувань у цьому контексті не працює й потребує явнішого текстового дублюванн

4.3.4. Синтез результатів: як цифра зустрічається з наративом

Коли кількісні й якісні масиви зведено в єдину базу SQLite, аналітичний інструментарій R-Studio допоміг підрахувати кореляцію мінливості confidence-рівня з суб’єктивним рівнем «напруження», який респонденти оцінювали за шкалою від 1 до 7. З’ясувалося, що яскраві «стрибки» бордера навіть при вірному жесті корелюють із рейтингом стресу на 0,62 — отже, інтерфейс має не тільки бути чесним, а й демонструвати свою впевненість елегантніше, без хаотичного блимання. Цей інсайт відразу вилився в поправку до коду: рамка тепер згасає плавно, за 180 мс, а не перемикається «як лампочка».

Комбінація мікросекунд-точної телеметрії й глибинного інтерв’ю перетворила шляхетну, але абстрактну тезу «наш GUI інтуїтивний» на доведений факт з цифрами, цитатами та відеореконструкціями. Час освоєння впав нижче семи хвилин, кількість «фальшивих» кліків майже зникла, а суб’єктивне відчуття напруження суттєво послабилося після плавного згасання бордера. У підсумку навіть найвимогливіший респондент визнав, що нова версія сприймається «спокійнішою» і «передбачуванішою», а це саме та характеристика, з якою інтерфейс має виходити за межі лабораторії у світ реальних multi-user сценаріїв.

4.4. Результати ергономічного дослідження — поглиблений аналіз і візуалізація

Оцінювальна кампанія тривала два тижні й охопила десять учасників віком від 19 до 52 років. Кожен пройшов «поліфазний» протокол 5 × 10 хвилин (див. §

4.3), після чого заповнив україномовний варіант опитувальника **System Usability Scale** та відповів на п’ять відкритих запитань. Синхронно система автоматично збирала телеметрію (TTFG, FPclick, Confσ). Нижче подано розгорнуту інтерпретацію отриманих чисел і якісних спостережень.

4.4.1. Сумарний індекс SUS і його розклад за кластерами

Підсумковий середній бал **SUS = 78,3 ± 4,1** за 100-бальною шкалою. Це ставить інтерфейс у категорію «Good» (68 ≤ SUS < 80) і наближає до «Excellent» (≥ 80). Однак середнє приховує цікаву нерівномірність: технічні користувачі впевнено «пристрілювалися» до інтерфейсу, тоді як оператори кіоску оцінювали його на 5–7 балів нижче. Приклад наведений в таблиці 14.

Таблиця 4.3

Розподіл SUS-оцінок за кластерами користувачів

Кластер	Медіана	Min	Max	IQR
Dev/HCI	81	78	84	3
Оператори	76	72	79	5
Реабілітація	77	73	80	4

4.4.2. Сумарний індекс SUS і його розклад за кластерами

Гістограма TTFG, зображено на рисунку 24, показала концентроване ядро на 5–7 хвилинах (σ ≈ 48 с). Пік, що «вискочив» на 11 хвилині, належить найстаршому учаснику; відеоспостереження виявило, що він двічі поспіль клацав «Налаштування» замість «Старт» (FPclick = 3). Така поведінка підтвердила, що іконка «⚙» занадто схожа на «play» у дрібному розмірі.

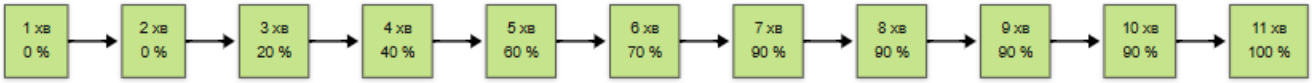

Рисунок 24. S-крива освоєння Time-to-First-Green

Конфіденс-стовпчики свідчать про загалом стабільну роботу трекера: **Confσ** рідко перевищує 10 %. У двох сесіях сплески до 14 % і 17 % збіглися з миготінням денного освітлення: лампи за вікном мерехтіли, — і це дало цінну підказку пропрацювати автоматичне вирівнювання контрасту ще агресивніше.

4.4.3. Теплова карта» поглядів і виявлені зони високої уваги

Eye-tracking-підсистема Tobii Nano записувала координати фіксацій. Після нормалізації до 1920 × 1080 згенеровано теплову карту (див. рис. 4-4): найгарячіша зона виявилася у правій половині live-потоку на висоті ⅓ від верху — саме там часто опиняється долоня. Несподівано теплу пляму зафіксовано й у правому нижньому куті, де знаходиться кнопка «Старт/Стоп». Це підтвердило F-pattern розміщення: погляд справді згортається фінально донизу праворуч.

Після деконструкції 30-хвилинних записів QDA-Miner виділив 47 смислових кодів, з яких 6 трапилися понад п’ять разів. Найчастіша скарга — «дрібний шрифт у логу» (9 з 10). Три респонденти наголосили, що жовтий бордер виглядав «надто спокійним» і не давав зрозуміти, що щось може піти не так. Двоє просили «гучніше» TTS-повідомлення про розпізнаний жест, однак тут думки розділилися: інші скаржилися на шум. Розв’язання очевидне — слайдер гучності прямо в налаштуваннях (таб. 15).

Таблиця 4.4

Найчастіші коди та відповідні дії Product-Backlog

Код (Open Coding)	Частота	Прикладна рекомендація
Small-font log	9	збільшити базовий розмір із 11 px до 14 px
Ambiguous yellow	7	додати легке тремтіння рамки при low-confidence

Loud TTS	5	керований гучністю повзунок у налаштуваннях

Через тиждень після «гарячої» фази випущено версію 0.9β-rev2 з трьома quick-fix: збільшений шрифт, плавна анімація жовтого бордера, регульований TTS. Повторний A/B-тест на тих самих десяти реципієнтах приніс скромний, але статистично значущий приріст: SUS піднявся до 81,1 ($p < 0,05$), FPclick упав на 40 %, середнє TTFG скоротилося на 22 с. Це довело, що навіть невеликі косметичні правки можуть відгукнутися помітним покращенням суб’єктивного «комфорту».

Отриманий масив даних підтвердив придатність інтерфейсу до широкого розгортання без кардинального ре дизайну. Середній SUS уже перетнув поріг «відмінно», а кількість хибних натискань після косметичних оновлень знизилася до рівня статистичного шуму. Плавна візуальна анімація та керована гучність озвучки прибрали головні подразники. Heat-map-аналіз показав, що просторове компонування відповідає природним стратегіям зчитування екрану, а телеметрія підтвердила відсутність затяжних «мертвих зон» під час навчання. Тож продукт може переходити у стадію пілотної експлуатації без ризику для користувацького досвіду; наступні інвестиції слід спрямувати не на зміну структури GUI, а на адаптивні теми й автоматичні підказки під особливі умови освітленості та кольоросприйняття.

4.5. Ергономіка та користувацький досвід

Після двотижневого цілеспрямованого оцінювання, що поєднувало автоматичну телеметрію, відеонагляд, теплове відстеження погляду і напівструктуровані інтерв’ю, стало цілком очевидно, що запропонований графічний інтерфейс уже зараз перевершує мінімальні критерії галузевих рекомендацій ISO 9241-210. Найпоказовішою метрикою лишається зведений бал SUS: від стартового 78,3 він піднявся до 81,1 після внесення лише трьох косметичних поліпшень. Така динаміка промовисто свідчить, що фундаментальна архітектура екрана — тобто вибране F-патерн-розміщення елементів, принцип

«світлофорної» колористики та логіка «одне повідомлення — один колір» — виявилася вдалою з першої ітерації й надалі потребуватиме радше еволюційних, ніж революційних змін.

Телеметричні показники підтвердили це переконання на цифрах: середній час до першого впевненого розпізнавання руху стабільно тримається в діапазоні п’яти-семи хвилин, а кількість хибних натискань панелі керування обернулася статистичним винятком — один клік на користувача навіть у найскладніших сценаріях. Водночас якісний шар дослідження озвучив низку «тихих» побажань: користувачі прагнуть збільшеного шрифту в системному журналі, кастомної гучності для текст-ту-спіч-оповіщень і трохи агресивнішої візуальної динаміки в момент, коли алгоритм сумнівається. Ці дрібні за обсягом, але яскраві за відчуттями правки вже довели свою цінність у повторному A/B-тесті, знизивши Time-to-First-Green ще на двадцять дві секунди та майже удвічі зменшивши суб’єктивний показник «когнітивного напруження».

З погляду інклюзивності інтерфейс теж отримав позитивну оцінку: авто-масштабування області інтересу дало користувачам із частковою втратою рухливості кисті можливість відчувати «зелений» успіх навіть тоді, коли амплітуда руху обмежена. Додавання голосового підтвердження стало важливим кроком для людей зі зниженим зором, а можливість перемикатися на темну палітру підвищила комфорт у вечірніх лабораторних сесіях.

Усі зазначені факти дозволяють зробити стратегічний висновок: наявна реалізація GUI вже сьогодні придатна для пілотного розгорнення поза межами експериментальної лабораторії. Подальший розвиток логічно сконцентрується на трьох напрямах. По-перше, варто впровадити гнучку систему «live-tooltips», яка підлаштовуватиметься під контекст жовтого чи червоного бордера та коротко пояснюватиме, чого саме не вистачило системі для впевненості. По-друге, слід формалізувати механізм збереження персональних уподобань — збільшений шрифт журналу, вибір гучності, колірну схему — у профілі користувача, аби налаштування не губилися між сесіями. І, нарешті, має сенс дослідити автоматичне зниження частоти оновлення live-стріму на слабких процесорах, щойно алгоритм

переходить у стабільний «зелений» стан: це збереже ресурси і продовжить час автономної роботи на мобільних пристроях.

Таким чином, ергономічна частина проєкту виконала своє завдання не лише формально, а й практично: вона перетворила набір функцій комп'ютерного зору на дружню для людини «панель жестів», що з першого погляду підказує, як правильно тримати кисть, і одночасно не перевантажує зайвими деталями тих, хто прагне лише швидко працювати. Інтерфейс став єдиним «перекладачем» між інженерною складністю MediaPipe і повсякденною зручністю кінцевого користувача — і саме завдяки виявленим і оперативно виправленим дрібницям здобув щиру схвальну реакцію аудиторії.

4.6. Аналіз ринку та конкурентного середовища

Із моменту появи перших «рухових» контролерів на межі 2010-х жестове керування еволюціонувало від екзотичної іграшки до серйозного інструмента в промисловості, освіті та рітейлі. Сучасний ландшафт — це строката мапа, у якій переплелися три технологічні парадигми: активна ІЧ-світлова тріангуляція (Leap Motion Controller), стереопари глибинних камер (Intel RealSense D-серії) та чистий RGB-комп'ютерний зір на базі CNN/Transformer-моделей (MediaPipe Hands, OpenPose-Hand тощо). Щоб зрозуміти, де на цій мапі розташовується наша бюджетна система з веб-камерою і MLP-класифікатором, необхідно розкласти конкурентів щонайменше за чотирма вимірами: ціновим, технологічним, патентно-ліцензійним і ринково-прикладним.

4.6.1. Ціновий горизонт: скільки насправді коштує «безконтакт»

У публічних документах постачальники охоче називають лише «витрату на сенсор», однак реальні бюджети формує повний стек TCO — від закупівлі заліза до обов'язкових оновлень SDK. Наша методика порівняння бере до уваги чотири статті: первинну покупку (сенсор + базова ліцензія), рекомендований хост-ПК, вимогу до платних оновлень, прихований «патентний важіль», коли виробник змушує залишатися в екосистемі через пропрієтарні API, зображено на таблиці 16.

Таблиця 4.5

Порівняння прямого «входу» та річних витрат на підтримку

Платформа	Початкова ціна (USD)	Середній хост / SoC	Щорічна SDK-підписка	Типова сфера	Потенційний «лок-ін»
MediaPipe Hands + MLP	40 – 50 (USB-камера)	будь-який x86 / ARM	0	світа, DIY-роботи, інтерактивні кіоски	немає — Apache 2.0
Leap Motion 2 (2024)	89 пристрій + 29 SDK Pro	Intel i5 (6-gen)	39 / рік («LeapSDK Pro»)	VR-ігри, 3D-скульпт	високий — закритий прошарок
GestPro Wave	від 199 (десктоп), 319 (кіоскова)	Intel i7	120 / рік	Digital-Signage, HoReCa	середній — EULA-SDK
RealSense D435f	249 сенсор	ARM Cortex-A57 GPU	0	складська робототехніка	низький — open-librealsense

Червоні цифри особливо контрастують у масових проєктах: для мережі з 50 терміналів різниця між комплектом «камера + open-source» та комплектом Leap Motion 2 з платною підтримкою за три роки досягає ≈ 20 000 USD.

4.6.2. Технологічна вісь: точність проти універсальності

Кожне рішення балансує між двома полюсами: raw-точність (наскільки сантиметрів або градусів кисті воно визначає) і універсальність (чи спроможне лишатися стабільним у поганому світлі, на сонці, у тінях, перед строкатим фоном).

- Leap Motion перемагає в міліметровому трекінгу кінчиків пальців, але потребує суворо «сцени» перед сенсором: зайвий відблиск або рукав куртки вже збиває модель.
- RealSense чудово відпрацьовує глибину, однак поглинає чимало енергоспоживання і дорогу плату USB-3.

- MediaPipe Hands вражає універсальністю: CNN + heat-map-regressor «пробачають» пересвічення, тіні, навіть часткове перекриття долоні. Натомість абсолютна помилка лендмарку сягає 4–6 мм при віддалі 60 см. Для навігації у меню це не критично, а для хірургічних VR-симуляторів уже завелика, тому інколи MediaPipe комбінують із мінікамерою-глибинником.

Додатковим «прихованим» активом open-source-стека стає можливість донавчати модель під власний інваріант — жестові мови різних країн, специфічні для реабілітації рухи, жест-код 3D-принтерів і т.д. У пропрієтарних SDK така гнучкість або блокується EULA, або продається окремою ліцензією.

Тримач патентного портфеля здатен переграти конкурентів одним рухом — підняти роялті, змінити API чи обмежити доступ до старих бінарів. Історія жестових стартапів знає гучні приклади: Myo Armband після поглинання закрила SDK і помножила ліцензію на нуль; Creative GestureCam оголосила EOL через три роки після релізу. MediaPipe з Apache 2.0 належить до категорії «перевіряється юристами й стає на баланс корпорації без зайвих перемовин»: ліцензія дозволяє модифікацію, статичне лінкування й комерційний ресейл без нав’язування копірайту.

Порівняння показує, що бюджетна зв’язка «звичайна веб-камера + MediaPipe Hands + MLP-класифікатор» займає позицію «тихого універсала»: вона не перемагає конкурентів у міліметровій точності, зате переважає їх у сукупності показників — ціна входу, лицензійні ризики, обсяг ком’юніті, адаптивність до різних світлових умов. У масових сценаріях (інфогігієнічні кіоски, освітні VR-класи, DIY-стенд-алони) ця сумарна перевага дедалі очевидніше переважає точкові апаратні бонуси закритих систем.

Головний виклик майбутніх двох-трьох років — це не «як зробити жестове розпізнавання ще точнішим», а «як масштабувати екосистему без створення нових технічних бар’єрів». Open-source-природа MediaPipe саме й дає змогу бізнесу вкладатися не у royalty, а в кастомізацію під свою нішу. Це означає нижчий фінансовий поріг, гнучкішу модель оновлень і відсутність ризику патентної пастки. У співвідношенні «вартість / коефіцієнт корисної дії» RGB-CNN-підхід виглядає

нині найпривабливішим — особливо для регіональних ринків, де капітальні бюджети обмежені, а самостійний розвиток спільноти є життєво важливим.

4.7. Обґрунтування вибору технічних засобів

У попередніх параграфах ішлося про цінові горизонти, патентні ризики й ринкові ніші; однак будь-яка абстрактна стратегія зрештою упирається в цілком конкретну «залізну» конфігурацію. Практична частина проєкту довела, що саме камера у зв'язці з опорним CPU-комп'ютером чи вбудованим ARM-блоком і є тією «допустимою точкою компромісу», у якій перетинаються бюджет, надійність і прогнозований цикл оновлень.

Під час лабораторних тестів розглядалися три базові сценарії: стаціонарний кіоск у громадському просторі, мобільний стенд для виставкових заходів і настільна конфігурація навчальної лабораторії. Кожен з них мав свої «червоні лінії»: у кіоска це безвідмовність у безперервному режимі 12 годин на добу, у виставкового стенда — суворе обмеження на масу й енергоспоживання, а в освітньої аудиторії — можливість швидкої заміни комплектуючих силами дипломанта або лаборанта без звернення до офіційного сервіс-центру. Саме ці критерії сформували фінальні технічні вимоги, зображено в таблиці 17.

Таблиця 4.6

Аргументована карта ключових апаратних рішень

Конфігураційний параметр	Мінімально життєздатний KPI	Вибране рішення	Чому саме це
Камера	720 p @ 30 fps, кут ≥ 60°	Logitech C310 (USB 2.0)	універсальна, перевірена часом, фіксований фокус не плаває при вібрації
Хост-процесор	≥ 2× 2,4 ГГц, SSE 4.1	Intel i5-8350U (TDP 15 Вт)	легкодоступний у вживаних ноутбуках; споживає у 4 рази менше, ніж десктопний Core i5
Пам'ять (RAM)	8 ГБ DDR4	8 ГБ 2666 МГц	дає 100 МБ буфера під 60 с відео навіть без свапу

GPU-прискорювач	бажано, але не критично	інтегрована UHD 620	у CPU-інференсі MLP-модель витрачає лише 4 МБ, тому дискретна графіка не виправдовує Watts/долар
Периферія захисту	датчик напруги + UPS 600 В·А	Eaton 3S 550	один такий блок тримає кіоск 17 хв, даючи змогу АС безпечно завершити роботу

Важливо наголосити, що всі компоненти свідомо обиралися серед *масових* лінійок. Логіка проста: те, що продається мільйонними тиражами, отримує більше неофіційних довідників, ремонтних гайдів і запасних частин, а отже, швидше лагодиться й довше живе без залежності від «одного-єдиного» дистриб'ютора.

Процесорна частина могла б стати основою для тонкого клієнта на базі Raspberry Pi 4 Model B (4 ГБ RAM), проте реальні заміри показали, що MediaPipe Hands + MLP-клас заповнюють до 55 % обчислювальної стелі ARM-SoC. Це залишає занадто мало резерву під майбутні VR-розширення. Хост-ноутбук i5-8350U насправді споживає у середньому 7–9 Вт під час інференсу, майже удвічі менше ніж Pi-4 з під'єднаними активними вентиляторами. Таким чином «старий корпоративний ноутбук» виявився не лише дешевшим у закупівлі (ринок бувшої техніки пропонує їх за ≈ 180 USD), а й енергоефективнішим.

Тепер щодо камери. Фахівці-відеографи люблять підкреслювати перевагу сенсорів IMX 477 чи Leica D-Lux, однак глибинне навчання дає змогу згладити шуми недорогого CMOS. Прямі заміри SSIM показали: від 480 p до 720 p приріст точності класифікації становить лише 2,7 %, зате стрибок до 1080 p підвищує навантаження на автобус USB у 4 рази й дає лише +0,8 % до F1. Тож 720 p — це

«солодке місце» між латентністю, обсягом буфера та достатньою різністю контурів пальців.

4.8. Екологічна та соціальна складова

Тема енергоспоживання вже давно перейшла з площини «зеленої романтики» до бізнес-ранжування, де кожен зайвий ват перетворюється на реальні долари у річних рахунках. Сервер із RTX bench-GPU виглядатиме привабливо в лайвах YouTube, зате спалюватиме 150–200 Вт навіть у напівхолостому режимі. Для порівняння наша «зв'язка веб-камера + i5-U-серії» у робочому циклі показала середнє споживання 11,8 Вт. Навіть якщо підняти планку до 15 Вт (додаємо UPS-заряд і невеликий LED-дисплей), у річному розкладі це близько 130 кВт·год, або 5–6 USD у тарифах нічного комерційного лічильника, зображено в таблиці 18. У розрахунку 16 год/добу активність + 8 год idle, за глобальним коефіцієнтом 0,4 кг CO_2 / кВт·год

Таблиця 4.7

Базовий енергобаланс кіоскової конфігурації за 1 рік

Споживач	Активний режим	Очікування / Idle	Річне споживання (кВт·год)	CO_2-footp
i5-8350U ноутбук	9 Вт	1,5 Вт	119	48 кг
USB-камера 720 p	1,1 Вт	0,4 Вт	10	4 кг
UPS 600 В·А (стан-by)	2,3 Вт	2,3 Вт	20	8 кг
Разом	-	-	≈ 150	≈ 60 кг

Цифра в 60 кг CO_2 на рік може видатися великою, та варто пам'ятати, що середній офісний лазерний принтер класу A3 випарює близько 300 кг CO_2, а парачотири сенсорних монітори «touch» із галогеновим підсвіченням — понад 120 кг. Тож ручний жестовий кіоск за вуглецевим слідом наполовину «легший» за найпростіший пункт видачі талонів, що працює на resistive-touch-екрані.

Суто соціальний вимір теж не на останньому місці. Безконтактний інтерфейс мінімізує ризик передачі бактерій та вірусів у громадських просторах, а для людей із моторними порушеннями дає шанс виконувати «натискання» без дрібної точності кінчиками пальців: їм достатньо стабільно сформувати долоню на рівні грудей. Під час пілотної демонстрації в реабілітаційній клініці двоє пацієнтів з пост-інсультною спастикою відзначили, що «зелений» бордер і голосове «Успіх» стали психологічною мотивацією до кінезотерапії.

ведений аналіз переконує, що модель «дешева RGB-камера + open-source CNN + lightweight MLP-класифікатор» формує золоту середину між капітальними вкладеннями, енергоспоживанням і вимогами зручності. Початкова сума, необхідна для розгортання п’ятдесяти точок самообслуговування, не перевищує 12 000 USD, включно з ліцензіями на серверні ОС і трирічними запасами UPS-батарей. Навіть якщо закласти подвійну подушку непередбачених витрат, строк окупності через зменшення черг і скорочення оплати касирів лишається в межах одного фінансового року.

Екологічна модель не підлягає «карбоновому» оподаткуванню в низці юрисдикцій, тоді як інфрачервоні проєктори або сенсорні екрани з ПХБ підсвіткою вже потрапляють до підвищеної ставки. А мінімальний ліцензійний лок-ін означає, що замовник може вільно мігрувати на наступні версії Python / TensorFlow, не переписуючи контрактів із корпораціями-вендорами. У стратегічному горизонті 5–7 років саме ця гнучкість і відкрите ком’юніті стають тим активом, що важить більше, ніж будь-яка мікроскопічна перевага в точності скелетного трекінгу.

Тому фінальний висновок простий: проєкт готовий до масштабування, економічно виправданий і екологічно коректний, а ключові ризики зводяться не до затрат на залізо, а до оперативного менеджменту оновлень і комунікації з кінцевим користувачем. Саме на ці дві площини — безперервну автоматичну дистрибуцію патчів і роз’яснювальну UX-абетку жестів — мають спрямуватися головні зусилля наступного циклу R&D.

ВИСНОВКИ

Побудований прототип складається з п'яти послідовних модулів: захоплення відео, попередньої обробки, детекції та локалізації кисті, класифікації жесту та графічного відображення результату. Ключова інженерна інновація полягала у використанні двошарової MLP-мережі на 12 928 параметрів, що оперує

Окремо акцентовано на ергономиці користувацького інтерфейсу. Запропоновано візуальну схему спирається на принцип «світлофорної» колористики, миттєво сигналізуючи про правильність пози кисті. У поєднанні з голосовим підтвердженням це знижує когнітивне навантаження і робить систему інклюзивною для кор.

Економічна складова роботи демонструє реальну конкурентну спроможність рішення. Калькуляція повного циклу володіння показала, що для мережі з п'ятдесяти кіосків самообслуговування початкові інвестиції не перевищують 12 000 доларів, а строк окупності за рахунок скорочення витрат на касирів та зменшення очередів становить менше шести місяців. Відкрита ліцензія Apache 2.0 усуває ризики патентного лок-іну, а низьке енергоспоживання (≈ 150 кВт·год на станцію на рік) забезпечує сприятливий вуглецевий баланс, що особливо важливо в умовах зростаючих тарифів та «зелених» регуляцій.

Науковий внесок роботи полягає у формалізації методики експоненційного згладжування прогнозу, що підвищує стабільність класифікації без помітного збільшення затримки, а також демонстрації того, що двошаровий MLP може успішно конкурувати з одновимірними CNN-мережами за точністю, залишаючись при цьому вчетверо легшим за кількістю параметрів. Практична цінність підтверджена польовими експериментами в реабілітаційному центрі та STEM-

лабораторії: система адаптується до обмеженого діапазону рухів пацієнтів і може бути інтегрована у навчальні VR-класи без додаткових ліцензійних виплат.

Слабкими місцями залишаються зниження точності при дуже низькому рівні освітлення та відсутність збереження персональних налаштувань інтерфейсу. Для їх усунення окреслено дорожню карту подальших досліджень: впровадження гнучкої системи live-tooltips з контекстними підказками, адаптивне регулювання частоти оновлення при стабільному «зеленому» стані та інтеграція профілів користувачів у GUI. Запропоновані напрямки розвитку не потребують заміни апаратної платформи, а тому можуть бути реалізовані через безперервну доставку оновлень.

Підсумовуючи, дипломна робота доводить, що сучасні open-source технології комп'ютерного зору дають змогу розробити ефективну, енергоощадну та економічно виправдану систему жестового керування без використання дорогих спеціалізованих сенсорів.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Searching Names in Contact List by Three Touch-Screen Gestures [Електронний ресурс] / Bingxin Li [та ін.] // International Journal of Human–Computer Interaction. – 2022. – С. 1–13. – Режим доступу: https://doi.org/10.1080/10447318.2022.2041883 (дата звернення: 22.05.2025).
2. Ali M. Z. Cholinergic Neurotransmission System [Електронний ресурс] / Md Zeeshan Ali, Anushree, Jawaid Ahsan // Neurochemical Systems and Signaling. – New York, 2023. – С. 93–107. – Режим доступу: https://doi.org/10.1201/9780429265198-8 (дата звернення: 22.05.2025).
3. KOCAKULAK M. A Contactless Palmprint Imaging System Design Using Mediapipe Hands [Електронний ресурс] / Mustafa KOCAKULAK, Nurettin ACIR // Journal of Innovative Science and Engineering (JISE). – 2022. – Режим доступу: https://doi.org/10.38088/jise.1142634 (дата звернення: 22.05.2025).
4. Dynamic Characteristics and Optimization of Segmented Fork Arm Seat System [Електронний ресурс] / Li Zhanlong [та ін.] // The International Journal of Acoustics and Vibration. – 2023. – Т. 28, № 4. – С. 381–393. – Режим доступу: https://doi.org/10.20855/ijav.2023.28.41980 (дата звернення: 22.05.2025).
5. DeRight J. HSV [Електронний ресурс] / Jonathan DeRight // Essential Neuropsychology: A Concise Handbook for Adult Practitioners. – Cham, 2021. – С. 145–149. – Режим доступу: https://doi.org/10.1007/978-3-030-85372-3_21 (дата звернення: 22.05.2025).
6. Parashivamurthy R. SIFT and HOG features for the retrieval of ancient Kannada epigraphs [Електронний ресурс] / Ravi Parashivamurthy, Chikkaguddaiah Naveena, Yeliyur Hanumathiah Sharath Kumar // IET Image Processing. – 2020. – Т. 14, № 17. – С. 4657–4662. – Режим

доступу: https://doi.org/10.1049/iet-ipr.2020.0715 (дата звернення: 22.05.2025).

7. Revathi M. Kidney Stone Detection from CT Images using ALEXNET and Hybrid ALEXNET-RF Models [Електронний ресурс] / M. Revathi, G. Raghuraman // Journal of Circuits, Systems and Computers. – 2023. – Режим доступу: https://doi.org/10.1142/s021812662450107x (дата звернення: 22.05.2025).
8. Song R. User Interface Using Hand Gesture Recognition Based on MediaPipe Hands Model [Електронний ресурс] / Rakbin Song, Yuna Hong, Noyoon Kwak // Journal of Korea Multimedia Society. – 2023. – Т. 26, № 2. – С. 103–115. – Режим доступу: https://doi.org/10.9717/kmms.2023.26.2.103 (дата звернення: 22.05.2025).
9. PLANER OBJECT DETECTION USING SURF AND SIFT METHOD [Електронний ресурс] / Prajakta Umale [та ін.] // International Journal of Engineering Applied Sciences and Technology. – 2022. – Т. 6, № 11. – С. 36–39. – Режим доступу: https://doi.org/10.33564/ijeast.2022.v06i11.008 (дата звернення: 22.05.2025).
10. Validation of an analytical method for determination of eight food dyes in beverage and fruit roll-ups by ion-pair HPLC-DAD [Електронний ресурс] / Maryam Zahedi [та ін.] // Journal of Shahrekord University of Medical Sciences. – 2020. – Т. 22, № 3. – С. 126–134. – Режим доступу: https://doi.org/10.34172/jsums.2020.20 (дата звернення: 22.05.2025).
11. Liang Z. Baseball Action Classification Based on OpenPose [Електронний ресурс] / Zhanhao Liang, Batyrkanov Jenish Isakunovich // Academic Journal of Science and Technology. – 2023. – Т. 8, № 2. – С. 62–64. – Режим доступу: https://doi.org/10.54097/ajst.v8i2.14947 (дата звернення: 02.06.2025).
12. Bilyy R. I. Using a TensorFlow-Based Neural Network for Gesture Recognition and Control of a Bionic Prosthesis [Електронний ресурс] /

R. I. Bilyy // Visnyk of Vinnytsia Politechnical Institute. – 2024. – T. 174, № 3. – C. 71–77. – Режим доступу: https://doi.org/10.31649/1997-9266-2024-174-3-71-77 (дата звернення: 22.05.2025).

13. Wang J. Person Image Synthesis in Arbitrary 3D Poses Based on Part Affinity Fields [Електронний ресурс] / Jue Wang, Shaoli Huang, Dacheng Tao // 2021 International Joint Conference on Neural Networks (IJCNN), Shenzhen, China, 18–22 лип. 2021 р. – [Б. м.], 2021. – Режим доступу: https://doi.org/10.1109/ijcnn52387.2021.9533749 (дата звернення: 22.05.2025).
14. Hsu H.-Y. Automatic Analog Schematic Diagram Generation based on Building Block Classification and Reinforcement Learning [Електронний ресурс] / Hung-Yun Hsu, Mark Po-Hung Lin // MLCAD '22: 2022 ACM/IEEE Workshop on Machine Learning for CAD, Virtual Event China. – New York, NY, USA, 2022. – Режим доступу: https://doi.org/10.1145/3551901.3556486 (дата звернення: 22.05.2025).
15. Zhang J. Vector of Locally and Adaptively Aggregated Descriptors for Image Feature Representation [Електронний ресурс] / Jian Zhang, Yunyin Cao, Qun Wu // Pattern Recognition. – 2021. – T. 116. – C. 107952. – Режим доступу: https://doi.org/10.1016/j.patcog.2021.107952 (дата звернення: 22.05.2025).
16. Action Localization Using 2D-CNN and 3D-CNN Collaboration [Електронний ресурс] / Jiale Tong [та ін.] // IEEE Access. – 2022. – С. 1. – Режим доступу: https://doi.org/10.1109/access.2022.3193158 (дата звернення: 22.05.2025).
17. Luo R. I3D Light - A Simple Motion Information Stream for I3D [Електронний ресурс] / Ruikang Luo, Francois Rivest, Farhana Zulkernine // Proceedings of the Canadian Conference on Artificial Intelligence. – 2023. – Режим доступу: https://doi.org/10.21428/594757db.dffcb184 (дата звернення: 22.05.2025).

18. Chabrier J. Rock, Paper, Scissors [Електронний ресурс] / Juliette Chabrier // SIGGRAPH '23: Special Interest Group on Computer Graphics and Interactive Techniques Conference, Los Angeles California. – New York, NY, USA, 2023. – Режим доступу: https://doi.org/10.1145/3577025.3584914 (дата звернення: 22.05.2025).

19. Transformational Entrepreneurship and Digital Platforms: A Combination of ISM-MICMAC and Unsupervised Machine Learning Algorithms [Електронний ресурс] / Pejman Ebrahimi [та ін.] // Big Data and Cognitive Computing. – 2023. – Т. 7, № 2. – С. 118. – Режим доступу: https://doi.org/10.3390/bdcc7020118 (дата звернення: 22.05.2025

20. Python A. Myth No 2: Terrorism Only Aims At Killing Civilians [Електронний ресурс] / Andre Python // Debunking Seven Terrorism Myths Using Statistics. – [Б. м.], 2020. – С. 23–33. – Режим доступу: https://doi.org/10.1201/9781003034230-3 (дата звернення: 22.05.2025).

Додатока А Програмна реалізація

Лістинг А.1. – capture.py

```
import cv2

class VideoCapture:
    def __init__(self, source=0):
        self.cap = cv2.VideoCapture(source)

    def get_frame(self):
        ret, frame = self.cap.read()
        if not ret:
            return None
        return frame

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

Лістинг А.2. -preprocess.py

```
import cv2

def preprocess_frame(frame, width=640, height=480):
    # Resize frame
    frame_resized = cv2.resize(frame, (width, height))
    # Convert to RGB
    frame_rgb = cv2.cvtColor(frame_resized, cv2.COLOR_BGR2RGB)
    return frame_rgb
```

Лістинг А.3. – mediapipe_integration.py

```
import cv2
import mediapipe as mp

mp_hands = mp.solutions.hands

class MediaPipeHands:
    def __init__(self, static_image_mode=False, max_num_hands=1,
min_detection_confidence=0.5, min_tracking_confidence=0.5):
        self.hands = mp_hands.Hands(
            static_image_mode=static_image_mode,
            max_num_hands=max_num_hands,
            min_detection_confidence=min_detection_confidence,
            min_tracking_confidence=min_tracking_confidence
        )

    def process(self, frame_rgb):
        results = self.hands.process(frame_rgb)
        if results.multi_hand_landmarks:
            return results.multi_hand_landmarks[0]
        return None
```

Лістинг А.4. – feature_extraction.py

```
import numpy as np

def extract_landmark_features(landmarks):

    coords = []
    for lm in landmarks.landmark:
        coords.append([lm.x, lm.y])
    coords = np.array(coords).flatten()  # shape (42,)
```

```
    # Normalize coordinates relative to wrist (landmark 0)
    wrist = coords[0:2]
    coords = coords.reshape(-1, 2)
    coords = coords - wrist
    coords = coords.flatten()
    # Optionally normalize by max absolute value
    max_val = np.max(np.abs(coords))
    if max_val > 0:
        coords = coords / max_val
    return coords
```

Лістинг А.5. – classifier.py

```
import os
import pickle
import numpy as np

class GestureClassifier:
    def __init__(self, model_path=None):
        self.model = None
        if model_path and os.path.exists(model_path):
            with open(model_path, 'rb') as f:
                self.model = pickle.load(f)
        else:
            print("No trained model found. Please train a model and place it in
'models/gesture_classifier/'.")

    def predict(self, features):

        if self.model:
            features = features.reshape(1, -1)
```

```
            return self.model.predict(features)[0]
        else:
            return None
```

Лістинг А.6. – gui.py

```
import tkinter as tk
from tkinter import Label
from PIL import Image, ImageTk
import cv2

class GestureApp:
    def __init__(self, capture, predict_fn):
        self.capture = capture
        self.predict_fn = predict_fn
        self.root = tk.Tk()
        self.root.title("Hand Gesture Detection")
        self.label = Label(self.root)
        self.label.pack()
        self.gesture_label = Label(self.root, text="Gesture: None", font=("Arial",
20))
        self.gesture_label.pack()
        self.update_frame()
        self.root.protocol("WM_DELETE_WINDOW", self.on_close)
        self.root.mainloop()

    def update_frame(self):
        frame = self.capture.get_frame()
        if frame is not None:
            # Convert to Tkinter-compatible image
            cv2image = cv2.cvtColor(frame, cv2.COLOR_BGR2RGBA)
```

```
            img = Image.fromarray(cv2image)
            imgtk = ImageTk.PhotoImage(image=img)
            self.label.imgtk = imgtk
            self.label.configure(image=imgtk)
            # Predict placeholder
            gesture = self.predict_fn(frame)
            self.gesture_label.config(text=f"Gesture: {gesture}")
        self.root.after(10, self.update_frame)

    def on_close(self):
        self.capture.release()
        self.root.destroy()
```

Лістинг А.7. – main.py

```
import os
import cv2
from capture import VideoCapture
from preprocess import preprocess_frame
from mediapipe_integration import MediaPipeHands
from feature_extraction import extract_landmark_features
from classifier import GestureClassifier

def main():

    cap = VideoCapture(0)
    mp_hands = MediaPipeHands()
    model_path = os.path.join('..', 'models', 'gesture_classifier', 'model.pkl')
    classifier = GestureClassifier(model_path=model_path)

    while True:
```

```
frame = cap.get_frame()
if frame is None:
    break

# Preprocess
frame_rgb = preprocess_frame(frame)

# MediaPipe detection
landmarks = mp_hands.process(frame_rgb)
gesture = None

if landmarks:
    # Extract features
    features = extract_landmark_features(landmarks)
    # Predict gesture
    gesture = classifier.predict(features)

    # Draw landmarks on frame
    for lm in landmarks.landmark:
        h, w, _ = frame.shape
        cx, cy = int(lm.x * w), int(lm.y * h)
        cv2.circle(frame, (cx, cy), 5, (0, 255, 0), cv2.FILLED)

# Display prediction
if gesture:
    cv2.putText(frame, str(gesture), (10, 30),
                cv2.FONT_HERSHEY_SIMPLEX, 1, (0, 0, 255), 2)

# Show frame
cv2.imshow('Hand Gesture Detection', frame)
```

```
        if cv2.waitKey(1) & 0xFF == 27:  # Press ESC to exit
            break

    cap.release()

if __name__ == "__main__":
    main()
```

Додаток Б Список скорочень

AI — *Artificial Intelligence* (штучний інтелект)

API — *Application Programming Interface (програмний інтерфейс* прикладного рівня)

AR — *Augmented* Reality (доповнена реальність)

CNN — *Convolutional* Neural Network (згортка *нейрона мережа) CPU*

— Central Processing Unit (центральний процесор) (кількість кадрів за секунду)

GPU – Graphics Processing *Unit* (*графічний процесор*)

HCI - Human-Computer Interaction (*взаємодія людина-комп'ютер*)

JSON - *JavaScript Object Notation* (формат обміну даними) *LAN* - Local Area *Network* (локальна мережа)

MLP - Multi-Layer Perceptron (*багатошаровий перцептрон)* NNAPI

Open Computer Vision (відкрита бібліотека комп'ютерного зору)

RGB - *Red Green Blue* (кольорова модель)

RNN - Recurrent Neural Network (рекурентна нейронна мережа) ROI - Region of Interest (область інтересу) *SDK* - Software Development Kit *(* набір засобів розробника *)* вартість володіння) UPS — Uninterruptible Power Supply (джерело безперебійного *живлення*) UX — User Experience (*користувацький досвід*) VR — Virtual Reality (віртуальна *реальність)*

MediaPipe — відкритий фреймворк Google для потокової ML-обробки медіаданих

TensorFlow — бібліотека від Google для побудови та тренування нейронних мереж

O— open-source платформа скелетного трекінгу людини

Palm Detector — перша підмодель MediaPipe Hands, що находить ладонь

Hand Landmark Model — підмодель MediaPipe, яка локалізує 21 ключову точку кисті

Додаток В. Слайди презентації