УДК 004.93.1 **doi: 10.26906/SUNZ.2026.2.069**

Д. В. Герасимчук, В. М. Федорченко

Харківський національний університет радіоелектроніки, Харків, Україна

ДОСЛІДЖЕННЯ ПРОГРАМНО-АПАРАТНИХ ЗАСОБІВ РОЗПІЗНАВАННЯ МОВИ ЖЕСТІВ У РЕАЛЬНОМУ ЧАСІ

Анотація. Актуальність. Реалізація розпізнавання жестів у реальному часі на доступних обчислювальних платформах (ноутбук/ПК) є важливою для створення інклюзивних інтерфейсів та систем взаємодії людини з комп'ютером. Практичний інтерес становить вибір такого програмно-апаратного конвеєра, який забезпечує прийнятний компроміс між точністю та швидкодією під час роботи з відеопотоком з веб-камери. **Об'єкт дослідження:** програмно-апаратні конвеєри розпізнавання статичних жестів зображення руки у відеопотоці в режимі реального часу. **Мета статті:** розробити прототип системи, що зчитує кадри з веб-камери, та виконати порівняльне дослідження трьох підходів (MediaPipe, OpenCV, YOLOv8n) за показниками якості розпізнавання і швидкодії на платформі класу Intel Core i3 + NVIDIA GeForce MX350. **Результати дослідження.** Реалізовано модульну програмну архітектуру, у якій кожен підхід оформлено як окремий конвеєр із уніфікованим виходом (клас жесту, довіра, затримка). Для MediaPipe використано landmarks-ознаки з подальшою класифікацією, для OpenCV — ознаки форми (контур, Hu-інваріанти, HOG) із SVM, для YOLOv8n — детекцію класу жесту на кадрі. Проведено оцінювання Accuracy/Precision/Recall/F1 і вимірювання FPS/latency для кількох роздільних здатностей; показано, що MediaPipe і YOLOv8n забезпечують близьку якість для статичних жестів, тоді як OpenCV-підхід більш чутливий до освітлення та складності фону, а також має помітні втрати швидкодії на високих роздільностях. **Висновки.** Встановлено, що для ноутбуків класу i3+MX350 найпрактичнішим за співвідношенням «якість/ресурси» є MediaPipe-конвеєр для статичних жестів, тоді як YOLOv8n доцільний у задачах, де потрібна вища стійкість до фону та більший контекст зображення; класичний OpenCV-підхід може бути корисним як легкий базовий варіант, але потребує ретельної нормалізації умов зйомки та доопрацювання сегментації.

Ключові слова: мова жестів, розпізнавання жестів, реальний час, веб-камера, MediaPipe, OpenCV, YOLOv8, комп'ютерний зір.

Вступ

Сучасні інтерфейси дедалі частіше спираються на комп'ютерний зір: жест може замінювати кнопку, голос або клавіатуру в ситуаціях, де контакт із пристроєм небажаний чи неможливий. Для систем реального часу критичним параметром стає не лише правильність розпізнавання, а й час реакції: затримка на рівні сотень мілісекунд сприймається користувачем як «підвисання».

На практиці застосовують три сімейства рішень: landmarks-підходи (наприклад, MediaPipe Hands) [1], класичні конвеєри на ручних ознаках з реалізацією в OpenCV [2] та нейромережеві детектори сімейства YOLO [3], зокрема сучасну реалізацію YOLOv8 [4].

Внесок роботи. У межах роботи:

1) створено прототип, що працює з веб-камерою та дозволяє перемикати режими MediaPipe [1], OpenCV [2] і YOLOv8 [4] в єдиному інтерфейсі;

2) запропоновано легкі ознаки для двох CPU-сценаріїв (landmarks та Hu+HOG) і виконано порівняння з детектором YOLOv8n [4];

3) сформовано компактний набір UkrSL-10 і наведено узгоджені метрики якості та швидкодії.

Аналіз публікацій. У фахових роботах з розпізнавання жестів найчастіше зустрічаються три напрями: геометрія за ключовими точками кисті, класичні конвеєри з сегментацією та ручними дескрипторами, а також моделі глибинного навчання. Узагальнення підходів і проблематики SLR наведено в оглядах [5] та [6].

Для статичних жестів поширені CNN-підходи, що навчаються на зображеннях або попередньо виділених ознаках [7]. Для безперервного розпізнавання та перекладу жестової мови застосовують трансформерні архітектури [8]. Альтернативою landmarks-підходам є позові/ключові моделі на кшталт OpenPose [9]. Базові принципи глибинного навчання систематизовано в [10].

Мета роботи – на одному програмному прототипі порівняти MediaPipe Hands [1], OpenCV підхід [2] та YOLOv8n [4] для розпізнавання жестів із веб-камери в реальному часі. Для цього сформовано UkrSL-10, реалізовано три конвеєри та виконано оцінювання якості (Accuracy/F1) і швидкодії (FPS/latency) за фіксованим протоколом експерименту.

Матеріали та методи

Експериментальну частину виконано для конфігурації ноутбука класу Intel Core i3 + NVIDIA GeForce MX350 (2 ГБ VRAM); характеристики набору даних UkrSL-10 подано в табл. 1.

Таблиця 1 – **Характеристики набору даних UkrSL-10**

Параметр	Значення
Кількість класів (жестів)	10 (статичні)
Кількість учасників	15
Загальна кількість зразків	≈30 000 кадрів
Розділення даних	70% навчання / 15% валідація / 15% тест
Камера та частота	веб-камера 30 кадр/с
Роздільна здатність	640×480 та 1280×720

Прототип написано на Python: OpenCV відповідає за захоплення кадру та базову обробку, MediaPipe Hands повертає landmarks кисті, а YOLOv8 використано як нейромережевий детектор жестів. Швидкодію MediaPipe/OpenCV вимірювали у CPU-режимі в

межах прототипу; для YOLOv8n наведено референсні метрики Ultralytics після експорту в ONNX (640×640) та окремо зазначено роль GPU-прискорення [4].

Якість оцінювали стандартними показниками класифікації (Accuracy, Precision, Recall, F1). Параметри реального часу фіксували як середній час кадру (latency) та FPS для трьох роздільностей вхідного відео: 320×240, 640×480 і 1280×720. Latency трактували як суму препроцесингу, інференсу та постобробки/візуалізації.

Додатково враховували практичні обмеження: пікове використання RAM/VRAM та розмір моделі (для YOLOv8), оскільки саме ці параметри часто визначають можливість розгортання на ноутбуках і вбудованих платформах.

Відеопотік оброблявся послідовно: кадр зчитувався через OpenCV VideoCapture і передавався у вибраний конвеєр. Для MediaPipe виконувалося перетворення BGR→RGB. У OpenCV-варіанті з центральної ROI формували зображення 224×224, переводили у grayscale, застосовували Gaussian blur та бінаризацію Оцу; найбільший контур описували Hu-моментами і HOG. Щоб зменшити «стрибки» класу між кадрами, для MediaPipe та OpenCV використовували голосування більшості у вікні 7 кадрів [2].

Метрики Accuracy/Precision/Recall/F1 обчислювали за загальноприйнятими визначеннями на основі TP/FP/FN/TN. FPS оцінювали як відношення кількості оброблених кадрів до сумарного часу вимірювання; додатково аналізували 95-й перцентиль затримки (p95) (рис. 1).

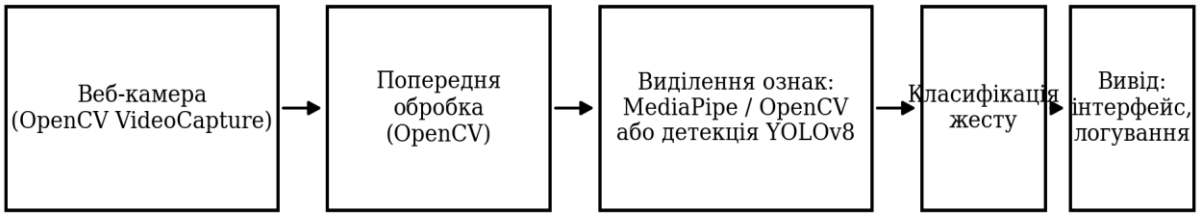

Рис. 1. Узагальнена програмна архітектура системи розпізнавання жестів у реальному часі

Опис алгоритмів розпізнавання.

Підхід 1 (MediaPipe Hands + MLP). MediaPipe виділяє кисть і повертає 21 точку (x, y, z). Ознаки формували як 63-вимірний вектор, нормалізуючи координати відносно зап'ястя та масштабу кисті. Далі застосовували компактний MLP-класифікатор (2 приховані шари), що швидко працює на CPU і легко перенавчається при зміні набору жестів [1].

Підхід 2 (OpenCV + SVM). Використано класичний конвеєр: фіксована ROI → grayscale → згладжування → порогування Оцу → найбільший контур. Для опису форми застосовано Hu-інваріанти, для текстури/градієнтів — HOG; об'єднаний вектор подавали на SVM з RBF-ядром. Метод простий і легкий, але чутливий до сегментації [2].

Підхід 3 (YOLOv8n). YOLOv8n розглядається як детектор, що на одному проході дає рамку руки та клас жесту. Для навчання потрібні дані з bounding boxes; для порівняння швидкодії використано еталон Ultralytics для CPU ONNX (640×640) [4]. У прикладних RTSLR-сценаріях бажано застосовувати прискорення на GPU (зокрема MX350) або оптимізовані бекенди інференсу.

Реалізація програмного забезпечення. Прототип побудовано як модульний конвеєр:

захоплення кадру → легкий препроцесинг → розпізнавання → візуалізація.

Кожен алгоритм оформлено як окремий модуль із однаковим виходом (label, confidence, latency), що спрощує порівняння. Час обробки вимірювали через time.perf_counter() після короткого «прогріву». Для кожного режиму фіксували середню затримку, p95 та FPS. MediaPipe і OpenCV оцінювалися в CPU-режимі; для YOLOv8n використано референсні дані Ultralytics для CPU ONNX [4]. Інтерфейс виводить поточний клас, confidence та службові показники (FPS, latency). Режим роботи задається параметром командного рядка, що робить експерименти відтворюваними.

Результати експериментів та їх обговорення

Вимоги до обчислювальних ресурсів для трьох підходів наведено в табл. 2. Візуальне зіставлення F1-міри подано на рис. 2. Показники швидкодії та затримки для різних роздільностей узагальнено в табл. 3; залежність FPS від роздільної здатності показано на рис. 3, а залежність середньої затримки — на рис. 4. Стійкість конвеєрів до освітлення та складності фону наведено в табл. 4; вплив умов освітлення на F1-міру узагальнено на рис. 5.

Таблиця 2 – **Порівняння якості розпізнавання на тестовій вибірці UkrSL-10**

Метод	Accuracy, %	Precision, %	Recall, %	F1, %
MediaPipe Hands + MLP	95.2	95.6	94.4	95.0
OpenCV (контур+HOG) + SVM	86.4	87.9	83.7	85.7
YOLOv8n	96.8	97.1	96.2	96.6

Таблиця 3 – **Швидкодія та затримка (CPU) для трьох роздільних здатностей (значення в комірках: FPS/мс; для YOLOv8n наведено еталонні дані Ultralytics для входу 640, CPU ONNX)**

Роздільність	MediaPipe (FPS/мс)	OpenCV (FPS/мс)	YOLOv8n (FPS/мс)
320×240	71.4 / 14.0	54.2 / 18.4	–
640×480	66.4 / 15.1	47.6 / 21.0	12.4 / 80.4
1280×720	52.8 / 18.9	54.1 / 18.5	–

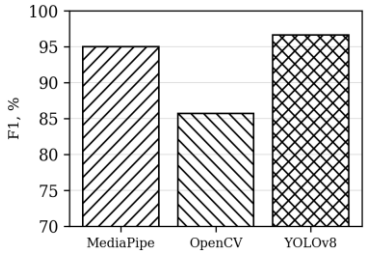

Рис. 2. Порівняння F1-міри для трьох підходів

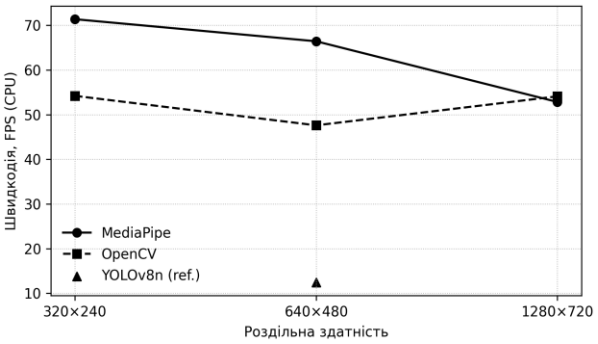

Рис. 3. Залежність швидкодії (FPS) від роздільної здатності (CPU)

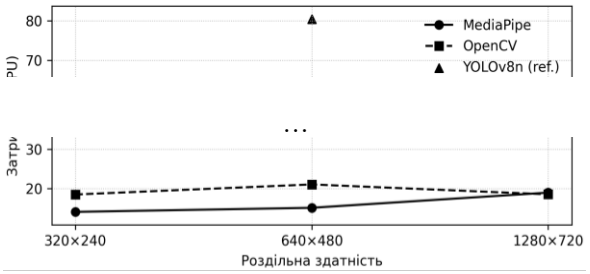

Рис. 4. Залежність середньої затримки (мс) від роздільної здатності (CPU)

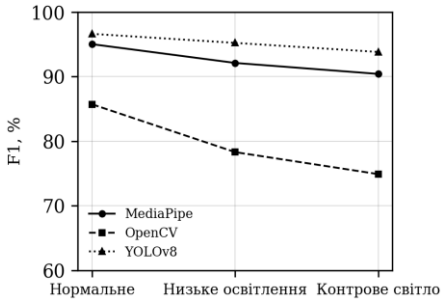

Рис. 5. Залежність F1-міри від умов освітлення

За показниками якості (табл. 2) MediaPipe+MLP і YOLOv8n дають близькі результати для статичних жестів, оскільки перший спирається на стабільні landmarks, а другий використовує піксельний контекст кадру. OpenCV+SVM виступає як базова «легка» лінія: він помітно програє за F1 через залежність від сегментації, але практично не потребує додаткових обчислювальних ресурсів.

Швидкодія (табл. 3, рис. 3 та 4) підтверджує, що MediaPipe та OpenCV можуть працювати у реальному часі на CPU в діапазоні 320×240–1280×720. Для YOLOv8n референс Ultralytics у CPU ONNX (640×640) становить 80.4 мс/кадр (≈12.4 FPS), тому для сценаріїв із вимогами 20–25 FPS доцільно застосовувати прискорення на GPU (MX350) або оптимізований бекенд інференсу.

Стійкість до умов зйомки (табл. 4) показує найбільшу деградацію в OpenCV-підході при низькому/контровому світлі; MediaPipe зберігає стабільність краще, а YOLOv8n має найменший спад на складному фоні. За ресурсами (табл. 5) OpenCV та MediaPipe придатні для систем без дискретного GPU, тоді як нейромережевий детектор вимагає більше пам’яті і виграє при використанні VRAM.

Таблиця 4 – **Стійкість до освітлення та складності фону (F1, %)**

Метод	Нормальне	Низьке освітлення	Контрове світло	Простий фон	Складний фон
MediaPipe+MLP	95.0	92.1	90.4	94.6	91.3
OpenCV+SVM	85.7	78.3	74.9	83.2	76.5
YOLOv8n	96.6	95.2	93.8	96.0	94.7

Таблиця 5 – **Порівняння вимог до обчислювальних ресурсів**

Метод	Параметри моделі, млн	GPU VRAM, ГБ	GPU пам’ять, МБ	RAM, МБ
MediaPipe+MLP	≈0.0	–	–	180
OpenCV+SVM	≈0.0	–	–	140
YOLOv8n	3.2	2.0	320	520

Обмеження дослідження

Дослідження спрямоване на статичні жести; моделювання динаміки (послідовностей) і безперервне розпізнавання фраз не входили до обсягу роботи.

Абсолютні значення швидкодії залежать від ОС, драйверів і бекенда інференсу.

Показник для YOLOv8n подано як еталон Ultralytics (CPU ONNX), тоді як MediaPipe/OpenCV вимірювалися у прототипі Python; тому результати слід інтерпретувати як практичні орієнтири для вибору стеку [4].

Висновки

Створений прототип RTSLR працює з веб-камерою та дозволяє порівнювати три популярні стеки в однакових умовах: MediaPipe, OpenCV і YOLOv8n.

Для UkrSL-10 найкращий баланс «якість/швидкість» у CPU-режимі демонструє MediaPipe+MLP; OpenCV+SVM придатний як максимально легкий варіант, але потребує контрольованого фону/світла; YOLOv8n найстійкіший у складних сценах, однак для стабільного realtime бажано використовувати GPU-прискорення (MX350) [4].

Найближчі напрями розвитку: перехід до динамічних жестів (послідовності), застосування часових моделей (TCN/Transformer) та оптимізація нейромережевого інференсу для вбудованих платформ (ONNX Runtime, квантування).

У практичних системах без дискретного GPU доцільно починати з MediaPipe як базового модуля виділення кисті; за потреби підвищеної робастності або розширення набору класів — переходити до YOLO-підходу з прискоренням.

Конфлікт інтересів. Автори декларують, що не мають конфлікту інтересів стосовно даного дослідження, в тому числі фінансового, особистісного характеру, авторства чи іншого характеру, що міг би вплинути на дослідження та його результати, представлені в даній статті.

Використання засобів штучного інтелекту. Автори підтверджують, що не використовували технології штучного інтелекту при створенні представленої роботи.

Список літератури

1. Zhang F., Bazarevsky V., Vakunov A. et al. MediaPipe Hands: On-device real-time hand tracking. arXiv:2006.10214, 2020. URL: https://arxiv.org/abs/2006.10214
2. Bradski G. The OpenCV Library // Dr. Dobb's Journal. 2000. URL: http://www.drdobbs.com/open-source/the-opencv-library/184404319
3. Redmon J., Farhadi A. YOLO9000: Better, Faster, Stronger // Proc. IEEE CVPR. 2017. P. 7263–7271. DOI: https://doi.org/10.1109/CVPR.2017.690
4. Ultralytics. YOLOv8: документація та бенчмарки швидкодії. 2023–2026. URL: https://docs.ultralytics.com
5. Koller O. Quantitative survey of the state of the art in sign language recognition. arXiv, 2020. URL: https://arxiv.org/abs/2008
6. Subburaj S., Murugavalli S. Survey on sign language recognition in context of vision-based and deep learning // Measurement: Sensors. 2022. Vol. 23. 100385. DOI: https://doi.org/10.1016/j.measen.2022.100385
7. Pigou L., Dieleman S., Kindermans P.-J., Schrauwen B. Sign language recognition using convolutional neural networks // ECCV Workshops (LNCS). 2015. P. 572–578. DOI: https://doi.org/10.1007/978-3-319-16178-5_40
8. Camgoz N. C., Koller O., Hadfield S., Bowden R. Sign Language Transformers: Joint End-to-End Sign Language Recognition and Translation // Proc. IEEE CVPR. 2020. URL: https://openaccess.thecvf.com/content_CVPR_2020/papers/
9. Cao Z., Simon T., Wei S.-E., Sheikh Y. OpenPose: Realtime Multi-Person 2D Pose Estimation Using Part Affinity Fields // IEEE TPAMI. 2021. Vol. 43(1). P. 172–186. DOI: https://doi.org/10.1109/TPAMI.2019.2929257
10. Goodfellow I., Bengio Y., Courville A. Deep Learning. MIT Press, 2016. URL: https://www.deeplearningbook.org

Received (Надійшла) 15.01.2026
Accepted for publication (Прийнята до друку) 22.04.2026
Publication date (Дата публікації) 22.05.2026

Відомості про авторів / About the Authors

Герасимчук Дмитро Вікторович – студент кафедри електронних обчислювальних машин, Харківський національний університет радіоелектроніки, Харків, Україна; e-mail: dmytro.herasymchuk@nure.ua.

Dmytro Herasymchuk – student of the Department of Electronic Computers, Kharkiv National University of Radio Electronics, Kharkiv, Ukraine;
e-mail: dmytro.herasymchuk@nure.ua; ORCID Author ID: https://orcid.org/0009-0009-2715-2614.

Федорченко Володимир Миколайович – PhD, доцент, доцент кафедри електронних обчислювальних машин, Харківський національний університет радіоелектроніки, Харків, Україна;

Volodymyr Fedorchenko – PhD, Associate Professor? Associate Professor of the Department of Electronic Computers, Kharkiv National University of Radio Electronics, Kharkiv, Ukraine;
e-mail: volodymyr.fedorchenko@nure.ua; ORCID Author ID: http://orcid.org/0000-0001-7359-1460.

Research of software and hardware tools for real-time sign language recognition

Dmytro Herasymchuk, Volodymyr Fedorchenko

Abstract. Relevance. Real-time hand-gesture recognition on affordable computing platforms (laptops/PCs) is important for building inclusive human–computer interaction interfaces and assistive technologies. A practical challenge is selecting a processing pipeline that maintains an acceptable trade-off between recognition accuracy and runtime performance when processing a webcam video stream. **Object of research:** software–hardware pipelines for real-time recognition of static hand gestures in a live video stream. **Purpose of the article:** to develop a webcam-based prototype and conduct a comparative study of three approaches (MediaPipe, OpenCV, YOLOv8n) in terms of recognition quality and real-time performance on a platform of the Intel Core i3 + NVIDIA GeForce MX350 class. **Research results.** A modular architecture was implemented where each approach is represented as an independent pipeline with a unified output (gesture class, confidence, latency). The MediaPipe pipeline uses hand landmarks followed by classification; the OpenCV pipeline relies on shape features (contours, Hu moments, HOG) with an SVM classifier; the YOLOv8n pipeline performs single-pass detection/classification of gestures in a frame. The prototype was evaluated using Accuracy/Precision/Recall/F1 and timed using FPS/latency at several input resolutions. The results indicate that MediaPipe and YOLOv8n achieve comparable performance for static gestures, whereas the OpenCV-based solution is more sensitive to illumination changes and complex backgrounds and shows a more pronounced speed drop at higher resolutions. **Conclusions.** For laptops of the i3+MX350 class, the MediaPipe-based pipeline provides the most practical balance between accuracy and computational cost for static gestures, while YOLOv8n is preferable when robustness to background clutter and stronger image context are required. The classical OpenCV pipeline can serve as a lightweight baseline but typically requires careful capture-condition normalization and additional improvements in segmentation to remain stable.

Keywords: sign language, gesture recognition, real-time, webcam, MediaPipe, OpenCV, YOLOv8, computer vision.