КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ТАРАСА ШЕВЧЕНКА

Факультет радіофізики, електроніки та комп'ютерних систем

Кафедра комп'ютерної інженерії

Системи розпізнавання жестів як інтерфейс користувача персонального комп'ютера

Дипломна робота магістра
студента II курсу ОР «Магістр»

спеціальності 123 «Комп'ютерна інженерія»

ОП «Комп'ютерні системи та мережі»

Олександра КУРТИ

Науковий керівник:
асистент **Юрій ЮРЧИК**

Рецензент:

Доцент кафедри конструювання
електронно-обчислювальної апаратури,
факультету електроніки
Київського політехнічного інституту
імені Ігоря Сікорського

Денис ЛЕБЕДЕВ

До захисту допускаю:
Завідувач кафедрою,
к. ф.-м. н., доцент Юрій БОЙКО

Ухвалено на засіданні кафедри "__ " _____ 2023 р., протокол No

Київ – 2023

РЕФЕРАТ

Робота присвячена створенню інтерфейсу людина-машина для подальшої імплементації в застосунок для персонального комп’ютера з використанням нейронних мереж сучасних архітектур. Імплементована модель розпізнавання жестів на основі моделі рекуррентної нейронної мережі, дозволяє досягнути середніх значень метрик точності розпізнавання 91.6%. Створено застосунок для взаємодії користувача з ПК за допомогою жестів. Проаналізувавши аналоги було виявлено, що застосунок має функціональну перевагу у вигляді можливості назначення власної дії на певний жест.

Дипломна робота магістра складається з 41 сторінку, 13 рисунків, 1 таблиці, 3 лістинги коду, 20 посилань на джерела, 3 додатки.

Ключові слова: розпізнавання жестів, модель руки, MediaPipe Hands, рекурентна нейронна мережа, LSTM, GRU, Bidirectional LSTM, макрос.

ЗМІСТ

РЕФЕРАТ **2**
ЗМІСТ **3**
СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАЧЕННЯ **5**
ВСТУП **6**
РОЗДІЛ 1. АНАЛІТИЧНИЙ ОГЛЯД ІНСТРУМЕНТІВ **7**
1.1 Основні етапи розпізнавання жестів 7
1.2 Методи збору початкових данних 8
1.3 Огляд інструментів для початкового збору даних 10
1.3.1 OpenPose 10
1.3.2 MediaPipe Hands 11
РОЗДІЛ 2. АНАЛІЗ ІСНУЮЧИХ ПІДХОДІВ ДО РОЗПІЗНАВАННЯ ЖЕСТІВ **15**
2.1 Аналіз різниці зображень 15
2.2 Гістограмма напрямків 16
2.3 Кольорові рукавички 17
2.5 Аналіз контуру зображень 18
2.6 Нейронні мережі 19
РОЗДІЛ 3. ЗАГАЛЬНИЙ ПРИНЦИП РОБОТИ ЗАСТОСУНКУ **21**
3.1 Застосування MediaPipe Hands 22
3.2 Попередня обробка навчальних даних 24
3.3 Використані інструменти 26
3.5 Набір тренувальних даних 28
3.6 Результати досліджень 30
3.7 Порівняння моделей з різними типами шарів 31
3.7.1 Аналіз метрик ефективності нейронної мережі з різними шарами 32
3.7.2 Матриця плутанини (Confusion Matrix) 34
3.8 Огляд застосунку 36
3.9 Порівняння з аналогами 37
ВИСНОВКИ **40**
ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ **41**

Додаток А **44**
Додаток Б..... **45**
Додаток В..... **48**

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАЧЕННЯ

2D/3D зображення – двовимірне/трьохвимірне зображення

GFLOP - величина, що використовується для вимірювання продуктивності обчислювальних систем

Mlaas(Machine Learning as a Service) - машинне навчання як послуга це група служб, які надають інструменти машинного навчання (ML) як складову служб хмарних обчислень. Постачальник послуг у машинному навчанні як службі надає такі інструменти, як глибоке навчання, візуалізація даних, прогнозний аналіз, розпізнавання тощо.

GPU(graphics processing unit) - окремий пристрій персонального комп'ютера або ігрової приставки, виконує графічний рендеринг.
CPU(Central processing unit) - функціональна частина комп'ютера, що призначена для інтерпретації команд
TPU(tensor processing unit) - це інтегральна схема специфічного застосування (ASIC) призначена для прискорення розрахунків штучного інтелекту
ASL(American Sign Language) - основна мова жестів в спільнотах глухих США і англомовній частині Канади

ВСТУП

Інформаційні технології стали неодмінною частиною нашого життя, і забезпечення ефективної комунікації є однією з головних задач сучасного світу.

Перспективним напрямком розвитку інформаційних технологій є розробка нових способів забезпечення інтерфейсу людина-машина. Перед розробниками подібних систем ставиться задача використання природних для людини способів для комунікації з комп'ютерами.

Враховуючи усі можливі перешкоди та наявність інформаційних шумів в оточуючому середовищі, перевага надається системам на основі комп'ютерного зору. Особливо перспективними для побудови інтерфейсів управління програмним та апаратним забезпеченням комп'ютерів є жести. Розпізнавання жестів дозволить людині спілкуватися і взаємодіяти з машинами та інженерними пристроями інтуїтивно зрозумілим шляхом. Іншим, не менш важливим плюсом такого інтерфейсу є те, що жести дозволяють розширити можливості обміну інформацією для людей з вадами слуху і мови, та забезпечити дистанційне управління різними побутовими пристроями.

Таким чином, метою дипломної роботи магістра є розробка застосунку, що реалізує інтерфейс перетворення жестів в комп'ютерні команди на основі рекурентних нейронних мереж.

Для досягнення мети поставлено такі задачі:

- Побудова моделі рекурентної нейронної мережі, що здатна класифікувати подані жести.
- Імплементація застосунку на основі даної моделі, що дозволить користувачу персонального комп'ютера транслювати жести в дії.

РОЗДІЛ 1. АНАЛІТИЧНИЙ ОГЛЯД ІНСТРУМЕНТІВ

1.1 Основні етапи розпізнавання жестів

Розглянувши та проаналізувавши наявні системи для розпізнавання жестів можна зробити наступні висновки.

Робота системи складається з основних етапів:

1. отримання зображення – об’єкт фіксується за допомогою камер, які підключені до комп’ютера;
2. локалізація руки на зображенні та виділення пальців – на отриманому зображенні знаходиться ділянка руки;
3. розпізнавання жестів – виділення характерних ознак;
4. класифікація жестів та присвоєння розпізнаному жесту певної реакції від системи.

Першим етапом будь-якої системи розпізнавання є початковий збір даних. Отримане зображення обробляється для відділення області руки від фону.

Після збору даних стає можливим використання первинної інформації про руку для фільтрації даних та видалення шумів на зображенні, які можуть виникнути, наприклад, через нерівномірне освітлення, а також видалення артефактів. Дана процедура є надзвичайно важливою, через те, що напряму впливає на якість розпізнавання та правдивість цих даних. На етапі розпізнавання рухів рук здійснюють виділення ознак. Вибір ознаки є важливим, через те що жести рук відрізняються варіативністю форм і рухів. Для розпізнавання статичного положення руки застосовують геометричні ознаки, такі як кінчики пальців, напрямок пальців. Але такі ознаки не завжди доступні та не завжди надійні через самозатінення та умови освітлення.

Наступним етапом буде визначення конкретних жестів на основі аналізу відфільтрованих даних, які несуть інформацію про рух руки. З цією метою проводиться процедура класифікації. Але перед цим систему необхідно «навчити» реагувати на жести, та адаптувати їх для конкретних рухів рук користувача.

В результаті навчання система поступово здобуває здатність зіставляти потрібні реакції та певні сукупності зовнішніх впливів. Об’єктами навчання можуть виступати візуальні зображення рук, зокрема набір жестів з алфавіту глухонімих, що розширить можливості інтерфейсу для людей з дефектами слуху.

1.2 Методи збору початкових данних

Для отримання інформації про виконуваний жест, найчастіше використовуються дві технології: контактні та безконтактні (оптичні методи). У контактних технологіях, використовується інформаційна рукавичка (data glove), яка фіксує рухи руки користувача та передає їх до комп'ютера, формуючи модель руки за допомогою програмних засобів. Незважаючи на високу завадостійкість та можливість здійснювати процеси в режимі реального часу, контактні технології мають недоліки у вимозі до одягання рукавички, яка накладає обмеження на користувача. З цієї причини, безконтактна технологія є більш практичною та ефективною, оскільки звільняє рух рук користувача від обмежень. Для розпізнавання жестів руки за допомогою безконтактних технологій, необхідно зібрати інформацію про руку за допомогою однієї або кількох камер.

Існують різні системи, які можна класифікувати наступним чином:

1. Стереографічна система надає детальну по піксельну інформацію для будь-якої точки в полі зору камер і забезпечує великий об’єм інформації про руку. Ця система дозволяє легко відокремити пальці руки на фоні шкіри, оскільки вони знаходяться

ближче до камери. Проте, для обчислення 3D даних знадобиться багато операційного часу, що виключає використання алгоритмів у режимі реального часу.

2. Багатокамерна 2D система спостереження буде видавати меншу кількість інформації у порівнянні зі стереографічною, але й потребуватиме менше часу на обробку. Два або більше 2D зображення руки, отримані з різних камер, можна об’єднати для розпізнавання жестів. Достатня кількість камер забезпечить повний об’єм інформації для визначення будь-якого жесту.

3. Однокамерна система буде забезпечувати значно меншу кількість інформації про руку. Наприклад, палець на фоні шкіри буде дуже важко визначити, оскільки буде відсутня інформація про його товщину. В основному точно може бути визначена тільки інформація про контур. Дані про контур будуть відносно позбавлені від шумів, а на їх обчислення знадобиться невелика кількість часу.

1.3 Огляд інструментів для початкового збору даних

1.3.1 OpenPose

OpenPose - бібліотека ідентифікації пози людини в режимі реального часу, розроблена дослідниками Університету Карнегі-Меллона. Це один з сучасних підходів для оцінки постави людини в реальному часі. База коду з відкритим вихідним кодом задокументована та доступна на публічному репозиторії GitHub. Одні з найбільш примітних аспектів бібліотеки OpenPose для виявлення пози людини:

- Тривимірне визначення ключових точок від однієї особи в режимі реального часу
- Виявлення ключових точок у 2D для кількох осіб у режимі реального часу
- Відстеження однієї особи для прискорення розпізнавання та згладження візуальних ефектів

Перші кілька шарів використовуються пакетом OpenPose для вилучення функцій із зображення. Потім деталі надсилаються на два паралельні шари згорткових мереж. Перший розділ передбачає набір із 18 карт впевненості, кожна з яких відповідає унікальній частині скелета людської позиції. Наступна гілка передбачає додаткові 38 полів спорідненості частин (PAF), які показують рівень зв’язку між частинами.

Основний недолік OpenPose полягає в тому, що його результати мають низьку роздільну здатність, що обмежує рівень деталізації в оцінках ключових точок. Як наслідок, OpenPose менш підходить для додатків, які вимагають високого рівня точності в оцінці кінематики руху, наприклад для елітного спорту та медичних оцінок. Крім того, OpenPose визнано надзвичайно неефективним, оскільки кожен

висновок коштує 160 мільярдів операцій із плаваючою комою (GFLOP)[16]. Попри ці проблеми, OpenPose все ще залишається популярною мережею для розпізнавання жестів для однієї людини, яка виконує безмаркерне захоплення руху.

1.3.2 MediaPipe Hands

MediaPipe — це міжплатформна конвеєрна структура для розробки рішень машинного навчання, які Google використовує всередині кількох продуктів і послуг. Спочатку він був створений для аналізу відео та аудіо публічного сервісу YouTube у реальному часі. Цей фреймворк із відкритим кодом наразі перебуває на стадії альфа-версії та охоплює операційні системи Android, iOS і вбудовані пристрої, такі як Raspberry Pi.

MediaPipe розроблено для команд машинного навчання як послуги (MLaas), інженерів-програмістів, студентів і дослідників, які публікують код і прототипи в рамках своїх дослідницьких проектів.

Фреймворк MediaPipe в основному використовується для швидкої розробки конвеєрів бачення, які включають багаторазові частини та моделі штучного інтелекту для висновків. Крім того, це полегшує інтеграцію програмного забезпечення комп’ютерного бачення в програми та демонстрації, що працюють на багатьох апаратних платформах. Команди можуть поступово розробляти конвеєри комп’ютерного бачення за допомогою мови конфігурації та інструментів оцінки.

MediaPipe ділиться на три основні частини:

- Структура для висновків на основі сенсорного введення
- Набір інструментів для оцінки продуктивності
- Бібліотека повторно використовуваних компонентів висновків і обробки.

Розробник може поступово прототипувати конвеєр за допомогою MediaPipe. MediaPipe пропонує зразки коду та демонстрації для MediaPipe у Python та MediaPipe у JavaScript. Щоб використовувати рішення MediaPipe, потрібен лише невеликий обсяг коду. MediaPipe підтримує операційні системи Mac OS X, Debian

Linux, iOS та Android. Фреймворк MediaPipe відносно легко адаптувати до інших систем, оскільки він побудований на бібліотеці C++.

Переваги:

- Наскрізне прискорення: логічний висновок і обробка відео швидко вбудовані за допомогою стандартного апаратного забезпечення, такого як GPU, CPU або TPU.
- Просте розгортання: уніфікована структура ідеально підходить для таких платформ, як мобільні операційні системи: Android, iOS, десктопні операційні системи: MacOS, Linux, Web та Інтернет речей (IoT).
- “Рішення з коробки”: весь потенціал середовища MediaPipe демонструється за допомогою готових додатків ML.
- Безкоштовний і відкритий код: фреймворк є повністю гнучким, розширюваним і випущеним за ліцензією Apache 2.0.

Наразі MediaPipe знаходиться в стадії активної розробки та містить обширну документацію, включаючи демонстрації та приклади використання вбудованих функцій. На момент тестування MediaPipe доступний в альфа-версії версії 0.7.

Обидва рішення мають хорошу точність у визначенні орієнтирів людського тіла при рендерингу щодо статичних зображень при поганому освітленні. Що стосується перекриття, MediaPipe та OpenPose працюють задовільно, але слід враховувати точність роботи у разі перекриття рук або обличчя рук, особливо вище 50% зони перекриття може бути нижчою. Найбільшою проблемою з обома рішеннями є розмиття руху, яке, зі збільшенням швидкості руху, призвело до великих помилок у поданні положення орієнтирів навіть до повної втрати виявлення. У цій галузі MediaPipe виявивляється набагато ефективнішим у боротьбі з завадами.

Дослідження[16] враховувало необхідну кількість доступних обчислювальних ресурсів у вигляді використання GPU/CPU, а також швидкість обробки відео в реальному часі, виражену в частоті кадрів за секунду [FPS]. Експеримент проводився на 6 різних вхідних роздільних здатностях. Використовувана вхідна роздільна здатність: -1×64px, -1×128px, -1x368px, -1x480px, -1x608px для OpenPose та 256x256px для MediaPipe. Знак -1 означає, що роздільна здатність буде адаптована для збереження співвідношення сторін джерела вхідного сигналу. Наприклад, -1x368, 656x-1 і 656x368 призведе до однакової точної роздільної здатності для вхідних зображень 720p і 1080p. Стандартна роздільна здатність моделі оцінки пози становить -1x368 для OpenPose та 256x256px для MediaPipe. Результати тесту[16] показані на діаграмах нижче.

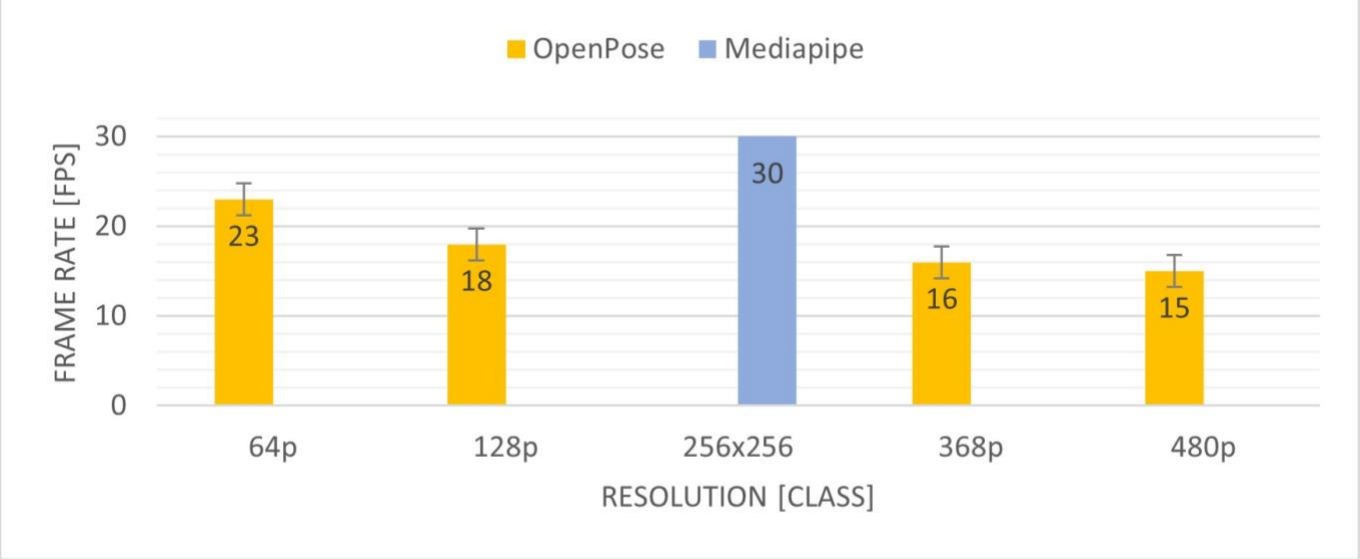

Рис.1 Порівняльна діаграма з різними значеннями роздільної здатності

Рис.2 Використання GPU залежно від роздільної здатності

Як видно з рис. 1, збільшення роздільної здатності вхідного зображення трохи вплинуло на швидкість обробки за допомогою OpenPose. Швидкість обробки зображення для налаштувань за замовчуванням становила 16 FPS і коливалася на 1-2 кадри під час обробки відео. Крім того, збільшення роздільної здатності вхідного зображення було пов'язане зі значним зростанням потреби в пам'яті графічного процесора (рис. 2), що унеможливлювало використання OpenPose із високим значенням параметра роздільної здатності вхідного зображення з камери. Зниження роздільної здатності вхідного зображення у своєю чергою погіршує отримані результати, бо на порозі (-1x64) відбулася повна втрата здатності визначати характерні точки тіла людини.

Обробка зображення за допомогою MediaPipe у налаштуваннях за замовчуванням не призвела до падіння FPS обробленого відео. Через відсутність зниження FPS параметр роздільної здатності не піддавався подальшому тестуванню в цьому дослідженні.

Беручи до уваги наведені вище дані, можна зробити висновок, що модель MediaPipe ефективніша за відповідну модель OpenPose та більше підходить для розробки власного блоку для розпізнавання жестів у власному застосунку.

Виявлення жестів з використанням результатів OpenPose можливе, але, робота з OpenPose була дуже повільною проти MediaPipe. Обробка відео займає багато часу і займає багато обчислювальної потужності навіть на машинах з більше ніж 30 ГБ пам’яті GPU, що досить багато. Також MediaPipe простий в інсталяції та має гарно задокументоване API для Python, що був вибраний основною мовою написання застосунку. Тому було вирішено використати MediaPipe для розв'язання проблеми розпізнавання ключових точок долоні на відеоматеріалах.

РОЗДІЛ 2. АНАЛІЗ ІСНУЮЧИХ ПІДХОДІВ ДО РОЗПІЗНАВАННЯ ЖЕСТІВ

2.1 Аналіз різниці зображень

Аналіз різниці кадрів відеоряду є потужним інструментом для визначення різних аспектів на відео. Цей підхід полягає у порівнянні пікселів на послідовних кадрах і виявленні змін між ними. Ці зміни можуть вказувати на рух об'єктів, зміни в освітленні, появу або зникнення об'єктів і багато іншого.

Одним з основних застосувань аналізу різниці кадрів є виявлення руху об'єктів. Це може бути корисно в системах відеоспостереження для автоматичного виявлення незвичайної активності або відстеження об'єктів у режимі реального часу. Зміни в різниці кадрів можуть слугувати ознакою руху об'єктів, і їх можна використовувати для активації сповіщень або подальшого аналізу.

Крім того, аналіз різниці кадрів може бути використаний для визначення статичних або динамічних об'єктів на відео. За допомогою порівняння пікселів на кадрах можна виділити області зі значними змінами та використовувати ці дані для сегментації об'єктів або виявлення їх контурів. Аналіз різниці кадрів також може бути використаний для виявлення руху камери (камери що тремтить або пересуваються). Це може бути корисно для компенсації руху камери або для стабілізації відео. Розрізнення між кадрами відеоряду можна використовувати для аналізу руху об'єктів у відео в реальному часі, навіть якщо зображення не є

однорідним. Такі технології, як, наприклад, Motion History Image (MHI), знайшли застосування в інтерактивних додатках, таких як віртуальний тренер з аеробіки та інтерактивна кімната для розповіді історій.

2.2 Гістограмма напрямків

У багатьох додатках комп'ютерного зору, які використовуються для розпізнавання позиції та орієнтації людини, потрібна додаткова інформація про параметри рук. Ця проблема вирішується за допомогою гістограм напрямків та карт напрямків зображень, які є методами аналізу текстурного зображення.

Гістограма напрямків є векторним представленням зображення, яке відображає розподіл орієнтацій локальних градієнтів пікселів. Цей підхід дозволяє виявляти особливості, такі як краї та текстури, на зображенні незалежно від освітлення. Для побудови гістограми напрямків, зображення зазвичай розбивають на малий набір блоків або клітинок. Для кожної клітинки обчислюються градієнти пікселів, а потім розподіл градієнтів групується за напрямками від 0 до 360 градусів, утворюючи гістограму напрямків. Наприклад, для розпізнавання жестів руки, можна побудувати гістограму напрямків для кожного блоку зображення, і потім порівняти ці гістограми з попередньо навченими шаблонами жестів. Це дозволяє визначити, який жест виконує користувач.

Додатки, які використовують цю технологію, можуть працювати в реальному часі та показувати добрі результати при незначних змінах розміру рук, але можуть бути вразливі до змін у рухах рук, тому для кожної особи, яка користується цими додатками, може бути потрібне окреме навчання.

За допомогою цієї технології можна розпізнавати жести людини в режимі реального часу, якщо виконуються такі умови:

- Рука повинна займати значну частину зображення.
- Зображення має однорідний фон.

- Жести обрані таким чином, щоб гістограми напрямків виявляли значні відмінності один від одного.

2.3 Кольорові рукавички

З метою розпізнавання жестів рук, використовуються кольорові рукавички, що дозволяють точно відстежувати рухи рук у реальному часі за допомогою вбудованої відеокамери. Конструкція рукавички включає двадцять сегментів різних кольорів, розташованих у визначеній послідовності.

Кожен сегмент має свій унікальний колір(рис.3), який може бути розпізнаний алгоритмом обробки зображень. Завдяки використанню різних кольорів та їх унікального розташування, можна точно визначати положення та орієнтацію кожного сегмента рукавички у просторі. Для розпізнавання жестів рук, використовується алгоритм, який аналізує зображення рукавички та виявляє кольорові сегменти. Цей алгоритм може використовувати методи комп'ютерного зору, такі як сегментація, виявлення контурів та аналіз розподілу кольорів, для точного визначення положення та орієнтації рукавички. Завдяки великій кількості кольорів у рукавичці, цей підхід дозволяє розпізнавати жести рук навіть при змінах у рівні освітлення середовища.

Рис. 3 Приклад кольорових рукавичок

Вона знаходить широке застосування у віртуальній реальності, взаємодії з комп'ютерними системами, іграх, задачах управління анімаційними персонажами та розпізнавання жестів ручної азбуки глухонімих ASL та інших областях, де точне визначення положення та рухів рук є важливим.

2.5 Аналіз контуру зображень

Аналіз контуру руки в кольоровому зображенні часто використовується для визначення конфігурації руки, зокрема для розпізнавання жестів ручної азбуки глухонімих ASL. Початково, в кольоровому зображенні руки виконується процес видалення точок, що не відповідають кольору шкіри, і отримане зображення перетворюється в бінарне зображення. Після цього застосовується фільтр Гауса для згладжування зображення. Для виявлення контурів руки використовується оператор Собеля. Після цього обчислюється описаний прямокутник навколо контуру руки, з центром у початку координат. Аналізуючи контур руки в кольоровому зображенні, можна визначити позицію кінчиків пальців, використовуючи аналіз локальних вигинів контуру.

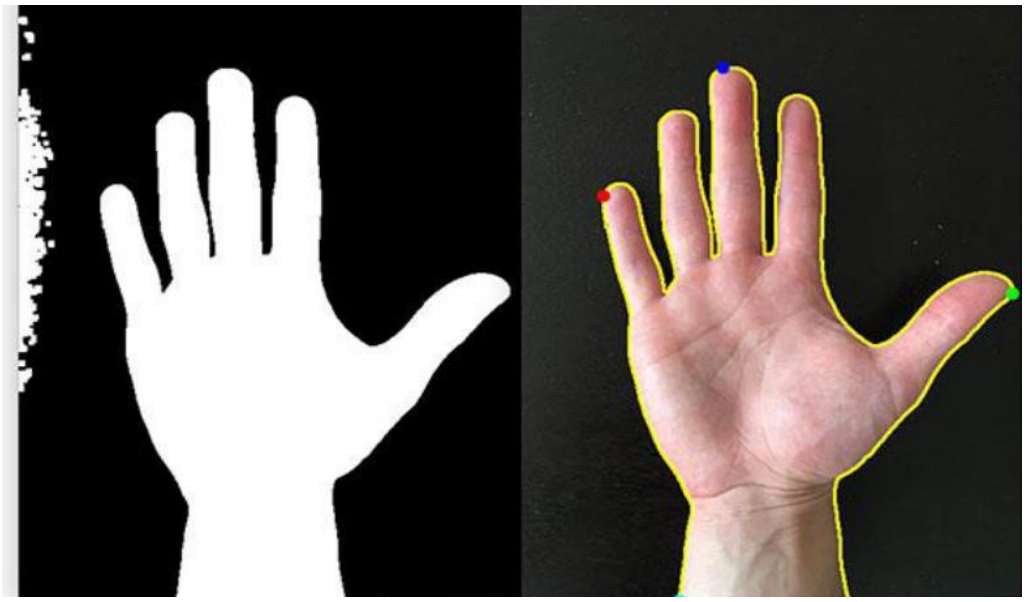

Рис.4 Приклад відокремлення контуру долоні від фону

На рисунку 4 представлена задача розпізнавання позиції кінчиків пальців руки, заснована на аналізі контуру руки в кольоровому зображенні. Перші етапи алгоритму, як і раніше, включають видалення фонових точок, згладжування і виділення зображення руки. Після отримання контуру руки, використовується аналіз локальних вигинів контуру для виділення пальців руки. Цей алгоритм був використаний для управління вебкамерою персонального комп'ютера.
На малюнку ліворуч – рука з видаленим фоном, праворуч – показана реалізація цього алгоритму. Проблема цього алгоритму: розпізнаються лише прості жести, через фон з різними природними шумами жест може бути не розпізнаним. На роботу алгоритму впливає й освітлення. Метод рідко використовується самостійно, найчастіше цей метод включають у процес попередньої обробки зображення.

2.6 Нейронні мережі

Математична модель штучних нейронних мереж базується на організації та функціонуванні біологічних нейронних мереж. Початково, основною метою цього підходу було вирішення задач таким самим способом, як це робить людський мозок. Штучні нейронні мережі використовуються в багатьох різноманітних задачах, таких як комп'ютерне бачення, розпізнавання мовлення, машинний переклад, соціально-мережеве фільтрування, настільні та відеоігри, медичне діагностування, а також задачі розпізнавання. Наприклад, при розпізнаванні жестів, вхідні математичні характеристики жесту передаються до штучної нейронної мережі, а на виході отримуємо номер класу розпізнаного жесту. Різні штучні нейронні мережі відрізняються своєю структурою, моделями та методами навчання. Серед найбільш поширених типів, що застосовуються у задачах розпізнавання, можна виділити мережі прямого поширення, рекурентні нейронні мережі, мережі Кохонена та згорткові нейронні мережі.

Рекурентні нейронні мережі (RNN) є потужним інструментом для обробки послідовностей даних, таких як мовлення, музика, текстові дані, а також для розпізнавання жестів на відео. Основною перевагою використання RNN для розпізнавання жестів на відео є їх здатність моделювати залежності між послідовними фреймами відео.

Одна з важливих переваг використання RNN для розпізнавання жестів на відео полягає в тому, що ці мережі можуть допомогти вирішити проблему з динамічним зміщенням (temporal shift) відео. Тобто, RNN здатний розпізнавати жести незалежно від того, коли саме вони були зроблені на відео.

Крім того, застосування RNN для розпізнавання жестів на відео дозволяє збільшити точність розпізнавання в порівнянні зі стандартними методами машинного навчання

Додатково до переваг рекурентних нейронних мереж можна зазначити такі аспекти:

- Обробка послідовностей довільної довжини: рекурентні нейронні мережі можуть обробляти послідовності довільної довжини, що робить їх ефективними для обробки жестів на відео. Крім того, вони можуть обробляти послідовності з різними розмірами кроків між кадрами, що дає можливість враховувати нерегулярність в зміні жестів.
- Здатність до запам'ятовування контексту: рекурентні нейронні мережі мають здатність запам'ятовувати контекст від попередніх кадрів і використовувати його для подальшого аналізу кадрів. Це дає можливість робити більш точну класифікацію жестів на відео.
- Можливість працювати з послідовними вхідними даними: рекурентні нейронні мережі можуть працювати з послідовними вхідними даними, такими як послідовність кадрів відео. Це дозволяє здійснювати аналіз жестів в режимі реального часу, тобто оброблювати відеопотік на льоту.
- Можливість враховувати контекст в часі: рекурентні нейронні мережі можуть враховувати контекст в часі, що дає можливість аналізувати динаміку зміни жестів.

Отже, рекурентні нейронні мережі мають багато переваг для розпізнавання жестів на відео. Вони можуть працювати з послідовностями довільної довжини, запам'ятовувати контекст від попередніх кадрів і враховувати контекст в часі. Тому для використання в застосунку було обрано саме цей вид мереж.

РОЗДІЛ 3. ЗАГАЛЬНИЙ ПРИНЦИП РОБОТИ ЗАСТОСУНКУ

Основні компоненти застосунку можна розбити на блоки.
Блокова модель побудови системи розпізнавання жестів має кілька переваг:

1. Модульність: кожен блок в системі можна легко модифікувати та вдосконалювати без впливу на решту системи. Це дозволяє швидко та ефективно вдосконалювати окремі компоненти системи та розширювати її функціональні можливості.
2. Гнучкість: застосування блокової моделі дозволяє з легкістю змінювати компоненти системи та їх послідовність, що робить систему більш гнучкою та адаптивною до змінних вимог користувачів.
3. Ефективність: блокова модель побудови системи дозволяє оптимізувати роботу окремих блоків та системи в цілому, що забезпечує високу ефективність та швидкодію системи.
4. Легкість у використанні: блокова модель робить систему більш легкою у використанні, оскільки окремі компоненти можуть бути прості у використанні та зрозумілі користувачам.

Отже, блокова модель побудови системи розпізнавання жестів має кілька переваг, які роблять її ефективною та гнучкою для використання.
Принцип роботи блоків для розпізнавання жестів та перетворення їх в комп'ютерні команди наступний(рис.5):

1. Модель орієнтації руки: Система спочатку визначає орієнтацію руки на зображенні або відео. Це зроблено шляхом використання технології MediaPipe. Модель орієнтирів руки допомагає визначити її на зображенні та правильно інтерпретувати жести, що зроблені користувачем.
2. Блок підготовки даних до нейронної мережі: Для того, щоб нейронна мережа могла розпізнати жести, необхідно підготувати дані. Цей блок включає в себе застосування функції перетворення та нормалізації даних. Дані підготовлені в реальному часі.
3. Нейронна мережа: Для розпізнавання використовується рекурентна нейронна мережа. Мережа була навчена на власному наборі даних з жестами, що створено на основі публічного набору жестів IPN Hand[2]. Після навчання модель може розпізнавати жести.
4. Контролер виконання команди на ПК: Після розпізнавання жесту нейронна мережа може класифікувати зображення, що контролер асоціює з відповідною дією ПК.

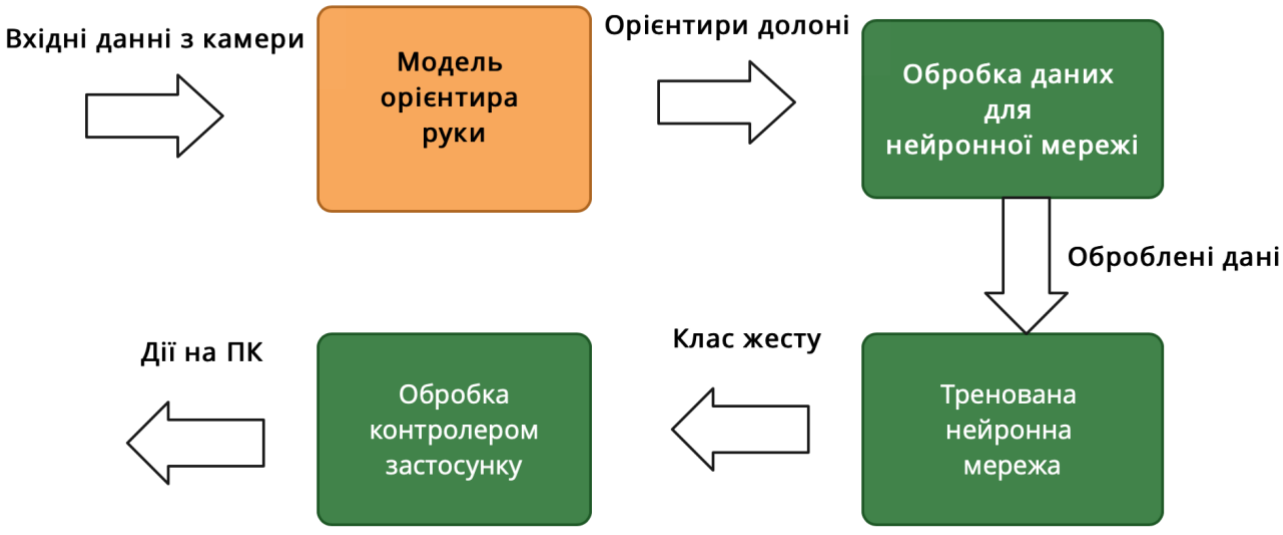

Рис.5 Блок-схема алгоритму роботи застосунку

3.1 Застосування MediaPipe Hands

Результатом використання MediaPipe Hands є MediaPipe Hand Landmark модель долоні що являє собою набір з 21 ключової точки, який описує руку. Кожна з цих точок має свої координати x, y та z, де x та y це піксельні координати на зображенні, а z - глибина точки відносно камери(лістинг 1). В лістингу коду відображено тільки 4 точки, через надмірність показу всіх 20. Ці точки використовуються для подальшого аналізу руху руки та жестів.

Лістинг 1. Вивід ключових точок моделі

```
[landmark
x: 0.6005926396569934
Y: 6.7575482063149048
Z: -1.280P0234645431366-87,
landmark
x: 0.5829388758056128
y: 0.7258450984954836
z: -0,0061848625458776 95,
landmark
x: 0.5717848048973085
Y: 0.6758704058064087
z: -0.0127853006124696 46,
landmark
X: 0.6022641062734507
y: 0.6225804354667664
2: -0.0052602764708142 28]
```

Результат роботи цього блоку коду можна побачити на рис.6 на якому виведено положення точок відносно зображення долоні.

Рис.6 Зображення моделі руки на зображенні руки

3.2 Попередня обробка навчальних даних

Передача до нейронної мережі відстаней між ключовими точками замість координат може підвищити ефективність розпізнавання жестів з кінцевих точок (наприклад, руки, пальці, обличчя тощо), оскільки вона дозволяє зберегти більше інформації про розмір та пропорції об'єктів на зображеннях.

Координати кінцевих точок можуть бути нерівномірними та залежати від розміру та форми об'єкта на зображенні, що може ускладнити процес розпізнавання та знизити точність моделі. Наприклад, якщо два зображення зображують один і той же жест, але з різною пропорцією та масштабом, то координати кінцевих точок можуть бути різними. Це може призвести до помилкових результатів розпізнавання.

У той же час, передача відстаней між ключовими точками дозволяє уникнути цих проблем та зберегти пропорції та розміри об'єктів на зображенні. Відстані між ключовими точками можуть бути нормалізовані та стандартизовані для кожного зображення, що дозволяє зробити їх більш стійкими до зміни масштабу та пропорцій об'єктів на зображеннях.

Крім того, передача відстаней між ключовими точками може допомогти знизити кількість параметрів моделі та зробити її більш ефективною. Оскільки відстані між

ключовими точками залежать лише від просторового розташування точок на зображенні, а не від їх координат, модель може використовувати менше параметрів для опису цієї інформації.

Отже, передача до нейронної мережі відстаней між ключовими точками може підвищити ефективність

Було створено інваріантний до масштабу та обертання інструмент для виділення ознак, інформацію про відстань від різних пар ключових точок в пікселях.

Фіолетовим кольором(рис.7) позначено відстані від фаланги великого пальця до вершини кожного з пальців.

Темно-зеленим кольором позначено відстань від зап’ястя до вершини кожного з пальців

Синім позначено відстань від початку фаланги до вершини фаланги

Жовтим позначено відстань від початку фаланги великого пальця до вершини фаланги кожного з пальців, крім власне великого.

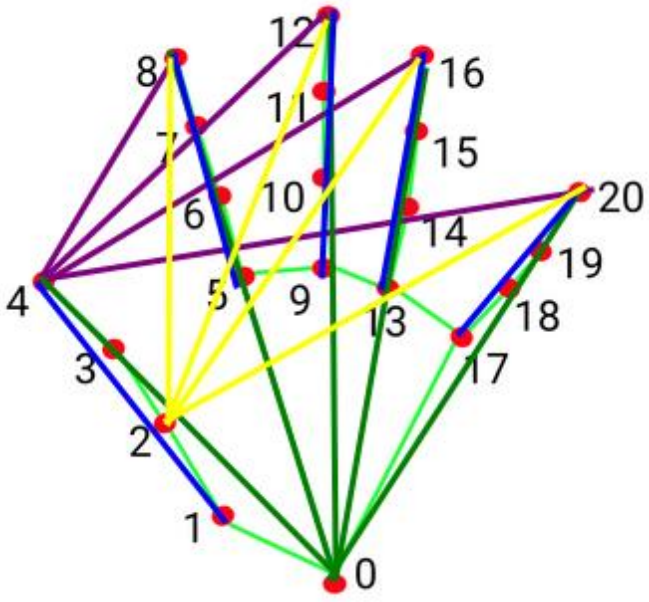

Рис.7 Зображення відстаней між ключовими точками

3.3 Використані інструменти

Мова програмування: Python 3.9

Інструмент (framework): Kivi 2.1.0

Бібліотеки для блоку нейронної мережі:

- Mediapipe - надає готові моделі та інструменти для обробки зображень та відео. Вона включає інструменти для розпізнавання облич, детекції ключових точок та відстеження рухів. Використовується для початкової обробки та визначення ключових точок руки.
- Numpy - бібліотека для обробки числових даних. Вона надає функції для маніпулювання масивами даних, що дозволяє ефективно робити математичні обчислення. Використовується для обробки ключових точок функцією екстракції.
- Matplotlib - бібліотека для візуалізації даних. Вона дозволяє створювати графіки, діаграми та інші візуалізації, що можуть бути корисними при відображенні результатів розпізнавання жестів. Використовується для побудови графіків метрик нейронної мережі
- Keras - високорівневий фреймворк для створення власний нейронних мереж на основі готових класів. Використовується для побудови та тренування моделей машинного навчання та для класифікації жестів рук.
- Scikit-learn - бібліотека машинного навчання. Містить імплементації різних алгоритмів класифікації та кластеризації, що можуть бути використані для розпізнавання жестів та групування даних.
- Moviepy - бібліотека для редагування та обробки відео. Вона надає функціональність для обрізки, з'єднання, додавання ефектів та експорту відео, що може бути корисним при обробці відеоматеріалів для розпізнавання. В

даному проєкті використовується для підготовки власного набору даних на основі IPN Hand набору.

- Tensorflow - бібліотека для чисельних обчислень, яка зосереджена на розробці та тренуванні моделей глибокого навчання. TensorFlow надає широкі можливості для побудови та виконання нейронних мереж, що можуть бути використані для розпізнавання жестів на відео.
- Pandas - бібліотека для маніпулювання та аналізу даних. Pandas надає структури даних та функції, що спрощують роботу з таблицями та числовими даними. В процесі розпізнавання жестів, Pandas може бути використана для організації, фільтрації та обробки вхідних даних, що допомагає в побудові та навчанні моделей розпізнавання.

3.5 Набір тренувальних даних

Вибір бази знань є важливою складовою побудови оптимальної моделі нейронної мережі, оскільки якість та достовірність даних, на яких модель навчається, може впливати на її точність та ефективність.
Основні фактори, які необхідно враховувати при виборі бази знань для навчання нейронної мережі, це:

1. Розмір та репрезентативність даних: база знань повинна містити достатню кількість даних, які є репрезентативними для задачі, щоб нейронна мережа мала змогу вивчити широкий спектр зразків.
2. Якість та чистота даних: дані повинні бути достовірними та чистими від шуму, аномалій та випадкових помилок.
3. Розподіл класів: база знань повинна мати достатній розподіл даних між різними класами, щоб уникнути перенавчання на одному класі та недонавчання на іншому.
4. Різноманітність даних: база знань повинна містити різноманітні дані, щоб нейронна мережа мала змогу вивчити різні особливості та характеристики зразків.
5. Відповідність реальному світу: база знань повинна відповідати реальному світу та містити дані, які корисні та зрозумілі для задачі.

Отже, правильний вибір бази знань для навчання нейронної мережі може покращити її точність та ефективність, тому слід звернути на це увагу при побудові моделі.
Нижче наведено список наборів даних у відкритому доступі.

1. Набір даних NVIDIA Dynamic Hand Gesture Dataset[17] включає динамічні жести рук, зафіксовані датчиками глибини, кольору та стерео-ІЧ-датчиками. Загалом 20 суб’єктів взяли участь у зборі набору даних, який включав 25 класів жестів, призначених для використання під час водіння для керування автомобільними пристроями в автомобілі.

2. Набір даних IPN Hand [2] включає 13 класів статичних і динамічних жестів для взаємодії з без сенсорними екранами, отриманих від 50 різних суб’єктів.
3. Jester Dataset[18] — це масштабний набір відеоданих реального світу для розпізнавання жестів. Це результат збору даних за допомогою краудворкінгу, в якому брали участь 1376 суб’єктів, які виконували набір із 27 жестів, які охоплюють найпоширеніші людські жести в контексті інтерфейсу взаємодії.
4. Набори даних, такі як ChaLearn LAP Continuous Gesture Dataset[19] та EgoGesture[20], зосереджені на жестах рук, не призначених для розпізнавання жестів. Набір ChaLearn містить 249 класів жестів щодо різних областей, включаючи мову жестів, мову тіла, символічні жести та італійські жести. У EgoGesture набір даних складається з 83 статичних і динамічних класів жестів, отриманих від 50 різних суб’єктів. У цьому випадку домен розпізнавання жестів є від першої особи, тобто розглянуті жести були спеціально розроблені для взаємодії з носимими пристроями.

Проаналізувавши наявні набори, можна зробити висновок, що автоматичне розпізнавання жестів рукою все ще є складним завданням через такі причини:

- Дефіцит загальнодоступних наборів даних жестів рукою
- Жести рук отримані з низькою якістю зображення
- Різноманітність у тому, як люди виконують жести

Для роботи було вибрано набір IPN Hand, через те, що наведений набір має одну з найбільших роздільних здатностей 640x480 з швидкістю 30 кадрів в секунду та включає найбільшу кількість безперервних жестів на відео та найбільшу швидкість варіації всередині класу.

З наведеного набору було створено власний тренувальний датасет, що складається з 1500 відео протяжністю від однієї до десяти секунд, що містить 13 класів жестів, що виконують різні суб’єкти в різних умовах.

Також було створено валідаційний датасет з 400 відео, для оцінки роботи класифікаційного елементу інтерфейсу.

Код застосунку для підготовки даних для тренування в Додатку

3.6 Результати досліджень

Шляхом підбору початкових параметрів було встановлено, що 100 епох із розміром партії 160 дають найкращі результати метрик після навчання мережі.

Лістинг коду 2. Тренувальна функція

```
verbose, epochs, batch_size = 1, 100, 160
model.fit(X_train, y_train,
          validation_data=(X_test, y_test),
          epochs=epochs,
          batch_size=batch_size,
          verbose=verbose,
          shuffle=True,
          callbacks=callbacks_list)
with open("test_model.pkl", "wb") as f:
    pickle.dump(model.history, f)
model.save("test_model")
```

Середній відсоток успішної класифікації жесту складає 91.6% на валідаційному наборі даних. В тому числі й такі на яких жесту немає. Обчислення проводилось за формулою(1).

$$avg = \frac{\sum_0^n accuracy_n}{n} (1)$$

Лістинг коду 3. Обчислення усередненого відсотку успішної класифікації жестів

```
with open(DIR_NAME / Path("model_w_bidir_history.pkl"), "rb") as f:
    landmark_npy_all = pickle.load(f)
print("Avg accuracy: {}".format(int(reduce(lambda a, b: a + b,
landmark_npy_all.history["accuracy"])) / 250))
```

Вивід консолі:

```
>>> Avg accuracy: 0.916
```

Візуалізація складових частин нейронної мережі зображена на рисунку 8.

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 bidirectional (Bidirectiona  (None, 50, 512)          567296
 l)

 dropout (Dropout)           (None, 50, 512)           0

 bidirectional_1 (Bidirectio  (None, 50, 512)          1574912
 nal)

 dropout_1 (Dropout)         (None, 50, 512)           0

 bidirectional_2 (Bidirectio  (None, 256)              656384
 nal)

 dense (Dense)               (None, 64)                16448

 batch_normalization (BatchN  (None, 64)               256
 ormalization)

 activation (Activation)     (None, 64)                0

 dense_1 (Dense)             (None, 14)                910

=================================================================
Total params: 2,816,206
Trainable params: 2,816,078
Non-trainable params: 128
_________________________________________________________________
```

Рис.8 Візуалізація складових частин

3.7 Порівняння моделей з різними типами шарів

LSTM (Long Short-Term Memory), GRU (Gated Recurrent Unit), та Bidirectional LSTM - це різновиди рекурентних нейронних мереж, які використовуються для обробки послідовностей даних, таких як текст, звук або відео.

LSTM та GRU мають подібні архітектури, що дозволяє їм зберігати та використовувати інформацію з попередніх кроків. Однак, LSTM має додатковий механізм пам'яті, який називається "клітинною пам'яттю". Це дає LSTM здатність зберігати та використовувати довгострокову інформацію краще, ніж GRU.
У GRU є менше параметрів, тому він може бути ефективнішим для моделей з обмеженою кількістю даних або при обмеженому обсязі обчислювальних ресурсів.
Bidirectional LSTM - це комбінація двох LSTM, що працюють паралельно, один - для обробки послідовності в прямому порядку, інший - в зворотньому порядку. Це дозволяє моделі звернути увагу на контекст з обох напрямків та використовувати цю інформацію для кращого прогнозування.

3.7.1 Аналіз метрик ефективності нейронної мережі з різними шарами

Метрики ефективності нейронної мережі відображають її здатність до точного передбачення результатів на відомих даних та здатність до узагальнення на нові дані. Дві основні метрики ефективності нейронної мережі - це accuracy та loss.
Model accuracy - це метрика, яка відображає відсоток правильних передбачень моделі на тестових даних. Чим вища точність, тим краща є модель. Однак, точність не завжди є найкращою метрикою, особливо коли дані мають незбалансовану розподіленість між класами. Наприклад, якщо маємо класифікувати 100 зображень, 99 з яких належать до класу A, а 1 - до класу B, то модель може відмінно відтворити клас A, але не мати здатності до передбачення класу B. Тому, доцільно використовувати інші метрики разом з точністю.

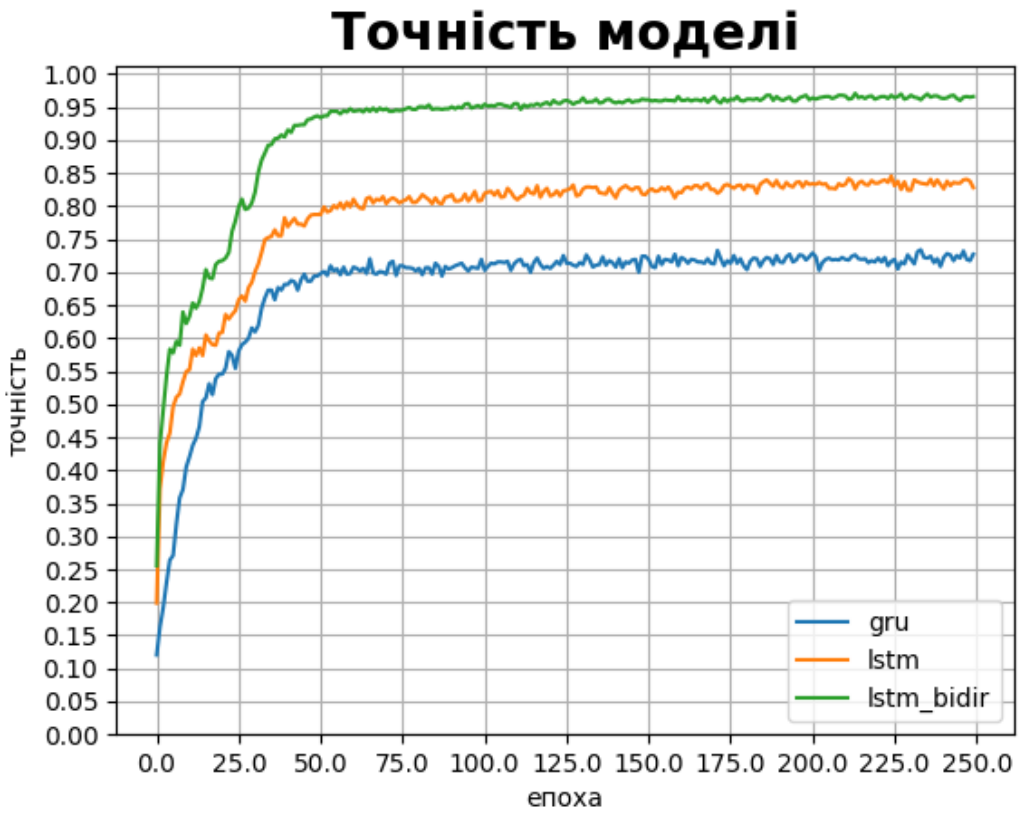

Рис. 9 Графік метрик точності моделі для різних типів шарів

Проаналізувавши графік(рис.9) можна помітити, що після 75 епохи, для усіх типів моделей мереж наступає процес перенавчання, бо точність класифікації майже не зростає із подальшим навчанням. Це явище негативне через те, що модель надто добре запам’ятовує тренувальні дані і буде не здатна коректно розпізнавати нові дані та втратить свою точність. Бачимо, що різні види шарів не дають виграшу в швидкості навчання моделі адже процес перенавчання (overfitting) починається приблизно після однакової кількості епох, але дає перевагу у точності розпізнавання, адже модель з шаром Bidirectional LSTM має більше значення точності на приблизно 25 та 15 відсотків відповідно до GRU та LSTM моделей.

Model loss - це метрика, яка відображає помилковість передбачень моделі на тестових даних. Чим менша втрата, тим краща є модель. Зазвичай, втрати обчислюються за допомогою функції втрат, такої як MSE (Mean Squared Error),

MAE (Mean Absolute Error), категоріальна крос-ентропія (Categorical Cross-Entropy) та інші. В випадку нашого застосунку це категоріальна крос-ентропія.

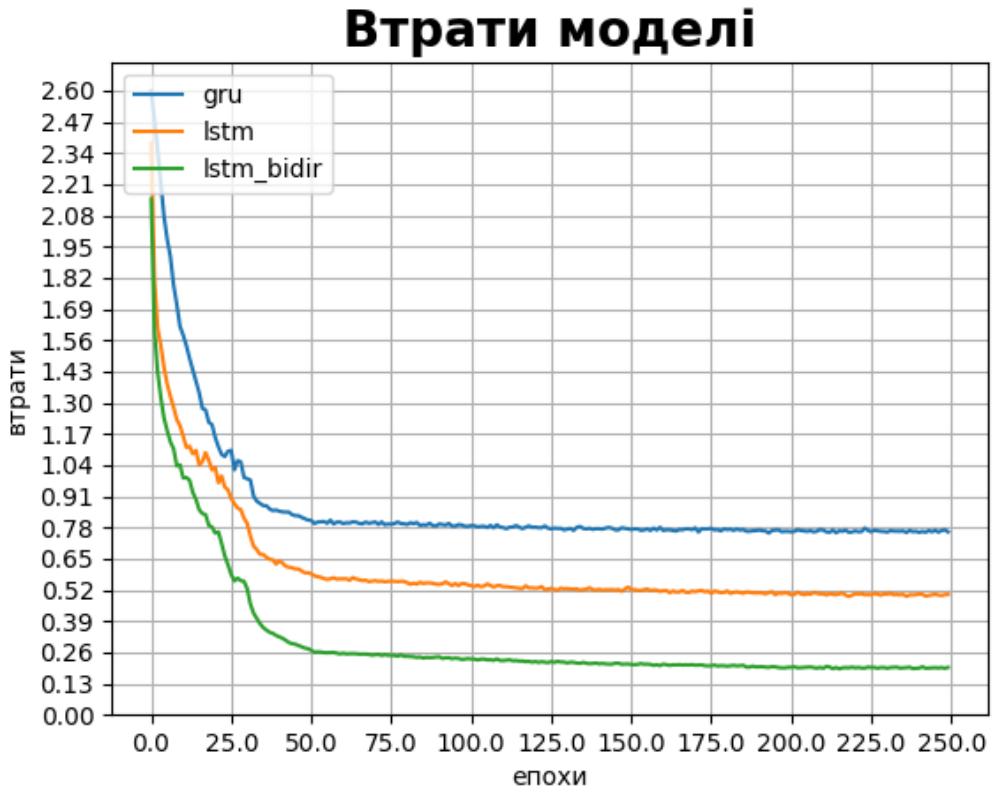

Рис.10 Графік втрат моделей з різними шарами

Проаналізувавши графік цієї метрики(рис.10) бачимо, що в цілому він обернений до графіку метрики точності через те, оцінює якість моделі на основі різниці між передбаченими значеннями та фактичними. Тому якщо б метрики не відповідали одна одній, можна було б зробити припущення, що модель занадто спрощена, або є проблеми з даними. В даному випадку можемо спостерігати, що метрики вказують на коректну роботу моделі.

3.7.2 Матриця плутанини (Confusion Matrix)

Матриця точності, також відома як матриця плутанини або confusion matrix, є візуалізацією результатів класифікації нейронної мережі. Вона дозволяє оцінити,

наскільки точно модель класифікує дані на певні класи. Матриця точності зображує кількість правильних та неправильних передбачень моделі для кожного класу. Матриця складається з двох рядків та двох стовпчиків. Рядки відповідають реальним класам, а стовпчики - передбачуваним класам. Кожний елемент матриці показує кількість випадків, коли реальний клас належить до одного з рядків, а передбачений клас - до одного зі стовпчиків. Таким чином, діагональні елементи матриці відображають кількість правильних передбачень, а недіагональні - кількість неправильних передбачень.

Матриця точності є важливим інструментом для оцінки ефективності моделі, особливо в задачах з незбалансованими класами. З її допомогою можна оцінити точність та помилковість класифікації для кожного класу окремо, а також оцінити загальну ефективність моделі в цілому.

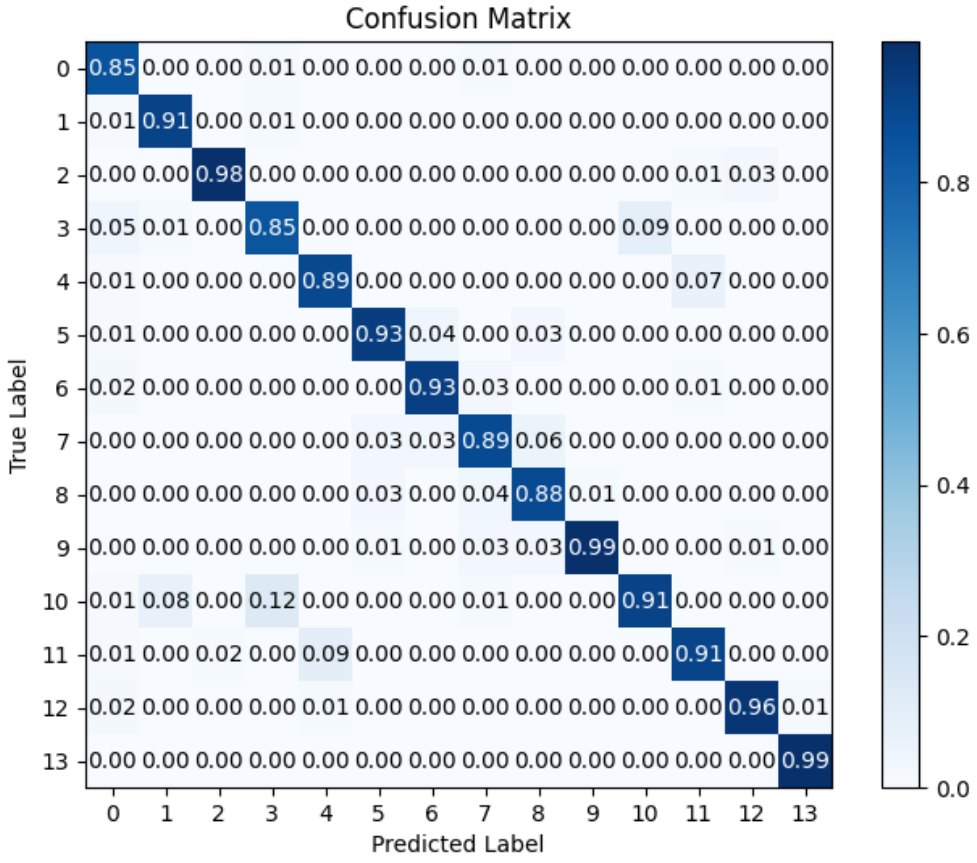

Рис.11 Матриця плутанини для моделі

Для випадку моделі з найкращими метриками(рис.11) правильно класифіковано майже всі класи зображень, найменший відсоток классифікації на жестах 3 і 4 та 10 і 11, через схожість жестів, це клік одним пальцем і клік двума пальцями та подвійний клік одним пальцем і подвійний клік двома пальцями відповідно.

3.8 Огляд застосунку

Розроблений застосунок реалізує надання користувачу персонального комп'ютера інтерфейсу для взаємодії з машиною та виконує роль контролера для перетворення розпізнаного блоком нейронної мережі жесту в команду. Крім цього, застосунок дозволяє робити налаштування для персоналізації цього інтерфейсу під власні потреби.

Головна сторінка(рис.12) містить наступні можливості:

- Бібліотеку доступних дій, жестів та макросів
- Додавання новий дій та макросів

Рис.12 Головне меню застосунку

Жестом може бути будь який жест з класів тренувального набору (кинути вліво, вправо, вгору, вниз тощо).

Дією може бути додана комбінація клавіш або консольна команда.(рис.13)

Макрос - зв’язка дії та жеста, після створення якого застосунок певний жест буде асоціювати з певною дією на ПК.

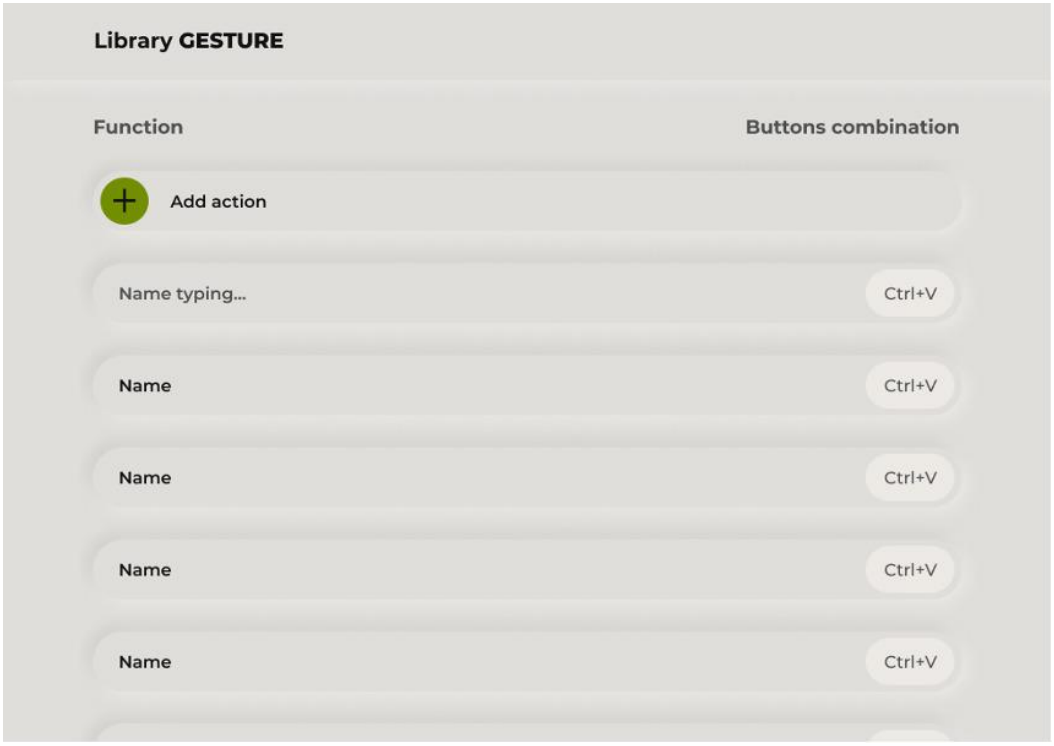

Рис.13 Меню додавання дії

3.9 Порівняння з аналогами

Zesture - безкоштовний застосунок, який використовує камеру для виконання розпізнавання жестів та їх наступну інтерпретацію в дії.
З відмінностей - підтримує застосування розпізнаних жестів тільки в визначеного списку жестів в обмеженій кількості застосунків і кількість жестів та призначення кожного жесту не можна змінити.

GestureSign - застосунок для Windows, містить широкий список можливих дій в системі, але не має можливості розширення.

Npointer - утиліта для Windows, що дозволяє жестами виконувати переміщення курсору миші та операціями натискання на її клавіші, основною відмінністю цього додатку є те що неможливо задати свої дії на певний жест

Застосунок	Управління системою	Можливість кастомізації	Можливість назначити власну дію на певний відомий жест
Zesture	Так	Ні	Ні
GestureSign	Так	Ні	Ні
Npointer	Частково	Так	Ні
Власний застосунок	Так	Ні	Так

Таблиця. 1 Порівняльна таблиця функціональниї можливостей

За таблицею можна зробити наступні висновки:
- Zesture і Npointer надають можливість управляти всіма застосунками або діями, тоді як GestureSign дозволяє управляти тільки певними діями.
- GestureSign і власний застосунок мають можливість кастомізувати виконання дій, тоді як у Zesture та Npointer ця можливість відсутня.
- GestureSign і власний застосунок дозволяють назначати власну дію на певний відомий жест, тоді як у Zesture ця можливість є, але в Npointer відсутня.

У загальному можна стверджувати, що кожен з цих застосунків має свої особливості та переваги, і вибір залежить від потреб користувача. Наприклад, якщо важлива можливість кастомізації виконання дій, то варто обрати GestureSign або власний застосунок. А якщо важливо мати можливість управління всіма застосунками, то Zesture або Npointer або власний застосунок можуть бути кращим вибором. Але власний застосунок моє одну перевагу над іншими в тому, що можна визначити дію самостійно та асоціювати її з певним жестом.

ВИСНОВКИ

У результаті виконання роботи магістра:

1. Проведено аналіз побудованих моделей нейронних мереж з різними архітектурами шарами, такими як: LSTM, GRU та Bidirectional LSTM. В результаті аналізу було виявлено, що Bidirectional LSTM дає найкращі результати відповідно до метрик accuracy та loss, а саме 91.6% для середнього значення точності. Про це свідчить матриця плутанини, та середнє значення точності розпізнавання.
2. Проведено порівняльний аналіз власного застосунку зі схожими за функціоналом застосунками, такими як Zesture, GestureSign та Npointer, внаслідок якого можна стверджувати, що власний застосунок має перевагу над іншими через можливість назначення власних дій на певні відомі жести.

Отриманий результат, свідчить про те, що використання технології Mediapipe з використанням рекурентних нейронних мереж є придатним для розробки інтерфейсу людина-машина з високою точністю та кастомізацією дій.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Офіційна документація Mediapipe [Електронний ресурс]. URL: https://google.github.io/mediapipe/solutions/hands.html (Дата звернення: 23.03.2023).
2. Набір даних IPN Hand [Електронний ресурс]. URL: https://gibranbenitez.github.io/IPN_Hand/ (Дата звернення: 23.03.2023)
3. Офіційна документація Keras [Електронний ресурс]. URL: https://keras.io (Дата звернення: 23.03.2023).
4. LSTM layer [Електронний ресурс] URL: https://www.mathworks.com/help/deeplearning/ref/nnet.cnn.layer.lstmlayer.html (Дата звернення: 24.03.23)
5. A Dynamic Hand Gesture Recognition Dataset for Human-Computer Interfaces [Електронний ресурс]. URL: https://www.researchgate.net/publication/357674123_A_Dynamic_Hand_Gesture_Recognition_Dataset_for_Human-Computer_Interfaces (Дата звернення: 23.03.2023)
6. DATASET FOR DYNAMIC HAND GESTURE RECOGNITION SYSTEMS [Електронний ресурс]. URL: https://ieee-dataport.org/documents/dataset-dynamic-hand-gesture-recognition-systems#files (Дата звернення: 24.03.23)
7. Офіційна документація Tensorflow [Електронний ресурс]. URL: https://www.tensorflow.org/guide/keras/rnn (Дата звернення: 24.03.23)
8. Офіційна документація OpenPose [Електронний ресурс]. URL: https://cmu-perceptual-computing-lab.github.io/openpose/web/html/doc/md_doc_00_index.html (Дата звернення: 24.03.23)
9. MediaPipe and OpenPose comparison [Електронний ресурс]. URL: https://www.hearai.pl/post/14-openpose/ (Дата звернення: 24.03.23)
10. Офіційна документація програмного каркасу kivy [Електронний ресурс]. URL: https://kivy.org/doc/stable/ (Дата звернення: 24.03.23)

11."Sequence to Sequence Learning with Neural Networks" (Sutskever et al., 2014) [Електронний ресурс]. URL: https://arxiv.org/abs/1409.3215 (Дата звернення: 24.03.23)
12."Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow" (Geron, 2019) https://www.oreilly.com/library/view/hands-on-machine-learning/9781492032632/ ISBN: 978-1492032649
13.S. Hochreiter and J. Schmidhuber, "Long short-term memory," in Neural computation, 1735-1780, 1997. ISBN: 9780262531995
14.A. Graves, "Generating sequences with recurrent neural networks," 2013. URL: https://arxiv.org/abs/1308.0850 (Дата звернення: 24.03.23)
15.Y. Bengio, P. Simard, and P. Frasconi, "Learning long-term dependencies with gradient descent is difficult," in IEEE transactions on neural networks, pp. 157-166, 1994. ISBN: 9780780314749
16. Detection of human body landmarks - MediaPipe and OpenPose comparison [Електронний ресурс]. URL: https://www.hearai.pl/post/14-openpose/ (Дата звернення: 24.03.23)
17.NVIDIA Dynamic Hand Gesture Dataset [Електронний ресурс]. URL: https://research.nvidia.com/publication/2016-06_online-detection-and-classification-dynamic-hand-gestures-recurrent-3d (Дата звернення: 24.03.23)
18.Jester набір даних [Електронний ресурс]. URL: https://developer.qualcomm.com/software/ai-datasets/jester (Дата звернення: 24.03.23)
19.ChaLearn LAP Continuous Gesture набір даних [Електронний ресурс]. URL: https://chalearnlap.cvc.uab.cat/dataset/22/description/ (Дата звернення: 24.03.23)
20.EgoGesture набір даних [Електронний ресурс]. URL: http://www.nlpr.ia.ac.cn/iva/yfzhang/datasets/egogesture.html (Дата звернення: 24.03.23)
21. Офіційна документація OpenCV [Електронний ресурс]. URL: https://docs.opencv.org/ (Дата звернення: 12.05.2023)

22. Офіційна документація PyTorch [Електронний ресурс]. URL: https://pytorch.org/docs/stable/index.html (Дата звернення: 12.05.2023)

23. Офіційна документація scikit-learn [Електронний ресурс]. URL: https://scikit-learn.org/stable/documentation.html (Дата звернення: 12.05.2023)

24. Офіційна документація TensorFlow [Електронний ресурс]. URL: https://www.tensorflow.org/api_docs/python/ (Дата звернення: 12.05.2023)

25. "Computer Vision: Algorithms and Applications" (Szeliski, 2010) [Електронний ресурс]. URL: http://szeliski.org/Book/ (Дата звернення: 12.05.2023)

26. Офіційна документація SciPy [Електронний ресурс]. URL: https://docs.scipy.org/doc/ (Дата звернення: 12.05.2023)

Додаток А

Блок коду що є демо-версією роботи програми

```
with mp_hands.Hands(max_num_hands=1,
                    min_detection_confidence=0.5,
                    min_tracking_confidence=0.5) as hands:
    while cap.isOpened():
        landmark_npy_single = []
        success, image = cap.read()
        if not success:
            print("Ignoring empty camera frame.")

        image = cv2.cvtColor(cv2.flip(image, 1),
cv2.COLOR_BGR2RGB)
        image.flags.writeable = False
        results = hands.process(image)
        # Draw the hand annotations on the image.
        image.flags.writeable = True
        image = cv2.cvtColor(image, cv2.COLOR_RGB2BGR)
        image = cv2.putText(image, name_of_gesture, org,
font,
                            fontScale, color, thickness,
cv2.LINE_AA)
        if results.multi_hand_landmarks:
            for hand_landmarks in
results.multi_hand_landmarks:

landmark_npy_single.append(landmark_to_distance(results))
                mp_drawing.draw_landmarks(
```

```
                    image, hand_landmarks,
mp_hands.HAND_CONNECTIONS)
            landmark_per_50.append(landmark_npy_single[0])

        if len(landmark_per_50) > 30:
            gest_type = predict_gesture(model,
landmark_per_50)[0]
            print(gest_type)
            name_of_gesture = id_to_name(gest_type)
            landmark_per_50.clear()

        cv2.imshow('MediaPipe Hands', image)
        if cv2.waitKey(5) & 0xFF == 27:
            break
cap.release()
```

Додаток Б

Код створення набору даних

```
import argparse
import os
import pickle

import cv2
```

```
import mediapipe as mp

from data_preparation_utils import landmark_to_distance

mp_drawing = mp.solutions.drawing_utils
mp_hands = mp.solutions.hands


def extract_video_data(script_args):
    arr = os.listdir(script_args.video_source_dir)
    arr.remove(".DS_Store")
    gestures_classes = []
    hands_model_data = []
    mp_hands_setup = mp_hands.Hands(max_num_hands=1,

min_detection_confidence=0.6,

min_tracking_confidence=0.6)
    for num, video_name in enumerate(arr):
        landmark_npy_single = []
        try:
            video_data =
cv2.VideoCapture(script_args.video_source_dir + video_name)
        except Exception as e:
            print("Something went wrong with file
{}".format(video_name))
            print("Next exception ocurred {} {}".format(e,
type(e)))
        else:
```

```
gestures_classes.append(int(video_name.split("_")[4]))
            while video_data.isOpened():
                success, image = video_data.read()
                if not success:
                    break
                image = cv2.cvtColor(cv2.flip(image, 1),
cv2.COLOR_BGR2RGB)
                image.flags.writeable = False
                results = mp_hands_setup.process(image)
                image.flags.writeable = True
                if results.multi_hand_landmarks:
                    for _ in results.multi_hand_landmarks:

landmark_npy_single.append(landmark_to_distance(results))
            hands_model_data.append(landmark_npy_single)
            video_data.release()
        if ((num + 1) % 10) == 0:
            print(f"Finished for {(num + 1)} videos")
    print(f"Finished for total {len(arr)} videos.
Completed.")
    with open(script_args.hand_model_source_file_name, "wb")
as f:
        pickle.dump(hands_model_data, f)
    with open(script_args.gestures_classes_file_name, "wb")
as f:
        pickle.dump(gestures_classes, f)


if __name__ == "__main__":
```

```
    parser = argparse.ArgumentParser()
    parser.add_argument("--video_source_dir", help="path to
videos for preparation",
default="edited_videos_validation/")
    parser.add_argument("--hand_model_source_file_name",
help="name of file for hand model data",
                        default="landmark_damp2.pkl")
    parser.add_argument("--gestures_classes_file_name",
help="name of file for gestures classes data",
                        default="video_classes_dump2.pkl")
    parser.add_argument("--max_num_hands", help="number of
hands", default=1)
    args = parser.parse_args()
    extract_video_data(args)
```

Додаток В

Код для функції підготовки даних та функцій класифікації

```
import json
import math
from pathlib import Path

import keras
```

```
import numpy as np
import pandas as pd
from google.protobuf.json_format import MessageToJson
from keras.utils import pad_sequences


def distance_between_points(results, p1, p2):
    jsonObj = MessageToJson(results.multi_hand_landmarks[0])
    lmk = json.loads(jsonObj)['landmark']
    p1 = pd.DataFrame(lmk).to_numpy()[p1]
    p2 = pd.DataFrame(lmk).to_numpy()[p2]
    squared_dist = np.sum((p1 - p2) ** 2, axis=0)
    return np.sqrt(squared_dist)


def landmark_to_distance(results):
    jsonObj = MessageToJson(results.multi_hand_landmarks[0])
    lmk = json.loads(jsonObj)['landmark']

    emb = np.array([
        # thumb to finger tip
        distance_between_points(results, 4, 8),
        distance_between_points(results, 4, 12),
        distance_between_points(results, 4, 16),
        distance_between_points(results, 4, 20),
        # wrist to finger tip
        distance_between_points(results, 4, 0),
        distance_between_points(results, 8, 0),
        distance_between_points(results, 12, 0),
        distance_between_points(results, 16, 0),
```

```
        distance_between_points(results, 20, 0),
        # tip to tip (specific to this application)
        distance_between_points(results, 8, 12),
        distance_between_points(results, 12, 16),
        # within finger joint (detect bending)
        distance_between_points(results, 1, 4),
        distance_between_points(results, 8, 5),
        distance_between_points(results, 12, 9),
        distance_between_points(results, 16, 13),
        distance_between_points(results, 20, 17),
        # distance from each tip to thumb joint
        distance_between_points(results, 2, 8),
        distance_between_points(results, 2, 12),
        distance_between_points(results, 2, 16),
        distance_between_points(results, 2, 20)
    ])
    # use np normalize, as min_max may create confusion that
the closest fingers has 0 distance
    emb_norm = emb / np.linalg.norm(emb)
    return emb_norm


def skip_frame(landmark_npy_all, frame=50):
    new_lmk_array = []
    for each in landmark_npy_all:
        if len(each) <= frame:
            new_lmk_array.append(each)
        else:
            to_round = math.ceil(len(each)/frame)
            new_lmk_array.append(each[::to_round])
```

```
    return new_lmk_array


def lr_cof(epoch):
    lr = 0.001
    if epoch > 200:
        lr *= 0.0005
    elif epoch > 120:
        lr *= 0.005
    elif epoch > 50:
        lr *= 0.01
    elif epoch > 30:
        lr *= 0.1

    print('Learning rate: ', lr)
    return lr


def frame_to_timecode(start_frame: int, finish_frame: int) -
> str:
    start_seconds = start_frame // 30
    finish_seconds = finish_frame // 30
    start_minute = start_seconds // 60
    finish_minute = finish_seconds // 60
    add_to_start = 1 if start_frame - start_seconds * 30 >
15 else 0
    add_to_finish = 1 if finish_frame - finish_seconds * 30
> 15 else 0
```

```
    return (start_minute, start_seconds % 60 +
add_to_start), (finish_minute, finish_seconds % 60 +
add_to_finish)


def load_model(path="educated_model_with_bidirectional"):
    MODEL_NAME = Path(path)
    return keras.models.load_model(MODEL_NAME)


def predict_gesture(model, landmark_npy_all):
    new_lmk_array = skip_frame([landmark_npy_all])
    train_x = pad_sequences(new_lmk_array, padding="post",
maxlen=50, dtype="float32")
    y_prediction = model.predict(train_x)
    return np.argmax(y_prediction, axis=1)
```