КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ТАРАСА ШЕВЧЕНКА

Факультет комп'ютерних наук та кібернетики
Кафедра інтелектуальних систем

Курсова робота

за спеціальністю 121 Інженерія програмного забезпечення
на тему:

ВИКОРИСТАННЯ ЗГОРТКОВИХ НЕЙРОННИХ МЕРЕЖ ДЛЯ СКЕЛЕТОНІЗАЦІЇ ТА РОЗПІЗНАВАННЯ ЖЕСТІВ

Виконав студент 3-го курсу

Ігор ОЛЬХОВАТИЙ

_____
(підпис)

Науковий керівник

кандидат фізико-математичних наук, доцент кафедри ІПС

Ярослав ЛІНДЕР

_____
(підпис)

Засвідчую, що в цій курсовій роботі немає запозичень з праць інших авторів без відповідних посилань.

Студент

_____
(підпис)

Київ – 2024

РЕФЕРАТ

Обсяг роботи 38 сторінок, 21 ілюстрація, 53 джерела посилань.

ЗГОРТКОВІ НЕЙРОННІ МЕРЕЖІ, ВИЯВЛЕННЯ ОБ'ЄКТІВ У РЕАЛЬНОМУ ЧАСІ, ВІДСТЕЖУВАННЯ ОБ'ЄКТІВ, СКЕЛЕТОНІЗАЦІЯ КІЛЬКОХ ОБ'ЄКТІВ У РЕАЛЬНОМУ ЧАСІ, СКЕЛЕТОНІЗАЦІЯ ЗВЕРХУ ВНИЗ.

Центральна тема дослідження полягає у застосуванні передових технологій штучного інтелекту для аналізу рухів рук людини, використовуючи дані, отримані з оптичних сенсорів, як-от камери. Основна мета полягає у розробці ефективних алгоритмів і моделей машинного навчання, які можуть ідентифікувати, відстежувати та аналізувати скелет рук, з подальшим розпізнаванням жестів для інтуїтивного дистанційного управління електронними пристроями.

Метою даної наукової роботи є глибоке дослідження та систематичне порівняння існуючих методологій і технологічних підходів до вирішення комплексної задачі розпізнавання жестів. Це включає аналіз окремих модулів, що відіграють критичну роль у процесі розпізнавання жестів, зокрема модулів скелетонізації, які дозволяють прецизійно інтерпретувати рухи користувача, та спеціалізованих алгоритмів розпізнавання, котрі здатні точно ідентифікувати специфічні жести на основі зібраних даних.

Для розробки програмного забезпечення в рамках цієї роботи було обрано наступні технології: мова програмування Python [1], бібліотека машинного навчання Tensorflow [2], бібліотека машинного навчання PyTorch [3].

Кінцевий продукт дослідження — це високоефективний програмний код, що забезпечує надійне виявлення та скелетонізацію рук людини в реальному часі, а також дозволяє розпізнавати жести для контролю над різними пристроями. Таким чином, дана робота має на меті не лише внести вклад у розвиток теоретичних основ розпізнавання жестів, але й сприяти появі нових практичних рішень для покращення інтерактивності користувацьких девайсів за рахунок ефективного впровадження передових технологій розпізнавання жестів.

Застосування цієї технології не обмежується лише однією сферою і має потенціал революціонізувати взаємодію між людиною та машиною в багатьох галузях, включаючи автоматизоване управління домашньою технікою, переклад

жестових мов, розвиток аугментованої реальності та багато іншого, відкриваючи нові можливості для більш інтуїтивного спілкування між людьми та технологіями.

ЗМІСТ

ВСТУП

Актуальність проблеми розпізнавання жестів

Актуальність проблеми розпізнавання жестів значно зросла в останні десятиліття, особливо з розвитком технологій машинного навчання, комп'ютерного зору та розумних пристроїв. Ця технологія відіграє важливу роль у сфері взаємодії людини і комп'ютера, надаючи нові способи роботи з цифровими пристроями та системами без фізичного контакту.

Розпізнавання жестів також є ключовою для створення інтуїтивних інтерфейсів у віртуальній та доповненій реальності, дозволяючи користувачам ефективно керувати віртуальним середовищем. Окрім того, ця технологія покращує взаємодію із розумними-пристроями, а також використовується у системах безпеки як альтернатива традиційним методам аутентифікації.

Стан сучасних досліджень у сфері розпізнавання жестів

Із приходом нейронних мереж усі сучасні дослідження у сфері комп’ютерного зору кардинально змінилися. Станом на сьогоднішній день згорткові нейронні мережі стали наріжним каменим досліджень розпізнавання жестів, відкривши новий, досі не бачений, простір для досліджень. Більше того, сучасні роботи у цій сфері змінили формулювання задачі, знайшовши тісний зв’язок з розпізнаванням об’єктів та їх скелетонізацією. Подібні методи вирішення задачі дають змогу досягти необхідної якості і дозволяють використання алгоритмів розпізнавання жестів у реальному часі на користувацьких девайсах.

Умови завдання та мета роботи

1. Дослідити і порівняти сучасні методи вирішення задачі розпізнавання жестів.
2. Обрати і реалізувати найбільш застосовний підхід до вирішення задачі.
3. Провести тестування реалізованої системи, отримавши відповідні метрики для точності та швидкості роботи.

1. ТЕОРІЯ

1.1 Нейронні мережі

Розпізнавання жестів рук за допомогою камери є складною задачею, яка включає в себе ідентифікацію та інтерпретацію рухів рук та пальців у реальному часі. Нейронні мережі є ідеальними для розпізнавання образів, оскільки вони здатні ефективно аналізувати візуальні дані, виділяючи важливі ознаки з великої кількості даних без потреби у їх попередньому явному визначенні. Завдяки своїй здатності вчитися на прикладах, нейронні мережі можуть адаптуватися до різноманітних умов освітлення, позицій рук та індивідуальних особливостей жестів користувача. Це дозволяє створювати гнучкі та надійні системи розпізнавання жестів, здатні працювати в реальному часі з високою точністю.

Нейромережа — це математична модель, що імітує роботу мозку біологічного організму для вирішення певних задач, певним чином перетворюючи вхідні дані у вихідні. В основі нейромережі лежить набір з'єднаних між собою нейронів, які можуть приймати, обробляти та передавати сигнали. Далі визначимо основні поняття, якими будемо оперувати далі у рамках цього дослідження.

Нейрон – основний обчислювальний елемент нейромережі, який моделюється як вагована сума його входів плюс зсув (bias), застосовуючи нелінійну активаційну функцію. Формула одного нейрону може бути представлена як:

$$f(\sum_{i=1}^{n} w_i x_i + b), \text{ де}$$

x_i — вхідні сигнали, w_i — ваги синапсів, b — зсув, f — функція активації.

Функція активації — функція, яка визначає вихід нейрону на основі суми вхідних сигналів. Приклади включають сигмоїду, гіперболічний тангенс, ReLU [13] та інші.

Архітектура — певна послідовність нейронів та їх шарів. Нейромережі можуть мати різні архітектури, залежно від задачі. Найпростіша — це одношаровий перцептрон, де всі входи з'єднані з кожним виходом. Більш складні мережі, такі як багатошарові перцептрони (MLP) [14], згорткові нейронні мережі (CNN) [15] та рекурентні нейронні мережі (RNN) [16], містять кілька шарів нейронів, кожен з яких виконує свої обчислення.

Навчання нейромережі — процес адаптації ваг нейромережі для мінімізації різниці між фактичними та прогнозованими даними. Оптимізація параметрів моделі машинного навчання у більшості випадків відбувається за допомогою алгоритму зворотного поширення помилки (backpropagation) [12], який використовує градієнтний спуск для оптимізації ваг.

Функція втрат — диференційована метрика, яка вимірює якість роботи нейромережі, тобто наскільки добре прогнози мережі відповідають дійсним даним, і використовується на покращення результатів роботи. Приклади включають середньоквадратичну помилку, перехресну ентропію тощо.

У контексті нейронних мереж, задача розпізнавання жестів може бути сформульована як задача класифікації, тобто задача оптимізації, де метою є мінімізація розбіжності між прогнозованими і фактичними мітками класів. Розглянемо нейромережу як композицію функцій, що приймає вхідний вектор $\boldsymbol{x}$ і перетворює його у вихід $\boldsymbol{y}$, що представляє прогнозовану категорію. У загальному випадку, це може бути виражено як:

$$y = f(x; \theta), \text{ де}$$

$\boldsymbol{x}$ — тензор вхідних даних, $\boldsymbol{y}$ — вихід, що представляє клас, до якого належать вхідні дані, $\boldsymbol{\theta}$ — параметри (ваги) моделі, що потрібно оптимізувати.

1.2 Згорткові нейронні мережі

Згорткові нейронні мережі (CNN, Convolutional Neural Networks) — це клас нейронних мереж глибокого навчання (deep learning), оптимізованих для аналізу візуальних зображень. Основним фактором, що відрізняє згорткові мережі від інших архітектур, є їх здатність автоматично та адаптивно вивчати просторові ієрархії ознак із зображень, завдяки операції згортки, що дозволяє їм виявляти складні візуальні ознаки на різних, у тому числі високих, рівнях абстракції. Основними компонентами більшості згорткових мереж є згорткові шари, шари агрегування та повнозв'язні шари.

Згортковий шар (Convolutional Layer) — виконує операцію згортки (convolution або точніше cross-correlation), яка передбачає застосування фільтру (ядра згортки) до вхідного зображення або вихідних даних попереднього шару для виявлення певних ознак. Формально, крос-кореляція для двовимірних даних (наприклад, зображення) визначається як:

$$(f*g)(i,j) = \sum_m \sum_n f(m,n) \cdot g(i-m, j-n), \text{ де}$$

f — фільтр розміром $F \times F$, g — вхідне зображення або вихід попереднього шару, тобто тензор розміром $W \times H$

Агрегувальний шар (Pooling Layer) — шар, що зменшує розмірність вхідних даних, зберігаючи при цьому важливі ознаки. Часто використовуються max pooling або average pooling.

Повнозв'язний шар (Fully Connected Layer) - шар, де всі входи з попереднього шару з'єднані з кожним нейроном. Він часто використовується в кінці нейронних мереж для класифікації вхідних даних на основі виявлених ознак.

Навчання згорткових нейронних мережі здійснюється через зворотне поширення помилки та оптимізацію ваг за допомогою алгоритмів, таких як градієнтний спуск. У процесі навчання мережі оптимізуються ваги фільтрів у

згорткових шарах, а також ваги у повнозв'язних шарах, для мінімізації функції втрат.

Згорткові нейронні мережі виявилися надзвичайно ефективними для різноманітних задач обробки зображень, включаючи класифікацію зображень, виявлення об'єктів, різного роду сегментацію та багато інших. Вони здатні виявляти ознаки на різних рівнях абстракції, що робить їх могутнім інструментом у глибокому навчанні.

1.3 Трансформерні нейронні мережі

Трансформерні архітектури [45], запозичені з сучасної обробки природної мови (NLP), досягли великого успіху в різних завданнях комп'ютерного зору, включно із задачею скелетонізації.

У класичному варіанті, представленому у роботі Attention is All You Need [45], механізм уваги виглядає наступним чином:

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

де Q, K, V — значення запитів, ключів та значень видповідно, d_k — розмірність вектора ключів, а функція softmax визначається наступним чином:

$$softmax(z)_i = \frac{e^{z_i}}{\sum_{j=1}^{N} e^{z_j}},$$

Цей механізм використовує три вагові матриці: W_Q, W_K, W_V, які є параметрами моделі машинного навчання. Ці матриці слугують для проектування вхідних даних на компоненти запиту (Q), ключа (K) та значення (V) відповідно. Ці компоненти є лінійними проєкціями тензора вхідних даних. Далі застосовують функцію softmax над нормалізованим добутком проєкцій матриць Q та K і множать на значень (V).

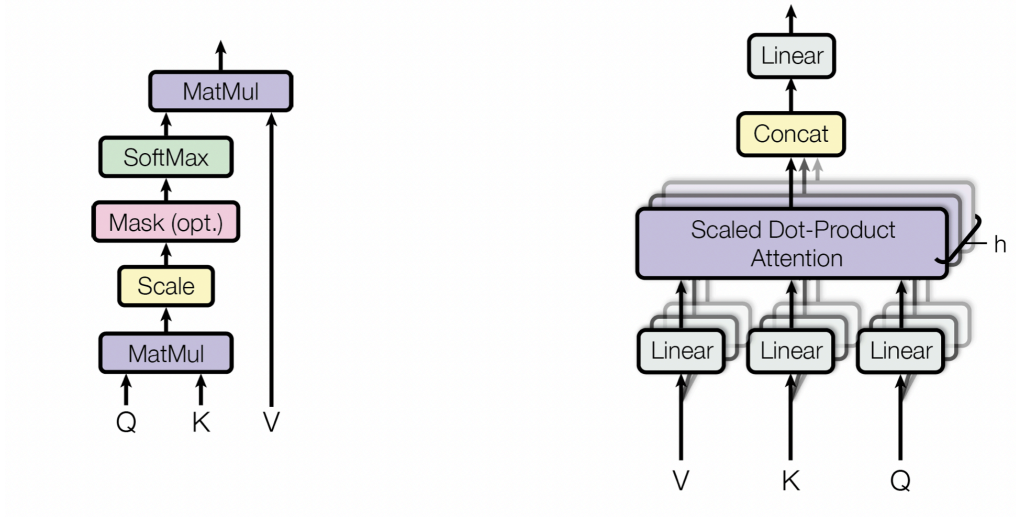

Рис. 1 — Схема роботи механізму уваги у роботі Attention is All You Need [45].

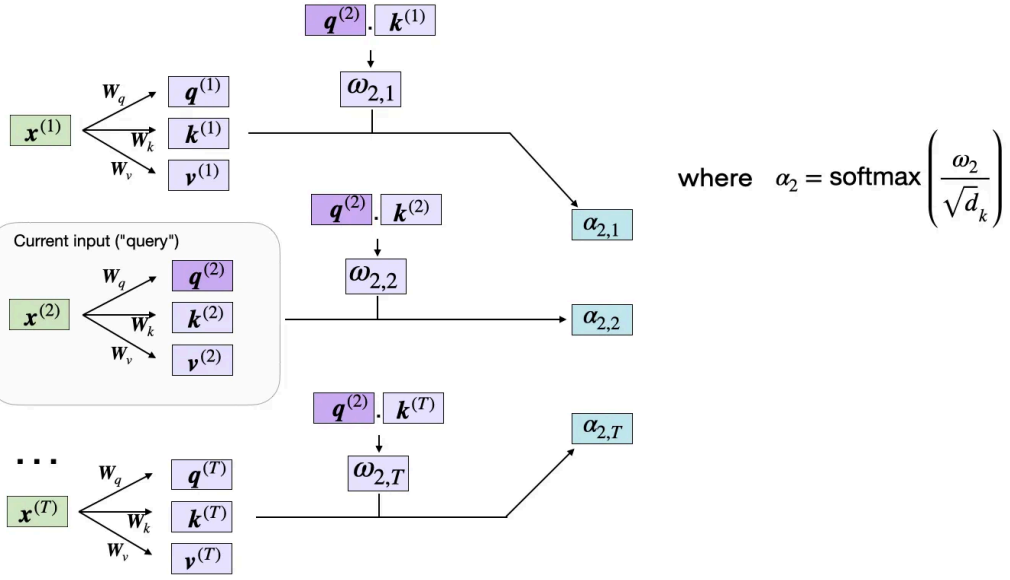

Рис. 2 — Візуалізація роботи механізму уваги [46].

Далі цю ідею перевинайшли в контексті обробки зображень, так з'явилась концепція зорових трансформерів [47]. Для цього ми розбиваємо зображення на ділянки фіксованого розміру, використовуємо лінійні ембедінги для кожної з них, додаємо позиційні ембедінги і подаємо отриману послідовність векторів на кодувальник трансформера. Для виконання класифікації ми використовуємо стандартний підхід - додаємо до послідовності додаткову "класифікаційну мітку", яка може бути використана для навчання.

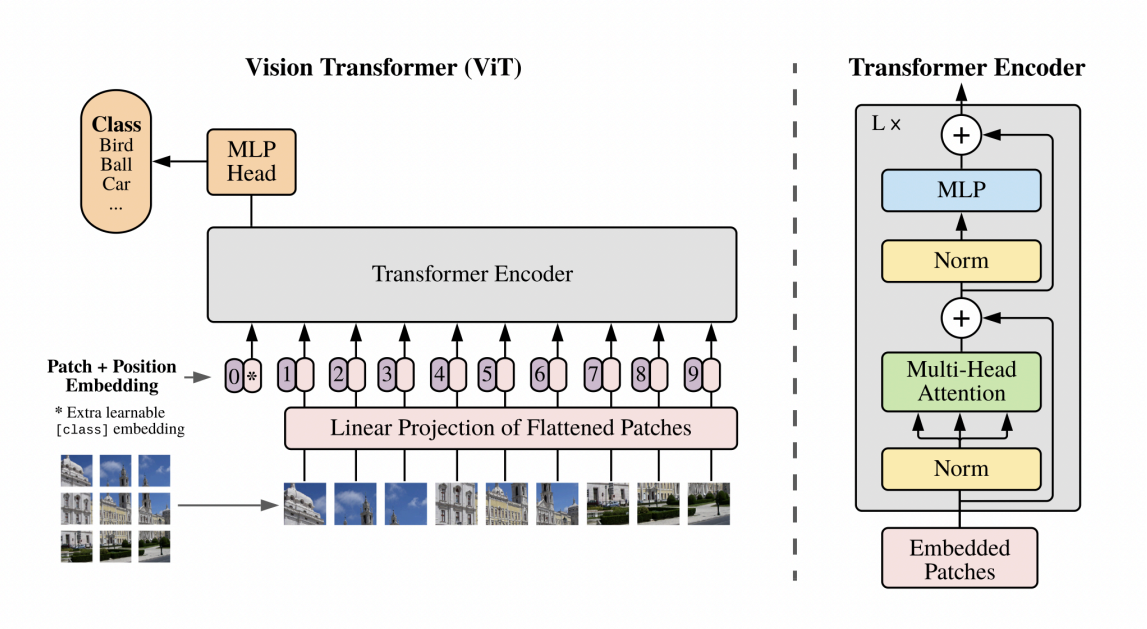

Рис. 3 — Візуалізація принципу роботи класичного зорового трансформера [47].

1.4 Виявлення об’єктів

Виявлення об'єктів - це задача в області комп'ютерного зору, яка включає ідентифікацію екземплярів реальних об'єктів. Її мета полягає в тому, щоб визначити, чи присутні на зображенні екземпляри конкретних категорій об'єктів і, якщо так, локалізувати кожен екземпляр, виводячи навколо нього обмежувальну рамку. Загалом існує дві основних парадигми для вирішення задачі виявлення об’єктів: на основі якорів (anchor based object detection) та без якорів (anchor-free object detection) уперше запропонована в FCOS [17].

Задача виявлення об’єктів без якорів можна сформулювати наступним чином:

$$f: X \to Y \text{ , де}$$

$$X\text{: } \{x \in X \mid x = I\},\ Y = \{(\Delta b_{a1},\ c_1,\ P_1),\ (\Delta b_{a2},\ c_2,\ P_2),\ \dots\ (\Delta b_{an},\ c_n,\ P_n)\},$$

I — тензор розмірності *(h, w, c),* де h — висота вхідного зображення, w — ширина вхідного зображення, c — кількість каналів,

$\Delta b_{ai} = (\Delta x_{ai},\ \Delta y_{ai},\ \Delta w_{ai},\ \Delta h_{ai})$ —Δx_{ai}, Δy_{ai} зміщення центру, Δw_{ai}, Δh_{ai} зміна масштабу для обмежувальної рамки якоря для i-того об’єкта , c_i — клас для i-того об’єкта, p_i — впевненість для i-того об’єкта, $p_i \in [0; 1]$.

Зміни Δb_{ai} застосовуються до попередньо визначеної рамки якоря, щоб отримати бажану обмежувальну рамку b_{ai}, яка точно відповідає фактичному об’єкту на зображенні.

Формально задачу виявлення об'єктів без якорів можна сформулювати наступним чином:

$$f: X \to Y \text{ , де}$$

$$X\text{: } \{x \in X \mid x = I\},\ Y = \{(B_1,\ c_1,\ P_1),\ (B_2,\ c_2,\ P_2),\ \dots\ (B_n,\ c_n,\ P_n)\},$$

I — тензор розмірності *(h, w, c),* де h — висота вхідного зображення, w — ширина вхідного зображення, c — кількість каналів, $B_i = (x_{min}, y_{min}, x_{max}, y_{max})$ або $(x_{center}, y_{center},\ width,\ height)$ — координати обмежувальної рамки (bounding

box) для *i*-того об’єкта, c_i — клас для *i*-того об’єкта, p_i — впевненість для *i*-того об’єкта, $p_i \in [0; 1]$.

Спочатку більшість систем виявлення об’єктів будували вихід моделі на основі якорів [20], натомість найбільш ефективні сучасні рішення із виявлення об’єктів [18], [19] у більшості випадків використовують саме методи без якорів.

1.5 Скелетонізація

Скелетонізація — задача оцінки пози одного або декількох об'єктів у межах зображення або послідовності зображень. У контексті оцінки пози людини, це включає знаходження просторового розташування ключових точок тіла. Формально, маючи зображення *I*, мета скелетонізації полягає у визначенні позицій *N* ключових точок, які визначають позу.

$$f: X \to Y,$$

$$X: \{x \in X \mid x = I\},$$

$$Y: \{y \in Y \mid x = [P_1, P_2, \dots P_S]\},$$

$$P = [p_1, p_2, \dots p_j],\ p = (x_1, x_2, \dots, x_n)\ ,$$ де

I — тензор розмірності *(h, w, c),* де *h* — висота вхідного зображення, *w* — ширина вхідного зображення, *c* — кількість каналів,

j - кількість суглобів (joints), $(x_1, x_2, \dots, x_n)$ — координати кожної з них у просторі R^n.

Загалом для вирішення задачі скелетонізації об’єктів існує два підходи: низхідний (top-down pose estimation) та висхідний (bottom-up pose estimation).

Низхідна скелетонізація

При низхідному підході спочатку виявляють кожен об'єкт окремо, а потім виконують скелетонізацію для кожного виявленого екземпляра окремо. Цей процес можна розбити на два основні етапи:

а) Виявлення об’єктів: ідентифікація та локалізація об'єктів на зображенні.

б) Скелетонізація: для кожного виявленого об'єкта модель прогнозує положення ключових точок у рамці.

Формально цей підхід передбачає послідовне застосування функції виявлення *D*, яка ідентифікує об'єкти, та функції оцінки пози *F*:

$$B = D(X; \theta_D),$$

$$Y = F(Crop(X, B); \theta_F), \text{ де}$$

B - обмежувальні рамки об’єктів, отримані з виходу моделі виявлення об’єкті, $Crop(X, B)$ - операція, яка дозволяє отримати область задану обмежувальною рамкою B із зображення X, θ_D та θ_F - параметри моделі виявлення та скелетонізації відповідно.

Висхідна скелетонізація

З іншого боку, висхідний підхід починається з виявлення всіх відповідних ключових точок на зображенні, незалежно від того, до якого екземпляра вони належать, а потім групує ці ключові точки в окремі екземпляри на етапі постобробки.

а) Виявлення всіх ключових точок на зображенні. Цей крок виводить набір ключових точок, не пов'язуючи їх з конкретними об'єктами або людьми.

б) Групування виявлених ключових точок у окремі екземпляри на основі просторової та, можливо, часової близькості або на основі вивченої спорідненості ознак.

Формально цей підхід передбачає послідовне застосування функції виявлення G, яка ідентифікує ключові точки, та функції групування ключових точок H:

$$K = G(X; \theta_G),$$

де K - виявлені ключові точки, а θ_G - параметри моделі виявлення ключових точок.

Потім функція групування H організовує ці точки в окремі множини, які представляють окремі пози:

$$Y = H(K; \theta_H), \text{ де}$$

θ_H - параметри моделі групування, які можуть включати вивчену спорідненість або геометричні обмеження між ключовими точками.

Обидва підходи мають свої переваги і недоліки. Наприклад, висхідний підхід може бути більш ефективним у сценаріях з великою кількістю об'єктів або людей, оскільки він не вимагає окремого кроку оцінки пози для кожного екземпляра, але він загалом гірше за точністю за низхідну скелетонізацію.

Загалом, більшість підходів до скелетонізації зазвичай розглядають подальшу локалізацію ключових точок як регресію їх координат (наприклад, [34, 35, 36]) або ж регресією за теплокартами (наприклад, [37, 38, 39, 40]).

1.6 Розпізнавання жестів

Розпізнавання жестів використовуючи дані з оптичних сенсорів формулюють у наступному вигляді: нехай *N* — кількість, можливих класів жестів, тоді, маючи набір з *S* вхідних зображень, кожне з яких це тензор *I* розмірності *(h, w, c),* де *h* — висота вхідного зображення, *w* — ширина вхідного зображення, *c* — кількість каналів, потрібно побудувати модель

$$f: X \to Y \text{ , де } X\text{: } \{x \in X \mid x = [I_1, I_2, \dots I_S]\},\ Y = \{1, 2, \dots, N\}$$

Дана функція використовує внутрішнє представлення зображень, щоб віднести кожне зображення до одного з *N* класів. Таке формулювання властиве раннім науковим роботам у сфері [4], [5]. Частина сучасних досліджень мають аналогічне формулювання вхідних даних, проте розглядають це не як задачу класифікації [6].

Більшість сучасних наукових робіт [7], [8], [9], [10] формулюють область визначення функції розпізнавання жестів як:

$$f: X \to Y,$$

$$X\text{: } \{x \in X \mid x = [P_1, P_2, \dots P_S]\},$$

$$P = [p_1, p_2, \dots p_j],\ p = (x_1, x_2, \dots, x_n) \text{ , де}$$

j - кількість суглобів (joints), $(x_1, x_2, \dots, x_n)$ — координати кожної з них у просторі R^n.

Відповідно, значення $P_i,\ i = \overline{1, S}$ можуть бути отримані за допомогою моделі скелетонізатора (pose estimator), що детальніше описані у пункті 1.5. Перевагами подібного підходу є спрощення моделі розпізнавання і зменшення часу роботи моделі. Але у подібного формулювання є також свої недоліки, тому деякі дослідження також фокусуються на більш інваріантному та стійкому представлення скелетона, наприклад DDNet [11].

2. ОПИС ОБРАНОГО РІШЕННЯ

2.1 Розпізнавання жестів

Враховуючи обмеження, яким повинна відповідати модель машинного навчання для розпізнавання жестів у реальному часі, було обрано DDNet (Double-feature Double-motion Network) [11]. Дана модель має надлегку архітектуру, що дозволяє використовувати її на широкому спектрі девайсів, причому вона є беззаперечним лідером за точністю у порівнянні з іншими рішеннями на таких наборах даних як DHG-14 [22], DHG-28 [22], JHMDB [23] згідно [21].

Зокрема, автори DDNet провели дослідження для вирішення наступних проблем скелетонізації: варіація точки зору, варіація швидкості руху, локальні та глобальні траєкторії руху, некорельовані індекси суглобів. Для вирішення проблем, спричинених цими властивостями, на відміну від попередніх рішень, вони вирішили спростити вхідні дані і структуру мережі. Запропонована функція JCD містить інваріантну інформацію про розташування суглобів. Порівняно з іншими подібними функціями, її можна легко обчислити і вона містить менше елементів.

Нехай K - кількість кадрів жесту, N - кількість суглобів (joints), стала для кожного кадру. Тоді для кадру k декартові координати в R^3 для суглоба n можна представити у вигляді $J_i^k = (x, y, z)$, а для простору R^2 їм буде відповідати представлення $J_i^k = (x, y)$. Таким чином, маємо множину суглобів:

$$S_k = \{J_{k_1}, J_{k_2}, ..., J_{k_N}\}$$

а $JCD^k(S_k)$ можна визначити наступним чином:

$$JCD^k = \begin{bmatrix} \left\|\overrightarrow{J_2^k J_1^k}\right\| & & \\ \vdots & \ddots & \\ \vdots & \cdots & \ddots \\ \left\|\overrightarrow{J_N^k J_1^k}\right\| & \cdots & \cdots & \left\|\overrightarrow{J_N^k J_{N-1}^k}\right\| \end{bmatrix}$$

де елементи матриці JCD^k — це евклідові відстані між J_i^k та J_j^k, причому $i \neq j$. На вході у модель машинного навчання дану трикутну матрицю перетворюють у вектор розмірності $(N,\ 2)$.

Оскільки JCD^k не містить інформації про швидкість руху, ми обчислюємо також часові різниці декартових координат. Оскільки жести можуть містити одночасно як швидкі, так і повільні рухи, то для формування глобальної функції руху DD-Net використовує обидва типи рухів:

$$M_{slow}^{\ k} = S^{k+1} - S^k,\ k \in \{1,\ 2,\ \ldots,\ K-1\},$$

$$M_{fast}^{\ k} = S^{k+2} - S^k,\ k \in \{1,\ 2,\ \ldots,\ K-2\}$$

Далі обидві матриці перетворюють у вектори відповідно і застосовують лінійну інтерполяцію аби отримати розмірності що відповідають матриці JCD^k.

Оскільки більшість нейронних мереж за своєю суттю припускають, що вхідні дані є локально корельованими, безпосередня обробка даних, де сусідні індекси можуть відповідати точкам, що не корелюють, призведе до неефективного навчання. Щоб вирішити цю проблему, DD-Net переводить JCD^k та двомасштабну ознаку руху у латентні вектори на кожному кадрі за допомогою ембедингів.

$$\varepsilon_{JCD}^{\ k} = Embedding_1(JCD^k)$$

$$\varepsilon_{M_{slow}}^{\ k} = Embedding_1(M_{slow}^{\ k})$$

$$\varepsilon_{M_{fast}}^{\ k} = Embedding_2(M_{fast}^{\ k})$$

Ембедингами загалом є послідовність шарів нейронів, що переводить дані з одного простору в інший. У даному випадку:

$$Embedding_1 = Conv1D(1,\ 2 * filters) \rightarrow Conv1D(3, filters) \rightarrow Conv1D(1,\ filters),$$

$$Embedding_2 = Conv1D(1,\ 2 * filters) \rightarrow Conv1D(3, filters) \rightarrow Conv1D(1,\ filters) \rightarrow$$

$$\rightarrow Max\ pooling(2),$$

де $filters$ — кількість фільтрів відповідно до варіанту архітектури DDNet, $Conv1D$ — одновимірна крос-кореляція, $Max\ pooling$ — агрегувальний шар.

Таким чином, кореляція між суглобами автоматично визначається завдяки ембедингам, до того ж, цей процес також зменшує вплив шуму у даних. Далі відбувається конкатенація ембедингів латентних ознак:

$$\varepsilon^{k} = \varepsilon_{JCD}{}^{k} \oplus \varepsilon_{M_{slow}}{}^{k} \oplus \varepsilon_{M_{fast}}{}^{k}, \quad \varepsilon^{k} \in R^{(K/2) \bullet filters}$$

Вищезгаданий параметр $filters$ фактично відповідає за розмір нейронної мережі, від нього лінійно залежить кількість параметрів моделі DDNet.

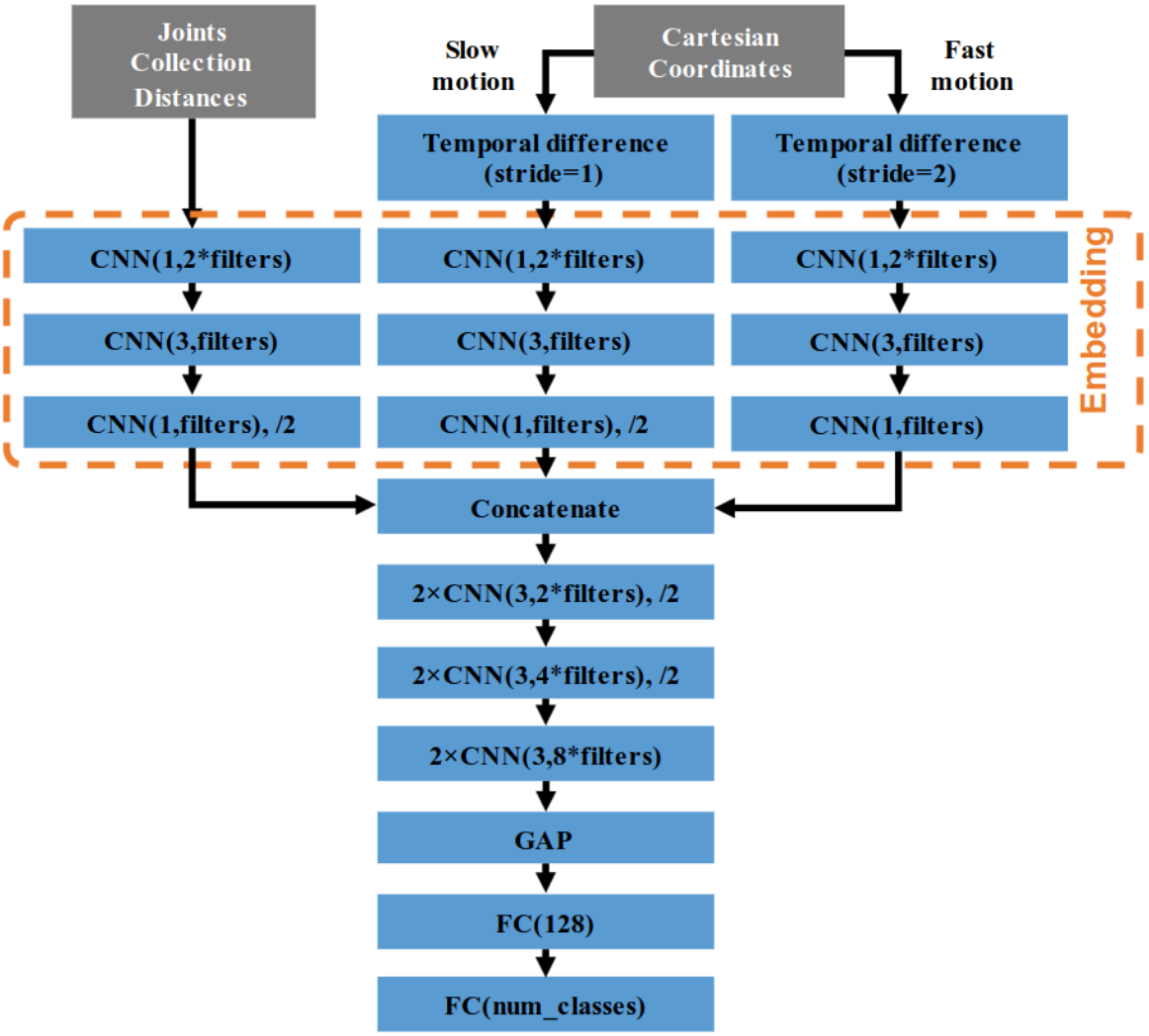

*Рис. 4 — Схема архітектури мережі DDNet. 2×CNN(3, 2*filters), /2" позначає два шари Conv1D із розміром ядра 3 та кількістю каналів 2*filters та Maxpooling із кроком 2. Інші шари ConvNet визначаються у тому ж форматі. GAP позначає Global Average Pooling. FC позначає повнозв'язні шари.*

Класами для навчання моделі (тобто жестами) були: свайп, початок кліку, кінець кліку, вказівник, відсутність жесту та інші. Для навчання моделі DDNet використовувався власний набір даних, що нараховує 15 людей.

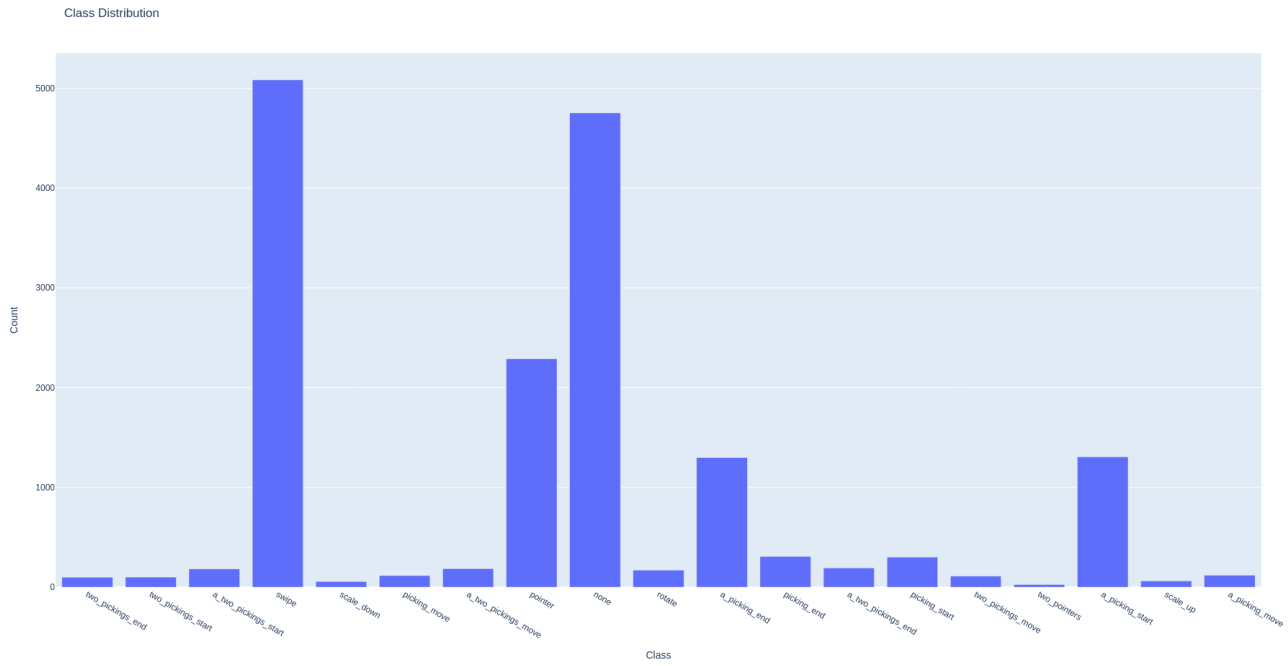

Рис 5. — розподіл кількості жестів серед зібраних даних. Основні жести налічуються від 1000 до 5000 виконань жесту.

2.2 Скелетонізація Mediapipe

Оскільки модель розпізнавання жестів DDNet приймає дані у форматі:

$$P = [p_1, p_2, \dots p_j],\ p = (x_1, x_2, \dots, x_n)$$

наступна задача полягає у тому, аби отримати відповідну інформацію за допомогою моделі скелетонізації. У рамках поставленої задачі необхідно працювати одночасно лише з розпізнаванням жестів рук однієї людини, тому найбільш логічним вибором у даному випадку є *низхідна* скелетонізація.

На початку цього дослідницького шляху для вирішення задачі була обрані моделі нейронних мереж MediaPipe Hands [24] із фреймворку MediaPipe [25]. Дане рішення складається із двох моделей: моделі виявлення об’єктів за один кадр і наступною регресійною моделлю.

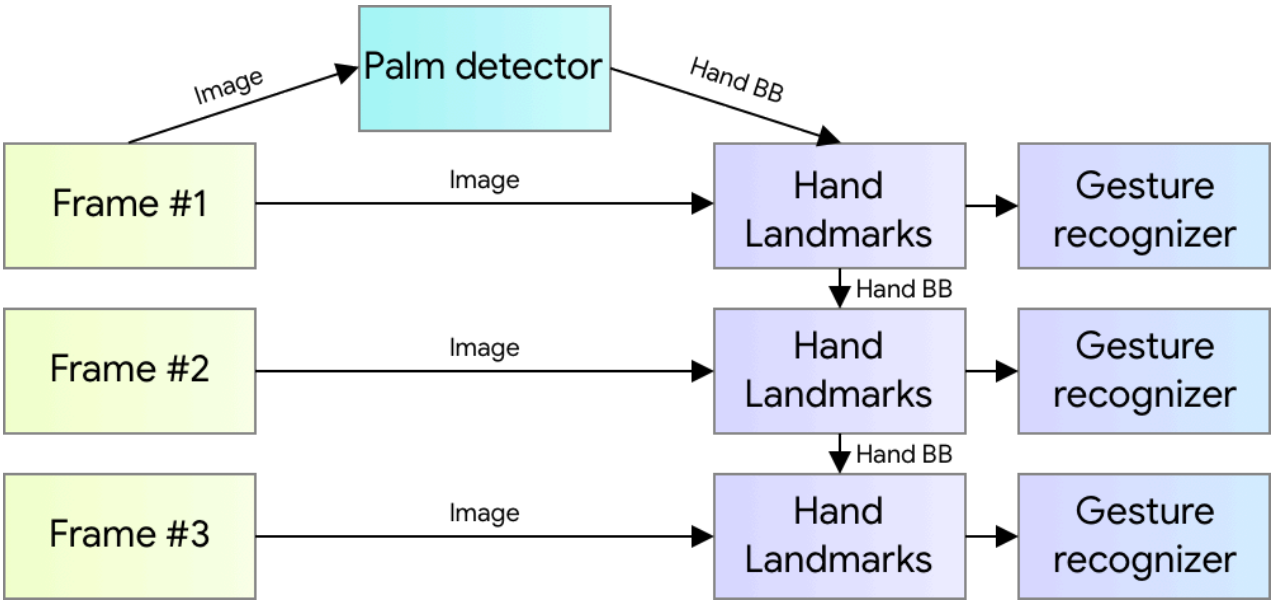

Рис. 6 — Огляд рішення з використанням MediaPipe [26].

Перша модель (під назвою BlazePalm [27]) — це згорткова нейронна мережа, що виявляє долоні, яка працює з повним зображенням, і повертає їх обмежувальні рамки. По-перше, оскільки долоні є досить малими об'єктами, використовується алгоритм *non-max suppression*, причому, долоні знаходять за допомогою квадратних обмежувальних рамок (анкерів), ігноруючи інші співвідношення сторін, що дозволяє зменшити кількість анкерів у 3-5 разів. По-друге, архітектура моделі складається з кодувальника та декодувальника, для

кращого розуміння контексту зображень, [28], що виникають внаслідок високої масштабної дисперсії.

Друга модель — скелетонізатор, яка оперує обрізаною областю зображення, і повертає 3D-орієнтири руки.

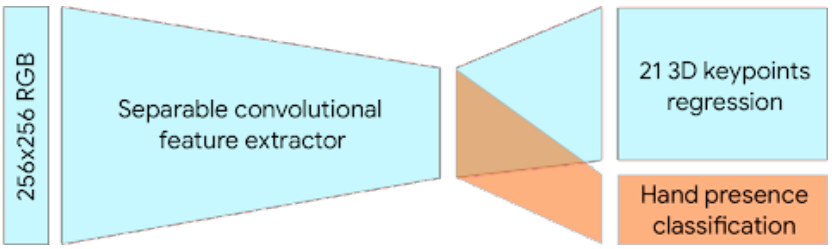

Рис. 7 — Архітектура моделі скелетонізатора у Mediapipe [26].

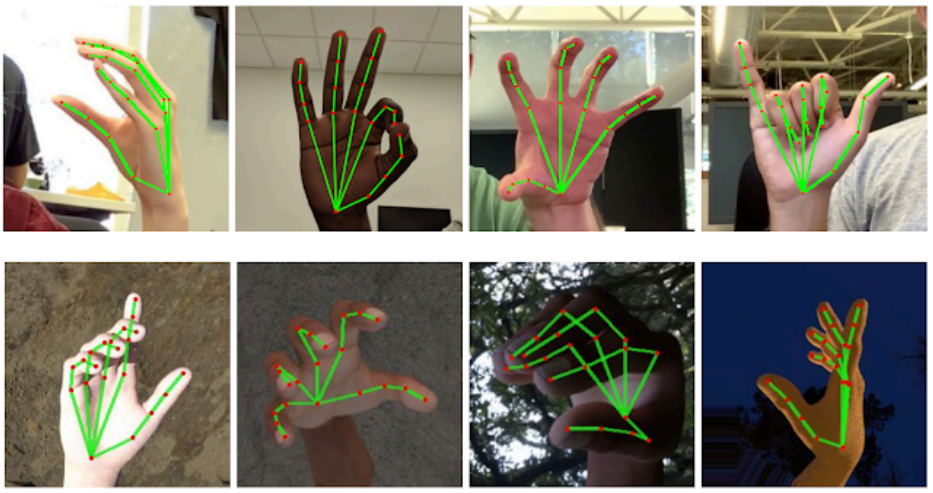

Рис. 8 — Приклад візуалізації роботи скелетонізатора Mediapipe [26].

Автори даної моделі зазначають, що середня квадратична помилка моделі становить 16.1%. Оскільки їх рішення розроблялось, враховуючи жорсткі обмеження щодо технічних вимог під час використання моделі, це досить непоганий результат. Проте, під час експериментів з навчанням моделі розпізнавання жестів DDNet на даних, отриманих за допомогою Mediapipe, виявилося, що це неймовірно сильно впливає на якість. Проблеми скелетонізації під час відносно швидких рухів або ж за неякісного освітлення (які часто зустрічаються в умовах реального використання) результують у розріджені вхідні матриці для моделі розпізнавання жестів і недостатню кількість інформації для якісного навчання.

2.3 Скелетонізація MMPose

Враховуючи недоліки MediaPipe, було прийнято рішення використати більш точні моделі скелетонізації у обмін на збільшення часу обробки. У межах цього дослідження вибір зупинився на моделі RTMPose [30] у рамках фреймворку MMPose [31]. На час публікації моделі RTMPose це рішення було найкращим за ефективністю на мобільних девайсах серед публічно доступних, зберігаючи достатній рівень якості (72.2% середньої влучності на наборі даних COCO [32]).

RTMPose — це модель низхідної скелетонізації. Низхідні алгоритми зазвичай вважаються точними, але повільними, через через додатковий процес виявлення, що впливає на продуктивність у сценах з великою кількістю об’єктів. Однак, завдяки чудовій ефективності детекторів, що працюють в режимі реального часу [29] і [33], частина виявлення більше не є вузьким місцем низхідних алгоритмів скелетонізації.

Спочатку потрібно вирішити задачу виявлення рук у кадрі, для цього було використано модель RTMDet [29]. У цій статті автори прагнули розробити ефективний детектор об'єктів у реальному часі, який перевершує серію YOLO і є легко застосовним для різних об'єктів, у тому числі із різним положенням та кутом повороту. Макроархітектура RTMDet є типовим одноступеневою моделлю виявлення.

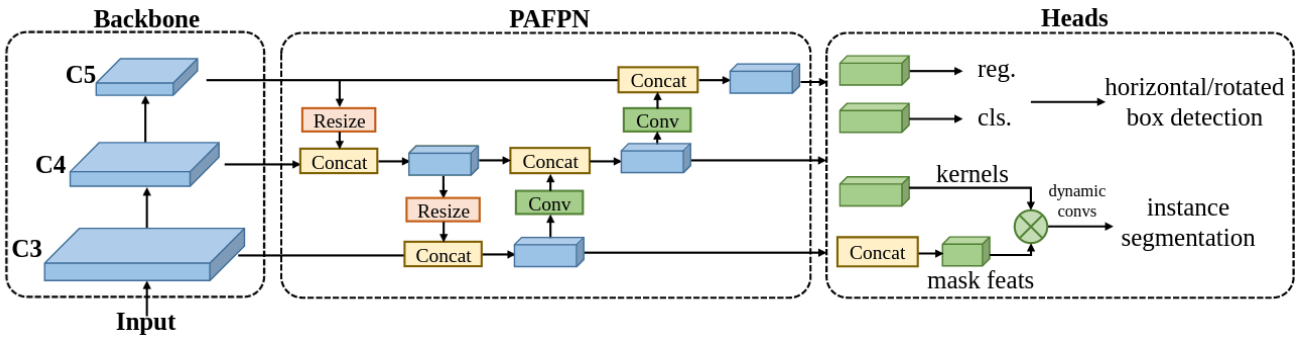

Рис. 9 — Макроархітектура RTMDet [29]. Використовуються CSP-блоки з великими ядрами depth-wise convolution для побудови кістяка. Багаторівневі ознаки, позначені як C3, C4 і C5 потім об'єднуються у шиї моделі CSP-PAFPN, який складається з того ж блоку, що і кістяк. Для виявлення використовуються голови зі спільними вагами згортки.

Класичні підходи серії YOLO зазвичай використовують CSPDarkNet [50] як базову архітектуру, яка містить чотири етапи, і кожен етап складається з декількох базових блоків. Тому автори RTMDet використовують схожий підхід, але зі згортками 5×5 за глибиною в базовому структурному блоці CSPDarkNet для збільшення ефективних рецептивних полів. Такий підхід дозволяє більш комплексне контекстне моделювання та значно підвищує точність. Оскільки, кількість шарів у базовому блоці також збільшується за рахунок додаткової згортки 1x1, це перешкоджає паралельній обробці. Щоб вирішити цю проблему, ми зменшуємо кількість блоків, але кожен з них має велику ширину.

У якості функції втрат використовується наступна, що основа на стратегії dynamic soft label assignment використаній у SimOTA [19]:

$$C = \lambda_1 C_{cls} + \lambda_2 C_{reg} + \lambda_3 C_{center},$$

де C_{cls} *— класифікаційна функція втрат,* C_{reg} *— регресійна функція втрат,* C_{center} *— функція втрат відносно центрів.*

$$C_{cls} = CE(P, Y_{soft}) \times (Y_{soft} - P)^2,$$

де P *— передбачення моделі,* CE *— категоріальна кросс ентропія,* Y_{soft} *— мітки класів із врахуванням IoU аналогічно до [51].*

$$C_{reg} = -log(IoU),$$

де $IoU = \frac{A \cap B}{A \cup B}$ *для обмежувальних рамок A та B.*

$$C_{center} = \alpha^{|x_{pred} - x_{gt}| - \beta},$$

де x_{pred} *— передбачений цент,* x_{gt} *— реальний центр, α та β — гіпер параметри.*

Дана модель є відкритою для використання, і у рамках даного рішення була використана уже натренована версія моделі RTMDet.

RTMPose також використовує модель CSPNeXt [42] як основу, що демонструє хороший баланс швидкості та точності (оскільки вона була розроблена як модель виявлення об’єктів) і є зручною для розгортання. Ця модель скелетонізації знаходить точки за допомогою алгоритму на основі SimCC [41], який розглядає локалізацію ключових точок як задачу класифікації. Порівняно з

алгоритмами на основі теплокарт [37, 38, 39, 40], алгоритм на основі SimCC досягає достатньої точності з меншою обчислювальною ефективністю.

Він являє собою нову схему, яка формулює прогнозування ключових точок як класифікацію з субпіксельних бінів для горизонтальних і вертикальних координат відповідно, що дає кілька переваг. По-перше, SimCC не залежить від теплокарт високої роздільної здатності, що дозволяє створити дуже компактну архітектуру, яка не потребує ні проміжних зображень високої роздільної здатності [42], ні дорогих шарів масштабування [43]. По-друге, SimCC вирівнює остаточну карту ознак для класифікації замість того, щоб використовувати глобальні агрегувальні шари [36], і, таким чином, уникає втрати просторової інформації. По-третє, похибка квантування може бути ефективно зменшена за рахунок класифікації координат на субпіксельному рівні, без необхідності додаткової постобробки [40]. Ці якості роблять SimCC привабливим для побудови легких моделей скелетонізації та їх подальше розгортання на різних бекендах.

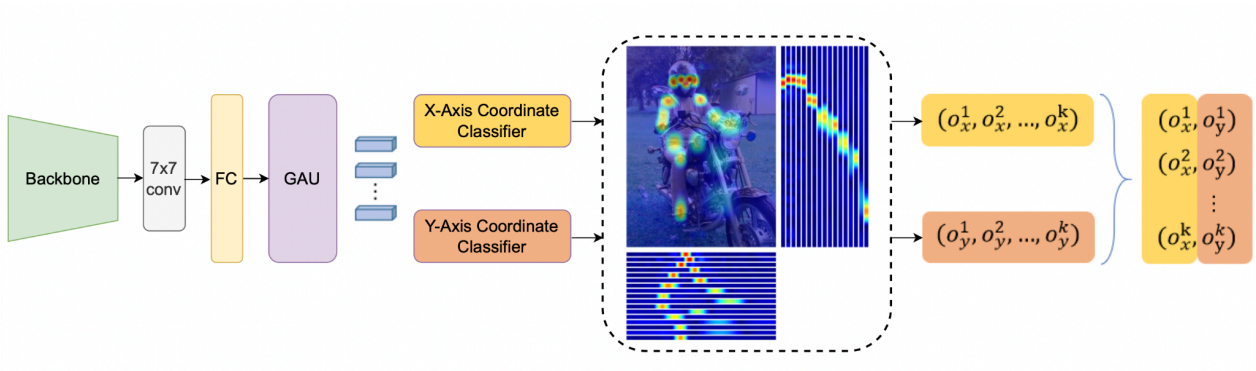

Рис. 10 — Архітектура RTMPose, яка містить згортковий шар, повнозв'язний шар і вентильний вузол уваги (Gated Attention Unit, GAU) для уточнення представлень K ключових точок. Після цього скелетонізація в R^2 розглядається як дві задачі класифікації для координат осі x та осі y для прогнозування горизонтального та вертикального розташування ключових точок [30].

RTMPose також використовує *вентильний вузол уваги* Gated Attention Unit [44], що є спрощеним механізмом уваги. Попередні підходи до оцінки пози за допомогою трансформерів або використовують представлення на основі теплокарт, або зберігають як токени пікселів, так і токени ключових точок, що призводить до високих обчислювальних витрат і ускладнює роботу в реальному

часі. На відміну від них, RTMPose використовує механізм самоуваги (self-attention) з компактним представленням на основі SimCC для фіксації залежностей ключових точок, що значно зменшує обчислювальне навантаження і дозволяє моделі працювати в реальному часі з підвищеною точністю та ефективністю.

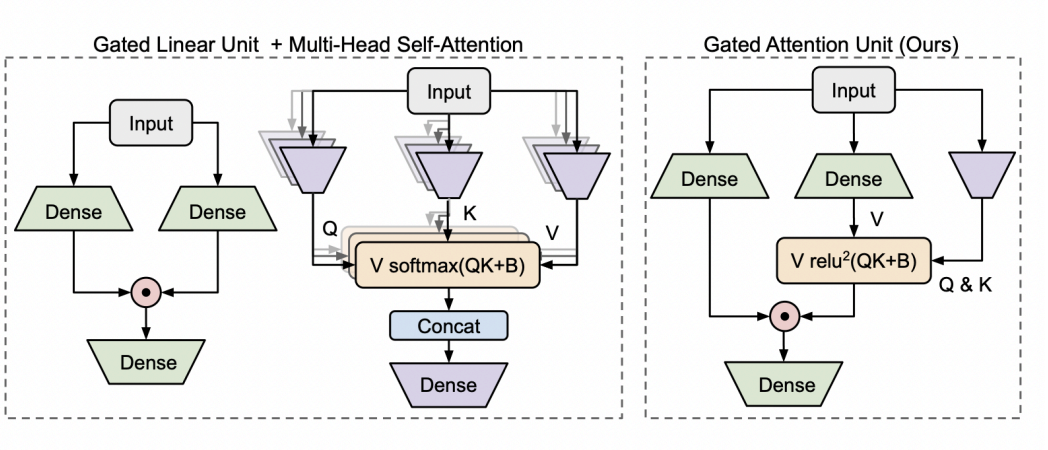

Рис. 11 — Схема роботи вентильного вузла уваги (GAU) [44].

Вентильний вузол уваги використовує менше пам'яті, має більшу швидкість, та кращу продуктивність порівняно з ванільним трансформером. Зокрема, GAU покращує мережі прямого поширення (Feed-Forward Networks) у шарах трансформера за допомогою вентильних лінійних блоків (Gated Linear Unit, GLU) [48], а також елегантним чином визначає механізм уваги:

$$U = \varphi_u(XW_u),$$

$$V = \varphi_v(XW_v),$$

$$O = (U \odot AV)W_o,$$

де φ — *нелінійна функція активації,* $\odot$ — *добуток Адамара,* W_u, W_v та W_o — *вагові матриці (тензори),* X — *тензор вхідних даних.*

Відповідно, самоувага у цьому контексті визначається наступним чином:

$$\Phi = \frac{1}{n} relu^2\left(\frac{Q(x)K(z)^T}{\sqrt{s}}\right),\ Z = \varphi_z(XW_z),$$

де $s = 128$, Q та K це лінійні проєкції тензора вхідних даних, а $relu^2(\cdot)$ це функція ReLU піднесена до квадрату.

Окрім того, RTMPose використовує додаткову оптимізацію під час використання мережі. Застосовуються алгоритм виявлення пропущених кадрів, запропонований у [49], щоб зменшити затримку та алгоритм *non-max suppression* та фільтри згладжування, аби покращити після обробку результатів.

Оскільки RTMPose також є відкритою, була використана її претренована версія.

2.4 Післяобробка

У рамках реального застосування даного рішення для розпізнавання жестів усього лише моделей машинного навчання недостатньо, оскільки їх результати завжди будуть містити неточності. Тому у межах даного дослідження також було перевірено емпіричним шляхом різні методики для покращення роботи комбінації даних моделей.

Для фільтрування результатів роботи моделі скелетонізації було застосовано лінійну інтерполяцію для викидів. Викидами *O* у даному випадку вважаються значення для кадру *i*:

$$P_i \in O \mid \sqrt{(P_i - P_{i-1})^2} > \alpha,\ i \in \{1,\ N\},$$

де α - порогове значення для фільтрування.

Подібний підхід успішно вирішує проблему явних викидів (помилок моделі детекції).

Для фільтрування результатів жеста вказівника використано згладжування за допомогою алгоритми Exponential Moving Weighted Average (EMWA).

$$EMA_t = \alpha x_t + (1 - \alpha) * EMA_{t-1},$$

x_t - координати у момент часу *t* , α - коефіцієнт згладжування ∈ [0; 1]

3. ТЕСТУВАННЯ АЛГОРИТМІВ

3.1 Метрики

Тренування мережі DDNet відбувалось використовуючи розбиття валідаційний даних на 5 на частин (*K-folds cross-validation, K=5*), де у кожній з них не було даних які відповідають одній і тій самій людині. Пошук найкращої моделі відбувався за допомогою сітки пошуку параметрів *(grid search*), а після вибору найкращої за результатами моделей для тренування використовувалися усі дані. Експериментальним чином було обрано тренувати моделі протягом 200 епох, далі метрика влучності (precision) перестає значно покращуватися. У якості оптимізатора градієнтного спуску обрано AdamW [53], що має додатковий параметр який використовується для зменшення ймовірності перенавчання, на відміну від Adam[52].

Нижче наведені графіки для моделі DDNet з використанням Mediapipe для скелетонізації при параметрах $batch\ size\ =\ 64,\ learning\ rate\ =\ 1\ *\ 10^{-4},$ $weight\_decay\ =\ 1\ *\ 10^{-4}$.

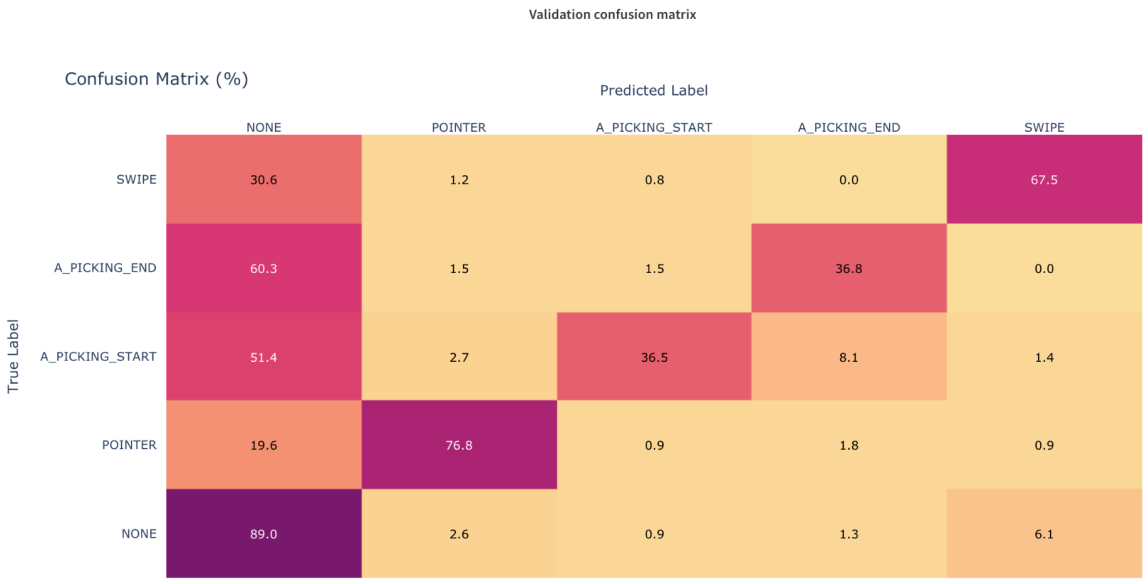

Рис. 12 — Матриця невідповідностей DDNet + Mediapipe.

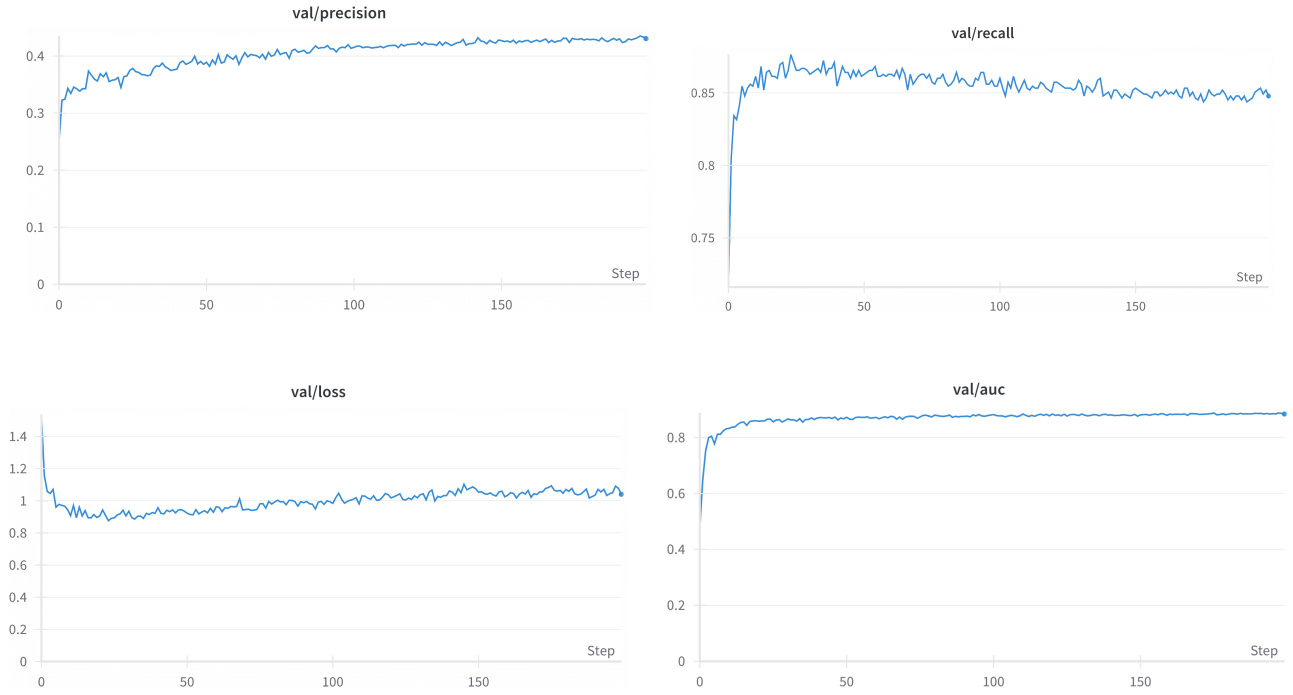

Рис. 13-16 — Графіки влучності, повноти, функції втрат та ROC-кривої відповідно для DDNet + Mediapipe.

Можна помітити, що модель має проблеми із розпізнаванням жесту “None”, тобто забагато хибних спрацювань. Це відбувається через низьку якість моделі скелетонізації Mediapipe та відповідно розріджені матриці вхідних даних.

Нижче наведені графіки для моделі DDNet із використанням MMPose при параметрах $batch\ size\ =\ 128,\ learning\ rate\ =\ 1\ *\ 10^{3},$ $weight\_decay\ =\ 1\ *\ 10^{-5}.$

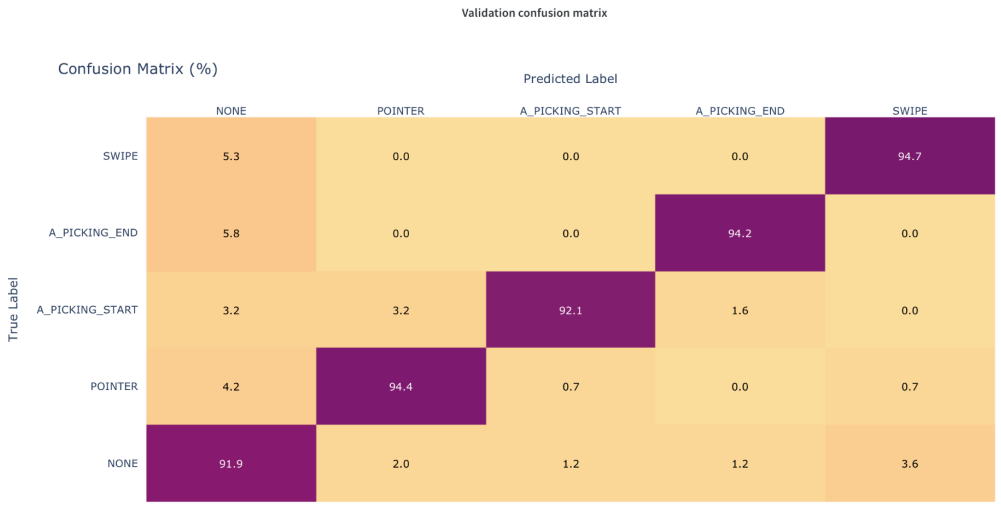

Рис. 17 — *Матриця невідповідностей DDNet + RTMPose.*

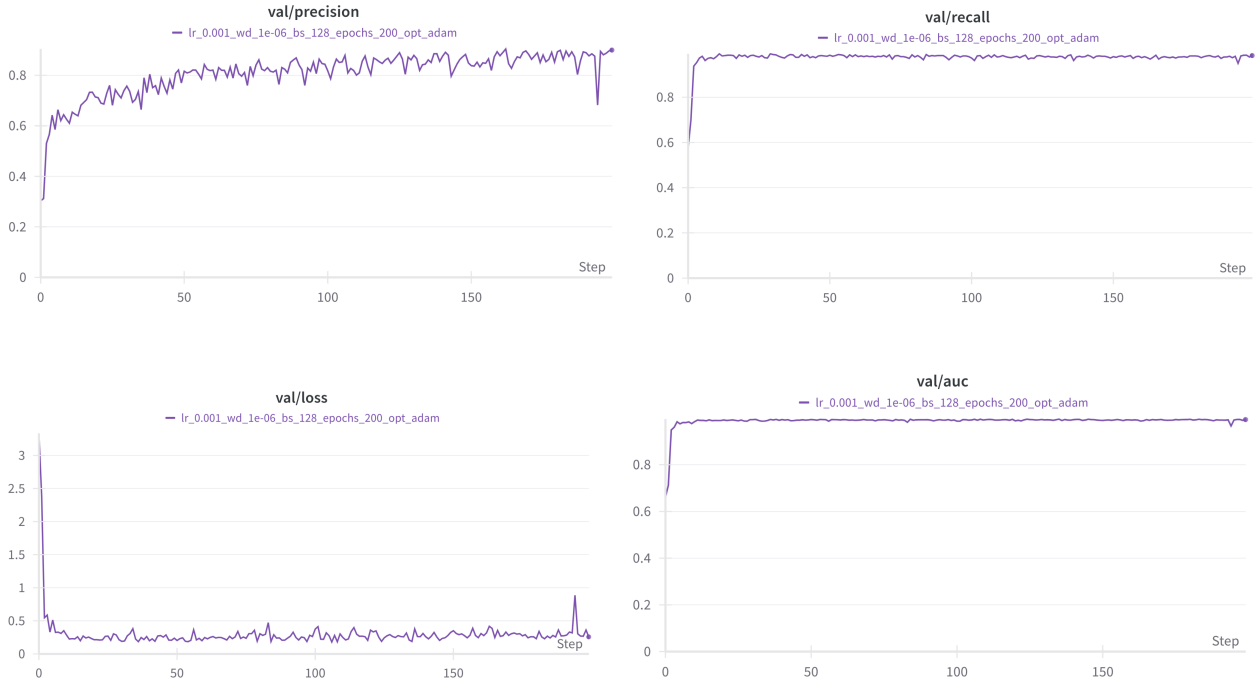

Рис. 18-21 — Графіки влучності, повноти, функції втрат та ROC-кривої відповідно для DDNet + RTMPose.

Очевидно, результати роботи DDNet із якісними вхідними даними є кращими, зокрема для хибно-позитивних результатів.

3.2 Обмеження

Попри усі переваги даного рішення, у нього також є обмеження. Одне з найбільш очевидних: запропонований алгоритм працює з обмеженою кількістю жестів, а для додавання нових потрібно перетреновувати мережу. Від цього важко позбавитися, але дана вимога напряму пов’язана із використанням моделей нейронних мереж.

Найбільший недолік реалізованого у цій роботі підходу пов’язаний з тим, що архітектура моделі DDNet не здатна відрізняти напрямки руху жестів. Тобто для того, аби розрізняти напрямок виконання жестів потрібно використовувати пост-обробку із допомогою алгоритмічних підходів. Це пов’язано із тим, що вхідні дані у тому вигляді, у якому їх приймає модель машинного навчання не містять інформацію про кут повороту відносно будь-якого центру у просторі, а всі координати мають відносну природу. Однак, варто зауважити, що хоч модифікація формату вхідних даних може виправити даний недолік, але це тема окремого дослідження.

Варто також зауважити, що модель виявлення RTMDet скелетонізації RTMPose у класичному вигляді не визначає тип руки (ліва чи права), тож за подібної потреби необхідно редагувати представлену авторами [29] архітектуру.

Критика якості розпізнавання жестів також має місце, адже запропоноване рішення на даний момент не пристосоване до нестандартних умов, наприклад: недостатнього освітлення або коли користувач знаходиться достатньо далеко від камери. Це пов’язано із тим, що помилка однієї з моделей у рішенні (наприклад під час виявлення чи скелетонізації) впливає на розпізнавання жестів. Це проблема властива для багатьох рішень, які розділяють загальну задачі на підзадачі, де останні виконують окремі моделі машинного навчання.

ВИСНОВКИ

У цій курсовій роботі було здійснено глибоке дослідження сучасних методів виявлення об'єктів, процесу скелетонізації та систем розпізнавання жестів, що працюють у реальному часі. Ретельно проведено аналіз переваг та обмежень провідних підходів до скелетонізації, і в контексті цього аналізу було здійснено порівняння результатів, отриманих за допомогою моделей Mediapipe та MMPose, з особливим акцентом на їх вплив на точність і ефективність подальшого розпізнавання жестів.

У ході роботи було розроблено та впроваджено програмне забезпечення на основі моделі DDNet для тренування системи розпізнавання жестів. Ця технологія відкриває значні перспективи для застосування у широкому спектрі сфер, починаючи з автоматизації побутових процесів і закінчуючи розробкою інтерактивних рішень в аугментованій реальності та перекладі жестових мов. Її впровадження має потенціал кардинально змінити способи взаємодії людей з машинами, роблячи їх більш інтуїтивно зрозумілими та зручними.

Загалом, ця курсова робота має на меті не тільки розширити теоретичну базу у сфері розпізнавання жестів, але й стимулювати розробку новітніх практичних рішень. Це, у свою чергу, сприятиме підвищенню рівня інтерактивності та ефективності використання користувацьких пристроїв за допомогою інноваційних методів розпізнавання жестів, відкриваючи нові горизонти для безперервної взаємодії між людиною та технологіями.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Van Rossum G, Drake FL. Python 3 Reference Manual. Scotts Valley, CA: CreateSpace; 2009.
2. Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Rafal Jozefowicz, Yangqing Jia, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Mike Schuster, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.
3. Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zach DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, Soumith Chintala. PyTorch: An Imperative Style, High-Performance Deep Learning Library; 2019.
4. J. K. Aggarwal and M. S. Ryoo, “Human activity analysis: a review,” ACM Computing Surveys, vol. 43, no. 3, pp. 16:1–16:43, 2011.
5. R. Poppe, “A survey on vision-based human action recognition,” Image and Vision Computing, vol. 28, no. 6, pp. 976–990, 2010.
6. Simone Undri Innocenti, Federico Becattini, Federico Pernici, Alberto Del Bimbo. Temporal Binary Representation for Event-Based Action Recognition (2020).
7. G. Devineau, W. Xi, F. Moutarde, J. Yang. Convolutional Neural Networks for Multivariate Time Series Classification using both Inter- & Intra- Channel Parallel Convolutions (2019).
8. Quentin de Smedt, Hazem Wannous, Jean-Philippe Vandeborre, Joris Guerry, Bertrand Le Saux, et al.. SHREC’17 Track: 3D Hand Gesture Recognition Using a Depth and Skeletal Dataset. 3DOR - 10th Eurographics Workshop on 3D Object Retrieval, Apr 2017, Lyon, France. pp.1-6, 10.2312/3dor.20171049. hal-01563505.
9. Hou, Jingxuan et al. “Spatial-Temporal Attention Res-TCN for Skeleton-Based Dynamic Hand Gesture Recognition.” *ECCV Workshops* (2018).
10. V. Choutas, P. Weinzaepfel, J. Revaud and C. Schmid, "PoTion: Pose MoTion Representation for Action Recognition," 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 2018, pp. 7024-7033, doi: 10.1109/CVPR.2018.00734. keywords: {Heating systems;Joints;Image color analysis;Streaming media;Computer architecture;Optical imaging;Standards}.
11. Fan Yang, Sakriani Sakti, Yang Wu, Satoshi Nakamura; Make Skeleton-based Action Recognition Model Smaller, Faster and Better (2019).

12. Kelley HJ. Gradient theory of optimal flight paths. Ars Journal. 1960;30(10):947–54.
13. Agarap AF. Deep learning using rectified linear units (relu). arXiv preprint arXiv:180308375. 2018;
14. Haykin S. Neural networks: a comprehensive foundation. Prentice Hall PTR; 1994.
15. Fukushima, K. (1979). Neural network model for a mechanism of pattern recognition unaffected by shift in position - Neocognitron. Trans. IECE, J62-A(10):658–665
16. Rumelhart, David E; Hinton, Geoffrey E, and Williams, Ronald J (Sept. 1985). Learning internal representations by error propagation. Tech. rep. ICS 8504. San Diego, California: Institute for Cognitive Science, University of California.
17. Zhi Tian, Chunhua Shen, Hao Chen, Tong He. FCOS: A Simple and Strong Anchor-free Object Detector.
18. Jocher, G., Chaurasia, A., & Qiu, J. (2023). Ultralytics YOLO (Version 8.0.0) [Computer software]. https://github.com/ultralytics/ultralytics
19. Zheng Ge, Songtao Liu, Feng Wang, Zeming Li, Jian Sun. YOLOX: Exceeding YOLO Series in 2021 (2021).
20. Ross Girshick. Fast R-CNN (2015).
21. https://paperswithcode.com/paper/make-skeleton-based-action-recognition-model-1
22. *Dynamic Hand Gesture Recognition using Skeleton-based Features* , Quentin De Smedt, Hazem Wannous and Jean-Philippe Vandeborre, 2016 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW).
23. Towards understanding action recognition. Hueihan Jhuang, Juergen Gall, Silvia Zuffi, Cordelia Schmid, Michael J. Black.
24. GHUM & GHUML: Generative 3D Human Shape and Aiculated Pose Models Proceedings of the IEEE/CVF Conference on Computer Vision and Paern Recognition, pages 6184-6193, (2020).
25. Camillo Lugaresi, Jiuqiang Tang, Hadon Nash, Chris McClanahan, Esha Uboweja, Michael Hays, Fan Zhang, Chuo-Ling Chang, Ming Guang Yong, Juhyun Lee, Wan-Teh Chang, Wei Hua, Manfred Georg, Matthias Grundmann. MediaPipe: A Framework for Building Perception Pipelines. https://developers.google.com/mediapipe
26. https://blog.research.google/2019/08/on-device-real-time-hand-tracking-with.html
27. Fan Zhang, Valentin Bazarevsky, Andrey Vakunov, Andrei Tkachenka, George Sung, Chuo-Ling Chang, Matthias Grundmann. MediaPipe Hands: On-device Real-time Hand Tracking (2020).
28. Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, Piotr Dollár. Focal Loss for Dense Object Detection (2017).
29. Chengqi Lyu, Wenwei Zhang, Haian Huang, Yue Zhou, Yudong Wang, Yanyi Liu, Shilong Zhang, Kai Chen. RTMDet: An Empirical Study of Designing Real-Time Object Detectors (2022).

30. Tao Jiang, Peng Lu, Li Zhang, Ningsheng Ma, Rui Han, Chengqi Lyu, Yining Li, Kai Chen. RTMPose: Real-Time Multi-Person Pose Estimation based on MMPose (2023).
31. Kai Chen, Jiaqi Wang, Jiangmiao Pang, Yuhang Cao, Yu Xiong, Xiaoxiao Li, Shuyang Sun, Wansen Feng, Ziwei Liu, Jiarui Xu, Zheng Zhang, Dazhi Cheng, Chenchen Zhu, Tianheng Cheng, Qijie Zhao, Buyu Li, Xin Lu, Rui Zhu, Yue Wu, Jifeng Dai, Jingdong Wang, Jianping Shi, Wanli Ouyang, Chen Change Loy, Dahua Lin. MMDetection: Open MMLab Detection Toolbox and Benchmark (2019).
32. Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Doll´ar, and C Lawrence Zitnick. Microsoft COCO: Common objects in context (ECCV), (2014).
33. RangiLyu. Nanodet-plus: Super fast and high accuracylightweight anchor-free object detection model. https://github.com/RangiLyu/nanodet (2021).
34. Jiefeng Li, Siyuan Bian, Ailing Zeng, Can Wang, Bo Pang, Wentao Liu, and Cewu Lu. Human pose regression with residual log-likelihood estimation. In ICCV (2021).
35. Weian Mao, Yongtao Ge, Chunhua Shen, Zhi Tian, Xinlong Wang, Zhibin Wang, and Anton van den Hengel. Poseur: Direct human pose regression with transformers. In European Conference on Computer Vision, pages 72–88. Springer, (2022).
36. Alexander Toshev and Christian Szegedy. Deeppose: Human pose estimation via deep neural networks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), (2014).
37. Ounjie Huang, Zheng Zhu, Feng Guo, and Guan Huang. The devil is in the details: Delving into unbiased data processing for human pose estimation. In CVPR, pages 5700–5709, (2020).
38. Bin Xiao, Haiping Wu, and Yichen Wei. Simple baselines for human pose estimation and tracking. In the European Conference on Computer Vision (ECCV), (2018).
39. Yufei Xu, Jing Zhang, Qiming Zhang, and Dacheng Tao. Vitpose: Simple vision transformer baselines for human pose estimation, (2022).
40. Feng Zhang, Xiatian Zhu, Hanbin Dai, Mao Ye, and Ce Zhu. Distribution-aware coordinate representation for human pose estimation. In IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), (2020).
41. Yanjie Li, Sen Yang, Peidong Liu, Shoukui Zhang, Yunx- iao Wang, Zhicheng Wang, Wankou Yang, and Shu-Tao Xia. Simcc: a simple coordinate classification perspective for hu- man pose estimation, (2021).
42. Ke Sun, Bin Xiao, Dong Liu, and Jingdong Wang. Deep high-resolution representation learning for human pose estimation. In CVPR, (2019).
43. Jiahong Wu, He Zheng, Bo Zhao, Yixin Li, Baoming Yan, Rui Liang, Wenjia Wang, Shipei Zhou, Guosen Lin, Yanwei Fu, Yizhou Wang, and Yonggang Wang. Ai challenger : A large-scale dataset for going deeper in image understanding. arXiv: Computer Vision and Pattern Recognition, (2017).
44. Weizhe Hua, Zihang Dai, Hanxiao Liu, and Quoc V. Le. Transformer quality in linear time. ArXiv, abs/2202.10447, (2022).

45. Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszko- reit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. Advances in neural information processing systems, 30, (2017).
46. https://magazine.sebastianraschka.com/p/understanding-and-coding-self-attention
47. Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, Neil Houlsby. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale (2020).
48. Noam Shazeer. Glu variants improve transformer, (2020).
49. Valentin Bazarevsky, Ivan Grishchenko, Karthik Raveendran, Tyler Zhu, Fan Zhang, Matthias Grundmann. BlazePose: On-device Real-time Body Pose tracking (2020).
50. Alexey Bochkovskiy, Chien-Yao Wang, and Hong-Yuan Mark Liao. YOLOv4: Optimal speed and accuracy of object detection. CoRR, abs/2004.10934, (2020).
51. Xiang Li, Chengqi Lv, Wenhai Wang, Gang Li, Lingfeng Yang, and Jian Yang. Generalized focal loss: Towards efficient representation learning for dense object detection. IEEE Transactions on Pattern Analysis and Machine Intelligence, pages 1–14, (2022).
52. Diederik P. Kingma, Jimmy Ba. Adam: A Method for Stochastic Optimization (2017).
53. Ilya Loshchilov & Frank Hutter. Decoupled Weight Decay Regularization (2017).