

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА»
ФАКУЛЬТЕТ ІНФОРМАЦІЙНО-КОМП'ЮТЕРНИХ ТЕХНОЛОГІЙ КАФЕДРА
ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

ПОЯСНЮВАЛЬНА ЗАПИСКА

до кваліфікаційної роботи освітнього ступеня «магістр»
за спеціальністю 121 «Інженерія програмного забезпечення»
(освітня програма «Інженерія програмного забезпечення»)
на тему:

**«Створення застосунку для перекладання мови
жестів на українську в реальному часі з
допомогою нейронних мереж»**

Виконав студент групи ІПЗм-22-2
ВАЩЕНКО Костянтин Ярославович

Керівник роботи:
ЄФРЕМОВ Юрій Миколайович

Рецензент:
ЧИЖМОТРА Олексій Володимирович

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА»
ФАКУЛЬТЕТ ІНФОРМАЦІЙНО-КОМП'ЮТЕРНИХ ТЕХНОЛОГІЙ КАФЕДРА
ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

«ЗАТВЕРДЖУЮ»

зав. кафедри інженерії
програмного забезпечення

Тетяна ВАКАЛЮК

«18» вересня 2023 р.

ЗАВДАННЯ
на кваліфікаційну роботу

Здобувач вищої освіти: ВАЩЕНКО Костянтин Ярославович

Керівник роботи: ЄФРЕМОВ Юрій Миколайович

Тема роботи: **«Створення застосунку для перекладання мови жестів на українську в реальному часі з допомогою нейронних мереж», затверджена Наказом закладу вищої освіти від «18» вересня 2023 р. № 479/с Вихідні дані для роботи: аналіз та пошук аномальних значень у роботі систем автоматизованого керування та впровадження нейронних мереж в процес пошуку аномалій; способи автоматизації процесу пошуку аномальних значень.**

Консультанти з магістерської кваліфікаційної роботи із зазначенням розділів, що їх стосуються:

Розділ	Консультант	Завдання видав	Завдання прийняв
1	Єфремов Ю. М	22.09.2023р.	22.09.2023р.
2	Єфремов Ю. М	22.09.2023р.	22.09.2023р.
3	Єфремов Ю. М	22.09.2023р.	22.09.2023р.

РЕФЕРАТ

Кваліфікаційна робота складається з розробку застосунку, який використовує нейронні мережі для перекладу мови жестів на українську мову в реальному часі. Пояснювальна записка міститиме 70 сторінок, у тому числі 72 ілюстрації та 13 таблиць, які демонструють архітектуру нейронної мережі, алгоритми обробки мови жестів та результати тестування.

Дипломний проект присвячений розробці інноваційного рішення, яке дозволяє людям з вадами слуху спілкуватися зі світом більш ефективно. Метою проекту є створення надійного, ефективного та точного механізму для перекладу мови жестів.

КЛЮЧОВІ СЛОВА: НЕЙРОННІ МЕРЕЖІ, КОМП'ЮТЕРНИЙ ЗІР, ОБРОБКА МОВИ ЖЕСТІВ, ПЕРЕКЛАД МОВИ ЖЕСТІВ, ІНКЛЮЗИВНІ ТЕХНОЛОГІЇ, МАШИННЕ НАВЧАННЯ, ШТУЧНИЙ ІНТЕЛЕКТ.

ABSTRACT

The qualification work consists of the development of an application that uses neural networks to translate sign language into Ukrainian in real time. The explanatory note will contain 70 pages, including 72 illustrations and 13 tables that demonstrate neural network architecture, sign language processing algorithms and test results.

The diploma project is devoted to the development of an innovative solution that allows people with hearing impairments to communicate with the world more effectively. The goal of the project is to create a reliable, efficient and accurate mechanism for sign language translation.

KEY WORDS: NEURAL NETWORKS, COMPUTER VISION, SIGN LANGUAGE PROCESSING, SIGN LANGUAGE TRANSLATION, INCLUSIVE TECHNOLOGIES, MACHINE LEARNING, ARTIFICIAL INTELLIGENCE.

Зміст

ВСТУП	7
РОЗДІЛ 1. АНАЛІЗ ІСНУЮЧИХ МЕТОДІВ РОЗПІЗНАВАННЯ ЖЕСТОВОЇ МОВИ.....	9
1.1. Аналіз сучасних систем розпізнавання жестової мови	9
1.2. Класифікація та аналіз існуючих методів розпізнавання жестової мови	11
1.3. Особливості азбуки та мови жестів	18
1.4. Аналіз літературних джерел щодо апробації результатів розпізнавання жестової мови.....	19
1.5. Постановка задачі дослідження.....	25
Висновки до розділу 1	26
РОЗДІЛ 2. ПРОЄКТУВАННЯ ДОДАТКУ	27
2.1. Інструменти реалізації.....	27
2.2. Створення власного набору даних.....	28
2.3. Проєктування згорткової нейронної мережі	29
2.4. Оптимізатор	31
2.5. Алгоритм виводу літери на екран	35
Висновки до розділу 2	37
РОЗДІЛ 3. ОПИС ПРОГРАМИ.....	39
3.1. Головний екран.....	39
3.2. Сервіс завантаження фотографії	43
3.3. Сервіс обробка фотографій.....	45
Висновки до розділу 3	49
ВИСНОВКИ	50
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ	51
ДОДАТКИ	54

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

ШНМ – штучна нейрона мережа

ЗНМ – згорткова нейрона мережа

КЗ – комп’ютерний зір

ДА – дактильний алфавіт

DPCNN – Deep Pyramid Convolutional Neural Networks (глибока піраміда згорткової нейронної мережі)

HCI – Human-Computer Interaction (взаємодія людини з комп’ютером)

NLP – Natural Language Processing (опрацювання природної мови)

ВСТУП

Актуальність теми. Спілкування осіб із порушеннями слуху є ключовою проблематикою сучасного суспільства. Відповідно до даних, наданих Всесвітньою організацією охорони здоров'я, приблизно 390 мільйонів людей у світі зазнають труднощів із слухом. В контексті України, статистика Українського товариства глухих, оновлена у 2016 році, вказує на наявність понад 60 000 громадян з порушеннями слуху та мовлення. Також існують неофіційні звіти, які підкреслюють, що кількість людей із такими проблемами в країні може перевищувати 2 мільйони. Що стосується української жестової мови, то, згідно з даними Всесвітньої організації глухих станом на 2022 рік, вона є основним засобом комунікації для приблизно 190 000 осіб в Україні.

Водночас, стикаємося з проблемою обмеженого доступу до навчальних матеріалів для освоєння української жестової мови. На даний момент, веб-сайт `languagebulletin` виступає як один із небагатьох ресурсів, але він не забезпечує повної та достовірної інформації в лінгвістичному контексті. Ця обмеженість суттєво ускладнює процес навчання та використання жестової мови для спілкування, особливо в Україні, де спільнота глухих та людей із вадами слуху потребує більш ефективних і доступних навчальних ресурсів.

Метою кваліфікаційної роботи є створення передового програмного забезпечення для розпізнавання української жестової мови.

Для реалізації вказаної мети нам необхідно виконати наступні **завдання**:

- проаналізувати існуючі методи розпізнавання жестової мови, зокрема сучасні системи, класифікацію та аналіз існуючих методів, особливості азбуки та мови жестів, аналіз літератури;
- спроектувати додаток, а саме інструменти реалізації, власний набір даних, згорткову нейронну мережу оптимізатор, алгоритми розпізнавання жестів та виводу літер на екран;
- описати програму (графічний інтерфейс);

Об’єктом дослідження у цій роботі є нейронна мережа, яка призначена для розпізнавання жестової мови. Основний фокус полягає у розробці та вдосконаленні цієї системи з метою створення високоефективного інструменту, здатного не лише розуміти, але й перекладати жестову мову. Таке дослідження є важливим кроком на шляху до реалізації більш доступного та інклюзивного цифрового простору. Ця ініціатива відкриває нові можливості для підвищення рівня залучення та спілкування осіб із порушеннями слуху в цифровому середовищі.

Предметом дослідження є підходи та алгоритми, які застосовуються для проектування, тренування та оцінки продуктивності нейронних мереж у контексті ідентифікації жестової мови.

Практичне значення роботи полягає у тому, що систематизований матеріал може бути використаний здобувачами вищої освіти технічних спеціальностей для написання власних досліджень, також дані напрацювання можуть бути використані для подальших розробок у сфері розпізнавання жестової мови.

Апробація роботи. Результати досліджень були опубліковані у вигляді тез:

Ващенко К. Я. Створення засобу розуміння мови жестів за допомогою нейронних мереж. *Теоретичні та практичні дослідження молодих вчених* : матеріали XVII Міжнародної науково-практичної конференції магістрантів та аспірантів, м. Харків, 28-30 листопада 2023 р. / НТУ «Харківський політехнічний інститут», 2023. С. 161 – 162.

Ващенко К. Я. Створення засобу розуміння мови жестів за допомогою нейронних мереж. *Сучасні комп’ютерні та інформаційні системи і технології* : матеріали III Всеукраїнської науково-практичної інтернет-конференції, м. Запоріжжя, 12-19 грудня 2022 р. / Таврійський державний агротехнологічний університет імені Дмитра Моторного, 2022. С. 412 – 414.

РОЗДІЛ 1. АНАЛІЗ ІСНУЮЧИХ МЕТОДІВ РОЗПІЗНАВАННЯ ЖЕСТОВОЇ МОВИ

1.1. Аналіз сучасних систем розпізнавання жестової мови

Жести включають у себе різноманітні рухи тіла, що є формою невербального спілкування, дозволяючи передавати інформацію без використання слів. До жестів належать рухи пальців, рук, голови, плечей, а також вирази обличчя. Жести можуть використовуватись для передачі окремих повідомлень, доповнення вербального спілкування, а також для вираження емоцій та почуттів. Вони поділяються на статичні, які сприймаються миттєво, та динамічні, що сприймаються протягом певного часового періоду, і мають своє значення у системах рукописного алфавіту та в безконтактних інтерфейсах людино-машинної взаємодії.

Система розпізнавання рукописних жестів представляє собою комплекс комп'ютерних технологій та математичних алгоритмів, що забезпечують розпізнавання конкретної групи жестів рук.

Застосування систем розпізнавання жестів охоплює такі сфери:

- Управління комп'ютером: конфігурація руки користувача відповідає певній команді комп'ютера. Коли людина робить жест, система ідентифікує рух і передає відповідну команду на комп'ютер.
- Взаємодія з курсором миші: положення долоні визначає положення курсора на екрані, а рух руки управляє рухом курсора. Різні конфігурації руки відповідають командам кліку миші.
- Управління побутовими приладами: розпізнані положення кінцівок пальців можуть бути використані для керування комп'ютером або побутовими приладами, використовуючи декілька пальців одночасно.

Створення інтуїтивних людино-машинних інтерфейсів для людей із порушеннями слуху та мовлення включає наступні аспекти:

- Використання систем розпізнавання жестів для текстового введення: це може бути значно простіше і зручніше для глухонімих осіб порівняно з традиційним введенням тексту через клавіатуру. Жести рук, які є природним способом спілкування для цієї групи людей, можуть бути перетворені на текст за допомогою спеціалізованого програмного забезпечення.
- Реалізація системи для онлайн-комунікації в умовах обмеженої пропускної спроможності мережі: замість відеозв'язку, можна передавати дані про жести рук, розпізнані на одному кінці мережі, а на іншому - візуалізувати їх як анімацію. Такий підхід знижує вимоги до швидкості мережі, дозволяючи ефективно передавати інформацію за допомогою жестів.

Управління тривимірними моделями: замість традиційного використання комп'ютерної миші, яка не завжди є зручною для таких задач, можна розробити систему людино-комп'ютерної взаємодії (НСІ), що використовує тривимірні координати руки та кінчиків пальців для управління моделями в 3D просторі. Це створює більш інтуїтивний і зручний спосіб взаємодії з тривимірними об'єктами.

Інтеграція систем розпізнавання жестів з технологіями віртуальної реальності, як наприклад із стереоскопічними окулярами, може створити новітні засоби взаємодії в віртуальному середовищі. Таке поєднання дозволить користувачам "взаємодіяти" з віртуальними об'єктами більш натуралістично та інтуїтивно.

1.2 Класифікація та аналіз існуючих методів розпізнавання жестової мови

Особливістю методів, які базуються на аналізі зовнішніх ознак жесту, є те, що вони обмежуються інтерпретацією лише зовнішнього вигляду цільового об'єкта. При розпізнаванні даного об'єкта не зберігається інформація про його фізичні характеристики. Нижче буде проаналізовано вже існуючі дослідження та методи, які спрямовані на розпізнавання жестів руки людини на основі аналізу зовнішніх ознаках[5].

Ці методи відрізняються від інших технік розпізнавання жестів тим, що вони фокусуються на аналізі зовнішнього вигляду руки та її рухів без необхідності зберігати детальну інформацію про фізичні властивості руки. Основні етапи такого аналізу включають в себе визначення форми та позиції руки, а також інших зовнішніх параметрів, таких як рухи пальців, напрямки жесту тощо.

Ці методи розпізнавання можуть бути корисними у великій кількості застосувань, включаючи розпізнавання жестів для управління електронними пристроями, віртуальною реальністю, медичними додатками та багатьма іншими областями. Дослідження у цьому напрямку є актуальними і можуть призвести до подальшого розвитку технологій розпізнавання жестів руки на основі аналізу зовнішніх ознак.

- Розпізнавання позиції та орієнтації

Позначимо через $M_{0,0}$, $M_{1,0}$, $M_{0,1}$, $M_{2,0}$, $M_{1,1}$, $M_{0,2}$ моменти нульового, першого і другого порядку зображення з функцією інтенсивності $f(x,y)$:

$$M_{0,0} = \sum_x \sum_y f(x, y), \quad (1.1)$$

$$M_{0,1} = \sum_x \sum_y y \times f(x, y), M_{1,0} = \sum_x \sum_y x \times f(x, y), \quad (1.2)$$

$$M_{1,1} = \sum \sum x \times y \times f(x, y), M_{0,2} = \sum \sum y^2 \times f(x, y), M_{2,0} = \sum_x \sum_y x^2 \times f(x, y). \quad (1.3)$$

Цей підхід використовує обмеження моментів зображення, щоб визначити певні жести руки і перетворити їх у команди для керування роботом. Він може бути корисним у створенні інтерфейсів для взаємодії людини з електронними пристроями, зокрема з іграшковими роботами. Результати таких досліджень можуть знайти застосування у різних галузях, де важлива гармонійна взаємодія між людиною і технологією.

Деякі обмеження моментів зображення можуть бути використані для розпізнавання простих жестів рук і створення на їх основі інструментальних засобів НСІ. Наприклад, у дослідженні[13] розглядається застосунок, який дозволяє керувати іграшковим роботом за допомогою рухів руки, де орієнтація руки визначає напрямки руху робота (рис. 1.1).

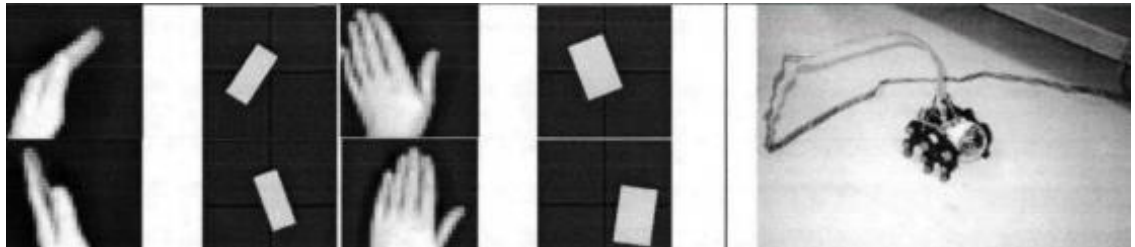


Рисунок 1.1 – Управління іграшковим роботом простими жестами руки: (а) – прямокутник показує розпізнаний напрямок, позицію і розмір зображення руки; (б) – іграшковий робот

Для розпізнавання позиції (x_c, y_c) і орієнтації руки Θ на бінарному зображенні використовується обчислення моментів зображення, при умові, що фон зображення однорідний і рука є домінуючим об'єктом на зображенні:

Цей підхід передбачає визначення розташування центра мас об'єкта (x_c , y_c) та визначення орієнтації об'єкта Θ шляхом обчислення моментів зображення. Метод передбачає, що фон зображення має однорідний колір чи текстуру, і що рука є головним об'єктом, який виділяється на зображенні.

Цей підхід може бути корисним у визначенні положення та орієнтації об'єкта на зображенні і знайти застосування у різних областях, включаючи комп'ютерне зорове сприймання, обробку зображень та автоматизоване визначення об'єктів на зображеннях.

- Розпізнавання рухів на основі аналізу різниць зображень

У практиці не завжди можливо забезпечити умови однорідного фону. Для таких ситуацій зазвичай використовується метод, який базується на аналізі центру маси різниць зображень, відомий як "Motion Energy Image" (MEI) руки в кадрах відеоряду[14].

Використання різниць між кадрами відеоряду (MEI) дозволяє в реальному часі аналізувати рух об'єкта на відеоряді при умові, що фон може бути стабільним, але не обов'язково однорідним. Цей метод дозволяє відокремити руку або інший об'єкт від фону і виділити його рух, що може бути корисним у визначенні жестів рук чи об'єктів, що рухаються на відеоряді.

Такий підхід є ефективним у випадках, коли фон змінюється або неоднорідний, і може бути застосований у різних областях, включаючи відеоспостереження, віртуальну реальність та інші сфери, де важливий аналіз руху об'єктів на відеозаписі.

На практиці ця технологія та її вдосконалені варіанти, такі як "Motion History Image" (MHI), знайшли широке застосування в різних областях. Наприклад, їх використовують у таких застосунках, як інтерактивний віртуальний тренер з аеробіки[15-16] і інтерактивна кімната для розповіді історій [17].

В інтерактивних віртуальних тренерах з аеробіки (згаданих в дослідженнях

[15-16]), ця технологія дозволяє відстежувати та аналізувати рухи користувача під час занять фізичними вправами, надаючи зворотний зв'язок та рекомендації для покращення техніки.

У інтерактивних кімнатах для розповіді історій (згаданих у дослідженні [17]), ця технологія може використовуватися для взаємодії з віртуальним оточенням та керування подіями у реальному часі на основі рухів користувача, що дозволяє створювати захоплюючі інтерактивні історії.

Отже, технологія розпізнавання руху та відстеження рухів, яка базується на МНІ та подібних методах, знаходить широке застосування у віртуальній реальності, інтерактивних іграх та інших сферах, де важлива взаємодія людини з комп'ютером чи віртуальним середовищем на основі рухів користувача.

- Розпізнавання конфігурації на основі аналізу гістограм комп'ютерній

У більшості програмних засобів комп'ютерного зору, крім визначення зображення, які менше чутливі до змін освітлення в навколишньому середовищі, що спостерігається.

Припустимо, що $f(x, y)$ представляє інтенсивність бінарного або напівтонового зображення руки. Локальний напрямок $\Theta(x, y)$ у точці (x, y) визначається таким чином[20]:

$$\theta(x, y) = \arctan^*(f(x, y) - f(x - 1, y), f(x, y) - f(x, y - 1)). \quad (1.4)$$

Цей метод дозволяє отримати інформацію про спрямованість пікселів на зображенні, що може бути використана для аналізу та визначення конфігурації руки. Ця інформація може бути корисною для подальших досліджень у галузі комп'ютерного зору та розпізнавання жестів.

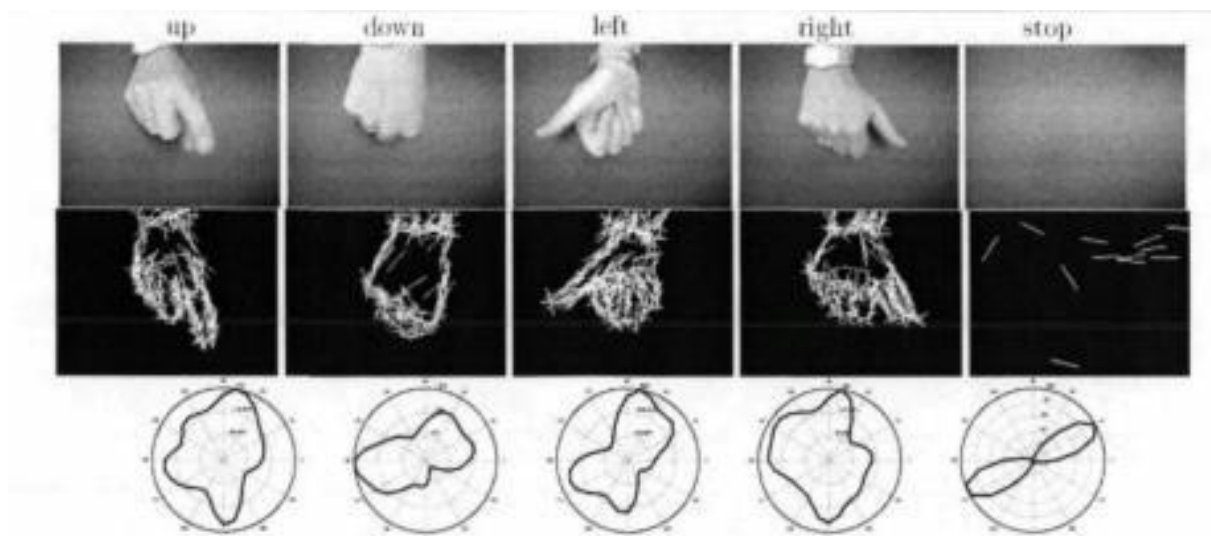


Рисунок 1.3 – Жести руки, карти та гістограми напрямків

У програмному засобі для управління анімаційним краном[20] використовуються наведені жести та технологія розпізнавання конфігурацій руки на основі гістограм напрямків. В роботі зазначається, що застосунок працює в реальному часі, він не відчуває незначні зміни розміру руки, але фіксує змінює напрямку руки. Крім цього, програмне забезпечення вимагає навчання для кожного оператора.

Ця технологія дозволяє розпізнавати конфігурацію руки в режимі реального часу за наступних умов:

- Рука є домінуючим об'єктом на зображенні.
- Фон зображення однорідний.
- Жести обрані таким чином, що гістограми напрямків мають важливі відмінності один від одного.

Цей підхід відмінно підходить для управління анімаційним краном та подібними застосунками, де важливо реагувати на рухи руки оператора в режимі реального часу, і де фон та жести руки відповідають встановленим умовам для ефективного розпізнавання.

- Розпізнавання конфігурації і позиції із застосуванням кольорових рукавичок

Для розпізнавання жестів руки часто використовують кольорові рукавички [17-21]. Метод, запропонований в роботі[17], дозволяє в реальному часі за допомогою однієї відеокамери розпізнавати конфігурацію руки і відстежувати рухи долоні в тривимірному просторі (рис. 1.4). Рукавичка сконструйована з двадцяти сегментів, десяти різних кольорів використовується в цій системі. Використання обмеженої кількості кольорів дозволяє розпізнати колір обраної точки на зображенні рукавички при зміні освітлення, і спеціальне розташування кольорових сегментів запобігає отриманню ідентичних зображень при різних конфігураціях руки.

Такий підхід дозволяє забезпечити надійне відстеження руки та розпізнавання жестів у реальному часі з використанням відеокамери і кольорової рукавички. Ця технологія може бути корисною в багатьох застосунках, включаючи віртуальну реальність, контроль жестами та інші області, де важлива точність та швидкість розпізнавання рухів руки.

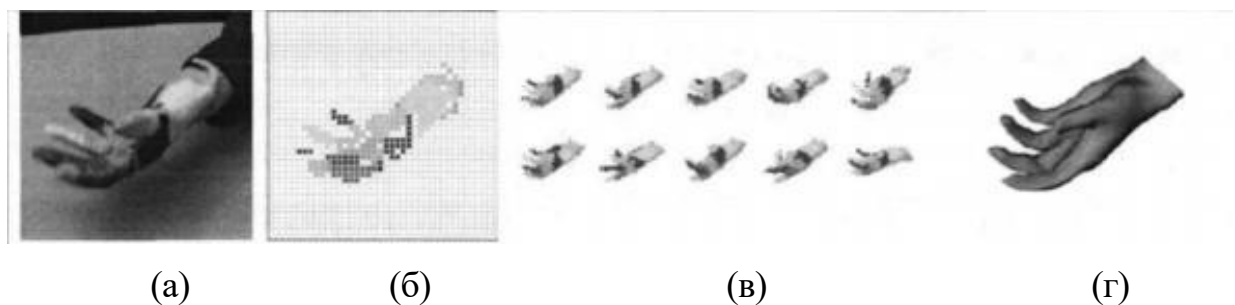


Рисунок 1.4 – Результат застосування кольорових рукавичок у завданні розпізнавання жестів руки:

(а) – вхідне зображення руки; (б) – растрове зображення руки розміром 40×40 ; (в) – схожі зображення руки в базі даних; (г) – знайдена конфігурація

Для розпізнавання конфігурації руки застосовується наступний процес: спочатку, початкове зображення піддається згладжуванню алгоритмом

двостороннього розмиття (Bilateral Filter). Далі, фонові точки видаляються, і цей процес відбувається на основі класифікації за допомогою моделі гаусової суміші (Gaussian Mixture Models). Отримане зображення перетворюється в растровий формат меншого розміру (рис. 1.4 (б)). Потім отримані результати порівнюються з еталонними зображеннями, які зберігаються в базі даних і відповідають різним формам руки (рис. 1.4 (в)).

Відстань $d(r^i, r^j)$ між растровими зображеннями r^i та r^j оцінюється наступним чином: (тут вставити формулу або пояснення щодо визначення відстані).

Цей процес дозволяє розпізнавати конфігурацію руки та порівнювати її з еталонними зображеннями для подальшої класифікації.

$$d(r^i, r^j) = \tilde{d}(r^i, r^j) + \tilde{d}(r^i, r^j), \quad (1.5)$$

$$S_{xy} = \{(u, v) | r^i(x, y) = r^j(u, v)\}; \quad (1.6)$$

$$C_i = \{(x, y) | r^i(x, y) \neq \text{фон}\}. \quad (1.7)$$

Описана вище система знайшла своє застосування в задачах управління анімаційним персонажем.

- Розпізнавання конфігурації і позиції на основі аналізу контуру зображення руки

Для опису конфігурації руки часто використовують контур її зображення. Наприклад, одне з досліджень пропонує систему розпізнавання жестів ручної азбуки для глухонімих, засновану на аналізі контуру зображення руки. У цьому процесі спрощується завдання шляхом нерозглядання двох динамічних жестів, а

деякі жести замінюються новими для полегшення розпізнавання[3].

В кольоровому зображенні руки видаляються всі точки, які не відповідають кольору шкіри. Зображення потім перетворюється у бінарне і згладжується за допомогою фільтра Гауса. Рука виділяється як окремий компонент. Для визначення контурів руки застосовується оператор Собеля. Обчислюється прямокутник, що описує контур руки, центр якого встановлюється як початок координат.

Схожість жестів оцінюється за допомогою відстані Евкліда між векторами дескрипторів жестів. У даній роботі наводиться оцінка якості розпізнавання – 90%. Важливо зауважити, що якість розпізнавання при показі жестів різними людьми не вивчалася.

У роботі[21] розглядається завдання розпізнавання позиції кінчиків пальців руки на основі аналізу контуру руки в кольоровому зображенні. Першими кроками алгоритму, аналогічно до[20], є видалення фонових точок, згладжування та виділення зображення руки. Після отримання контуру руки, пальці руки виділяються шляхом аналізу локальних вигинів контуру. Цей метод був використаний для управління вебкамерою персонального комп'ютера.

Обидва згадані методи використовують аналіз контуру руки для розпізнавання жестів та позицій пальців і можуть мати різні застосування, включаючи системи контролю жестами та взаємодії з комп'ютерами.

1.3 Особливості азбуки та мови жестів

Сьогодні для вивчення будь-якої мови достатньо мати доступ до інтернету з будь-якого гаджету. Інтернет щодня поповнюється великою кількістю сайтів, що пропонують різноманітні методики та уроки для вивчення іноземних мов, включаючи відеоуроки.

Розробляються також мобільні додатки, які пропонують захоплюючий підхід до вивчення мов. Однак, існує унікальна мова, яка відрізняється від інших своїм невербальним способом комунікації - мова жестів. Жести мови жестів

включають рухи рук, пальців і вирази обличчя, де один жест може символізувати букву, слово або навіть емоцію.

Ручна азбука або дактильний алфавіт, який використовується в дактилології, дозволяє відтворювати орфографічну форму слова за допомогою пальців рук. Комунікація за допомогою дактиля вважається вербальною, оскільки кожна буква представлена певним положенням пальців.

Більшість символів у дактильному алфавіті схожі на їх друковані еквіваленти, що спрощує процес вивчення цієї азбуки. Володіння дактильною азбукою може сприяти спілкуванню між людиною з порушеннями слуху та тією, хто чує. Проте, ускладнення можуть виникати, коли співрозмовник не знає мову жестів. Вирішення цієї проблеми включає дослідження розробки автоматизованих систем сурдоперекладу та розробку більш природних людино-машинних інтерфейсів для глухих.

Ручна азбука включає тільки частину жестів, використовуваних у мові глухонімих, тому автоматичне розпізнавання цих жестів може бути ключем до створення більш інтуїтивних людино-машинних інтерфейсів. Це допоможе подолати обмеження у спілкуванні, з якими стикаються глухі люди у їхньому повсякденному житті.

Люди, які народилися глухими, часто віддають перевагу мові жестів замість письмового тексту, на відміну від тих, хто втратив слух внаслідок нещасного випадку або хвороби. Для них вираження думок та емоцій через жести є простішим та зрозумілішим способом спілкування, ніж читання чи введення тексту за допомогою клавіатури комп'ютера або телефону.

1.4 Аналіз літературних джерел щодо апробації результатів розпізнавання жестової мови

За інформацією Всесвітньої організації охорони здоров'я, у світі налічується близько 360 мільйонів осіб із порушеннями слуху та глухих. Попри стрімкий розвиток комунікаційних технологій, виклик спілкування з людьми, які

мають вади слуху, залишається важливим. Існує все більше досліджень, спрямованих на поліпшення комунікації для цієї групи людей, зокрема в сфері машинного перекладу з мови жестів[1].

Одна з напрямків у перекладі мови жестів - це використання робототехніки. Починаючи з 1977 року, коли була створена роботизована рука, машинний переклад з мови жестів став реальністю. Одним з ефективних методів є використання кібер-рукавичок, які дозволяють комп'ютеру відстежувати рухи людини та перетворювати їх на усну або письмову мову.

В 2002 році Райан Паттерсон у США розробив спрощену модель рукавички. Цей пристрій міг відстежувати жести дактильної азбуки глухих та передавати дані бездротово на портативний пристрій, який відображав текст на екрані. Рукавичка потребувала адаптації до руки конкретного користувача.

З плином часу, використання кібер-рукавичок з датчиками руху набуло значної популярності, що сприяло появі таких проектів як CyberGlove та VPL Data Glove.

CyberGlove - це тип рукавички, обладнаної 18-22 датчиками, що дозволяє захоплювати детальні знаки. Ця рукавичка підключається до основного комп'ютера за допомогою послідовного кабелю.

VPL Data Glove оснащений датчиками, які функціонують на основі оптоволокна і здатні вимірювати кути згину пальців. Це дозволяє точно відтворювати рухи рук, що є ключовим для точного перекладу жестової мови.



Рисунок 1.5 – Приклад застосування рукавички CyberGlove



Рисунок 1.6 – VPL Data Glove рукавичка

Одним з практичних застосувань цих технологій є проект SignAloud, реалізований студентами Університету Вашингтона. Вони використали пару рукавичок, обладнаних датчиками руху, які передають дані в комп'ютерну систему через Bluetooth. Завдяки цьому розробники вдало здійснили переклад американської мови жестів (ASL) на англійську мову[9].



Рисунок 1.7 – Приклад однієї з рукавичок проєкта SignAloud

Хоча ці технології мають високу точність, вони також ставлять певні обмеження для користувачів, зокрема, необхідність використовувати спеціалізовані технічні пристрої і неможливість враховувати нюанси артикуляції.

Як альтернатива, в 2012 році китайська команда у співпраці з Microsoft Research Asian розробила Kinect Sign Language Translator, що є прикладом застосування іншого підходу до перекладу мови жестів.

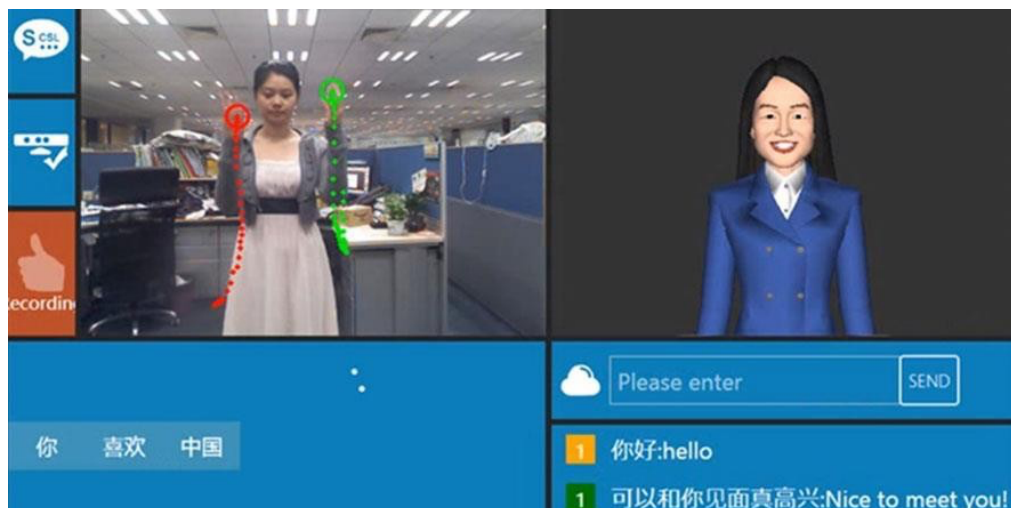


Рисунок 1.8 – Kinect Sign Language Translator у режимі спілкування

Перекладач має два основні режими роботи: режим перекладу та режим спілкування. Режим перекладу вміє конвертувати окремі жести у письмовий текст (слова) і навпаки. Він також здатний розпізнавати пози, форми рук та траєкторії руху, використовуючи технології машинного навчання, розпізнавання образів та комп'ютерного зору. Режим спілкування дозволяє перекладати повні фрази, при цьому розмову можна автоматично перекласти з використанням 3D-аватара. Жести зчитуються за допомогою периферійного пристрою Kinect, який призначений для ігрової консолі Xbox 360 від Microsoft.

SignAll - це автоматична система, розроблена угорською компанією Dolphio Technologies для перекладу американської мови жестів (ASL). Дослідження проводиться з 2015 року і базується на комп'ютерному зорі та обробці природної мови (NLP). Система використовує Kinect від Microsoft та вебкамери з датчиками глибини (один датчик глибини і три вебкамери), які підключаються до комп'ютера.

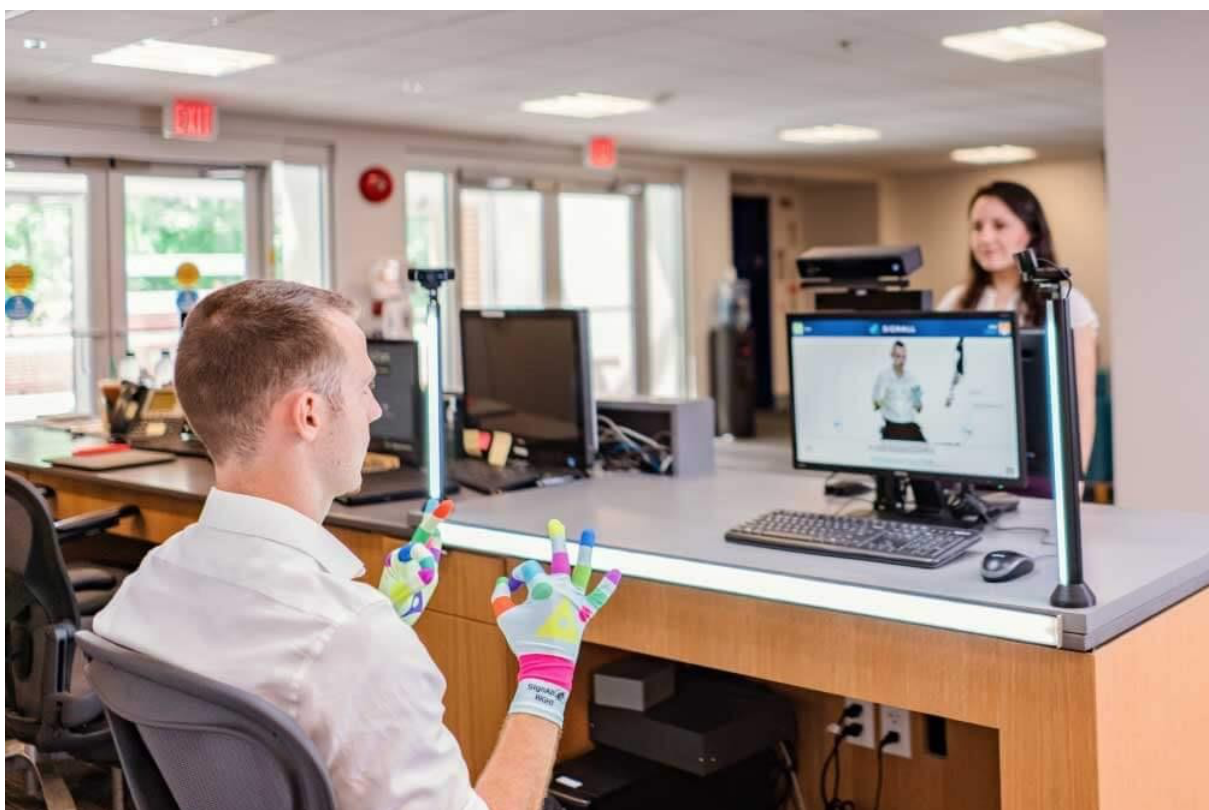


Рисунок 1.9 – Використання SignAll рукавичок

У цій технології комп'ютерного зору використовується для зчитування жестів та міміки людей з порушеннями слуху, а система обробки природної мови (NLP) перетворює ці дані в зрозумілі англійські фрази. Однак, основним недоліком є необхідність встановлення складної конструкції, що включає три камери, сенсор та комп'ютер, для зчитування мови жестів.

MotionSavvy - це пристрій, створений для перекладу американської мови жестів і навпаки. Розробка була здійснена у 2012 році групою з Рочестерського технологічного інституту (Національного технічного інституту для глухих). Для реалізації проекту команда використала планшетний комп'ютер з датчиком руху Leap Motion.



Рисунок 1.10 – Застосування MotionSavvy пристрою

Контролер Leap Motion 3D виявляє рухи рук, що виконуються користувачем над поверхнею планшета, і в результаті на екрані з'являється візуалізація жесту, який потім перекладається в текст або озвучується.

Sign2 Conversion System, розроблена для конвертації американської мови

жестів (ASL) на англійську мову (у письмовій та усній формах), наразі обмежена розпізнаванням жестів пальців у контрольованому середовищі за допомогою комп'ютерного зору. Однак, довгострокова мета полягає в перекладі повних фраз у природних умовах, з можливістю інтеграції з персональними цифровими помічниками (PDA) та смартфонами.

1.5 Постановка задачі дослідження

Важливістю в сучасному світі набуває створення систем розпізнавання жестової мови, що забезпечить легше розуміння людей з порушеннями слуху або автоматичне розпізнавання жестів, які сигналізують про потенційну небезпеку чи загрозу життю.

Дослідження фокусується на вивченні зображень рукописних жестів з метою розробки та дослідження методів розпізнавання жестової мови. Це сприятиме більш легкому розумінню осіб з вадами слуху та здатності автоматично розпізнавати спеціальні жести[4].

Цілі дослідження передбачають:

- Аналіз сучасних систем розпізнавання жестової мови.
- Класифікація та аналіз існуючих методів розпізнавання жестової мови.
- Вивчення особливостей азбуки та мови жестів.
- Аналіз літературних джерел з приводу апробації результатів розпізнавання жестової мови.
- Вивчення особливостей розробки методів розпізнавання жестової мови.
- Вибір інструментальних засобів для програмної реалізації методу розпізнавання.
- Програмна реалізація вибраного методу.
- Тестування та аналіз результатів розробленого застосунку.
- Визначення перспектив подальших досліджень у цій області.

Висновки до розділу 1

В даному розділі було проаналізовано існуючі методи розпізнавання жестової мови і було виявлено багато цікавих та ефективних підходів до розпізнавання рухів та жестів рук. Ці методи варіюються від аналізу контуру руки до використання кольорових рукавичок та комп'ютерного зору для виявлення рухів.

Один із способів розпізнавання полягає в аналізі контуру руки та визначенні певних дескрипторів, таких як позиція, орієнтація та форма руки. Ці дескриптори можуть бути використані для класифікації жестів та розпізнавання мови жестів.

Інший підхід використовує кольорові рукавички, які допомагають відстежувати рухи долоні в тривимірному просторі та визначати їх конфігурацію. Цей метод особливо ефективний у віртуальній реальності та системах контролю жестами.

Деякі методи використовують алгоритми обробки зображень та машинного навчання для аналізу контуру руки та виявлення жестів. Вони можуть вимагати попереднього навчання та створення бази даних для розпізнавання.

РОЗДІЛ 2. ПРОЄКТУВАННЯ ДОДАТКУ

2.1. Інструменти реалізації

Python - це інтерпретована мова програмування високого рівня, призначена для різноманітних завдань, з особливостями строгої динамічної типізації та автоматизованого управління пам'яттю. Ця мова підтримує ряд парадигм програмування: імперативне, процедурне, структурне, об'єктно-орієнтоване, метапрограмування та функціональне. Вона дозволяє вирішувати завдання загального програмування завдяки динамічній типізації. Аспектно-орієнтоване програмування частково реалізоване через декоратори і може бути повніше підтримане за допомогою додаткових фреймворків. Техніки контрактного і логічного програмування можна втілити з використанням бібліотек або розширень.

Щодо архітектури, Python має динамічну типізацію, автоматизоване управління пам'яттю, повну інтроспекцію, систему обробки виключень та підтримку багатопоточних обчислень із глобальним блокуванням інтерпретатора, а також високорівневі структури даних. Вона також підтримує структурування програм у модулі та пакети.

Стандартна бібліотека Python включає широкий набір функцій, від текстової обробки до інструментів для створення мережових застосунків.

Для розробки додатку було обрано Python, завдяки його простому синтаксису та широкому спектру доступних модулів. Проект розроблявся у MacOS з використанням Python версії 3.9.5.

Для написання та виконання коду було обрано інтегроване середовище розробки Vscode, яке забезпечує аналіз коду, графічний відлагоджувач, інтегровані засоби для запуску юніт-тестів, інтеграцію із системами контролю версій, а також підтримку веб-розробки з React Native Expo та Xeroxu.

Додатково у проекті використовувалися такі бібліотеки:

- PyTorch – бібліотека машинного та глибинного навчання, застосовувана в задачах комп'ютерного зору та обробки природньої мови.
- OpenCV – бібліотека алгоритмів комп'ютерного зору, обробки зображень та чисельних алгоритмів загального призначення з відкритим кодом.
- NumPy – бібліотека для роботи з матрицями та багатовимірними масивами, містить великий набір математичних функцій.
- Matplotlib – бібліотека для візуалізації даних та побудови графіків. Pandas – бібліотека для обробки та аналізу даних.
- PyQt – бібліотека для створення графічного інтерфейсу користувача.

2.2 Створення власного набору даних

Кількість літер в українському дактильному алфавіті відповідає кількості букв в стандартному українському алфавіті. Алфавіт містить як статичні, так і динамічні жести. Оскільки датасет буде складатися зі статичних зображень, деякі динамічні жести, які у статичному вигляді співпадають із іншими статичними жестами, будуть виключені з набору. В результаті, датасет включатиме 29 жестів, причому літери «г», «ї», «й», «щ» будуть замінені на візуально схожі «г», «і», «и», «ш» відповідно.

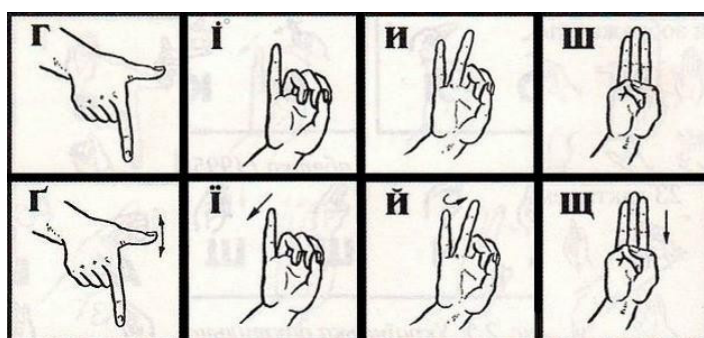


Рис. 2.1. Порівняння статичних та динамічних жестів

Таким чином, розроблюваний датасет буде містити 29 класів, представляючи різні жести, та один додатковий клас, який представлятиме фон. В кожному з цих класів буде знаходитися по 300 фотографій, загальна кількість

зображень складе 8700.

З метою уніфікації, всі зображення в наборі будуть мати однакові розміри, 60x60 пікселів. Для спрощення та оптимізації процесу обробки, кольорові зображення будуть перетворені у відтінки сірого. Це дозволить обмежити обробку одним кольоровим каналом, замість стандартних трьох.

2.3 Проектування згорткової нейронної мережі

- Архітектура згорткової нейронної мережі

Архітектура запропонованої нейронної мережі включає в себе тривимірну структуру, яка складається з трьох згорткових шарів, трьох агрегуювальних (пулінгових) шарів, та двох повнозв'язних шарів. Згорткові та пулінгові шари чергуються, створюючи ефективну послідовність для обробки вхідних даних.

У кожному згортковому шарі налаштовуються певні параметри: кількість вхідних та вихідних каналів. Ці параметри визначають, як інформація перетворюється в процесі пропускання через шар. Перший згортковий шар розрахований на один вхідний канал, що відповідає одному кольоровому каналу в зображенні, та вісім вихідних каналів. Другий згортковий шар приймає вісім вхідних каналів, відповідаючи вихідним каналам першого шару, та має 16 вихідних каналів. Третій згортковий шар, у свою чергу, має 16 вхідних каналів, збільшуючи їх кількість до 32 на виході. Кожен з цих шарів оснащений ядром згортки розміром 5x5 пікселів, що дозволяє детально обробляти зображення.

Пулінгові шари, які розміщені після кожного згорткового шару, використовують фільтри розміром 2x2. Вони здійснюють процес агрегування даних, зменшуючи розміри вхідних зображень та виділяючи найважливіші особливості, що важливо для ефективності процесу навчання мережі.

Така архітектура оптимізує процес обробки даних у мережі, забезпечуючи ефективне виявлення та класифікацію особливостей зображень, що є ключовим для роботи з великими обсягами даних та складними завданнями, як-от

розпізнавання жестів рук у дактильному алфавіті.

- Активація нейронів

В архітектурі цієї нейронної мережі (ЗНМ), функція активації ReLU (Rectified Linear Unit) застосовується до кожного згорткового та повнозв'язного шару, крім вихідного шару. ReLU є типом лінійного випрямлювача, який працює за принципом: якщо вхідний аргумент є від'ємним, функція повертає 0, а якщо аргумент є додатним, то повертається саме це число. Ця функція активації використовується для додавання нелінійності в процес обробки даних, що допомагає нейронній мережі ефективніше навчатися та вирішувати складніші завдання.

$$f(x) = \max(0, x).$$

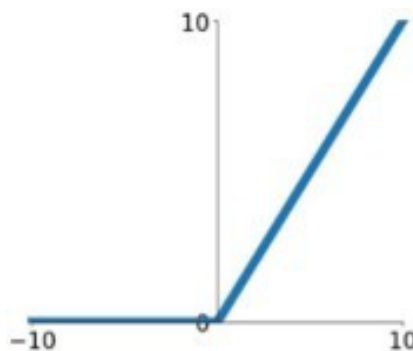


Рис. 2.2. Графік функції *ReLU*

- Оптимізатор

У даній нейронній мережі для оптимізації процесу навчання використовується алгоритм Adam (Adaptive Moment Estimation) із швидкістю навчання, встановленою на рівні 0.001. Алгоритм Adam є методом стохастичної оптимізації, який використовує адаптивні оцінки першого та другого моментів градієнтів для визначення швидкості навчання.

Алгоритм Adam ефективно поєднує переваги двох інших відомих методів оптимізації: RMSprop (Root Mean Square Propagation) і Stochastic Gradient Descent з моментом. Він автоматично коригує швидкість навчання для кожного параметра окремо, засновуючись на оцінці першого (середнього) та другого (невизначеного середнього квадрату) моментів градієнтів. Це дозволяє алгоритму бути більш ефективним у різних умовах, особливо у випадках, де оптимальні швидкості навчання сильно різняться між різними параметрами.

Основною перевагою Adam є його здатність адаптуватися до різних умов навчання, що робить його особливо корисним у задачах з великою кількістю даних та параметрів. Це робить його популярним вибором серед дослідників і розробників, які працюють над складними задачами машинного навчання, такими як розпізнавання образів, обробка природної мови та інші. Його здатність швидко адаптуватися та ефективно оптимізувати нейронні мережі є ключовою для досягнення високої точності та ефективності в процесі навчання.

2.4 Оптимізатор

Для задачі розпізнавання жестів було вибрано використання модуля OpenCV, який є важливим інструментом у галузі комп'ютерного зору. OpenCV надає широкі можливості для роботи з зображеннями та відео, включаючи відкриття, запис, відображення, а також можливість використання веб-камери для захоплення відео та зображень. Цей модуль також дозволяє виконувати різноманітні операції обробки зображень, такі як фільтрація, перетворення, виявлення об'єктів та інше.

Завдяки своїм функціональним можливостям, OpenCV стає ідеальним вибором для задач розпізнавання жестів. Цей модуль забезпечує високий рівень точності та ефективності у виявленні та класифікації жестів, що важливо для створення надійних і точних систем розпізнавання жестової мови. Крім того, можливість прямої інтеграції з веб-камерами та іншими пристроями захоплення зображень дозволяє розробникам створювати більш динамічні та інтерактивні

додатки. Використання OpenCV у проектах, пов'язаних з розпізнаванням жестів, значно розширює можливості дослідників та розробників, даючи їм в руки потужний інструмент для створення інноваційних рішень в цій сфері.

Розпізнавання жестів з веб-камери

Процес розпізнавання жестів з веб-камери зображений на рис. 2.3 та складається з трьох кроків:



Рисунок 2.3. Процес розпізнавання жестів з веб-камери

- **Захоплення зображення:** Користувач ініціює роботу програми та демонструє жести перед веб-камерою. Програма налаштована таким чином, що веб-камера фіксує 30 кольорових зображень кожну секунду. Кожне з цих зображень надсилається для подальшої обробки.
- **Попередня обробка зображень:** Виконується перетворення отриманих кольорових зображень у монохромні (сірі). Це не тільки спрощує обробку, але й зменшує обсяг даних для аналізу. Далі, змінюється розмір зображення для забезпечення однаковості формату обробки. Нормалізація значень пікселів проводиться для приведення їх до діапазону від 0 до 1, що сприяє кращому навчанню та обробці даних нейронною мережею.
- **Розпізнавання жестів:** Оброблені зображення подаються на вхід нейронної мережі. Мережа аналізує зображення, визначає виконаний жест і повертає його переклад або відповідне значення. Цей процес

включає виявлення ключових характеристик жесту, його класифікацію та трансляцію у відповідну символіку чи дію, що робить систему здатною розпізнавати та інтерпретувати жести в реальному часі.

- Розпізнавання жестів з відео

Процес розпізнавання жестів із відео, як зазначено на схемі 2.4, є процедурою, що має багато спільного з попередньо описаними методами. Основна відмінність полягає у вступному етапі, який включає вибір відеоматеріалу для подальшого аналізу. Цей процес починається з того, що користувач запускає відповідну програму та вибирає відео, яке необхідно аналізувати. Такий підхід дозволяє адаптувати процес розпізнавання до конкретних потреб та умов використання.

Далі, відібрані кадри з відео проходять етап попередньої обробки. На цій стадії відбувається первинний аналіз зображень, який може включати корекцію кольору, контрастності, а також видалення шумів та інших візуальних спотворень. Це критично важливо для підвищення точності подальшого розпізнавання жестів.



Рис. 2.4. Процес розпізнавання жестів з відео



Рис. 2.5. Процес розпізнавання жестів з теки з зображеннями

- Розпізнавання жестів з відео

Процес розпізнавання жестів з набору зображень детально ілюстрований на схемі 2.5. Цей процес має багато спільного з описаними раніше методами, але він має свою унікальність, особливо на початковому етапі - виборі теки з зображеннями. Користувач ініціює цей процес, запускаючи відповідну програму та вибираючи теку, що містить потрібні зображення. Вибір правильної теки є ключовим, оскільки він визначає датасет, який буде використовуватися для розпізнавання жестів.

Після вибору теки, зображення передаються на стадію попередньої обробки. На цій стадії відбувається перше фільтрування та оптимізація зображень для підготовки їх до подальшого аналізу. Це може включати коригування рівнів яскравості та контрасту, видалення шумів або небажаних елементів на фотографіях, а також перетворення зображень у формат, який найкраще підходить для обробки алгоритмами машинного навчання.

Після попередньої обробки, зображення проходять через складний процес розпізнавання жестів. Програма аналізує характеристики жестів на зображеннях, порівнюючи їх з відомими шаблонами та моделями. Використовуючи алгоритми глибокого навчання, система вчиться розрізняти

різні жести, їх значення та нюанси. Це включає ідентифікацію форми руки, положення пальців, а також контексту жесту, що може бути важливим у різних сценаріях застосування.

2.5 Алгоритм виводу літери на екран

У контексті обробки відео за допомогою OpenCV, кадрова частота системи встановлена на рівні 30 Гц. Це означає, що для процесу розпізнавання жестів, як з відео, так і з камери, неймережа оброблятиме 30 окремих зображень за кожную секунду. Ця висока частота кадрів є важливою для забезпечення плавності та точності в розпізнаванні рухів, але вона також створює виклик: як вивести на екран лише одну літеру, уникнувши появи декількох літер за короткий проміжок часу.

Відповідь на це питання полягає у розробці специфічного алгоритму. Користувач має можливість вибрати тривалість часу, протягом якого він повинен показувати жест, щоб він був коректно ідентифікований системою. Наприклад, якщо користувач вибирає двосекундний інтервал, система буде обробляти 60 кадрів (30 кадрів за секунду, помножене на дві секунди) для кожного потенційного жесту.

Алгоритм працює наступним чином: він реєструє кожную розпізнану літеру в окремий список. Якщо поточна літера відрізняється від попередньої, список очищається. Це допомагає уникнути помилкових визначень або короткочасних змін у позиції руки, які можуть відбуватися природно під час руху. Тільки коли в списку накопичується більше 60 однакових літер, система вважає цей жест достатньо стабільним та виводить відповідну літеру на екран.

Детальний опис цього алгоритму представлено на рисунку 2.6. Він демонструє кроки обробки від первинного вибору кадрів до кінцевого виведення результату. Такий підхід забезпечує, що система розпізнавання

жестів є точною та водночас достатньо гнучкою, щоб адаптуватися до різних стилів та швидкостей виконання жестів користувачами.

Висновки до розділу 2

У цьому розділі детально розглянуто аспекти проєктування додатку для розпізнавання знаків дактильного алфавіту. Проект включає в себе вибір технічних інструментів та ретельне планування архітектури системи.

Для реалізації було обрано мову програмування Python через її гнучкість та велику кількість бібліотек. PyCharm використовувався як основне середовище розробки завдяки його зручності для налагодження та інтеграції з іншими інструментами. PyTorch був обраний як основний фреймворк для глибокого навчання завдяки його ефективності в роботі з нейронними мережами, а OpenCV застосовувалась для обробки зображень.

Процес створення набору даних був ретельно продуманий:

- Збір даних (фотографій) відбувався власноруч, що дозволило забезпечити високу якість та релевантність зображень.
- Для тренування мережі були обрані лише статичні жести алфавіту, щоб спростити процес навчання та підвищити точність розпізнавання.
- Зображення були стандартизовані до розмірів 60×60 пікселів, що забезпечило уніфікацію вхідних даних.
- Усі кольорові зображення були перетворені у сірі монохромні, що дозволило зменшити обчислювальні вимоги та спростити обробку.

Архітектура нейронної мережі була спроектована з урахуванням вимог до точності та ефективності. Мережа складається з трьох згорткових шарів для видобутку ознак зображення, трьох пулінгових шарів для зменшення розмірів та двох повнозв'язних шарів для класифікації жестів. Для кожного шару були

визначені оптимальні кількості входів та виходів. Також були обрані функція активації нейронів (ReLU) для ефективності навчання, оптимізатор (Adam) для швидкого та стабільного збіження та функція втрат (Categorical Cross-Entropy Loss) для точної класифікації.

РОЗДІЛ 3. ОПИС ПРОГРАМИ

3.1. Головний екран

Після встановлення застосунку в AppStore або PlayMarket користувач бачить стартову сторінку. Головна сторінка зображена на рис. 3.1.

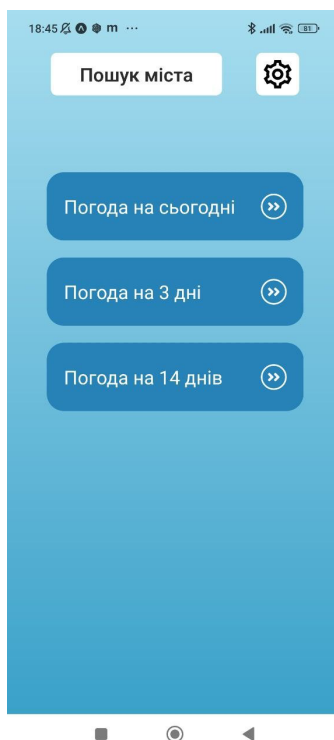


Рисунок 3.1 Головний екран

Відразу після запуску програми, відкриється модальне вікно де буде запропоновано підтвердити місце розташування користувача. Зображено на рис. 3.2.

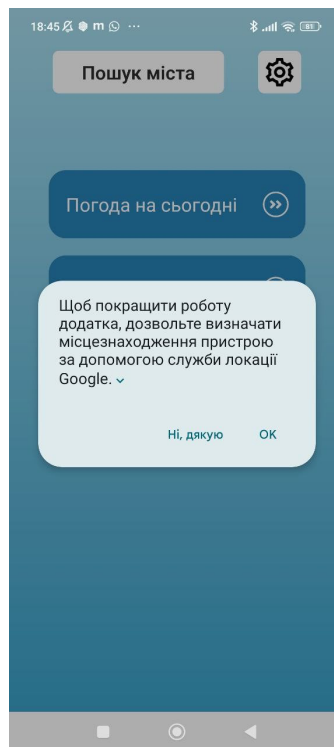


Рисунок 3.2 Модальне вікно з дозволом на геолокацію

Якщо користувач погоджується надати доступ до своє геолокації то на екрані з'явиться населений пункт де зараз знаходиться користувач. Зображено на рис. 3.3.

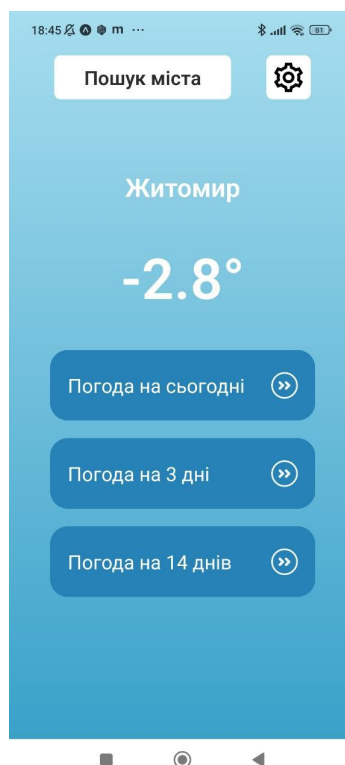


Рисунок 3.3 Погода в населеному пункті користувача

Інформація про погоду на сьогодні або на інші дні можна побачити обравши одну з трох кнопок (сьогодні, 3 дні, 14 днів) на рис. 3.4.



Рисунок 3.4 Погода на сьогодні

Для зручності користувача в програмі є можливість змінювати тему застосунку на денну або вечірню. Для цього користувач переходить в налаштування і в модальному вікні обирає зручний йому варіант , зображено на рис.3.5.

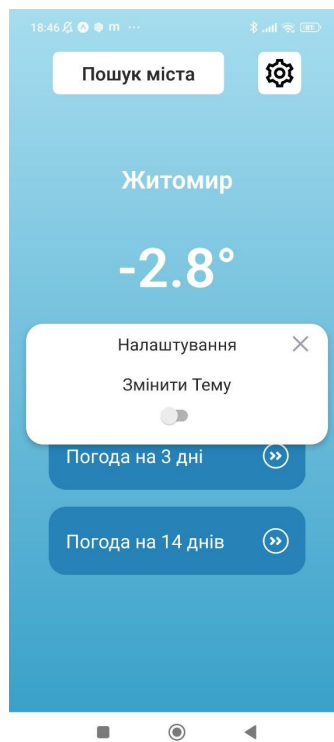


Рисунок 3.5 Модальне вікно зі зміною теми

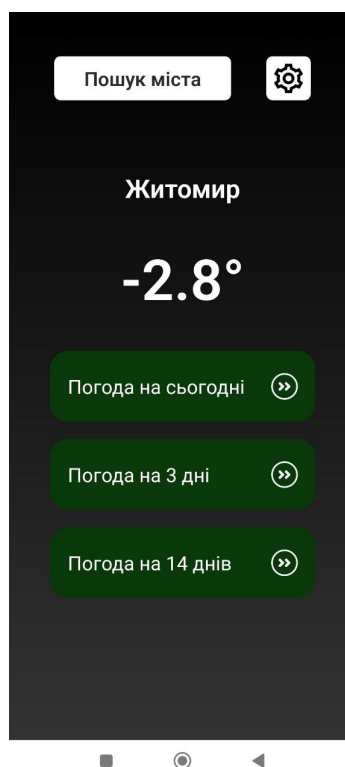


Рисунок 3.6 Головний екран застосунку при обираї тесної теми

Для того, щоб переглянути погоду в іншому населеному пункті натискаємо на кнопку Пошук Міста. Та переходимо на екран де ми можемо шукати обраний пункт Зображено на рис.3.7.

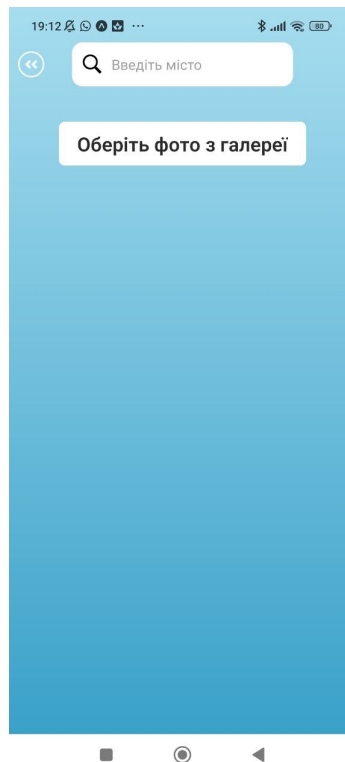


Рисунок 3.7 Екран пошуку міста

3.2. Сервіс завантаження фотографії

Далі пошук міста користувач може вести двома варіантами. Перший варіанти використовувати клавіатуру або використати алгоритм створений завдяки нейронній мережі щоб програма могла розуміти яка саме в нас на фото буква алфавіту та вводити її в строку пошуку. Тому натискаємо на кнопку Оберіть фотграфію і бачимо екран на рис. 3.8.

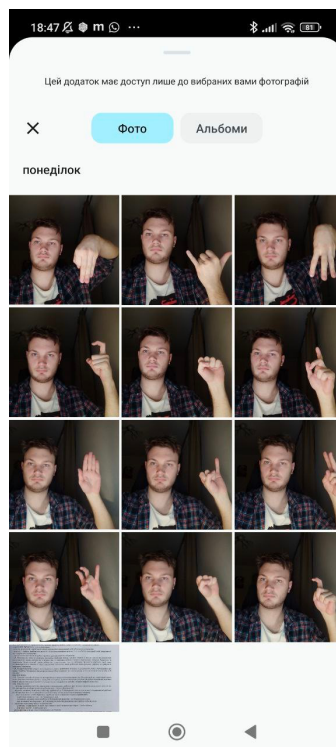


Рисунок 3.8 Вибір фотографії

Після вибору фотографії відкривається екран де ми можемо перед збереженням в програмі обрізати або змінити кут нахилу фотографії і це зображено на рис. 3.9.

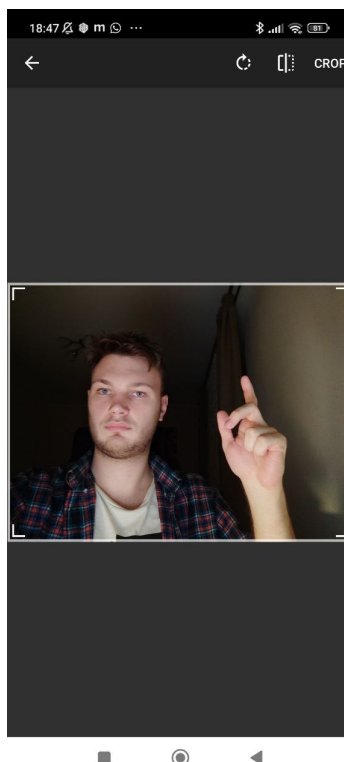


Рисунок 3.9 Підготовка фотографії до збереження в програмі

Після підтвердження розміру та куту нахилу ми можемо побачити фотографію на екрані. Далі натискає кнопку Відправка запиту на сервер і фотографія відправляється на хостинг HeroKu де буде зроблено оброблення та переклад рис. 3.10.

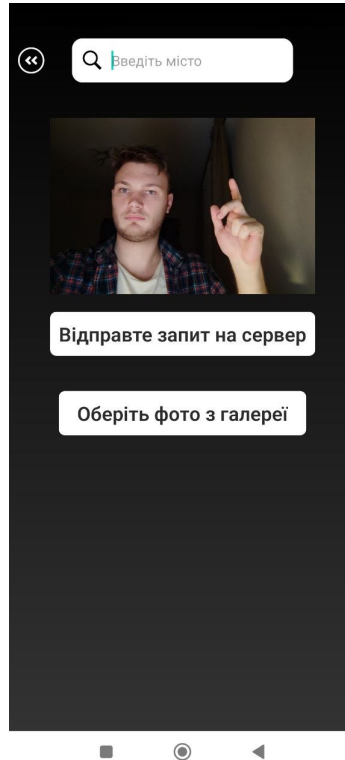


Рисунок 3.10 Відображення фотографії на екрані

3.3. Сервіс обробка фотографій

В разі успішного запиту на сервер в нас буде написано статус операції та літера буде написана в пошуковій строці, що зображено на рис. 3.11.

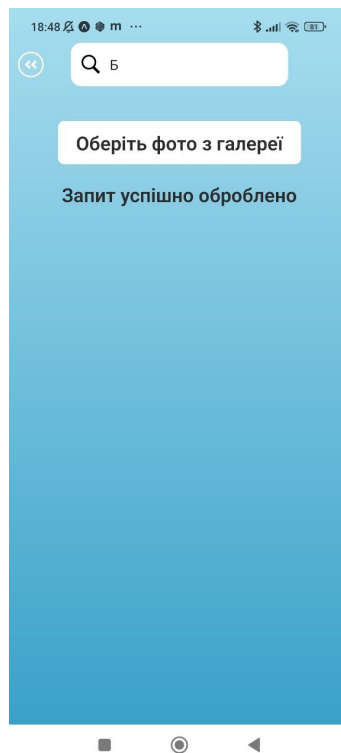


Рисунок 3.11 Інформація про статус відправлення зображення на сервер

Після написання або завантаження фотографії, коли набирається 3 або більше літери ми можемо натиснути кнопку Пошук, зображено на рис. 3.12. В нашому прикладі ми ввели 6 фотографій які нам видали назву населеного пункту Бахмут, після чого ми можемо натиснути кнопку Пошук для переходу на головний екран з результатом, зображено на рис. 3.13.

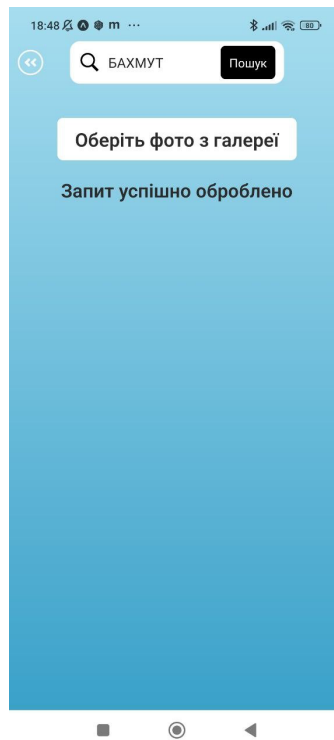


Рисунок 3.12 Появляється кнопка Пошук

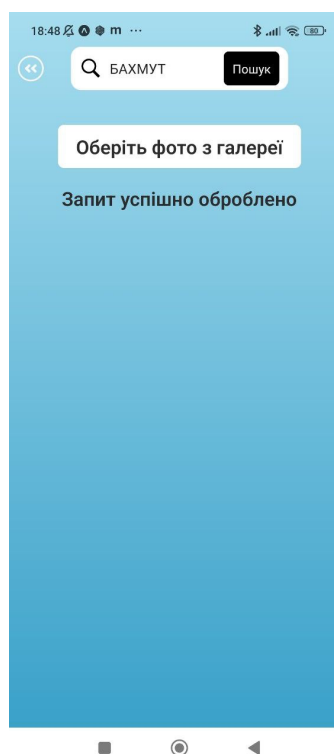


Рисунок 3.13 Вводимо повну назву населеного пункту

Переходимо на основний екран та бачимо погоду в реальному часі в населеному пункті який шукав користувач, зображено на рис. 3.14.

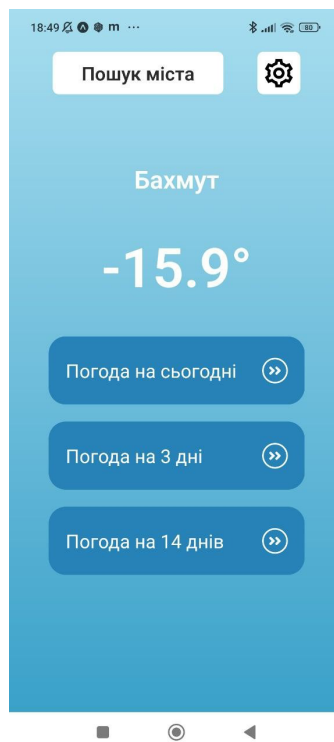


Рисунок 3.14 Інформація про погоду в шуканому населеному пункті

Висновки до розділу 3

У цьому розділі описано інтерфейс мобільного додатку для перегляду погоди. Після запуску додатку користувач спочатку взаємодіє зі стартовою сторінкою та модальним вікном для підтвердження геолокації. У додатку є опції для перегляду погоди на різні терміни та можливість зміни теми інтерфейсу. Функція "Пошук міста" дозволяє знайти погоду в іншому місці, з опцією вводу назви міста вручну або через розпізнавання фотографій. Після вибору фотографії, вона обробляється та використовується для введення інформації у пошукову строку. На головному екрані відображається погода для обраного місця.

ВИСНОВКИ

У даному проєкті була створена нейромережева програма для розпізнавання знаків дактильного алфавіту, використовуючи Python, Teachable Machine для глибинного навчання та бібліотеку OpenCV для обробки зображень. Додаток здатен розпізнавати жести через завантаження зображень з файлової системи.

У рамках проєкту проводилось вивчення тематики, аналіз робіт вітчизняних та іноземних дослідників, а потім був створений унікальний набір даних для тренування нейромережі. Процес розпізнавання жестів та їх перетворення у текст був детально розроблений та схематично представлений у проєкті.

Додаток проходив обширне тестування. Оскільки первинний набір даних містив жести лише однієї особи, для тестування були залучені додаткові учасники, що дозволило підвищити універсальність та точність розпізнавання.

Проєкт важливий через відсутність подібних публічно доступних програм. Оскільки це система з відкритим вихідним кодом і набором даних, користувачі можуть самостійно редагувати та доповнювати датасет, налаштовуючи мережу під свої індивідуальні потреби.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. T. Starner, J. Weaver, A. Pentland. Real-time American sign language recognition using desk and wearable computerbased video. Pattern Analysis and Machine Intelligence – Режим доступу до ресурсу: <https://ieeexplore.ieee.org/document/735811/>.
2. Y. LeCun et al. Deep learni – Режим доступу до ресурсу: <https://www.nature.com/articles/nature14539>
3. M.Z. Islam, M.S. Hossain, R. Islam and K. Andersson. Static Hand Gesture Recognition using Convolutional Neural Network with Data – Режим доступу до ресурсу: <http://www.diva-portal.org/smash/get/diva2:1299000/FULLTEXT01.pdf/>
4. V. Nair, G.E. Hinton. Rectified Linear Units Improve Restricted Boltzmann Machines – Режим доступу до ресурсу: <http://www.diva-portal.org/smash/~fritz/absps/reluICML.pdf/>
5. K. Simonyan, A. Zisserman. Very deep convolutional neural networks for large-scale image recognition Network – Режим доступу до ресурсу: <https://arxiv.org/abs/1409.1556/>
6. M.D. Zeiler, R. Fergus. Visualizing and Understanding Convolutional Neural Networks – Режим доступу до ресурсу: <https://arxiv.org/abs/1409.1556/>
7. M. Lin, Q. Chen, and S. Yan. Network In Network – Режим доступу до ресурсу: <https://arxiv.org/pdf/1312.4400/>
8. O. Russakovsky et al. ImageNet Large Scale Visual Recognition Challenge. International Journal of Computer Vision – Режим доступу до ресурсу: <https://arxiv.org/abs/1409.0575/>
9. J. Gua et al. «Recent Advances in Convolutional Neural Networks» – Режим доступу до ресурсу: <https://arxiv.org/abs/1512.07108/>
10. A.A. Hosain, P.S. Santhalingam, P. Pathak, H. Rangwala, J. Kosecka. FineHand: Learning Hand Shapes for American Sign Language Recognition

- [*Electronic resource*]. – Режим доступа до ресурсу:
<https://arxiv.org/abs/2003.08753/>
11. B. Kang, S. Tripathi, T.Q. Nguyen. *Real-time Sign Language Fingerspelling Recognition using Convolutional Neural Networks from Depth map* [Электронный ресурс] – Режим доступа до ресурсу:
<https://arxiv.org/abs/1509.03001/>.
 12. Pigou, L., Dieleman, S., Kindermans, P.-J., Schrauwen, B.: *Sign language recognition using convolutional neural networks* [Электронный ресурс] – Режим доступа до ресурсу: <https://ieeexplore.ieee.org/document/5204291/>.
 13. S. Liwicki, M. Everingham. *Automatic recognition of fingerspelled words in British Sign Language* [Электронный ресурс] – Режим доступа до ресурсу: <https://ieeexplore.ieee.org/document/5204291/>.
 14. CS231n: *Convolutional Neural Networks for Visual Recognition* [Электронный ресурс] – Режим доступа до ресурсу: <https://cs231n.github.io/convolutional-networks/>
 15. MotionSavvy (2021) [Электронный ресурс] – Режим доступа до ресурсу: <http://motionsavvy.com>.
 16. Machine learning for anomaly detection [Электронный ресурс] – Режим доступа до ресурсу: <https://www.geeksforgeeks.org/machine-learning-for-anomaly-detection/>.
 17. Statistics [Электронный ресурс] – Режим доступа до ресурсу: <https://www.britannica.com/science/statistics/Numerical-measures>.
 18. Machine Learning A-Z™: Python & R in Data Science [2022] [Электронный ресурс] – Режим доступа до ресурсу: <https://ua.udemy.com/course/machinelearning/learn> .
 19. Clow M. *Learn Google Flutter Fast: 65 Example Apps* // Mark Clow, 2019. – 476 с.

20. Zammetti F. Practical Flutter: Improve your Mobile Developemnt with Google's Latest Open-Source SDK // Frank Zammetti, 2019. – 416 с.
21. Zaccagnino C. Programming Flutter: Native, Cross-Platform Apps the Easy Way// Carmine Zaccagnino, 2020. – 275 с.
22. Sells C. What's new in Flutter 2.2. [Электронный ресурс] / Sells Chris 2021. - Режим доступа до ресурсу: <https://medium.com/flutter/whats-new-in-flutter-2-2-fd00c65e2039>
23. Kuppusamy K.S. What makes Dart an easy, scalable and multi-purpose programming language: [Электронный ресурс] // opensourceforu.com. – 2017. – Режим доступа до ресурсу: <https://www.opensourceforu.com/2017/06/dart-easy-scalable-multi-purpose-programming-language/>
24. Napoli M. Beginning Flutter: A Hand on Guide to App Development / Marco Napoli, 2019. – 528 с.
25. Rap P. Beginning App Development with Flutter: Create Cross-Platform Mobile Apps / Rap Payne, 2019. – 309 с.

ДОДАТКИ

Код для створення фотографій кожної літери

```
import cv2
from cvzone.HandTrackingModule import HandDetector
import numpy as np
import math
import time

cap = cv2.VideoCapture(0)
detector = HandDetector(maxHands=1)

offset = 20
imgSize = 300

folder = "Data/C"
counter = 0

while True:
    success, img = cap.read()
    hands, img = detector.findHands(img)
    if hands:
        hand = hands[0]
        x, y, w, h = hand['bbox']

        imgWhite = np.ones((imgSize, imgSize, 3), np.uint8) * 255
        imgCrop = img[y - offset:y + h + offset, x - offset:x + w + offset]

        imgCropShape = imgCrop.shape
```

```

aspectRatio = h / w

if aspectRatio > 1:
    k = imgSize / h
    wCal = math.ceil(k * w)
    imgResize = cv2.resize(imgCrop, (wCal, imgSize))
    imgResizeShape = imgResize.shape
    wGap = math.ceil((imgSize - wCal) / 2)
    imgWhite[:, wGap:wCal + wGap] = imgResize

else:
    k = imgSize / w
    hCal = math.ceil(k * h)
    imgResize = cv2.resize(imgCrop, (imgSize, hCal))
    imgResizeShape = imgResize.shape
    hGap = math.ceil((imgSize - hCal) / 2)
    imgWhite[hGap:hCal + hGap, :] = imgResize

cv2.imshow("ImageCrop", imgCrop)
cv2.imshow("ImageWhite", imgWhite)

cv2.imshow("Image", img)
key = cv2.waitKey(1)
if key == ord("s"):
    counter += 1
    cv2.imwrite(f'{folder}/Image_{time.time()}.jpg', imgWhite)
print(counter)

```

Код приймання запитів на сервері

```
from flask import Flask, request, jsonify
import cv2
from cvzone.HandTrackingModule import HandDetector
from cvzone.ClassificationModule import Classifier
import numpy as np
import math
import os
from werkzeug.utils import secure_filename

app = Flask(__name__)
app.config['UPLOAD_FOLDER'] = 'uploads/'

detector = HandDetector(maxHands=1)
classifier = Classifier("Model/keras_model.h5", "Model/labels.txt")

labels = ["A", "Б", "В", "Г", "Д", "Е", "Є", "Ж", "З", "И", "І", "Й", "К", "Л", "М",
          "Н", "О", "П", "Р", "С", "Т", "У", "Ф", "Х", "Ц", "Ч", "Ш", "Ъ", "Ю", "Я"]

@app.route('/process_image', methods=['POST'])
def process_image():
    if 'file' not in request.files:
        return jsonify(error='No file part'), 400
    file = request.files['file']
    if file.filename == '':
        return jsonify(error='No selected file'), 400
    if file:
        filename = secure_filename(file.filename)
```

```

        file_path = os.path.join(app.config['UPLOAD_FOLDER'], filename)
        file.save(file_path)

        result, label = process_frame(file_path)
        return jsonify(label=label, result=result)

def process_frame(file_path):
    img = cv2.imread(file_path)
    imgOutput = img.copy()
    hands, img = detector.findHands(img)

    if hands:
        hand = hands[0]
        x, y, w, h = hand['bbox']

        offset = 20
        imgSize = 300
        imgWhite = np.ones((imgSize, imgSize, 3), np.uint8) * 255
        imgCrop = img[y - offset:y + h + offset, x - offset:x + w + offset]

        aspectRatio = h / w

        if aspectRatio > 1:
            k = imgSize / h
            wCal = math.ceil(k * w)
            imgResize = cv2.resize(imgCrop, (wCal, imgSize))
            wGap = math.ceil((imgSize - wCal) / 2)
            imgWhite[:, wGap:wCal + wGap] = imgResize
        else:
            k = imgSize / w

```

```
        hCal = math.ceil(k * h)
        imgResize = cv2.resize(imgCrop, (imgSize, hCal))
        hGap = math.ceil((imgSize - hCal) / 2)
        imgWhite[hGap:hCal + hGap, :] = imgResize

    prediction, index = classifier.getPrediction(imgWhite, draw=False)
    label = labels[index]

    return (prediction, label)
else:
    return ("No hand detected", "None")

if __name__ == '__main__':
    app.run(debug=True)
```

Код відправлення фотографії на сервер

```
import React, { useState } from 'react';
import { View, StyleSheet, Image, TouchableOpacity, Text } from 'react-native';
import { useSelector } from "react-redux";
import CustomInput from '../components/CustomInput';
import { LinearGradient } from "expo-linear-gradient";
import * as ImagePicker from 'expo-image-picker';

const CitySearchScreen = ({ navigation }) => {
  const [city, setCity] = useState('');
  const [image, setImage] = useState(null);
  const [requestStatus, setRequestStatus] = useState(null);

  const isDarkTheme = useSelector((state) => state.theme.isDarkTheme);

  const onSearch = () => {
    onCitySelect();
  };

  const onCitySelect = () => {
    navigation.navigate('MainScreens', { selectedCity: city });
  };

  const sendRequestToServer = async () => {
    if (!image) return;

    const formData = new FormData();
```

```

formData.append('image', {
  uri: image,
  type: 'image/jpeg',
  name: 'photo.jpg',
});

try {
  setRequestStatus('Отправка запроса...');
  const response = await fetch('https://morning-escarpment-40202.herokuapp.com/process_image', {
    method: 'POST',
    body: formData,
    headers: {
      'Content-Type': 'multipart/form-data',
    },
  });
};

const data = await response.json();

if (data.letter) {
  setCity(prevCity => prevCity + data.letter);
} else {
  console.log('Буква не была получена');
}
} catch (error) {
  console.error('Ошибка при отправке запроса:', error);
} finally {
  setRequestStatus(null);
  setImage(null);
}

```

```

};

const pickImage = async () => {
  let result = await ImagePicker.launchImageLibraryAsync({
    mediaTypes: ImagePicker.MediaTypeOptions.All,
    allowsEditing: true,
    aspect: [4, 3],
    quality: 1,
  });

  if (!result.canceled) {
    setImage(result.assets[0].uri);
    setRequestStatus(null);
  }
};

return (
  <LinearGradient
    colors={isDarkTheme ? ["#000", "#333"] : ["#a7ddef", "#3aa1c9"]}
    style={styles.container}
  >
    <View style={styles.header}>
      <TouchableOpacity onPress={() => navigation.goBack()}>
        </TouchableOpacity>
        <CustomInput city={city} setCity={setCity} onSearch={onSearch} />
        <View style={{ width: 30, height: 30 }} />
      </View>
      {image} && <Image source={{ uri: image }} style={styles.image} />
      {image} && (
        <TouchableOpacity style={styles.sendButton} onPress={sendRequestToServer}>

```

```

        <Text style={{ fontSize: 22, fontWeight: 'semibold' }}>Відправте запит на
сервер</Text>
      </TouchableOpacity>
    )}
    <TouchableOpacity style={styles.acceptButtonWrapper} onPress={pickImage}>
      <Text style={{ fontSize: 22, fontWeight: 'semibold' }}>Оберіть фото з
галереї</Text>
    </TouchableOpacity>
    {requestStatus && <Text style={{ fontSize: 22, fontWeight: 'semibold',
marginVertical: 20 }}>{requestStatus}</Text>}
  </LinearGradient>
);
};

const styles = StyleSheet.create({
  container: {
    flex: 1,
    width: '100%',
    paddingTop: 40,
    alignItems: 'center',
  },
  image: {
    width: 300,
    height: 200,
    marginTop: 20,
  },
  sendButton: {
    backgroundColor: '#fff',
    borderRadius: 7,
    width: 300,

```

```

    height: 50,
    justifyContent: "center",
    alignItems: "center",
    marginVertical: 20,

  },
  acceptButtonWrapper: {
    backgroundColor: "#fff",
    borderRadius: 7,
    width: 280,
    height: 50,
    justifyContent: "center",
    alignItems: "center",
    marginVertical: 20,
  },
  goBackButton: {
    backgroundColor: "#fff",
    borderRadius: 7,
    width: 280,
    height: 50,
    justifyContent: "center",
    alignItems: "center",
    marginBottom: 80,
  },
  header: {
    width: '100%',
    flexDirection: 'row',
    alignItems: 'center',
    justifyContent: 'space-between',
    paddingHorizontal: 10,

```

```
        marginBottom: 20,  
      },  
  
    ));  
  
export default CitySearchScreen;
```

Код головного екрану

```
import React, { useState, useEffect } from "react";
import { StyleSheet, Text, View, TouchableOpacity } from "react-native";
import { useSelector, useDispatch } from "react-redux";
import { setWeather } from "../redux/actions/weatherActions";
import { fetchCurrentLocation } from "../components/location";
import { LinearGradient } from "expo-linear-gradient";
import WeatherWindows from "../components/WeatherWindows";
import { Typography } from "../Typography";
import WeatherIcon from "../components/WeatherIcon";
import Settings from "../assets/icons/settings.svg";
import SettingsModal from "../components/SettingsModal";
import { useNavigation } from "@react-navigation/native";
import { fetchWeatherForLocation, fetchWeatherForCity } from
"../components/weatherService";

const MainScreens = () => {
  const navigation = useNavigation();
  const [city, setCity] = useState("");
  const weather = useSelector((state) => state.weather.weatherData);
  const locationName = useSelector((state) => state.weather.locationName);
  const dispatch = useDispatch();
  const [modalVisible, setModalVisible] = useState(false);
  const isDarkTheme = useSelector((state) => state.theme.isDarkTheme);

  const fetchWeatherForCurrentLocation = async () => {
    const location = await fetchCurrentLocation();
    if (location) {
```

```

    const { latitude, longitude } = location.coords;
    const weatherData = await fetchWeatherForLocation(latitude, longitude);
    if (weatherData) {
      dispatch(setWeather(weatherData));
    }
  }
};

useEffect(() => {
  fetchWeatherForCurrentLocation();
}, []);

useEffect(() => {
  const unsubscribe = navigation.addListener('focus', () => {
    const routeParams = navigation.getState().routes.find(route => route.name ===
'MainScreens')?.params;
    if (routeParams?.selectedCity) {
      setCity(routeParams.selectedCity);
      fetchWeather(routeParams.selectedCity);
    }
  });

  return unsubscribe;
}, [navigation]);

const fetchWeather = async (cityToSearch) => {
  const weatherData = await fetchWeatherForCity(cityToSearch || city);
  if (weatherData) {
    dispatch(setWeather(weatherData));
  }
};

```

```

    }
  };

  return (
    <LinearGradient
      colors={isDarkTheme ? ["#000", "#333"] : ["#a7ddef", "#3aa1c9"]}
      style={styles.container}
    >
      <View style={{ flexDirection: "row", gap: 40 }}>
        <TouchableOpacity
          style={styles.inputWrapper}
          onPress={() => navigation.navigate('CitySearchScreen')}
        >
          <Typography f22 semibold >Помык миста</Typography>
        </TouchableOpacity>
        <View style={styles.settingsWrapper}>
          <TouchableOpacity onPress={() => setModalVisible(true)}>
            <Settings width={40} height={40} />
          </TouchableOpacity>
        </View>
      </View>

      {locationName && (
        <Typography f29 semibold color="#fff">
          {locationName}
        </Typography>
      )}
      {weather && (
        <View style={styles.weatherContainer}>
          <WeatherIcon conditionText={weather.current.condition.text} />

```

```

        <Typography
          f64
          semibold
          color="#fff"
          textAlign="center"
          marginTop={20}
          marginBottom={30}
        >
          {weather.current.temp_c}°
        </Typography>
      </View>
    )}

    <TouchableOpacity onPress={() => navigation.navigate("TodayWeather")}>
      <WeatherWindows title="Погода на сьогодні" />
    </TouchableOpacity>

    <TouchableOpacity onPress={() => navigation.navigate("ThreeDayWeather")}>
      <WeatherWindows title="Погода на 3 дні" />
    </TouchableOpacity>

    <TouchableOpacity
      onPress={() => navigation.navigate("FourteenDayWeather")}
    >
      <WeatherWindows title="Погода на 14 днів" />
    </TouchableOpacity>

    <SettingsModal
      visible={modalVisible}
      onClose={() => setModalVisible(false)}
    >

```

```

    />
  </LinearGradient>
);
};

const styles = StyleSheet.create({
  container: {
    flex: 1,
    alignItems: "center",
    justifyContent: "flex-start",
    width: "100%",
    paddingHorizontal: 40,
    paddingTop: 50,
  },
  weatherContainer: {
    marginTop: 20,
  },
  settingsWrapper: {
    backgroundColor: "#fff",
    borderRadius: 7,
    width: 50,
    height: 50,
    justifyContent: "center",
    alignItems: "center",
    marginBottom: 80,
  },
  inputWrapper: {
    backgroundColor: "#fff",

```

```
borderRadius: 7,  
width: 200,  
height: 50,  
justifyContent: "center",  
alignItems: "center",  
marginBottom: 80,  
},  
));  
  
export default MainScreens;
```