Київський національний університет імені Тараса Шевченка
Факультет радіофізики, електроніки та комп'ютерних систем
Кафедра комп'ютерної інженерії

ДИПЛОМНА РОБОТА МАГІСТРА

на тему:

«Розпізнавання української жестової мови на основі ключових точок з використанням рекурентних нейронних мереж»

Студентки ІІ курсу ОР «Магістр»
спеціальності 123 «Комп'ютерна інженерія»
Ірини БУРХАН

_____

Науковий керівник:
Кандидат фіз.-мат. наук, доцент
Олександр БАРАБАНОВ

_____

Рецензент:
Кандидат фіз.-мат. наук, доцент
Анатолій ШКАВРО

_____

До захисту допускаю
Завідувач кафедри
кандидат фіз.-мат. наук, доцент **Юрій БОЙКО**

Протокол засідання кафедри від
"___" _____2023 р., протокол №_____

Київ 2023

ЗМІСТ

РЕФЕРАТ

Робота присвячена побудові класифікаторів розпізнавання української жестової мови (УЖМ) з архітектурою LSTM; створенню навчального набору даних у вигляді відеозаписів жестів УЖМ; розробці графічного інтерфейсу користувача для практичного застосування кращого з побудованих класифікаторів.

Дипломна робота складається з трьох частин. Перша частина присвячена аналізу існуючих напрацювань у сфері розпізнавання жестів, та вибору подальшої методики роботи на основі розглянутих переваг та недоліків різних методів. Друга частина містить опис інструментів та методів, які використовуються для проведення досліджень. Третя частина роботи складається з викладення і пояснення розроблених функцій для виконання поставлених завдань, а також результати досліджень.

Робота містить 72 сторінки, 28 рисунків, 9 таблиць, 1 додаток.

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

АЦП – Аналогово-Цифровий Перетворювач

ЖМ – Жестова Мова

УЖМ – Українська Жестова Мова

СПЖМ – Спілка Перекладачів Жестової Мови

США – Сполучені Штати Америки

КМОН – Комплементарна структура Метал-Оксид-Напівпровідник

Adam – Adaptive Movement Estimation

ASL – American Sign Language

BANZSL – British, Australian and New Zealand Sign Language

BGR – Blue Green Red

CNN – Convolutional Neural Network

HTML – HyperText Markup Language

JSON – JavaScript Object Notation

LSTM – Long Short-Term Memory

PCB – Printed Circuit Board

PDF – Portable Document Format

ReLU – Rectified Linear Unit

RGB – Red Green Blue

RNN – Recurrent Neural Network

SDK – Software Development Kit

SeLU – Scaled Exponential Linear Unit

SVM – Support Vector Machine

YSSA – Yarn-based Stretchable Sensor Arrays

3D – 3-Dimensional

ВСТУП

За даними Всесвітньої організації охорони здоров’я станом на 2021 рік понад 5% населення світу потребують реабілітації для усунення інвалідної втрати слуху. За оцінками експертів, до 2050 року понад 700 мільйонів людей – або кожна десята людина матимуть вади слуху [1].

Створення системи автоматичного перекладу з жестової мови на словесну дозволить забезпечити комунікативну взаємодію людей з обмеженими можливостями. Це необхідно через недостатню кількість персоналу в галузі сурдоперекладу, а також через не завжди бажану медіацію в спілкуванні глухих і чуючих громадян, наприклад, у питаннях медицини, освіти, або особистих відносин.

Об’єктом дослідження виступають динамічні жести української жестової мови. Розпізнавання жестів у відеозаписах є складним завданням комп'ютерного зору через схожість візуального змісту, зміни кута огляду на одні і ті ж дії, масштабу, різних умов освітлення, а найголовніше відсутності готових наборів даних для навчання. Українська жестова мова не уніфікована, у ній також наявні специфічні для окремих регіонів діалекти. Крім того, через відсутність стандартизації, спостерігається брак загальнодоступних наборів даних УЖМ. Тому першочерговим завданням роботи є збір, систематизація і запис достатньої кількості відео жестів.

Для повноцінного розпізнавання одного слова з мови жестів потрібно класифікувати емоції на обличчі та позицію всього тіла, виявляти швидкість рухів рук і враховувати багато інших факторів. Сучасні технології комп’ютерного зору надають можливість відслідковування всіх вищезгаданих параметрів за допомогою ключових точок скелету людини. Підхід на основі опорних точок дозволяє узагальнити відеозаписи жестів за рахунок перетворення їх у сукупність файлів з координатами, які створюють необхідний базис для навчання моделі нейронної мережі.

Другим етапом роботи є створення класифікатора з LSTM архітектурою, навченого на власному наборі даних для розпізнання жестів УЖМ. Для неупередженої оцінки роботи моделі нейронної мережі передбачається розробка мінімального графічного інтерфейсу користувача для проведення тестування системи.

Отже, в рамках цієї роботи планується зробити перший крок на шляху до конструювання системи автоматичного сурдоперекладу з української жестової мови на українську словесну мову, використовуючи штучні нейронні мережі.

РОЗДІЛ 1. АНАЛІТИЧНИЙ ОГЛЯД ІСНУЮЧИХ ПІДХОДІВ ДО РОЗПІЗНАВАННЯ ЖЕСТОВИХ МОВ

1.1 Специфіка жестових мов та базові методи їх розпізнавання

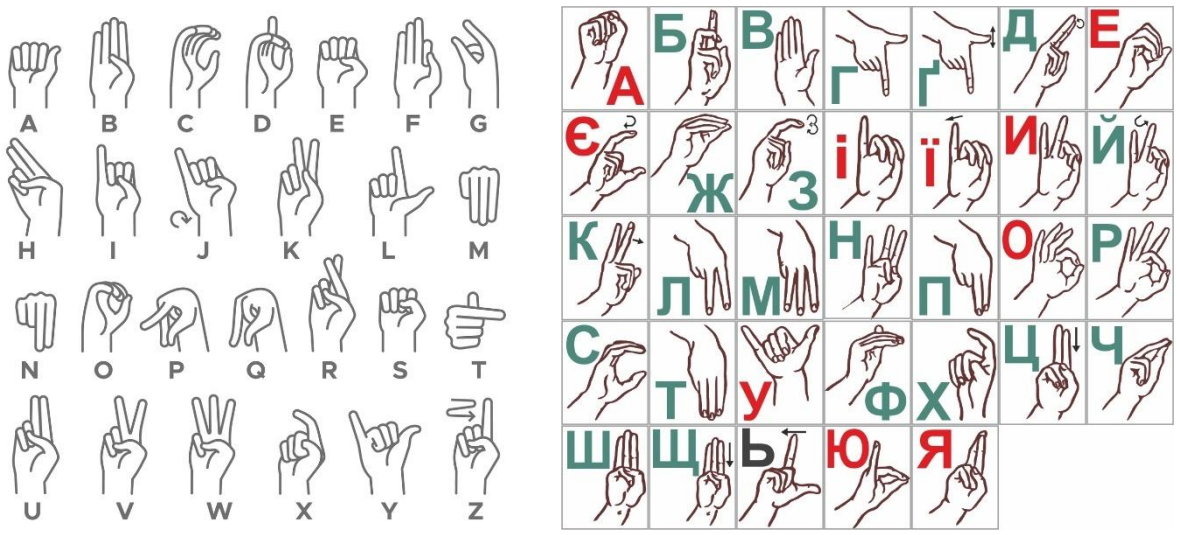
Жестові мови – це повноцінні природні мови із власною граматикою і лексикою, у яких замість вербальних слів використовуються візуальні образи – жести. Останні являють собою деякий артикуляційний рух, у формуванні якого можуть бути задіяні руки, голова, плечі та міміка обличчя.

Як і в будь-якій словесній мові, що включає в себе алфавіт, слова, речення, фрази і розділові знаки, в жестовій мові використовуються аналогічні елементи, тільки в образотворчій формі. Національні мови характеризуються рядом відмінних ознак, до яких належать, перш за все, система писемності (кирилиця, латиниця і т.д.) та літери алфавітів, що є для кожної з мов унікальними. При цьому, деякі мови можуть бути схожими між собою за окремими елементами, наприклад, за буквами алфавітів, або слів. У багатьох сучасних мовах із великим числом носіїв можна зустріти слова, що збігаються звучанням зі словами іншої мови, що пояснюється постійним впливом їх один на одну. Іншою помітною ознакою національних мов є наявність у кожній з них діалектних різновидів територіального та соціального характеру.

Усі перелічені міжмовні особливості мають місце і у жестових мовах. Кожна країна має власну, сформовану специфічними умовами ЖМ, у якій використовуються різні жести та їх діалекти, що варіюються залежно від регіону та спільнот глухих. Однак, на відміну від словесних мов, різні жестові мови меншою мірою різняться між собою, оскільки їх носії при спілкуванні часто використовують прості для розуміння образотворчі жести. Така форма спілкування створює комунікативний міст між різними культурами глухих, що дозволяє їм краще розуміти одне одного.

Розвиток ЖМ також не завжди базується на державній чи національній мові. Наприклад, не дивлячись на те, що у Сполучених Штатах Америки та Великобританії є спільна англійська мова, їх жестові мови відрізняються кардинально і навіть не є спорідненими за мовною групою. Це пов'язано з історичним розвитком цих країн, у результаті якого формування жестових мов відбувалося відокремлено. Як наслідок, на британській мові жестів BANZSL (British, Australian and New Zealand Sign Language) спілкуються в Австралії, Новій Зеландії, Великобританії, а також ЮАР і на півночі Ірландії. Натомість американська мова жестів ASL (American Sign Language) використовується в США і Канаді [2].

Важливе місце в жестовому спілкуванні глухих займає дактильна абетка (рис. 1.1), що є особливою системою конфігурацій пальців рук для позначення букв алфавіту тієї чи іншої національної мови.



а) б)

Рисунок 1.1 Дактильна абетка ASL – а, УЖМ – б

На відміну від жестової мови, що служить для передачі слів і цілих понять, пальцевий алфавіт відіграє допоміжну роль, передаючи окреме слово по-літерам, при цьому букви у свідомості миттєво вишиковуються в слово, що сприймається співрозмовником цілком, аналогічно сприйняттю

послідовності букв при читанні тексту. У всіх країнах зараз існують свої дактильні абетки та жести, а також правила їх передачі у реченні. Деякі дактильні системи побудовані за принципами, що відбивають специфіку національної писемності (ієрогліфіка). Форма передачі в ЖМ дактильних знаків може бути одноручною, дворучною та комбінованою, що характерно, наприклад, українській ЖМ.

Двома основними методами дослідження жестів руками є підходи на основі рукавичок та камери [3].

Базовий модуль розпізнавання ЖМ складається з чотирьох етапів: отримання зображення, сегментація руки, виділення ключових ознак і класифікація жесту, як схематично показано на рисунку 1.2

Рисунок 1.2 Схема базового модуля розпізнавання жестів рук

Розпізнавання жестів рук за допомогою рукавичок – це один із ранніх методів, який використовує переносні датчики, прикріплені безпосередньо до руки в рукавичках. Ці датчики виявляють фізичну реакцію, відповідно до руху руки або згинання пальців.

У рукавичках використовуються різні типи датчиків для фіксації руху та положення руки шляхом визначення правильних координат розташування долоні та пальців. Серед них можна виділити датчик кривизни, який використовується для вимірювання кута згину пальців; датчик кутового

переміщення, який визначає положення кінцівки пальців та повороти в зап'ясті; оптичний перетворювач, який визначає точну позицію кінчиків пальців; датчики гнучкості, які вимірюють ступінь гнучкості та еластичність матеріалу рукавички; датчик акселерометра, який вимірює прискорення та орієнтацію руки в просторі. Кожен з цих датчиків використовує різні фізичні принципи відповідно до свого типу.

Розпізнавання жестів на основі камери використовує різні методи для виявлення рук на зображенні, отриманому з різних типів камер. Для цього розробляються алгоритми, які намагаються виявити та відрізнити руки на зображенні, використовуючи характеристики руки, такі як колір шкіри, зовнішній вигляд, рух, скелет, глибина зображення, 3D-модель тощо. Деякі алгоритми також використовують камеру глибини, щоб створити 3D-модель руки та визначити її положення і орієнтацію в просторі. Основна мета таких алгоритмів – сегментувати та розпізнавати руки на зображенні. Для цього вони використовують навчальні набори даних, які містять велику кількість зображень різних жестів. Таким чином, розпізнавання жестів на основі камери може бути дуже ефективним та точним, але вимагає значної кількості обчислювальних ресурсів та навчання системи на великій кількості даних.

Автоматичне розпізнавання жестів рук можна використовувати в широкому діапазоні додатків, таких як клінічні операції, мова жестів, керування роботами, віртуальні середовища, домашня автоматизація, керування персональним комп’ютером, ігри та ін.

Перший розділ цієї роботи націлений на аналіз найбільш популярних підходів до автоматизації систем розпізнавання жестів з акцентом на переклад ЖМ.

1.2Системи автоматичного перекладу ЖМ, засновані на зображеннях

1.2.1 Системи на основі детекторів шкіри

Виявлення кольору шкіри є одним із популярних методів сегментації рук і використовується в широкому діапазоні застосувань, таких як класифікація об'єктів, відновлення фотографій із погіршенням якості, відстеження рухів людини, відеоспостереження, розпізнавання обличчя, пошук руки та ідентифікація жестів. Детектування шкіри досягається двома основними методами:

Перший метод – це виявлення шкіри на основі пікселів, у якому кожен піксель зображення класифікується як шкіра або ні, окремо від сусіднього.

Другий метод – це виявлення ділянки шкіри, у якому пікселі шкіри просторово обробляються на основі такої інформації, як інтенсивність і текстура.

У роботі [4] фотографія жесту отримується за допомогою веб-камери з роздільною здатністю 160×120 пікселів. Далі виконується пошук області руки за допомогою детектора шкіри, конвертація зображення у двійковий вигляд та застосування фільтра Гауса 5×5 для згладжування. Класифікація конкретного жесту відбувається шляхом пошуку евклідової відстані між вектором ознак вхідного зображення в реальному часі та вектором ознак кожного навчального зображення. Для цього використовується наступна формула:

$$L_{SC_t} = \sqrt{(x_{t+1} - x_1)^2 + (y_{t+1} - y_1)^2} \qquad (1.1)$$

де L_S – найменша Евклідова відстань для вектору ознак C_t, сформованого за допомогою центроїда; $x_{t+1} - x_1$ – різниця координат по осі абсцис між векторами навчального та цільового зображення; $y_{t+1} - y_1$ – аналогічна операція по осі ординат.

Таким чином, найменша Евклідова відстань використовується для розпізнавання відповідного жесту руки.

У цій роботі для демонстрації результату використовували спрощені, тобто статичні, жести дактильної абетки ASL. У таблиці 1.1 наведені результати роботи алгоритму для 26-ти літер ЖМ, що у відсотковому представленні демонструють успішність системи для розпізнавання статичних жестів у реальному часі.

Таблиця 1.1 Результати розпізнавання 26-ти жестів ASL [4]

Жест	Точність, %	Жест	Точність, %	Жест	Точність, %
A	90	*J*	86	*S*	93
B	86	*K*	90	*T*	90
C	90	*L*	92	*U*	85
D	85	*M*	90	*V*	100
E	100	*N*	95	*W*	96
F	85	*O*	86	*X*	85
G	90	*P*	90	*Y*	92
H	92	*Q*	92	*Z*	90
I	85	*R*	90		

Алгоритми на основі кольорів стикаються з важким завданням розрізнення об'єктів, таких як руки та обличчя людини, які мають схожий колір. Обмеження також накладаються на кольори інших об’єктів і навколишнє середовище в цілому, що опиняється у кадрі. При застосуванні детектору шкіри важливою умовою є низький інформаційний шум оточуючих об’єктів, іншою широкою практикою є використання спеціального контрастного одягу із довгими рукавами.

Ще одним недоліком таких підходів є чутливість системи до варіацій кімнатного освітлення. Коли освітлення не відповідає особливим вимогам, алгоритми на основі кольорів зазвичай дають збій.

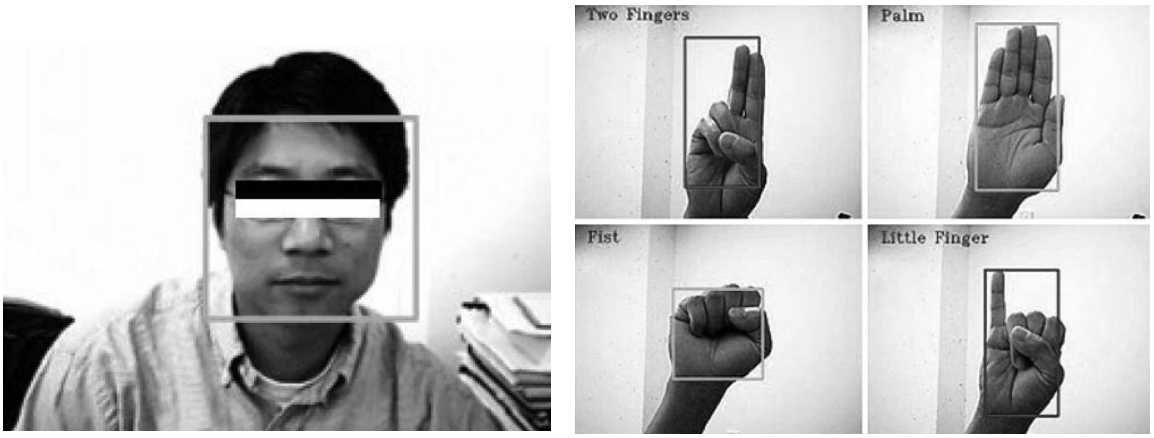
1.2.2 Системи на основі ознак Хаару

За для оминання перелічених у підрозділі 1.2.1 обмежень робота [5] представляє підхід на основі наборів ознак подібних до Хаара.

Ознаки Хаара, більшою мірою, зосереджені на пошуку інформації в межах певної області, а не на кожному пікселі окремо, вони описують співвідношення між темними та світлими ділянками у відповідній частині картинки.

Типовий приклад полягає у тому, що зона очей на обличчі людини завжди темніша за щоки і одна ознака Хаара може ефективно вловити цю характеристику, як зображено на рисунку 1.2 (а).

Порівняно з людським обличчям, що має відносно стабільну морфологію зображення (тобто положення очей, носа та рота на людському обличчі є відносно стабільними), жести рук включають різні форми, які потрібно не тільки виявляти, але й класифікувати. В аналізованій роботі тестуються чотири жести рук: «два пальці» – 480 зразків, «долоня» – 412 зразків, «кулак» – 400 зразків і «мізинець» – 420 зразків. Приклади жестів зображено на рисунку 1.2 (б).



а) б)

Рисунок 1.3 а) Демонстрація успішного детектування зони обличчя, що містить ознаку Хаара [5]; б) Приклад жестів рук для розпізнання [5].

У якості «негативних» зразків в процес навчання класифікатора були включені 500 випадкових зображень, які не мають у собі жестів рук, здебільшого це були фотографії природи та архітектури.

Оскільки система на основі ознак Хаара обчислює різницю в рівнях сірого між білими та чорними прямокутниками – це робить її відносно стійкою до шумів та змін освітлення. Для класифікації жестів у даній роботі використовується алгоритм навчання Adaptive Boost, який може адаптивно змінювати ваги на кожному кроці. На основі навчених каскадних класифікаторів реалізовано паралельну каскадну структуру для класифікації різних поз рук. Експериментальні результати показують, що ця структура може досягти задовільних показників у реальному часі та високої точності класифікації. Таблиця 1.2 показує продуктивність чотирьох навчених класифікаторів і час для обробки всіх тестових зображень.

Таблиця 1.2 Результати розпізнавання навчених класифікаторів [5]

Назва жесту	Успішність розпізнання	Негативне зображення	Час опрацювання (секунди)
Два пальця	100	29	3.049
Долоня	90	0	1.869
Кулак	100	1	2.829
Мізинець	93	2	2.452

Результат експерименту показує, що система може правильно розпізнавати жести рук в реальному часі. Однак, у розглянутій роботі експерименти проводилися із досить тривіальними жестами. Використання ознак Хаара для розпізнавання динамічних слів жестової мови нетривіальна задача з точки зору створення набору характеристик Хаара для комбінацій положення пальців та руки загалом. Для розпізнавання 4 жестів у роботі [5] було використано 14 ознак (4 кутових, 2 центральних, інші – лінійні), зі збільшенням примірників буде зростати і складність комбінацій ознак.

1.2.3 Системи на основі згорткових нейронних мереж

Штучний інтелект пропонує хорошу та надійну техніку, яка використовується в широкому діапазоні сучасних додатків завдяки принципу навчання. Глибоке навчання використовує багатошаровість для вивчення даних і дає хороший прогнозований результат. Найбільше проблем, з якими стикається ця техніка, пов’язано з необхідним набором даних для навчання мережі, що може вплинути на час обробки.

У дослідженні [6] для отримання даних використовується звичайна веб-камера, зображення з якої попередньо перетворюється на чорно-біле фільтром градацій сірого, а також позбавляється шумів розмиттям за Гаусом. Для мінімізації часу навчання нейронної моделі, отримані картинки обрізаються до формату 128×128 пікселів, після чого подаються на вхід багатошарової згорткової нейронної мережі CNN. Остання являє собою систему глибокого навчання, яка використовує фотографії у якості вхідних даних і призначає ваги, або зміщення різним компонентам зображення, дозволяючи розрізняти їх.

На рисунку 1.3 зображено схематичну архітектуру згорткової нейронної мережі.

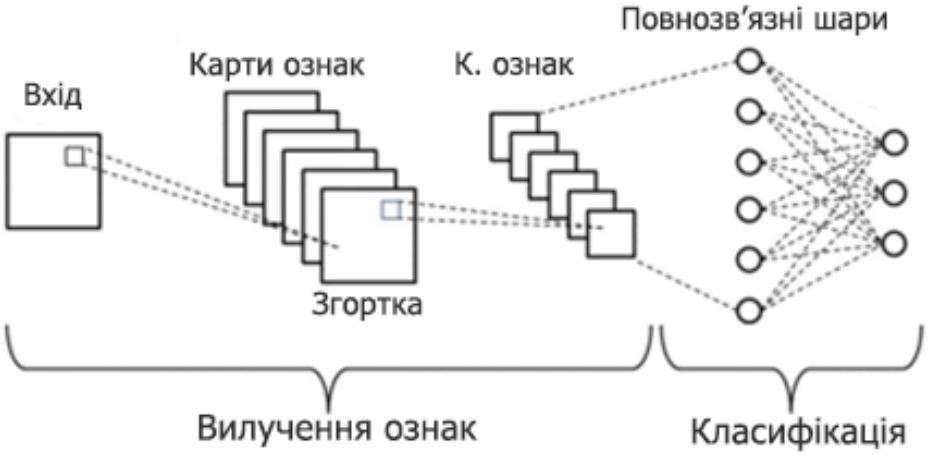


Рисунок 1.3 Базова архітектура CNN

Міцність CNN визначається кількістю прихованих шарів, що використовуються між вхідним і вихідним шарами. Кожен шар виділяє набір функцій. Серія фільтрів застосовується до вхідних даних для створення карт об’єктів.

У цій роботі архітектура нейронної мережі складається з одного шару згортки, який відповідає за виділення меж і домінуючих ознак жестів у бінарних зображеннях, та з 3 повністю повнозв’язних шарів з 128, 96, 64 нейронами відповідно, а також вихідного шару. В результаті модель може класифікувати 26 статичних дактилей ASL з середньою точністю – 98%.

Основним обмеженням використання мереж згортки є їх непристосованість до аналізу послідовностей зображень, більш детально цей аспект аналізується у розділі 2.1.

1.3 Системи автоматичного перекладу ЖМ, засновані на використанні спеціалізованих рукавичок

Спеціалізовані рукавички із датчиками використовуються для фіксації руху та положення руки. Крім того, вони можуть легко надати точні координати розташування долоні та пальців.

Наприклад, у роботі [7] описана методологія розробки системи перетворення жестів у текст, що являє собою спеціальний пристрій-рукавичку, яка складається з сенсорних матриць на основі ниток, що здатні розтягуватися (YSSA), а також бездротової друкованої плати PCB (рис. 1.5).

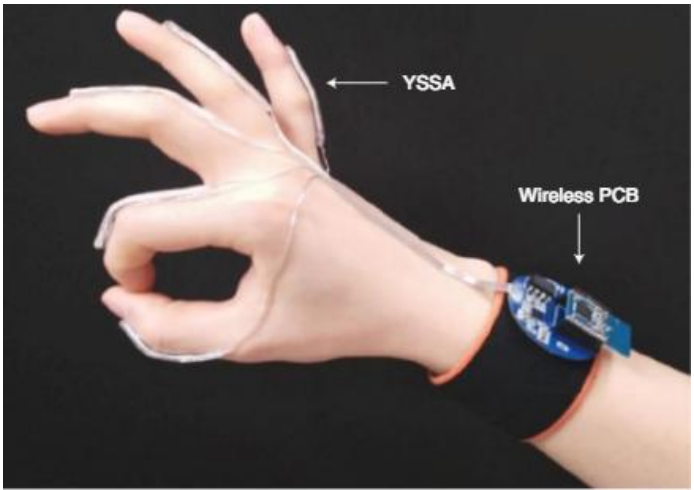


Рисунок 1.5 Фотографія руки з прикріпленою матрицею YSSA та бездротовою друкованою платою PCB [7]

Зазначається, що завдяки унікальному структурному дизайну та використанню м’яких матеріалів сенсорна матриця YSSA здатна підлаштовуватися під еластичну будову шкіри у станах розтягнення та спокою.

Датчики сенсорної матриці YSSA засновані на електризації через періодичну зміну площі контакту двох різнорідних м’яких матеріалів з протилежними трибоелектричними поляризаціями, які можуть ефективно перетворювати невелику силу розтягування або тиск в електрику. Компоненти інтегральної схеми бездротової друкованої плати, що закріплюється на зап’ясті, об’єднують у собі функції формування, обробки та бездротової передачі сигналу. Останнє реалізовано за допомогою bluethooth інтерфейсу, який надсилає дані до налаштованого мобільного додатку, у який вбудовано алгоритм машинного навчання для співставлення невербального жесту з його текстовим представленням.

За допомогою аналогової схеми сигнали, що надходять від датчиків посилюються та позбавляються від шумів навколишнього середовища фільтрами низьких частот, це гарантує отримання точної інформації про відповідний жест для подальшої обробки аналогово-цифровим перетворювачем (АЦП). У мобільний додаток вбудовано алгоритм машинного навчання на основі класифікатора багатокласової опорної векторної машини (SVM). Передані з рукавички ознаки слугують вхідними даними для навченого багатокласового класифікатора SVM.

На рисунку 1.6 наведено структурну схему сценарію роботи переносної системи автоматичного жестового перекладу в реальному часі. Тут зеленим зазначено шляхи отримання аналогових сигналів з кожного пальця руки, синім – обробку та передачу отриманих сигналів та жовтим – розпізнавання жестів за допомогою машинного навчання.

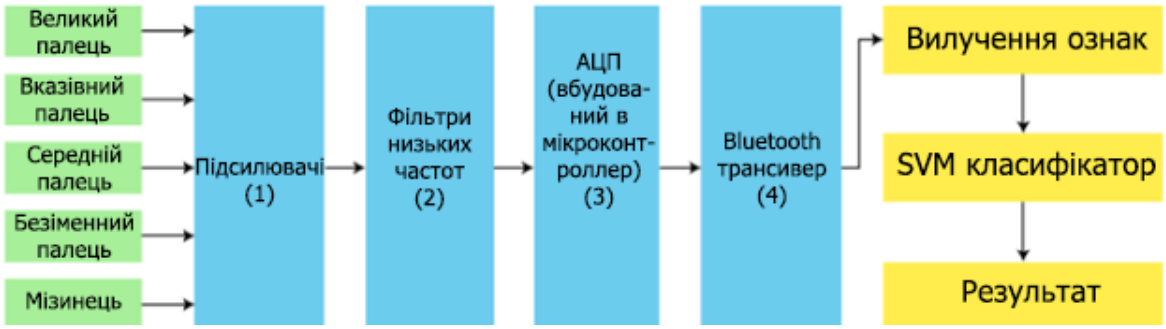


Рисунок 1.6 Схема сценарію роботи переносної системи перекладу ASL [7]

Щоб продемонструвати ефективність сенсорних матриць YSSA для виконання задачі по розпізнаванню компонентів мови жестів у роботі було проведено ряд експериментів по оцінці продуктивності блоку у відповідь на рухи пальців.

Також був проаналізований варіант, коли декілька пальців рухаються одночасно – канали збору даних п’яти пальців незалежні один від одного, тому у результаті інформація про рух кожного об’єднується як складений електричний сигнал, що відповідає унікальному жесту.

У результаті цього дослідження авторам вдалося досягти високих стандартів щодо рівня і часу розпізнавання жесту, а саме – 98,63% і < 1с, відповідно.

Іншою привабливою особливістю є низька собівартість пристрою, у статті заявлено, що усі компоненти коштують менше 50$, що створює колосальну конкуренцію в порівнянні з комерційними системами подібного типу – CyberGlove – 40000$, 5DT Glove – 1990$.

Незважаючи на те, що методи, на основі рукавичок, дали хороші результати, вони мають ряд обмежень, які роблять їх непридатними для певної групи людей. Сюди можуть відноситись люди похилого віку, ті, хто страждають від хронічних захворювань, які призводять до втрати функції м’язів, люди з чутливою шкірою або ті, хто отримав опіки. Деякі з цих проблем були розглянуті в дослідженні Ламберті і Камастри [8], які розробили систему комп’ютерного зору на основі кольорових маркованих

рукавичок. Хоча це дослідження не вимагало приєднання датчиків безпосередньо до шкіри, воно все одно вимагало носіння допоміжного пристрою для виявлення та класифікації жестів.

1.4Системи автоматичного перекладу ЖМ, засновані на використанні технології Microsoft Kinect

З 2012 року дослідники з Китайської академії наук, а також спеціалісти зі сфери освіти глухих з Пекінського університету співпрацюють з Microsoft Research для створення системи автоматичного перекладу з різних ЖМ на словесні мови і навпаки за допомогою Kinect камери [9].

Kinect камера спочатку створювалася як бездротовий контролер для ігрової приставки xbox-360, який дозволяє керувати консоллю та деякими іграми за допомогою просторових жестів, або голосом. Технічно камера складається з двох сенсорів глибини, відеокамери та мікрофону. Датчик глибини являє собою поєднання інфрачервоного проектору та монохромної КМОН матриці, що дозволяє отримувати псевдо трьох вимірне зображення [10]. При використанні цього підходу, освітлення, тінь і колір не впливають на отримане зображення.

Для обрахування глибини отримане з камери інфрачервоне зображення поділяється на зони кольоровим градієнтом – від білого (близькі об’єкти) до синього (об’єкти, що розташовані на віддаленні).

У статті [11] стисло описані головні тези дослідження Microsoft Research Asia присвячені розпізнаванню мови жестів на основі зображень глибини та кольору, наданих Kinect. На рисунку 1.7 наведена блок-схема головного алгоритму:

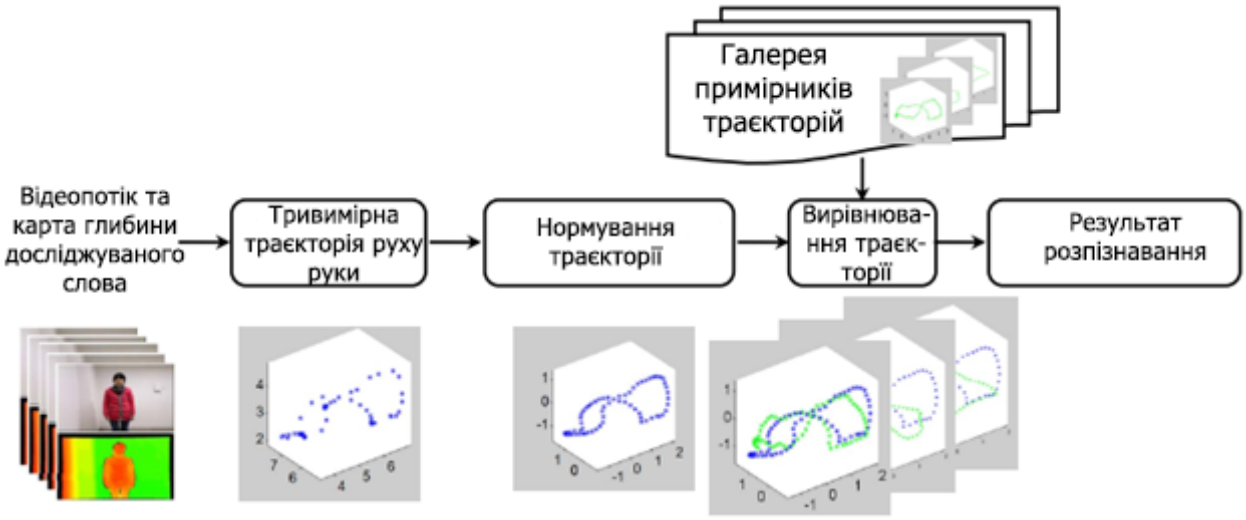


Рисунок 1.7 Блок-схема тривимірного методу розпізнавання мови жестів на основі зіставлення траєкторії [11]

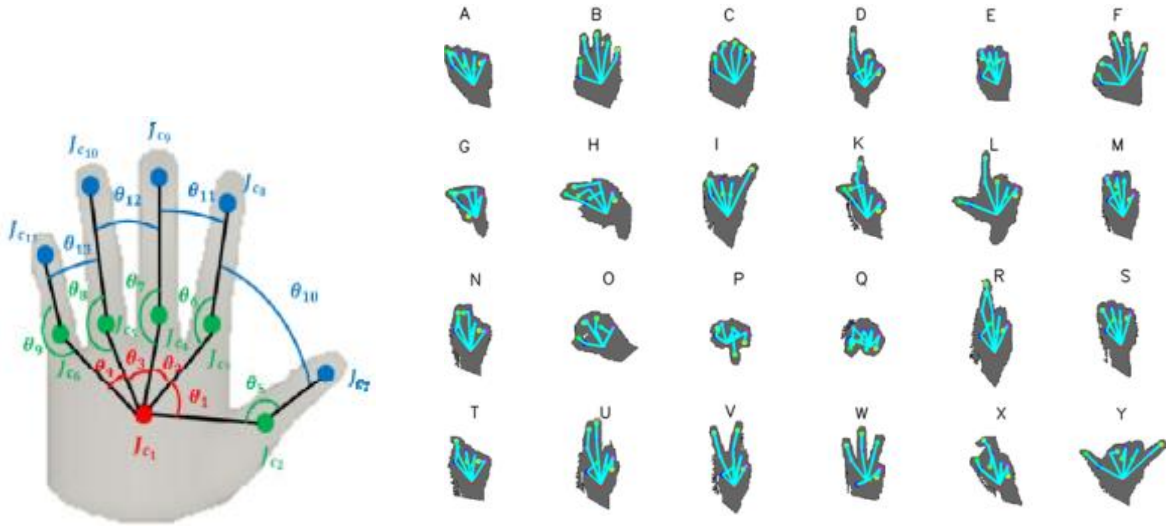
По-перше технологією відстеження руки Kinect Windows SDK генерується тривимірний опис траєкторії, що відповідає вхідному слову мови жестів. Оскільки рука може рухатися нерівномірно, що спричиняє розриви траєкторії, далі виконується її нормалізація шляхом усереднення накопиченої довжини всього вектору. Ідентифікація конкретного жесту відбувається порівнянням отриманої траєкторії з векторами, що наявні в галереї. Останнім пунктом алгоритму є обчислення оцінок відповідності шляхом вимірювання евклідової відстані. Для перевірки продуктивності розпізнавання жестів у цій роботі експерименти проводяться з базою даних, що містить 239 жестових слів китайської ЖМ. Досягнений рівень розпізнання становить 96.32%.

Розроблена система складається з двох режимів: режим перекладу, у якому здійснюється переклад з мови жестів у текст або мову, та режим спілкування, у якому будь-хто може спілкуватися з людиною із вадами слуху через аватар. Режим перекладу включає розпізнавання ізольованих слів і речень, де підняття та опускання рук визначаються як кінець кожного слова мови жестів, яке потрібно розпізнати.

В останні роки фінансова доступність високопродуктивного датчика Microsoft Kinect приваблює все більше дослідників займатися питанням

розпізнавання жестів. В роботі [12], дані про глибину отримані від датчика камери додатково обробляються накладанням на руку макету суглобів з обмеженими діапазонами рухів.

Отримані 13 кутів скелета (рис. 1.8 а, б) були використані у якості ознак жесту, які подавалися на вхід моделі класифікатора на основі алгоритму випадкових лісів. В результаті було реалізовано класифікатор для 24 дактилів ASL з точністю 92%.



а) б)

Рисунок 1.8 а) Використана у роботі [12] модель скелету кисті з позначеними кутами. б) Зразки розпізнаних дактилей ASL, суглоби та скелет показані кольором

Таким чином, використання камери глибини дає хороші результати розпізнання як статичних так і динамічних жестів. Основним недоліком цієї системи є пропрієтарність програмного забезпечення Kinect Windows, через що неможливо отримати повний доступ до налаштувань камери, та користуватися функціоналом обладнання без спеціального доступу.

1.5 Системи автоматичного перекладу ЖМ, засновані на опорних точках скелету кисті

Розпізнавання жесту на основі опорних точок скелету визначає параметри моделі, які можуть покращити виявлення складних ознак. У

задачах, де для класифікації можна використовувати різні представлення скелетних даних моделі руки, опорні точки описують геометричні атрибути та обмеження, які легко транслюються в функції кореляції даних.

Останні декілька років були досить плідними на досягнення у сфері нейронних мереж, більшість сучасних алгоритмів розпізнавання мови жестів так чи інакше включають у себе нейронні моделі. Згорткові нейронні мережі успішно можуть ідентифікувати окремі дактилі, проте, це стосується лише статичних жестів, аналіз динамічних рухів на основі картинок занадто громіздкий та ресурсозатратний процес. Тому ідея вилучати з серії кадрів динамічного просторового жесту окремі ключові точки для подальшого аналізу є привабливою з точки зору точності та економії пам’яті – адже координати ключових точок руки можна компактно зберігати у вигляді матриць не оперуючи початковою фотографією.

У роботі [13] пропонується архітектура системи розпізнавання динамічних жестів з відеозапису. Спочатку відео розбивається на кадри, після чого на кожному кадрі шукаються ключові точки скелету за допомогою фреймворку Alphapose, загалом використовується 55 ключових точок з 136 доступних (без нижньої частини тіла та деталей обличчя). На рисунку 1.9 наведені основні параметри для побудови ключових точок за допомогою Alphapose.

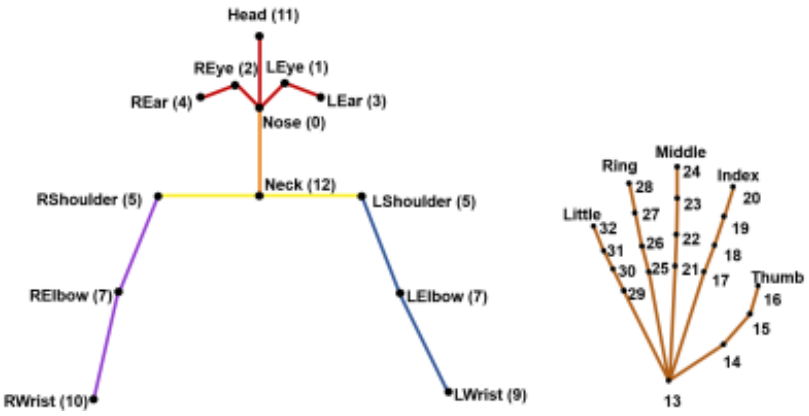


Рис. 1.9. Побудова опорних точок скелету Alphapose [13]

Для доведення продуктивності алгоритму були проведені експерименти з різними загальнодоступними наборами даних, а саме – набором KETI та RWTH-PHOENIX 2014. Набір KETI складається з відео високої чіткості корейською мовою жестів – 105 речень та 419 слів. Набір RWTH-PHOENIX-Weather-2014-T складається з 7096 тренувальних, 519 валідативних та 642 тестових відео німецької ЖМ, створених на основі телевізійних записів прогнозу погоди.

Нейронна мережа для цього проекту побудована за допомогою фреймворку Pytorch, у якості оптимізатора обраний Adam і функція помилки CrossEnthropy. В результаті роботи авторам вдалося досягти точності 85% на всіх наборах даних, практично це дає змогу розрізняти окремі динамічні жести і навіть невеликі словосполучення та речення.

ВИСНОВКИ ДО РОЗДІЛУ 1

В першому розділі дипломної роботи були детально проаналізовані різні підходи до розпізнавання жестів з акцентом на жестові мови. В ході дослідження були розібрані основні переваги та недоліки цих методів.

Оскільки метою роботи є розпізнавання динамічних жестів УЖМ, для проведення досліджень не підходять системи, орієнтовані виключно на пошук жестів в межах статичного зображення.

Використання детекторів шкіри для виявлення контурів руки є недоцільним в умовах високого інформаційного шуму.

Розпізнавання жестів за допомогою ознак Хаара потребує наддетального аналізу кожного жесту окремо, з використанням різних математичних алгоритмів для створення відповідних ознак. Кількість таких ознак для одного складного жесту може перевищувати 50 примірників. Тому, хоча цей алгоритм і дає хороші результати для простих жестів (приклади наведені у розділі 1.2.2), його використання недоцільно для великої кількості складних просторових жестів.

До недоліків використання спеціалізованих рукавичок можна віднести низьку доступність і практичність застосування цих приладів. Адже зазвичай, переклад з якоїсь мови, байдуже словесної чи жестової, потребує швидкого результату тут і зараз.

Аналіз напрацювань у сфері використання камер із датчиками глибини показав перспективність та багатонапрямленість цього методу. Однак, на поточний час хоча камера Microsoft Kinect і є цілком доступною для проведення досліджень, пропрієтарність програмного забезпечення не дає змоги вільно користуватися нею.

Серед всіх проаналізованих методів найбільш пристосованим до реальних умов є метод на основі нейронних мереж, що використовує ключові точки скелету руки для розпізнавання жестів (підрозділ 1.5). Для його впровадження не потрібно ніяких додаткових пристроїв, окрім будь-якої

відеокамери для запису жесту. Оскільки сучасні технології комп’ютерного зору для виявлення на кадрі частин тіла знаходяться на досить високому рівні, в теорії такий підхід повинен мати високі результати за умови достатньої кількості навчальних даних.

РОЗДІЛ 2 ПРОЕКТУВАННЯ СИСТЕМИ

2.1 Застосування нейронних мереж для розпізнання жестової мови

У розпізнаванні жестової мови зазвичай використовуються дві архітектури – згорткові нейронні мережі (CNN) та рекурентні нейронні мережі (RNN).

CNN призначені для ідентифікації просторових візерунків (ознак) у зображеннях, шляхом застосування до останніх згорткових фільтрів. При проектуванні нейронної мережі з цією архітектурою передбачають низку шарів, що досліджують різні типи ознак на різних рівнях абстракції. Перший шар, зазвичай, працює з загальними ознаками, наприклад, краї та кути зображення, наступні рівні ускладняються за допомогою комбінацій більш простих функцій та можуть визначати більш конкретні ознаки. Вихідний шар моделі є повнозв’язним та поєднує усі попередньо визначені ознаки для отримання остаточної класифікації.

У контексті розпізнавання жестів CNN може приймати як вхідні дані фотографію жесту, або послідовність кадрів, що представляє жест. Однак, хоча мережі згортки є потужним інструментом для виявлення об’єктів та класифікації зображень, вони мають деякі обмеження в контексті розпізнавання динамічних жестів, наприклад, жестів з відеоданих. Ключовим обмеженням є фіксований розмір входів, що може спричиняти проблеми для відео різної довжини та роздільної здатності. Крім того, CNN не можуть моделювати часову природу відеоданих, через що досягти «розуміння» часової динаміки жесту може бути досить важко.

Для усунення вищенаведених обмежень при роботі з послідовно пов’язаними даними використовуються рекурентні мережі. Ключовою особливістю цього типу мереж є їх здатність «запам’ятовувати» попередню інформацію. Така пам’ять реалізована наявністю ланцюжка зі зворотнім зв’язком в середині комірки RNN, який дозволяє передавати інформацію від

одного кроку до іншого. У процесі навчання мережі для обчислення результату на поточному кроці навчання використовується прихований стан, що оновлюється на основі попереднього стану. Оновлення як і у традиційних нейронних мережах виконується на основі значень ваг і зміщень, однак у RNN ваги та зміщення розподіляються між усіма кроками, що дозволяє вивчати довгострокові залежності.

Найпоширенішим типом рекурентних мереж є мережі архітектури Long Short-Term Memory (LSTM). Їх популярність пов’язана з вирішенням проблеми затухання градієнтів, що притаманна звичайним RNN мережам. Ця архітектура була обрана для виконання поставлених завдань дипломної роботи, більш детально її розглянуто у наступному підрозділі 2.2.

2.2 LSTM нейронні мережі та її компоненти

2.2.1 Структура LSTM

В порівнянні з традиційними рекурентними мережами, LSTM має більш складну та комплексну структуру, яка дозволяє їй запам’ятовувати, або забувати інформацію з попередніх часових кроків.

Базова архітектура LSTM складається з чотирьох компонентів (шлюзів, або фільтрів), що взаємодіють між собою. Шлюзи являють собою шари сигмоподібної нейронної мережі, які контролюють протікання інформації через LSTM комірку.

Фільтр забуття (Forget Gate) приймає рішення стосовно того, яку інформацію зі стану комірки слід прибрати, тобто, «забути». Він приймає вхідні дані попереднього прихованого стану та поточних вхідних даних і пропускає їх через сигмоїдну функцію. Остання повертає числа 0 або 1, де 0 означає «забути і видалити», а 1 – «запам’ятати і пропустити далі».

Вхідний фільтр (Input Gate) приймає рішення стосовно яку нову інформацію зберігати в стан комірки. На його вхід також подається значення попереднього стану та поточні вхідні дані, які пропускаються через

сигмоїдну функцію. Одержане значення надходить на вхід функції гіперболічного тангенсу, яка повертає значення від -1 до 1. Цей сигнал множиться на той, що був отриманий на виході сигмоїдного шлюзу, створюючи, таким чином, нову інформацію яку необхідно додати до стану комірки.

У стані комірки LSTM зберігається попередній стан, змінений фільтром забуття та нова інформація із вхідного фільтру. Новий стан комірки є сумою поелементного множення (добуток Адамара) вектору фільтру забуття на попередній стан комірки та вектору вхідного фільтра на поточні вхідні дані.

Вихідний фільтр (Output Gate) визначає яким має бути результат комірки LSTM на основі попереднього прихованого стану і поточних вхідних даних.

У підсумку, комірка LSTM вибірково контролює потік інформації, змінюючи та оновлюючи свій стан на основі фільтрів вводу, забуття та виводу. Це дозволяє краще розпізнавати довгострокові залежності, уникаючи затухання градієнту.

2.2.2 Основні компоненти нейронної мережі

При проектуванні архітектури LSTM мережі слід враховувати наступні параметри:

Кількість шарів LSTM. Вибір оптимального значення для цього параметру повинен засновуватися на аналізі та експериментуванні з конкретним набором даних та архітектурою. Оскільки передбачий у роботі набір даних для навчання по міркам глибоких нейронних мереж невеликий (представляє собою 1984 відео з розмірністю вхідного вектора 1662), можна зробити припущення що для поставленої задачі підійде двошарова, або тришарова архітектура.

Кількість нейронів у кожному шарі також має засновуватися на експериментах. Зазвичай, розмірність прихованого шару збільшують

поступово на значення, що відповідають ступеням числа 2. Це пов’язано з особливостями апаратної реалізації комп’ютерних обчислень, які мають обмеження на використання ресурсів оперативної пам’яті. Використання ступеней двійки є поширеною практикою, адже такі значення спрощують обрахунок гіперпараметрів моделі за рахунок використання оптимізованих бібліотек, що орієнтовані на роботу з числами у двійковій системі.

Значення шару відкидання (Dropout rate). Додання шарів Dropout між рекурентними та повнозв’язними шарами дозволяє знизити ймовірність перенавчання моделі. Цей коефіцієнт випадковим чином «відключить» зазначену частку нейронів в процесі навчання, що змусить працювати більш ефективно ті, які залишилися. В залежності від поставленого завдання може знадобитися різне значення шару Dropout. Оскільки, у випадку розпізнавання жестової мови різниця між деякими словами може бути невеликою, передбачається що 20%-30% відсіву буде достатньо.

Функції активації застосовуються до вихідного сигналу кожного LSTM, або Dense шару, та визначають діапазон можливих значень. У Розділі 3 даної роботи проводилися експерименти з наступними функціями активаціїї:

1) **Tanh** (гіперболічний тангенс): зазвичай використовується у прихованих шарах нейронних мереж та приймає значення в діапазоні від -1 до 1 та має форму S-видної кривої. Формула tanh:

$$\tanh(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}} \tag{2.1}$$

2) **Sigmoid**: стискає вхідні дані в діапазоні від 0 до 1, ця функція широко застосовується у задачах бінарної класифікації, а також, коли потрібно отримати ймовірність. Формула сигмоїдної функції активації:

$$\text{sigmoid}(x) = \frac{1}{1 + e^{-x}} \tag{2.2}$$

3) **ReLU** (Rectified Linear Unit): повертає вхідне значення без змін, якщо воно більше за нуль, та нуль, якщо вхідне значення менше, або дорівнює нулю.
4) **SeLU** (Scaled Exponential Linear Unit): це модифікація ReLU, передбачена для протидії занепаду градієнту у глибоких нейронних мережах. Вона має нелінійний характер та повертає значення в діапазоні від -1 до нескінченності. Слід також зазначити, що ця функція складна в обрахуванні та потребує більше ресурсів і додаткового налаштування гіперпараметрів. Формула SeLU:

$$\text{SeLU}(x) = f(x) = \begin{cases} \alpha \times (e^x - 1), & x < 0 \\ x, & x \geq 0, \end{cases} \tag{2.3}$$

де α – константа, що налаштовується, її часто обирають рівною 1.673

Алгоритм оптимізації використовується для оновлення деяких параметрів мережі під час навчання, зокрема, у роботі використовується метод стохастичної оптимізації Adam (Adaptive Movement Estimation). Його основна ідея полягає в підлаштуванні швидкості навчання параметрів на основі моментів градієнту.

Для завдань багатокласової класифікації підходить категоріальна функція втрат. Вона використовується для оцінки різниці між прогнозованим та реальним результатом на валідаційних даних в процесі навчання. Також для оцінки продуктивності моделі на навчальній та тестовій вибірці у роботі використовується категоріальна метрика точності. Вона визначається як відношення правильних передбачень до загальної кількості передбачень міток класів.

Усі описані у цьому підрозділі компоненти нейронних мереж, а також їх можливі значення підбираються емпіричним та експериментальним методами для роботи із визначеним набором даних. Розроблені архітектури найуспішніших моделей та результати їх тестувань на нових даних наведені у розділах 3.4 та 3.5 відповідно.

2.3 Інструментарій

2.3.1 Вибір середовища розробки

Інтерактивні середовища розробки на кшталт Jupyter [14] або Google Collaboratory [15] стали класикою при роботі з нейронними мережами. Обидва варіанти дозволяють писати та виконувати програмний код поділений на окремі комірки та відразу отримувати результат обчислень. Нижче представлена порівняльна таблиця 2.1 цих двох інструментів:

Таблиця 2.1 Порівняння функцій Jupyter та Google Collaboratory

Функція	Jupyter	Google Collaboratory
Розміщення ресурсів	На локальному комп’ютері або сервері	Google Cloud
Модель поширення	Безкоштовний та відкритий код	Безкоштовно з обмеженою кількістю ресурсів
Підтримка GPU	Вимагає конфігурації та налаштування	Доступно для преміум-акаунтів
Можливість колаборації	Локально або за допомогою хмарних служб	В реальному часі за допомогою облікових записів Google
Збереження даних	Локально або на сервері	Google Drive або інші хмарні сервіси
Інтерфейс	Вимагає певних технічних знань	Максимально простий у використанні

Як видно з таблиці, найбільш принципова відмінність полягає у можливості встановлення середовища локально. Хоча Google Collab надає

доволі зручний користувацький інтерфейс і повну взаємопов'язаність з екосистемою Google Drive, Gmail та Google Cloud, у період дії графіків стабілізаційних та екстрених відключень електроенергії ці переваги виявилися різко недоступними. Тож, для виконання завдань роботи було вирішено використовувати Jupyter.

Документи Jupyter мають розширення `ipynb` та можуть містити вхідні та вихідні дані інтерактивного сеансу, а також додатковий текст, який супроводжує код, але не призначений для виконання. Ці документи є внутрішніми файлами JSON. А оскільки JSON – це звичайний текстовий формат, їх можна контролювати версіями та ділитися без зусиль: експортувати в ряд статичних форматів, включаючи HTML reStructuredText, LaTeX, PDF і слайд-шоу.

2.3.2 Основні бібліотеки та інструменти

Для виконання поставлених завдань, а саме: створення власного набору даних, що складається з примірників відеозаписів жестів УЖМ; розробки і навчання класифікатора для розпізнавання жестів з архітектурою LSTM; створення користувацького інтерфейсу для візуалізації проведеної роботи, було обрано мову програмування Python, версії 3.9.

У таблиці 2.2 перелічені основні використані бібліотеки та їх сенс у рамках цієї роботи:

Таблиця 2.2 Основні бібліотеки та їх характеристики

Бібліотека	Короткий опис
Запис відеоданих та їх попередня обробка	
os	Взаємодія з файлами операційної системи
cv2	Запис відео з відповідними налаштуваннями, захоплення та вивід зображення з камери
datetime	Додання часових міток для ведення логування
numpy	Збереження масивів файлів опорних точок та

	робота з ними
mediapipe	Обробка мульти-медіа даних, пошук на відеокадрі опорних точок пози, рук та обличчя

Продовж. табл. 2.2

Бібліотека	Короткий опис призначення
Розробка та навчання моделі нейронної мережі	
tensorflow.keras.models /layers/optimizers/ /callbacks/utils	Надання моделей, шарів, оптимізаторів, зворотних викликів та різноманітних утиліт для роботи з даними моделі, її створення і навчання
sklearn.model_selection	Функціонал для розділення масиву даних на тренувальні, валідаційні та тестові вибірки
Створення графічного користувацького інтерфейсу	
PIL	Відображення поточного кадру в інтерфейсі tkinter
tkinter	Створення вікон для графічного інтерфейсу
ImageTK	Конвертація масиву пікселів Numpy кадру у об’єкт зображення
threading	Створення окремого потоку для паралельного запису, збереження та обробки відео
ScrolledText	Створення текстового поля з можливістю прокрутки для виведення службової інформації

Окрему увагу варто приділити інструменту MediaPipe [16]. З огляду на те, що створення чогось на кшталт моделі відстеження рук займає багато часу та ресурсів, було вирішено використовувати готовий інстумент від Google.

Це бібліотека, яка надає розробникам доступ до декількох моделей машинного навчання націлених на відслідковування людської фігури у статичних зображеннях, або динамічних відео.

Практичне застосування цього інструменту наведено у розділі 3.3. У роботі використовуються всі три доступних моделі:

- MPHands для знаходження у кадрі обох рук;
- MPPose для відслідковування всієї пози загалом;

- MPFaceMesh, що утворює спеціальну 3Д сітку обличчя для відстеження змін міміки.

У таблиці 2.3 описана кількість опорних точок, які утворюються після обробки зображення цим інструментом для відстеження вищезгаданих параметрів тіла людини:

Таблиця 2.3 Параметри MediaPipe

Параметр	Кількість точок
Обличчя	468
Ліва рука	21
Права рука	21
Поза	33
Visiability	11
Загалом для 1 кадру відео	**554**

Слід також підкреслити, що при передачі масиву координат контрольних точок в модель нейронної мережі його потрібно зробити одновимірним. Отже, вхідний шар моделі буде приймати відео з 90 кадрів та із 1662 (554 × 3) опорними точками у кожному кадрі, оскільки кожна точка має 3 координати.

На рисунку 2.1 демонструється як виглядає кадр відео з нанесеними опорними точками:

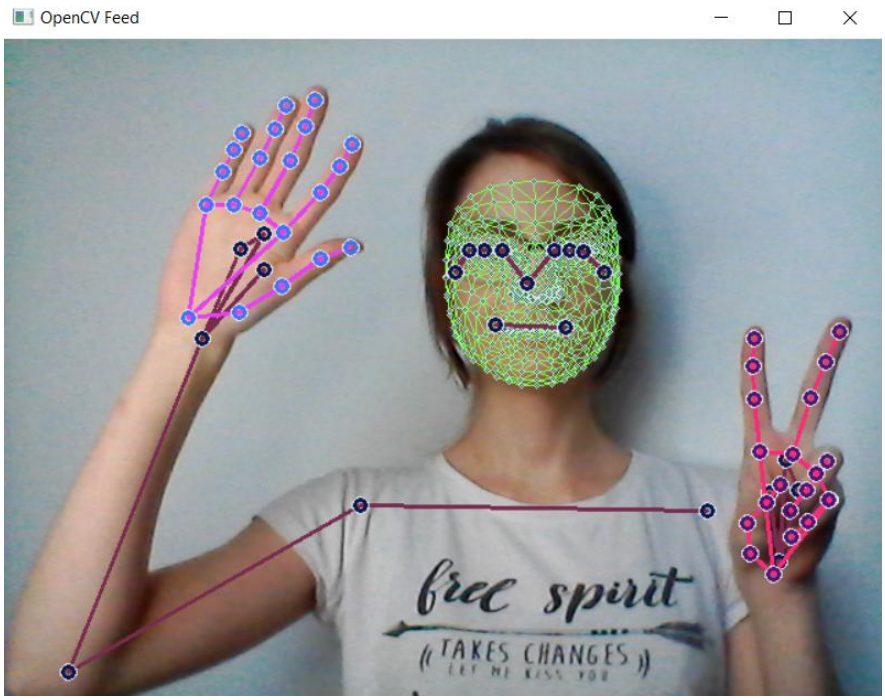


Рисунок 2.1 Демонстрація нанесення ключових точок на відеокадр

2.4 Проектування системи

Вхідними даними, з якими працює розроблена система розпізнавання жестів є відеозаписи динамічних жестів української жестової мови з наперед визначеного набору певних категорій. Так як запис усіх примірників відео для роботи виконувався власноруч – всі дані були стандартизовані, тобто мають однакову тривалість – 3 секунди, кількість кадрів на секунду – 90 та розширення – 1280×720 пікселів.

Для попередньої обробки відеозаписів вирішено використовувати бібліотеку MediaPipe, яка дозволяє отримувати просторові координати ключових опорних точок скелету людини. Для кожного кадру відео будуються 554 точки.

Аналіз можливих алгоритмів класифікації жестів, проведений у розділі 1 виокремив нейронні мережі як інструмент, що не потребує інших додаткових пристроїв, окрім будь-якої відеокамери. Натомість, тип архітектури LSTM був обраний як найбільш пристосований до виконання поставлених цілей роботи, шляхом аналізу у розділах 2.1.1 – 2.1.2.

На рисунку 2.2 наведено схему роботи розробленої системи:

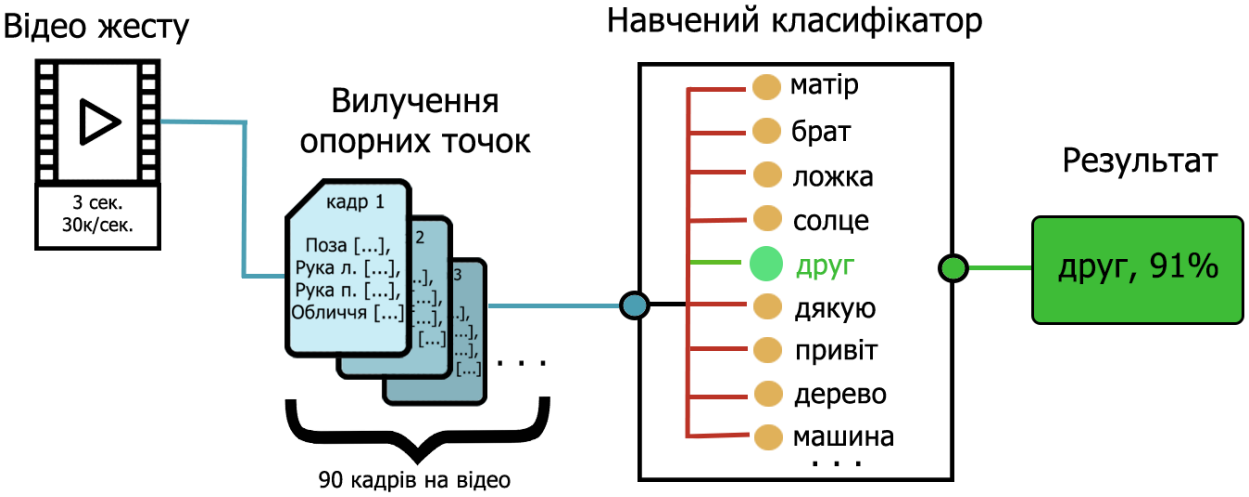


Рисунок 2.2 Схема алгоритму роботи системи розпізнавання жестів УЖМ

Алгоритм роботи розробленої системи полягає у наступному:

1) Відбувається запис відео з усіма необхідними налаштуваннями.
2) Перед подачею відеозапису до моделі нейронної мережі він обробляється шляхом пошуку та вилучення з кадру ключових точок скелету за допомогою інструменту MediaPipe.
3) Масив координат, отриманих на попередньому етапі, подається на вхід навченого LSTM класифікатора, де відбувається розпізнавання жесту.
4) Одержаний у минулому пункті результат виводиться разом зі своєю точністю. Після цього процедура розпізнавання може бути повторена або припинена.

2.5 Постановка задачі

Задачею дипломної роботи є створення та навчання моделі LSTM класифікатора для розпізнавання жестів УЖМ. Для цього передбачено створення власного навчального набору відеофайлів, та їх попередня обробка у вигляді вилучення з кадрів координат опорних точок.

Для демонстрації успішної роботи класифікатора необхідно розробити відповідне програмне забезпечення з графічним інтерфейсом користувача. До функцій якого будуть входити:

- Захоплення зображення з підключеної відеокамери;
- вивід на екран розпізнаного жесту;
- можливість надання зворотного зв’язку щодо правильності розпізнаного жесту.

РОЗДІЛ 3 РЕАЛІЗАЦІЯ РОЗПІЗНАВАННЯ ЖЕСТІВ УЖМ

3.1Пошук та вибір базису жестів для навчання класифікатора

3.1.1 Ресурси

Як вже зазначалося у вступній частині роботи – українська жестова мова не уніфікована та не має єдиної бази, у якій би були збережені всі жестові слова. На зараз, одним з найкращих варіантів вивчення жестової мови є проходження спеціалізованих курсів, оскільки специфіка ЖМ вимагає чіткого просторового уявлення про формування кожного, навіть найпростішого, жесту.

У зв’язку з вищенаведеним виникає багато труднощів, пов’язаних із пошуком даних для створення та навчання відповідного класифікатора для УЖМ. Також, розпізнавання динамічних жестів висуває основну вимогу для типу даних – це мають безперечно бути відеозаписи. На відміну від американської, китайської та німецької жестових мов, для яких розроблені набори даних [17, 18, 19] для дослідницьких цілей, нічого подібного не було створено в українському медіапросторі.

Ресурсів для пошуку і визначення базису жестових слів небагато. По-перше цю роль можуть відігравати відповідні відеоролики на платформі YouTube, які націлені на людей з обмеженими можливостями. Існує багато записів новин та телепередач, що на державному рівні вимагають синхронного сурдоперекладу і як наслідок, можуть слугувати якісним матеріалом для створення датасету. Проте, «телевізійна» жестова мова не є природньою у тому сенсі, що хоча окремі слова у ній дійсно співпадають із тією ЖМ на якій спілкуються між собою її носії, порядок слів та побудова речень при синхронному перекладі повністю повторюють сказане на словесній мові. Така ЖМ називається калькувальною, тобто вона копіює словесну у буквальний переклад. Хоча, для створення набору даних з окремих слів такий підхід не принциповий існують і інші вади, які на даному

етапі не дозволяють використовувати напрацювання записів телевізійних передач. Сюди відноситься і формат запису – зазвичай віконце сурдоперекладача займає приблизно 10-20% всього екрану через це якість відтворення обрізаного та збільшеного кадру буде страждати від шумів при зміненні формату. Ідеальним варіантом для створення детального та багато напрямленого набору даних буде отримання прав від телеканалів на використання оригінальних відеозаписів сурдоперекладу. Такий підхід може бути використаний у багатьох дослідницьких роботах, проте на цьому етапі він недоступний.

Іншим ресурсом, який використовується в роботі є відеозаписи на платформі YouTube навчального характеру. Під час пандемії 2020-2021 років, на початку локдауну в Україні, співробітники житомирської обласної універсальної наукової бібліотеки ім. Олега Ольжича розпочали проект «Жестова мова» [20]. Сюди входять 10 відео уроків УЖМ у яких були розібрані особливості відображення найнеобхідніших жестів для спілкування з людьми з обмеженими можливостями.

Другим ресурсом для створення власного набору даних українських жестових слів є матеріали курсу лекцій з базових навичок жестової мови благодійного фонду імені Сергія Горового [21], який читає Маріна Ліферрова – президент СПЖМ (спілка перекладачів жестової мови) та перекладач жестової мови телеканалу «Прямий». Цей онлайн курс включає в себе 6 розгорнутих лекцій, у яких детально описуються особливості положення рук і тіла, наводиться багато прикладів, а також пояснюються основи формування речень та лінгвістичних особливостей УЖМ.

Останнім та основним ресурсом жестових слів виступає проект Spreadthesign [22] – це онлайн словник жестів різних ЖМ, що адмініструється неприбутковою організацією «Європейський центр жестових мов» («European Sign Language Center») [23]. Проект вже зібрав та задокументував понад 400000 жестів і знаходиться у процесі постійного розвитку. На сайті Spreadthesign надається можливість перегляду

відеоматеріалів будь-якого з наявних у базі даних жестів, а також їх можливих варіацій. На сайті можна знайти не тільки жести для позначення окремих понять, а і деякі популярні фрази та речення. З усіх наявних у публічному просторі матеріалів Spreadthesign надає найбільш зручний користувацький інтерфейс в поєднанні з обширною базою даних.

3.1.2 Формування словника співвідношень жестів та слів

Для дослідницьких цілей роботи з ресурсів, описаних у пункті 1.1 даного розділу було відібрано 38 динамічних жестових слів. Всі вони наведені у таблиці 3.1.

Таблиця 3.1. Перелік жестів та їх особливості

№	Жест (англ.)	Жест (укр.)	Одноручний/ дворучний	Варіації домінуючої руки
1	*apple*	*яблуко*	1	+
2	*ball*	*м'яч*	2	-
3	*bike*	*велосипед*	2	-
4	*book*	*книга*	2	-
5	*bowl*	*миска*	2	-
6	*boy*	*хлопчик*	1	+
7	*brother*	*брат*	1	+
8	*butterfly*	*метелик*	2	-
9	*car*	*автомобіль*	2	+
10	*chess*	*шахи*	2	-
11	*day*	*день*	2	-
12	*evening*	*вечір*	2	+
13	*father*	*батько*	1	+
14	*friend*	*друг*	1	+
15	*fruit*	*фрукти*	1	+
16	*good*	*добрий*	2	+
17	*goodbye*	*до побачення*	1	+
18	*guitar*	*гітара*	2	+
19	*hello*	*привіт*	1	+
20	*home*	*дім*	2	-
21	*I*	*я*	1	+
22	*interpreter*	*перекладач*	2	-
23	*island*	*острів*	1	+

24	*medal*	*медаль*	1	+
25	*morning*	*ранок*	1	+

Продовж. табл. 3.1

№	Жест (англ.)	Жест (укр.)	Одноручний/ дворучний	Варіації домінуючої руки
26	*mother*	*матір*	1	+
27	*my*	*мій*	1	+
28	*night*	*ніч*	1	+
29	*only*	*тільки*	2	+
30	*salt*	*сіль*	1	+
31	*skydiver*	*парашутист*	2	+
32	*spoon*	*ложка*	1	+
33	*sun*	*сонце*	1	+
34	*team*	*команда*	2	+
35	*thanks*	*дякую*	1	+
36	*tiger*	*тигр*	2	-
37	*tree*	*дерево*	2	+
38	*voice*	*голос*	1	+

У таблиці стовпчик «Одноручний/дворучний» відповідає кількості рук, задіяних у відображенні жесту, а стовпчик «Варіації домінуючої руки» за симетричність відображення (зазвичай всі одноручні жести можуть показуватися як лівою так і правою рукою, тому для них виділяється дві категорії).

При виборі жестів увага зверталася на їх складність для повторювання початківцем, тривалість жестового слова (стандарт прийнятий в роботі передбачає тривалість до 3 секунд), а також на схожість між собою. Наприклад, демонстрація жесту «яблуко» [24] в цілому схожа на «фрукти» [25], жест «батько» [26] на жест «дякую» [27], а жест «я» [28] на «мій» [29]. Правильне розпізнання класифікатором візуально схожих жестів дозволить робити комплексні висновки про його якість.

3.2Створення набору даних

Для створення великої множини відео однакової довжини, кількості кадрів та параметрів роздільної здатності була розроблена функція `create_video`, рисунок 3.1.

```
def create_video(name, no_sequences=2, sequence_length=91, root='videos'):
    # Create the directory for the video if it doesn't exist
    dir_path = os.path.join(root, name)
    os.makedirs(dir_path, exist_ok=True)
```

Рисунок 3.1. Функція `create_video` (1)

Вона отримує на вхід назву відео `name`, кількість відео, яку треба записати – `no_sequences` та довжину кожного відео у кадрах – `sequence_length`. За замовчуванням кількість відео дорівнює 2, а кількість кадрів – 91. Функція також містить аргумент `root`, що вказує на каталог для збереження результату. За замовчуванням відео зберігаються у папку `videos`. Далі робиться перевірка на існування відповідного каталогу з переданою назвою. При його відсутності – він створюється у папці `root` з іменем `name` за допомогою модуля `os`.

Далі головна підфункція `record_sequence` (рис. 3.2) приймає аргумент `sequence`, що являє собою порядковий номер відео, яке буде записано. Напочатку відкривається камера за допомогою `cv2.VideoCapture(1)` (порядковий номер у скобках відповідає за номер підключеної камери, за замовченням – 0, у даному випадку – 1 камера смартфону) і встановлюється бажане розширення відео у 1280×720 пікселів.

```
def record_sequence(sequence):
    # Open the camera and set the resolution
    cap = cv2.VideoCapture(1)
    cap.set(3, 1280)
    cap.set(4, 720)
    width = 1280
    height = 720
    # Create a video writer to save the frames
    writer = cv2.VideoWriter(
        '{}/{}_{}.mp4'.format(dir_path, name, str(len(os.listdir(dir_path))).zfill(2)),
        cv2.VideoWriter_fourcc(*'DIVX'), 30, (width, height))
```

Рисунок 3.2. Функція `create_video` (2)

Створений об’єкт `writer` класу `cv2.VideoWriter` почергово записує кадри відео до файлу з назвою, що складається з аргументу `name`, порядкового номера та розширення `.mp4`.

У наступному фрагменті коду (рис. 3.3) відбувається цикл, що проходить по всім кадрам, що отримуються з камери за допомогою методу `cap.read()`. Перший з 91 кадрів має службовий характер та слугує для виводу на екран повідомлення «GET READY for RECORDING collection» з назвою відео та номером його послідовності. Після чого функція очікує 2 секунди, щоб користувач зміг підготуватися до запису.

```
# Loop through the frames
for j in range(sequence_length):
    ret, frame = cap.read()

    # Show a "Get Ready" message before the first frame
    if j == 0:
        blank_image = cv2.imread('blank.jpg')
        cv2.putText(blank_image, 'GET READY for RECORDING collection {}'.format(name),
                    (10, int(height/2)),
                    cv2.FONT_HERSHEY_SIMPLEX, 0.9, (0, 255, 0), 2, cv2.LINE_AA)
        cv2.putText(blank_image, 'Collecting frames for {} Video {}'.format(name, sequence),
                    (int(width/2 - 80), 60),
                    cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0, 0, 255), 1, cv2.LINE_AA)

        cv2.imshow('frame', blank_image)
        cv2.waitKey(2000)
```

Рисунок 3.3. Функція `create_video` (3)

Для всіх інших кадрів відбувається запис до файлу (рис. 3.4). Також, при натисканні кнопки `q` можна перервати процедуру: всі створені в функції об’єкти вивільнюються за допомогою методів `writer.release()` і `cv2.destroyAllWindows()`. В кінці викликається цикл для створення декількох відео. На цьому алгоритм завершується.

```
            else:
                # Write the frame to the video writer
                writer.write(cv2.resize(frame, (width, height)))
                cv2.imshow('frame', frame)
            # Quit if the user presses "q"
            if cv2.waitKey(10) & 0xFF == ord('q'):
                break
    # Release the video writer, camera, and close the window
    writer.release()
    cap.release()
    cv2.destroyAllWindows()
# Loop through and record multiple sequences
for n in range(no_sequences):
    record_sequence(n)
```

Рисунок 3.4. Функція `create_video` (4)

Таким чином, після виконання цієї функції на локальному комп'ютері створюється дерево каталогів для кожної категорії жестів. Наприклад, для першого відео з категорії «друг», воно має наступний вигляд: `os.path\Папка_проекту\videos\friend\friend_00.mp4`.

Створений набір відео складається з відтворених 38-ми популярних жестів УЖМ. Запис одного жесту відбувався з різних ракурсів для більш точного навчання моделі, також одноручні та деякі дворучні жести, спосіб виконання яких залежить від домінуючої руки, були записані у різний спосіб. Враховуючи останнє – кількість категорій зросла до 66-ти.

Згодом, для неупередженого тестування моделей додатково було записано ще по 10 примірників для 38-ми загальних категорій які зовсім не використовувалися для навчання.

Загалом було записано 2364 відео, загальною тривалістю 118 хвилин, тобто, майже 2 години.

Нижче, у таблиці 3.2 наведено зведену інформацію щодо результатів проведеної роботи по створенню навчального та тестового наборів даних.

Таблиця 3.2. Результати щодо створеного набору даних

Кількість жестів	Кількість категорій	Відео у категорії	Розширення
38	66	30	1280×720
Фреймрейт	Тривалість 1 відео	Всього відео	Загальна тривалість
30 к/сек.	3 сек.	2364	118 хв.

3.3 Підготовка даних. Виділення ключових ознак

Цей етап передбачає попередню обробку всіх тренувальних відео, з метою визначення характерних ознак кожного жестового слова. У конкретному випадку це означає, що за допомогою інструментів комп'ютерного зору, наданих бібліотекою Mediapipe визначаються опорні

точи скелету людини для кожного кадру відео, координати яких будуть записані до відповідного файлу.

Перш за все необхідно визначити об'єкт моделі `mp_holistic` для детектування ключових точок на зображенні для областей обличчя, рук та пози тіла загалом (рис. 3.5).

```
mp_holistic = mp.solutions.holistic # Holistic model
mp_drawing = mp.solutions.drawing_utils # Drawing utilities
```

Рисунок 3.5. Визначення об'єктів моделі

Ця модель передається у якості аргументу до функції `mediapipe_detection`, що виконує знаходження ключових точок на зображенні (рис. 3.6).

```
def mediapipe_detection(image, model):
    # Convert image color space from BGR to RGB
    image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
    # Set the image as non-writeable
    image.flags.writeable = False
    # Use the specified model to make predictions on the image
    results = model.process(image)
    # Set the image as writeable again
    image.flags.writeable = True
    # Convert the image color space back from RGB to BGR
    image = cv2.cvtColor(image, cv2.COLOR_RGB2BGR)

    # Return the image and the results of the model prediction
    return image, results
```

Рисунок 3.6. Функція `mediapipe_detection`

Напочатку функція приймає зображення – `image` в колірному просторі BGR, який за замовчуванням використовується у бібліотеці OpenCV. Після чого зображення конвертується у колірний простір RGB, з яким працює модель `mp_holistic`.

Для покращення показників продуктивності флаг можливості запису `writeable` встановлюється на False – це робить зображення незмінним, що в свою чергу впливає на результативність, оскільки обробка незмінюваних даних відбувається швидше, ніж змінюваних.

Результати передбачень положення ключових точок скелету людини записуються у змінну `results`, а флаг можливості запису та колірний

простір повертаються у початковий стан. В результаті функція повертає оброблене зображення та передбачення моделі.

Наступною функцією, яка вже безпосередньо працює із створеними відеофайлами є `parse_all_frames` (рис. 3.7).

```
# parse video and create pictures for each frame with landmarks on it
def parse_all_frames(video_basename, basename, number):

    # Get the path of the video to parse
    video_path = 'videos/{}/{}.mp4'.format(basename, video_basename)
    # Set the directory path to save the keypoints
    dir_path = 'keypoints/{}/{}_{}'.format(basename, basename, str(number).zfill(2))
    print('in parse:', basename, number, video_path, dir_path)

    try:
        # Create a directory to save the keypoints if it doesn't exist
        os.makedirs(os.path.join(dir_path))
    except:
        # If the directory already exists, return
        return

    # Open the video for reading
    cap = cv2.VideoCapture(video_path)

    # If the video cannot be opened, return
    if not cap.isOpened():
        return
```

Рисунок 3.7. Функція `parse_all_frames` (1)

Вона читає відеофайл за вказаним шляхом, створює каталог для збереження ключових точок (якщо він відсутній) і після вилучення ключових точок із кожного кадру відео зберігає їх у форматі numpy.

Для вилучення ключових точок для всіх необхідних параметрів (поза, руки, обличчя) використовується модель Holistic з бібліотеки Mediapipe (рис 3.8).

```
# Set the base path for saving keypoints
base_path = os.path.join(dir_path, basename)
# Get the number of digits for file names
digit = len(str(int(cap.get(cv2.CAP_PROP_FRAME_COUNT))))
# Set the initial frame number to 0
n = 0
# Create a holistic object for detecting keypoints
with mp_holistic.Holistic(min_detection_confidence=0.5, min_tracking_confidence=0.5) as holistic:
    while True:
        # Read a frame from the video
        ret, frame = cap.read()

        if ret:
            # Make detections on the frame
            image, results = mediapipe_detection(frame, holistic)
            # Save the extracted keypoints
            np.save('{}_{}'.format(base_path, str(n).zfill(digit)), extract_keypoints(results))
            # Increment the frame number
            n += 1
        else:
            # If there are no more frames to read, return
            return
```

Рисунок 3.8. Функція `parse_all_frames` (2)

На кожному кадрі відео, що зчитуються за допомогою `cap.read()` виконується виклик попередньо розглянутої функції `mediapipe_detection`.

На основі результатів детектування моделі для кожного кадру викликається допоміжна функція `extract_keypoints` (додаток А), яка відповідає за вилучення координат ключових точок і їх збереження в окремий файл формату numpy, з номером поточного кадру у якості імені. Цикл продовжується доки не залишиться кадрів з відео для читання, після чого функція завершує роботу.

Допоміжна функція `parse_all_videos_for` (додаток А) відповідає виключно за правильну систематизацію ключових точок зі всіх тренувальних відеофайлів з каталогу `videos/{name}`, отриманих за допомогою попередньої функції. Після її виконання на локальному комп'ютері утворюється система каталогів для збереження файлів numpy з координатами ключових точок. Наприклад, утворене дерево каталогів для зберігання ключових точок 2-го кадру 1-го відео жесту «друг» для лівої руки має наступний вигляд: `os.path\Папка_проекту\keypoints\friend(l)\ \friend(l)_00\friend(l)_01.npy`.

Таким чином було реалізовано попередню роботу з вилучення ключових ознак у вигляді координат орієнтирів для пози, рук та обличчя для їх подальшої передачі в обробку моделі машинного навчання.

3.4 Розробка архітектури моделей нейронних мереж

Особливості типу та доцільність використання нейронних мереж архітектури LSTM були детально проаналізовані у Розділі 2. На даному етапі роботи ставиться завдання по визначенню розмірності та кількості прихованих шарів нейронної моделі, а також підбору гіперпараметрів. Далі у цьому підрозділі будуть розібрані п'ять найбільш вдалих із розроблених

моделей, а у наступному підрозділі 5 продемонструється їх тестування, для наочного підкріплення якості.

Незмінними параметрами, що були використані для всіх описаних нижче прикладів є оптимізатор Adam, функція втрат categorical_crossentropy, а також відповідна метрика точності – categorical_accuracy.

Адаптивний оптимізатор Adam є одним з найпопулярніших та оптимальних оптимізаторів для навчання нейронних мереж. Його особливість, і разом з тим універсальність, полягає в обчисленні адаптивних швидкостей навчання для кожного параметру на основі середнього першого і другого градієнтних моментів. Підхід, використаний в алгоритмі дозволяє йому швидше наближатися до оптимальних значень ваг.

Вибір функції втрат, що відповідає за розходження між передбаченими та фактичними значеннями, та метрики точності засновується на поставленій задачі багатокласової класифікації – кожен приклад жесту може відноситися тільки до одного класу.

Іншою константою для всіх представлених у підпунктах 4.1-4.5 моделей є аргумент форми вхідних даних для першого шару LSTM. А саме – `input_shape = (90, 1662)`, де 90 – це довжина послідовності кадрів у навчальних відео, а 1662 – кількість контрольних точок у кожному кадрі, отриманих на попередньому етапі обробки даних.

3.4.1 Модель 1

Архітектура Моделі 1 складається з трьох шарів LSTM з функцією активації гіперболічний тангенс (`tanh`), за якими йдуть два повнозв’язні (Dense) шари з активацією `relu`, між якими розміщено шари вилучення Dropout з аргументом 0.2 (20% нейронів попереднього шару будуть «відключені», що змусить брати на себе більше відповідальності ті, що залишилися), це зменшує вірогідність перенавчання моделі. Вихідний шар

має функцію активації `softmax`, яка дозволяє отримувати відсоткове представлення точності класифікації жесту.

Архітектура Моделі 1 наведена на рисунку 3.9 а):

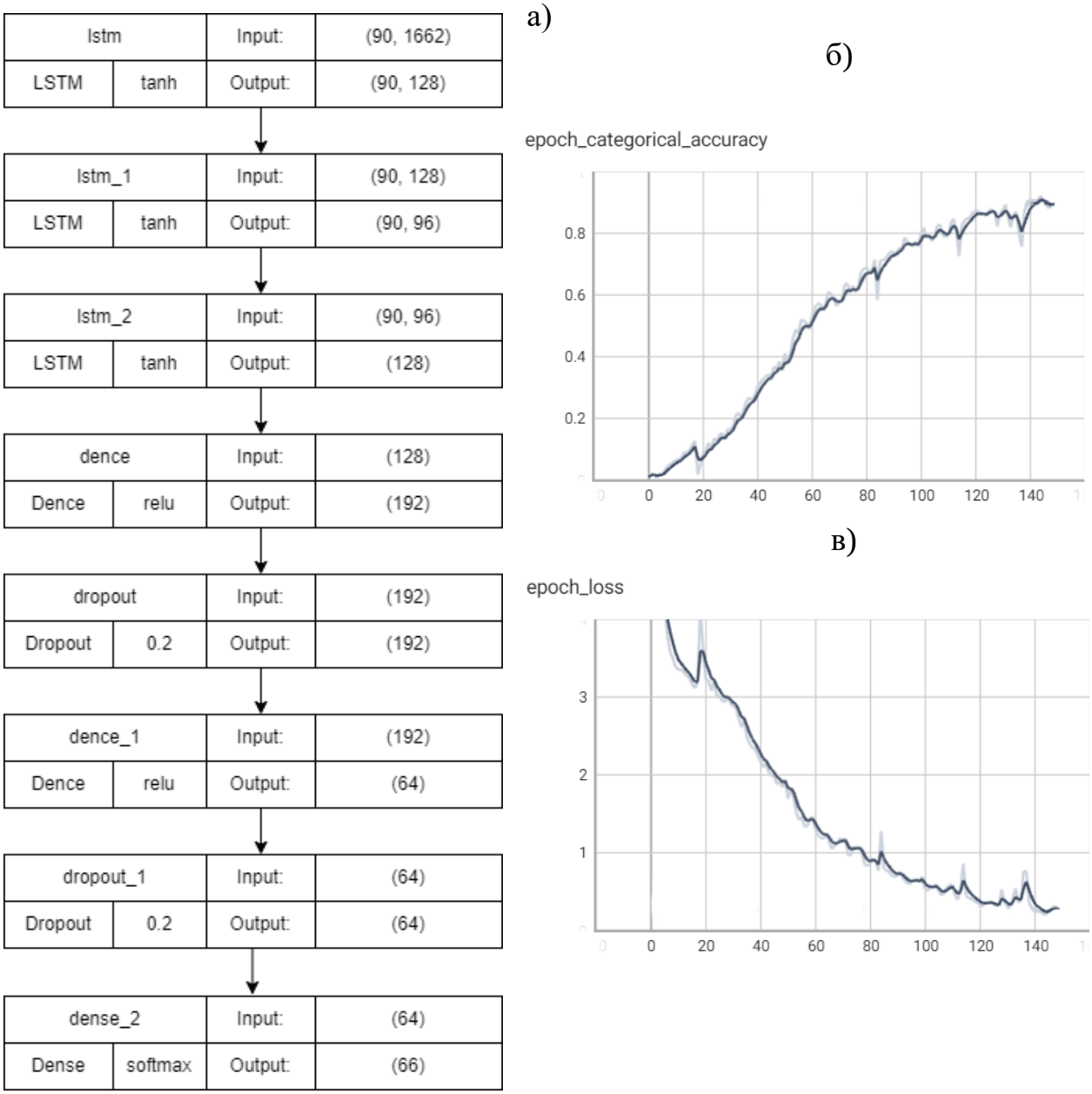


Рисунок 3.9. Схема архітектури Моделі 1 (а), залежність метрики точності (б) та функції втрат (в) від кількості епох

Описані гіперпараметри (кількість шарів та нейронів, функції активації тощо) були визначені шляхом експериментів для досягнення оптимальної продуктивності на конкретному наборі даних у завданні по розпізнаванню жестів УЖМ.

Для даної архітектури був явно визначений параметр кроку оновлення ваг на кожній ітерації навчання – `learning_rate = 0.0008` (зменшений зі стандартного 0.001), що означає, що оптимізатор буде використовувати саме цю фіксовану швидкість навчання. Теоретично, зменшений крок оновлення ваг може допомогти алгоритму точніше зійтися до оптимальних значень, проте серед ризиків є збільшення ймовірності загальмовування навчання.

На досягнення точності у 0.91 із втратами 0.22 на валідаційних даних моделі знадобилося 150 епох навчання (рис. 3.9 б, в). На тестовій вибірці досягнута точність становить 0.86.

3.4.2 Модель 2

Різниця між результатами розпізнавання тестового та валідативного наборів Моделі 1 з попереднього підпункту у 5% наштовхує на міркування про те, що розроблена архітектура погано сприймає нові дані. Одним із варіантів з чим це може бути пов’язано є те, що через порівняно невелику кількість навчальних даних мережа почала «запам’ятовувати» особливості конкретних відеозаписів, а не спільні ознаки притаманні категорії. Можливим варіантом вирішення такої проблеми є додання більшої кількості шарів вилучення не тільки між повнозв’язними, а і між LSTM шарами.

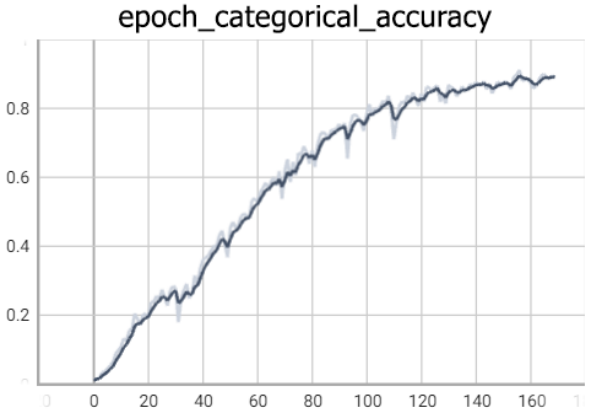
Архітектура Моделі 2 складається з трьох шарів LSTM з функцією активації гіперболічний тангенс. Кількість прихованих нейронів для вхідного шару було вирішено збільшити, в порівнянні з попередньою структурою, до 256. Між перешим та другим шаром додано Dropout з коефіцієнтом 0.2 для запобігання перенавчання. Після цього додані два повнозв’язні Dense шари з 256 та 64 нейронами відповідно та функцією активації `relu`, між якими також розміщено рівень виключення випадкових 20% нейронів. Вихідний шар залишився аналогічним Моделі 1 – він містить кількість нейронів відповідно до кількості передбачених класів (66) та активацію `softmax`.

Модель 2 має наступну архітектуру, наведену на рисунку 3.10, а:

а)

lstm		Input:	(90, 1662)
LSTM	tanh	Output:	(90, 256)
dropout		Input:	(90, 256)
Dropout	0.2	Output:	(90, 256)
lstm_1		Input:	(90, 256)
LSTM	tanh	Output:	(90, 64)
dropout_1		Input:	(90, 64)
Dropout	0.2	Output:	(90, 64)
lstm_2		Input:	(90, 64)
LSTM	tanh	Output:	(64)
dence		Input:	(64)
Dence	relu	Output:	(256)
dropout_2		Input:	(256)
Dropout	0.2	Output:	(256)
dense_1		Input:	(256)
Dense	relu	Output:	(64)
dense_2		Input:	(64)
Dense	softmax	Output:	(66)

б)



в)

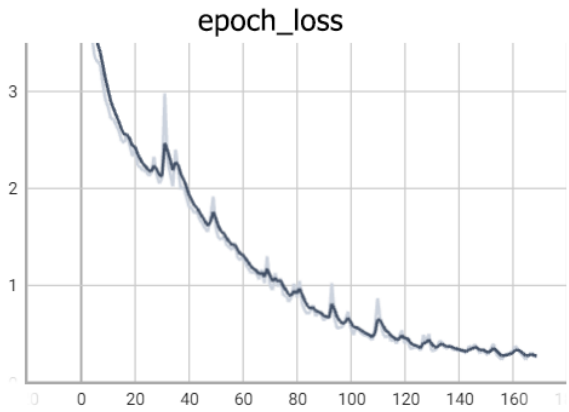
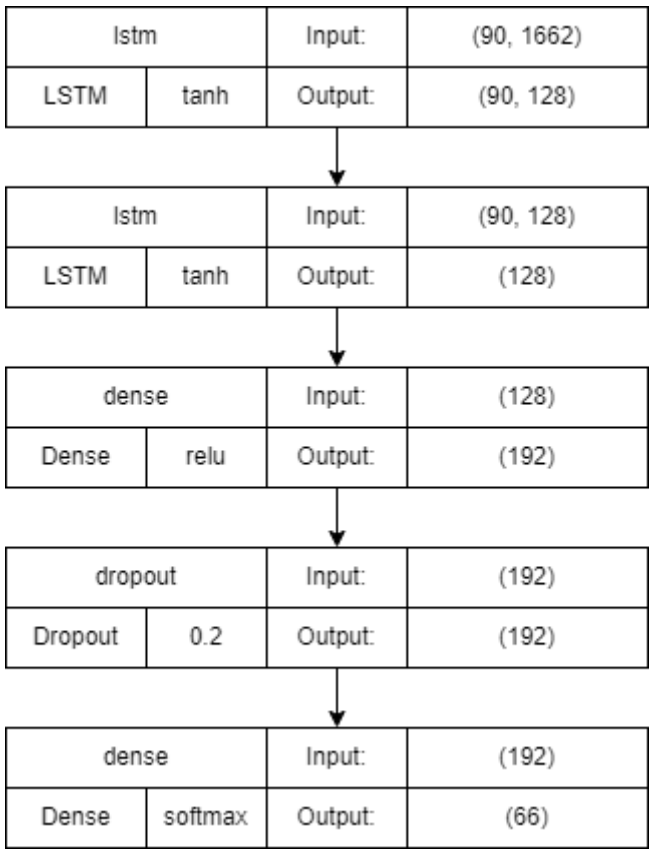


Рисунок 3.10. Схема архітектури Моделі 2 (а), залежність метрики точності (б) та функції втрат (в) від кількості епох

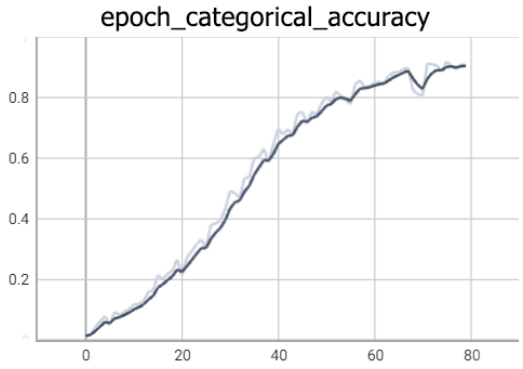
Для даної моделі параметр швидкості оновленні ваг було вирішено змінити на адаптивний. За 180 епох була досягнута точність – 89% при втратах – 0.27. Сенсу продовжувати навчання до 90 і більше відсотків не було, оскільки спостерігалася чітка тенденція «стрибків» точності на останніх 30 епохах (рис. 3.10 б, в). На тестовій вибірці точність склала 0.82. Одержаний результат виявився гіршим за попередню модель. Можливо, збільшення кількості випадаючих шарів призвело до втрати необхідної для точної класифікації жесту інформації.

3.4.3 Модель 3

Архітектура для Моделі 3 наведена на рисунку 3.11 а) а).



б)



в)

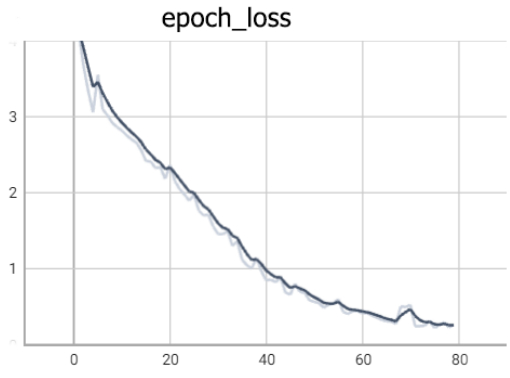


Рисунок 3.11. Схема архітектури Моделі 3 (а), залежність метрики точності (б) та функції втрат (в) від кількості епох

При проектуванні Моделі 3 було вирішено зменшити кількість як LSTM так і повнозв’язних шарів. Розроблена архітектура складається з вхідного LSTM шару на 128 нейронів с функцією активації гіперболічний тангенс, аналогічного за параметрами другого шару LSTM, повнозв’язного шару на 192 нейрони з активацією `relu`, після якого розміщений регуляризаційний рівень Dropout з коефіцієнтом 0.2 та вихідний шар.

Завдяки зменшенню кількості шарів у мережі швидкість навчання зросла майже удвічі. За 80 епох досягнена точність Моделі 3 становить 0.90 при втратах 0.26 (рис. 3.11, б, в). На тестових даних точність понизилася незначною мірою до 0.88.

Як висновок можна сказати, що двох LSTM шарів вистачає для поставленої задачі класифікації 66 жестів. Тому у наступних етапах будуть проводитися експерименти з гіперпараметрами для покращення даної архітектури.

3.4.4 Модель 4

Архітектура Моделі 4, наведена на рисунку 3.12 а):

а) б)

в)

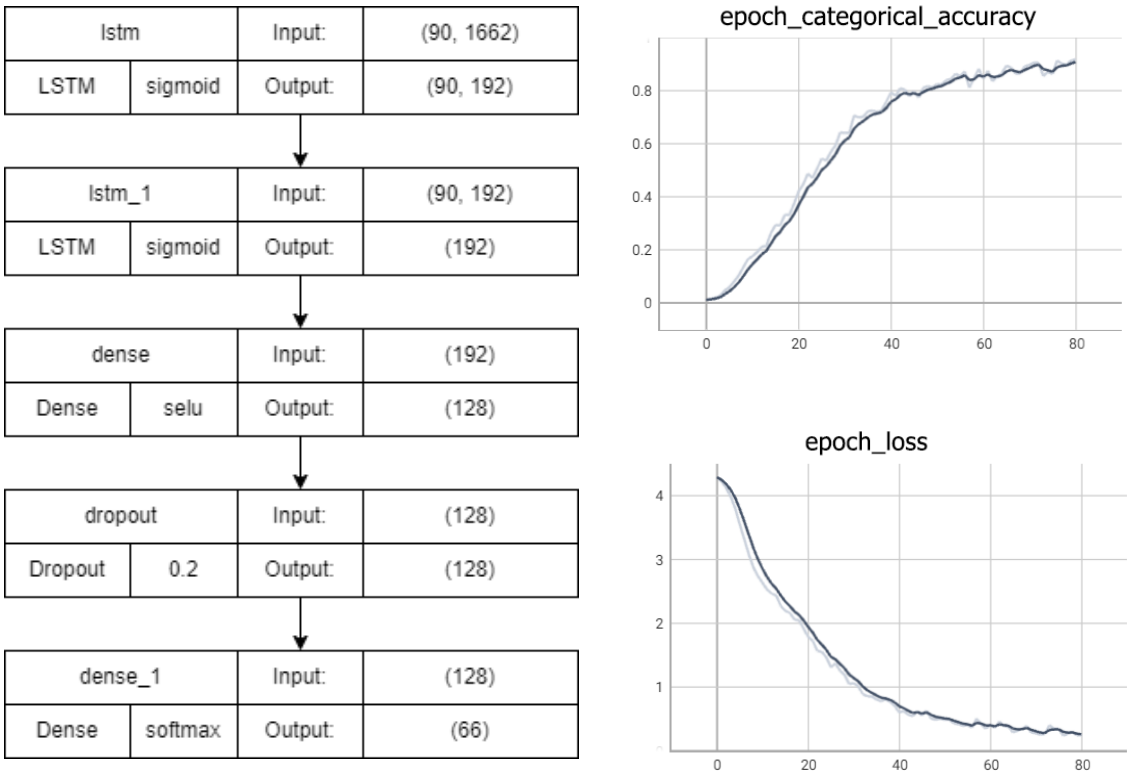


Рисунок 3.12. Схема архітектури Моделі 4 (а), залежність метрики точності (б) та функції втрат (в) від кількості епох

Як було виявлено на попередньому етапі – глибини у 2 LSTM шари для гнучкого підлаштовування під нові дані та визначення складних ознак достатньо. Модель 4 має більшу кількість нейронів у шарах LSTM, а саме – 198, також було вирішено поекспериментувати з функцією активації та замінити `tanh` на `sigmoid`. Активація повнозв'язного Dense шару також замінена на `selu`, а кількість нейронів у ньому знижена до 128. Вихідний шар та Dropout залишилися аналогічними Моделі 3.

Слід зазначити що за фактичним часом, що був витрачений на навчання Модель 4 працює повільніше за Модель 3, проте за ті ж самі 80 епох була досягнена точність у 0.92 із втратами 0.25 на валідаційних даних. На тестовому наборі отримана точність становить 0.9. Отже був отриманий невеликий прогрес, пов'язаний з заміною функції активації на `sigmoid`.

3.4.5 Модель 5

Архітектура Моделі 5 є наслідком закріплення позитивної тенденції попереднього підпункту. Емпіричним шляхом було виявлено, що збільшення кількості прихованих нейронів не призводить до значних покращень, а іноді і погіршує продуктивність.

На цей раз було прийняте рішення зменшити розмірність двох перших рекурентних шарів до 128, та збільшити повнозв'язний шар до 192. Також для архітектури моделі 5 було проведено багато експериментів з різними значеннями параметрів `batch_size` (16, 32, 64, 128), `learning_rate` (статичні 0.0005, 0.0007, 0.0008, та адаптивний 0.001).

Оптимальні комбінації гіперпараметрів були знайдені за допомогою пошуку по сітці (grid search) у попередньо визначеному просторі пошуку.

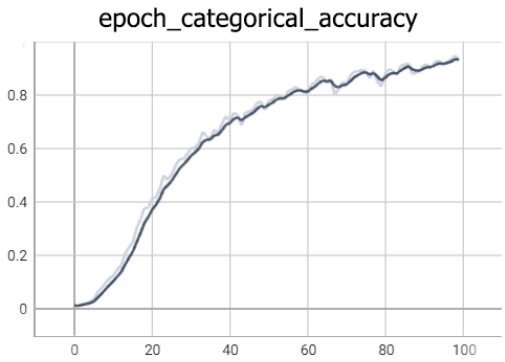
Найкращі результати були досягнені при розмірі пакету – 32, а також при адаптивній швидкості оновлення ваг, за замовчуванням для оптимізатора Adam – 0.001.

Схема архітектури Моделі 5 наведена на рисунку 3.13 а):

а)

lstm		Input:	(90, 1662)
LSTM	sigmoid	Output:	(90, 128)
lstm_1		Input:	(90, 128)
LSTM	sigmoid	Output:	(192)
dense		Input:	(192)
Dense	selu	Output:	(128)
dropout		Input:	(128)
Dropout	0.2	Output:	(128)
dense_1		Input:	(128)
Dense	softmax	Output:	(66)

б)



в)

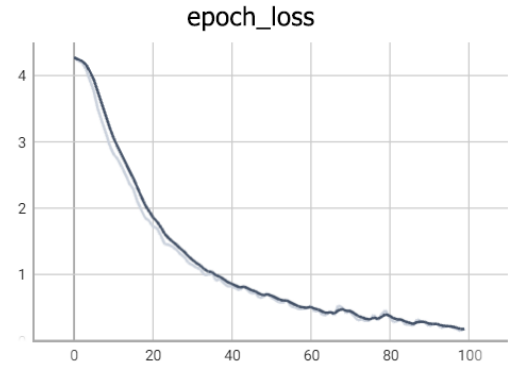


Рисунок 3.13. Схема архітектури Моделі 5 (а), залежність метрики точності (б) та функції втрат (в) від кількості епох

За 100 епох навчання Модель 5 досягла точності 0.93 при втратах 0.19. Це кращий результат серед всіх розроблених нейронних мереж у рамках даної роботи. На тестових даних точність розпізнавання становить 0.94, що навіть дещо перевищує очікування, в порівнянні з валідаційними даними.

3.4.6 Підсумки

У цьому пункті роботи представлені схеми архітектур, діаграми функцій втрат та метрики точності в залежності від кількості епох навчання та особливості розробки п’яти моделей нейронних мереж для розпізнавання жестів УЖМ.

Найкращою з розроблених моделей є Модель 5, описана у підрозділі 4.5, її точність на валідаційних даних становить 93% і на тестових – 94%.

Ваги для всіх розроблених моделей збережені у форматі `.h5` з відповідним номером моделі. З усіма створеними наробками можна ознайомитися через репозиторій, наведений у додатку А.

У таблиці 3.3 наведена зведена інформація щодо розроблених моделей, а також назви файлів збережених ваг.

Таблиця 3.3. Результати навчання класифікатора

Назва	Модель 1	Модель 2	Модель 3	Модель 4	Модель 5
Епохи	150	170	80	80	100
Точність в.	0.91	0.89	0.90	0.92	0.93
Втрати	0.22	0.27	0.26	0.25	0.19
Точність т.	0.86	0.82	0.88	0.90	0.94
Файл ваг	MODEL1.h5	MODEL2.h5	MODEL3.h5	MODEL4.h5	MODEL5.h5

У цій таблиці під скороченнями «Точність в.» і «Точність т.» розуміється точність моделі на валідаційній та тестовій вибірці відповідно.

3.5 Тестування розроблених моделей нейронних мереж

3.5.1 Тестування

Для проведення неупередженого тестування розроблених моделей необхідно мати набір даних, який не використовувався для навчання. Це дозволяє оцінити реальну ефективність моделей в різних умовах та визначити їхню загальну придатність для використання в реальних задачах.

У даному випадку було записано 380 нових відео жестів, по 10 для кожної загальної категорії (без розділення на домунуючі руки).

Для розпізнавання жестів з тестових відео була розроблена функція `sign_from_video` (рис. 3.14). Функція приймає шлях до відеофайлу як вхідні дані, та зчитує відео за допомогою OpenCV.

```
def sign_from_video(path):
    # Open video file
    cap = cv2.VideoCapture(path)

    # Check if video file was successfully opened
    if not cap.isOpened():
        return 'unknow (1)'  # Return string indicating video file was not successfully opened

    # Initialize empty list to store keypoints
    keypoints = []

    # Initialize Holistic model for full-body pose detection and tracking
    with mp_holistic.Holistic(min_detection_confidence=0.5, min_tracking_confidence=0.5) as holistic:
        while True:
            # Read a frame from the video file
            ret, frame = cap.read()

            if ret:
                # Make detections
                image, results = mediapipe_detection(frame, holistic)

                # Extract keypoints from the results of the pose detection
                keypoints.append(extract_keypoints(results))
            else:
                # No more frames to read
                break
```

Рисунок 3.14. Функція `sign_from_video` (1)

Далі за допомогою моделі MediaPipe з відео витягуються координати ключових точок, допоки всі кадри відеозапису не будуть оброблені.

Після попередньої обробки кадрів, ключові точки пози, обличчя та рук подаються до навченої моделі, для розпізнавання виконуваного жесту (рис. 3.15).

```
# Use the trained model to predict the sign language gesture from the extracted keypoints
res = model.predict(np.expand_dims(keypoints, axis=0))[0]
first = np.argmax(res)
second = secondmax(res, first)

# Release the video file
cap.release()

# Return a string indicating the predicted sign language gesture, as well as the top two predicted
#gestures and their corresponding probabilities
return ('{}({:.0f}%); {}({:.0f}%)'.format(actions[first], res[first] * 100,
actions[second], res[second] * 100) or 'unknow (2)', actions[first], actions[second])
```

Рисунок 3.15. Функція `sign_from_video` (2)

У даному фрагменті коду змінні `first` та `second` відповідають за жести, що мають найбільшу ймовірність за розпізнанням. Також функція повертає відсоткове представлення ймовірності розпізнаних жестів.

У таблиці 3.4 наведені результати тестувань розглянутих у підпунктах 4.1-4.5 моделей на нових відеозаписах жестів. Цифри в основних стовпчиках таблиці від 1 до 10 показують скільки правильних відповідей було надано по кожній з 38 категорій. В двох останніх рядках наведено сумарну кількість «балів» та результуючу точність вираховану за середнім арифметичним.

Таблиця 3.4. Результати проведення тестувань

№	Жест	Модель 1	Модель 2	Модель 3	Модель 4	Модель 5
1	*яблуко*	9	9	7	8	8
2	*м'яч*	7	9	7	10	10
3	*велосипед*	9	3	5	8	10
4	*книга*	8	9	7	10	9
5	*миска*	6	5	8	8	8
6	*хлопчик*	7	4	9	4	7
7	*брат*	7	10	8	10	10
8	*метелик*	3	6	5	8	8
9	*автомобіль*	8	7	9	10	10
10	*шахи*	3	2	9	3	10
11	*день*	10	10	10	10	10
12	*вечір*	8	9	9	10	10
13	*батько*	5	6	7	5	9
14	*друг*	9	7	7	10	9
15	*фрукти*	9	10	10	10	8
16	*добрий*	7	6	8	10	7
17	*до побачення*	8	5	10	7	8
18	*гітара*	9	7	9	7	9
19	*привіт*	3	4	6	6	7
20	*дім*	6	3	10	8	10
21	*я*	10	8	9	10	10
22	*перекладач*	3	5	6	2	4
23	*острів*	9	10	8	9	10
24	*медаль*	8	7	9	6	10
25	*ранок*	10	10	10	10	10
26	*матір*	9	9	10	10	9
27	*мій*	1	3	1	7	3
28	*ніч*	8	8	8	10	7
29	*тільки*	10	10	10	10	9
30	*сіль*	3	3	7	8	7
31	*парашутист*	9	7	9	9	9
32	*ложка*	9	7	6	7	9

Продовж. табл. 3.4

№	Жест	Модель 1	Модель 2	Модель 3	Модель 4	Модель 5
33	*сонце*	9	9	9	10	10
34	*команда*	7	8	7	8	9
35	*дякую*	6	8	8	10	5
36	*тигр*	9	8	7	10	9
37	*дерево*	9	10	10	10	10
38	*голос*	9	7	7	7	9
Підсумок		279	268	301	315	326
Точність		**73.42%**	**70.52%**	**79.21**	**82.90%**	**85.79%**

Кращий результат на новому тестовому наборі даних показала Модель 5, набрана сумарна кількість балів дорівнює 326 з 380, що у відсотковому представленні становить 85.79%.

3.5.2 Підсумки

Подальше підвищення точності, безперечно, потребує збільшення кількості матеріалів для навчання. З таблиці 3.4 також можна зробити висновок про принципове «несприйняття» усіма моделями певних жестів. Як от точність розпізнавання жесту «перекладач» ніколи не перевищує 60%. Схожа ситуація із жестом «мій», «дякую», «дім», «метелик»... Однією з ймовірних причин такої тенденції є образотворча схожість деяких жестів між собою.

З іншої сторони точність розпізнавання жестів, таких як «ранок», «день» на тестовому наборі для всіх моделей складає 100%. Позитивна ситуація також спостерігається з жестами «дерево», «матір», «машина»… З цього можна зробити висновок, що відео, які належать до даних категорій достатньо унікальні в межах створеного набору даних та їх ключові особливості добре зчитуються та класифікуються моделлю.

Іншою можливою причиною, по якій не вдається підвищити точність є використання інструменту MediaPipe для виявлення координат опорних точок скелету. Хоча, в порівнянні з іншими бібліотеками того ж напрямку,

MediaPipe є безсумнівним лідером, під час перевірки результатів обробки було виявлено, що для деяких окремих кадрів відео контрольні точки будуються некоректно.

Більшою мірою, повна відсутність, або неправильність ключових точок пов'язана зі швидкістю виконання жесту. З програмної точки зору за це відповідають два параметри, що використовуються для визначення граничних значень впевненості детектування (`min_detection_confidence`) і відслідковування (`min_tracking_confidence`) точок. В теорії – чим вище значення даних параметрів тим більш точно та надійно буде працювати алгоритм виявлення. З іншого боку, якщо вірогідність визначення ключової точки з боку бібліотеки буде низька, наприклад через «розмите» зображення, при високих значеннях цих параметрів вона не зможе відслідкувати, або виявити координати. Низка експериментів з параметрами відслідковування і виявлення показала що оптимальними значеннями є середнє – 0.5.

На якість попередньої обробки може позитивно вплинути збільшення кількості кадрів на секунду (у роботі використовується 30). Такий підхід зробить швидку зміну положення рук у відео більш плавною для програми, хоча для людського зору це може бути малопомітно. Однак, зворотнім боком буде збільшення часу і обчислювальних ресурсів, що будуть використовуватись для обробки відеозаписів.

Отже, досягнута точність розпізнавання загалом ілюструє вірність обраного шляху у питанні архітектури моделі нейронної мережі та допоміжних інструментів. Деякі з можливих причин зниженої точності розпізнавання жестів можна усунути завдяки розширенню набору тренувальних даних. Інші, пов'язані з недосконалістю алгоритмів комп'ютерного зору, потребують більшої кількості ресурсів, детального аналізу та проведення додаткових досліджень.

3.6 Розробка графічного інтерфейсу користувача для системи розпізнавання жестів УЖМ

3.6.1 Формування технічного завдання

Обумовленість розробки мінімального користувацького інтерфейсу спричинена необхідністю практичного тестування системи розпізнавання жестів. Хоча у минулому розділі і була представлена частина тестування, вона працює хоч і з новими, але наперед записаними та збереженими відеозаписами. Подібні наробки погано підходять для «живої» демонстрації можливостей системи.

Інша причина для прямої візуалізації роботи полягає у необхідності перевірки реакції на фактори, які можуть призвести до нестабільності та некоректній реакції моделі нейронної мережі. Оскільки всі примірники відео були записані виключно автором роботи, а отже і всі контрольні точки засновуються на власних параметрах рук, пози та обличчя існує ймовірність погіршення результатів при використанні системи іншими користувачами.

При проектуванні інтерфейсу користувача основними вимогами були зручність, зрозумілість та надання можливості бачити прогрес роботи програми.

Згодом після початку роботи над інтерфейсом стало зрозуміло, що досягти синхронного розпізнавання у реальному часі не вдасться. Одночасна обробка кожного кадру у вигляді додання ключових точок, відправка пакету кадрів до нейронної мережі, а також паралельне розпізнавання спеціально визначених жестів, що сигналізують про початок та кінець жестового слова, яке необхідно розпізнати, споживала надмірну кількість обчислювальних ресурсів.

Тому було прийняте рішення створити програму, яка б записувала відео, тривалістю 3 секунди при натисканні кнопки після чого тимчасово зберігала його. Після розпізнання та повернення результату користувачу необхідно надати можливість залишити відгук щодо правильності відповіді програми,

адже ніяких інших орієнтирів у даному випадку немає. Також у інтерфейсі потрібно реалізувати рядок логування та при натисканні відповідної кнопки повертати статистику точності розпізнавання жестів за сесію роботи.

3.6.2 Розробка

Першим етапом розробки інтерфейсу є визначення необхідних інструментів. На рисунку 3.15 наведено імпортування всіх бібліотек, які знадобляться для роботи програми.

```
from PIL import Image, ImageTk # for handling and displaying images
import tkinter as tk # for creating graphical user interfaces
import datetime # for getting current time
import cv2 # for computer vision tasks like image processing and video capture
import os # for interacting with the file system
import threading # for performing multiple tasks simultaneously
import time # for measuring time and delaying execution
from tkinter.scrolledtext import ScrolledText # for creating a text widget with scrollbars
```

Рисунок 3.15. Імпорт необхідних бібліотек

Оскільки повний код програми є досить громіздким – він наведений у репозиторії, доступному за посиланням у додатку А.

Метою програми є захоплення відеопотоку з камери та розпізнавання мови жестів у потоці за допомогою попередньо навченої моделі машинного навчання, а також відображення розпізнаного жесту в інтерфейсі.

Графічний інтерфейс користувача складається з двох загальних секцій. Ліва секція містить кілька віджетів, включаючи кнопки для розпізнавання і збереження відео та виводу статистики. Також тут розміщене текстове поле для відображення розпізнаного жесту, мітки стану роботи програми та кнопка відгуку. У правій секції відображається відеопотік, знятий камерою.

Клас програми містить декілька функцій, основні з яких:

`decode_video` – призначена для захоплення відео з веб-камери та розпізнавання мови жестів. У методі ініційована модель MediaPipe для виявлення та відстеження контрольних точок скелету. Визначені ключові точки передаються в вже розглянуту у розділі 5.1 функцію

`sign_from_video`, яка повертає два перших передбачених жести та їх ймовірності. Також метод змінює мітки стану обробки відео, які показуються у інтерфейсі. Всього передбачено 3 мітки. Перша інформує про те, що зараз відбувається запис відео. Після натискання кнопки «Recognise» у правій секції виводиться попередження приготуватися, яке триває 2 секунди, далі починається запис. Друга мітка статусу відповідає за процес розпізнавання, вона також виводить тривалість у секундах записаного відео. І остання мітка повідомляє, що жест розпізнано і повертає можливість натискання кнопки запису.

`rec_prop_routine` – викликається, коли користувач натискає кнопку «Recognised properly?». Остання відповідає за встановлення зворотного зв'язку і дозволяє інформувати програму стосовно правильності результату роботи. Отже, при натисканні кнопки (що означає, що жест розпізнаний коректно) збільшується значення змінної, що відповідає за успішність, а також змінюється стан кнопки для збереження відео з `disabled` на `normal`. Тобто, якщо відео розпізнано правильно – користувачу надається можливість його зберегти на локальному комп'ютері, поповнюючи таким чином можливий майбутній набір даних.

`save_routine` викликається, коли користувач натискає кнопку «Save video». Цей метод зберігає відеопотік у файл, використовуючи поточну дату, час, а також результат розпізнавання жесту як назву. За замовчуванням файл зберігається у каталог `temp`.

3.6.3 Результати

На рисунку 3.16 наведений розроблений графічний інтерфейс користувача. Всі поставлені завдання та вимоги до інтерфейсу були реалізовані. Зліва знаходиться службова форма для керування функціоналом, у секції справа під час очікування команд транслюється відеопотік із камери.

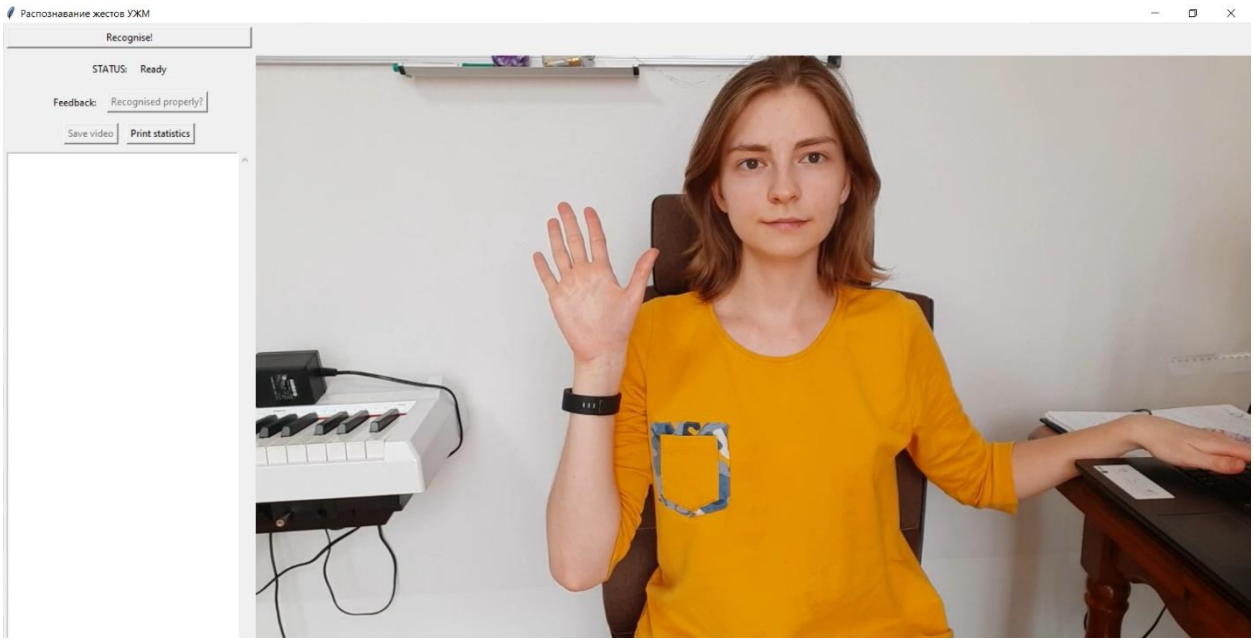
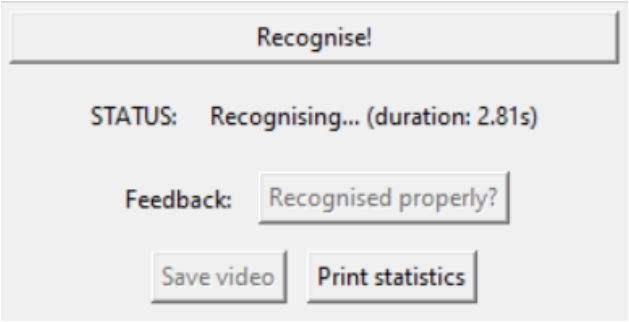


Рисунок 3.16. Демонстрація інтерфейсу користувача

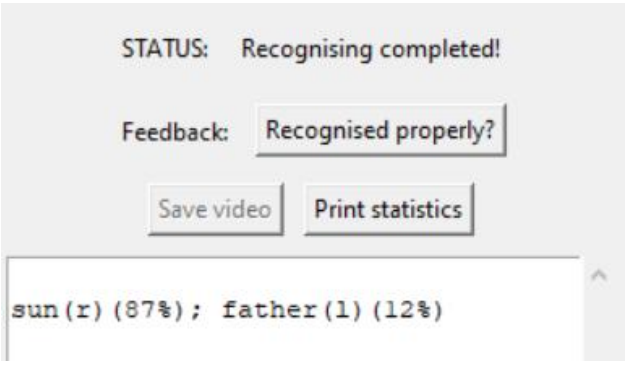
При натисканні кнопки «Recognise» відбувається запис відео (90 кадрів), після успішного запису статус змінюється як це показано на рисунку 3.17 а).

Далі, після проходження певного часу, необхідного для побудови ключових точок та розпізнання жесту, статус знову змінюється як це показано на рисунку 3.17 б). Крім цього у текстовому полі з’являється назва жесту, який класифікувала модель та активується кнопка зворотного зв’язку «Recognised properly?». При її натисканні користувачу надається можливість або продовжити користування програмою, або зберегти відео. При натисканні кнопки «Save video» записане відео зберігається у папку `temp` зі зміненою назвою, як це показано на рисунку 3.17 в). У користувача є можливість переглянути статистику розпізнавання, натиснувши кнопку «Print statistics» (рис. 3.17 г).

а)



б)



в)

г)

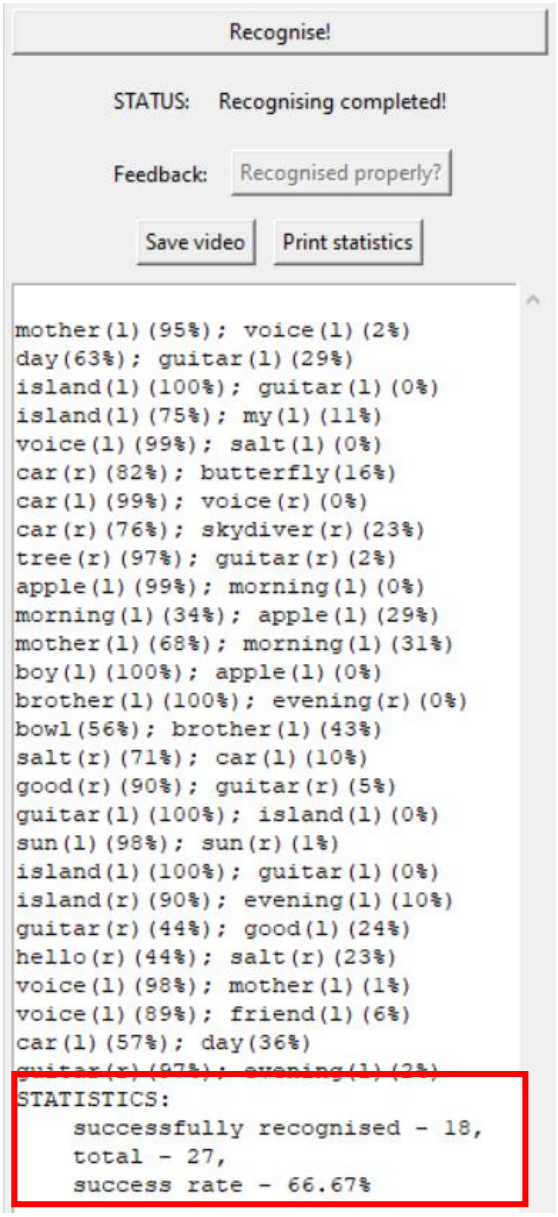


Рисунок 3.17. Демонстрація інтерфейсу користувача: а) зміна статусу після запису відео; б) вивід результату передбачення; в) коректність зберігання відео; г) вивід статистики розпізнавання

Демонстрація тестування розробленої системи розпізнавання жестів УЖМ двома різними виконавцями наведена у вигляді відеозаписів у репозиторії, доступному за посиланням у додатку А. Результати для виконавця 1 – 74.07%, для виконавця 2 – 70.83%.

ВИСНОВКИ

Першим результатом виконання дипломної роботи є створення власного набору даних у вигляді 2364 відеозаписів жестів УЖМ, загальною тривалістю 118 хвилин. Основним ресурсом прикладів жестів виступає онлайн-бібліотека жестів Spredthesign. До зазначеного набору входить 38 різних жестових слів, у процесі підготовки поділених на 66 категорій. Оскільки українська мова жестів є дворучною, більшість жестів може показуватися як правою, так і лівою рукою, для цих випадків були передбачені окремі категорії. Навчальний набір складається з 1980 відео (по 30 примірників на категорію). Попередня обробка відеозаписів для подальшої передачі на вхід нейронної мережі полягає у вилученні з кожного кадру опорних точок скелету людини. Цей процес відбувається за допомогою інструменту MediaPipe, який дозволяє виявляти та відслідковувати ключові точки рук, тіла і обличчя. На основі сукупності координат точок, для кожного навчального відео створюються відповідні файли у форматі numpy. Ці файли є базисом для навчання моделі нейронної мережі з LSTM архітектурою.

У роботі представлено процес розробки, створення та навчання 5 моделей нейронних мереж. Результати їх тестування на новому наборі відеозаписів, які зовсім не використовувалися для навчання наведено у розділі 3.5. Точність кращої з розроблених моделей (Модель 5) на тестовому наборі складає 85.79%. Архітектура моделі 5 складається з двох шарів LSTM на 128 нейронів з функцією активації `sigmoid`, аналогічного за розмірністю повнозв’язного шару з активацією `selu`, після якого слідує шар вилучення з коефіцієнтом 0.2 та вихідний шар.

Для неупередженого тестування створеної моделі машинного навчання був розроблений графічний інтерфейс користувача. До функціоналу останнього входить запис відео жесту і його подальша класифікація, обробка наданого користувачем зворотного зв’язку щодо правильності розпізнання

жесту. Інтерфейс передбачає повернення статистики на основі відповідей користувача протягом програмної сесії. При умові, що жест розпізнано правильно у користувача також існує можливість зберегти його, що може бути корисно для майбутнього розширення набору навчальних даних.

Всі створені файли координат ключових точок, розроблені моделі нейронних мереж, процес їх навчання та тестування, файли ваг, а також код графічного інтерфейсу користувача було задокументовано та опубліковано на ресурсі github.com, що доступний за посиланням у додатку А. Створені навчальні та тестові набори відеофайлів доступні за вимогою і можуть бути використані в дослідницьких цілях для інших робіт. Відео-демонстрація процесу тестувань системи через інтерфейс, проведена двома різними виконавцями, також доступна у вищезгаданому репозиторії.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Сайт Всесвітньої організації охорони здоров'я «Глухота і втрата слуху» [Електронний ресурс]. – (дата звернення: 09.02.2023) – Режим доступу: https://www.who.int/news-room/fact-sheets/detail/deafness-and-hearing-loss
2. K. Cormier, A. Schembri, M. E. Tyrone Nativisation of fingerspelling in ASL and BANZSL [Електронний ресурс] – (дата звернення: 09.02.2023) // Sign Language & Linguistics. – 2008. – №11. – С. 3-44. – Режим доступу: https://www.researchgate.net/publication/233582184_One_hand_or_two_Nativisation_of_fingerspelling_in_ASL_and_BANZSL
3. S. Mitra, T. Acharya. Gesture recognition: A survey. // IEEE Trans. Syst. Man Cybern. Part C. – 2007. – №37. – С.311-324.
4. J. Pansare, S. Gawande, M. Ingle. Real-Time Static Hand Gesture Recognition for American Sign Language (ASL) in Complex Background. // Journal of Signal and Information Processing. – Vol. 3. – № 3. – 2012. – С. 364-367.
5. Q. Chen, N. Georganas, E.Petriu. Hand Gesture Recognition Using Haar-Like Features and a Stochastic Context-Free Grammar. // IEEE Transactions on Instrumentation and Measurement. – Vol. 57. – № 8. – 2008. – С. 1562 - 1571.
6. Prof. P. Patil, R. Bhagwat, P. Padale, Y. Shah, H. Surwade. Sign Language Recognition System. // International Journal for Research in Applied Science & Engineering Technology (IJRASET). – 2022. – С. 1772-1776.
7. Zhihao Zhou, Kyle Chen, Xiaoshi Li, Songlin Zhang. Sign-to-speech translation using machine-learning-assisted stretchable sensor arrays. // Nature Electronics. – №3. – 2020. – С. 571-578.

8. L. Lamberti, F. Camastra. Real-time hand gesture recognition using a color glove. // In Proceedings of the International Conference on Image Analysis and Processing. – 2011. – С. 365-373.
9. K. Biswas, S. Basu. Gesture recognition using Microsoft Kinect. // The 5th International Conference on Automation, Robotics and Applications. – 2011. – С. 06-08.
10. S. Desai, A. Desai. Human Computer Interaction through hand gestures for home automation using Microsoft Kinect. // In Proc. of the International Conference on Communication and Networks. – 2017. – С. 19-29.
11. Xiujuan Chai, Guang Li, Yushun Lin, Zhihao Xu, Yili Tang, Xilin Chen. Sign Language Recognition and Translation with Kinect. // Microsoft Research Asia. – 2014.
12. Cao Dong, Ming C. Leu. American Sign Language alphabet recognition using Microsoft Kinect. // IEEE Conference on Computer Vision and Pattern Recognition Workshops. – 2015.
13. Youngmin Kim, Minji Kwak, Dain Lee, Yeongeun Kim, Hyeongboo Baek Keypoint based Sign Language Translation without Glosses [Електронний ресурс] – 2022 – (дата звернення: 16.04.2023) – Режим доступу: https://paperswithcode.com/paper/keypoint-based-sign-language-translation
14. Офіційний сайт некомерційної організації Project Jupyter. [Електронний ресурс]. – (дата звернення: 11.03.2023) – Режим доступу: https://jupyter.org/
15. Офіційний сайт-блокнот Google Collaboratory. [Електронний ресурс]. – (дата звернення: 11.03.2023) – Режим доступу: https://colab.research.google.com/?hl=ru.#scrollTo=GJBs_flRovLc
16. Сайт Google MediaPipe. [Електронний ресурс]. – (дата звернення: 11.03.2023) – Режим доступу: https://google.github.io/mediapipe/
17. Набір даних WLASL (World Level American Sign Language) Video. [Електронний ресурс]. – (дата звернення: 15.04.2023) – Режим доступу: https://www.kaggle.com/datasets/risangbaskoro/wlasl-processed

18. Chinese Sign Language Recognition Dataset, University of Science and Technology of China. [Електронний ресурс]. – (дата звернення: 15.04.2023) – Режим доступу: http://home.ustc.edu.cn/~pjh/openresources/cslr-dataset-2015/index.html
19. Набір даних «German Sign Language (DGS)». [Електронний ресурс]. – (дата звернення: 16.04.2023) – Режим доступу: https://www.kaggle.com/datasets/moritzkronberger/german-sign-language
20. Сайт Житомирської обласної універсальної наукової бібліотеки ім. Олега Ольжича, проект «Жестова мова online». [Електронний ресурс]. – (дата звернення: 10.03.2023) – Режим доступу: https://www.lib.zt.ua/ua/newsua/node/9156
21. Благодійний фонд імені Сергія Горового, курс лекцій з базових навичок жестової мови. [Електронний ресурс]. – (дата звернення: 06.02.2023) – Режим доступу: https://www.youtube.com/watch?v=8DHyf1nuMMc
22. Сайт онлайн-словника жестів «Spreadthesign». [Електронний ресурс]. – (дата звернення: 06.02.2023) – Режим доступу: http://www.spreadthesign.com/uk.ua/search/
23. Інформаційна сторінка некомерційної організації Європейський центр жестових мов. [Електронний ресурс]. – (дата звернення: 15.04.2023) – Режим доступу: https://www.signlanguage.eu/en/
24. Онлайн-словник жестів УЖМ Spreadthesign, жест «яблуко». [Електронний ресурс]. – (дата звернення: 16.04.2023) – Режим доступу: https://media.spreadthesign.com/video/mp4/31/119468.mp4
25. Онлайн-словник жестів УЖМ Spreadthesign, жест «фрукти». [Електронний ресурс]. – (дата звернення: 16.04.2023) – Режим доступу: https://media.spreadthesign.com/video/mp4/31/119659.mp4
26. Онлайн-словник жестів УЖМ Spreadthesign, жест «батько». [Електронний ресурс]. – (дата звернення: 16.04.2023) –Режим доступу: https://media.spreadthesign.com/video/mp4/31/390845.mp4

27. Онлайн-словник жестів УЖМ Spreadthesign, жест «дякую». [Електронний ресурс]. – (дата звернення: 16.04.2023) – Режим доступу: https://media.spreadthesign.com/video/mp4/31/119451.mp4
28. Онлайн-словник жестів УЖМ Spreadthesign, жест «я». [Електронний ресурс]. – (дата звернення: 16.04.2023) – Режим доступу: https://media.spreadthesign.com/video/mp4/31/109853.mp4
29. Онлайн-словник жестів УЖМ Spreadthesign, жест «мій». [Електронний ресурс]. – (дата звернення: 16.04.2023) – Режим доступу: https://media.spreadthesign.com/video/mp4/31/329420.mp4

ДОДАТКИ

Додаток А

Репозиторій з усіма створеними наробками:
https://github.com/mataamegafuru/sign_language_translation

- sign_recognition-Copy1.ipynb – файл з кодом функцій для запису відео, їх попередньою обробкою, розробленими моделями, процесом їх навчання та тестування, а також код графічного користувацького інтерфейсу.
- MODEL1.h5, MODEL2.h5, MODEL3.h5, MODEL4.h5, MODEL5.h5 – файли ваг для всіх розроблених моделей.
- Каталог keypoints з файлами координат ключових точок для всіх відеозаписів.
- Testing_Ira.mp4 – процес тестування системи виконавцем 1.
- Testing_Anton.mp4 – процес тестування системи виконавцем 2.