МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Чорноморський національний університет імені Петра Могили

Факультет комп'ютерних наук

Кафедра інженерії програмного забезпечення

ДОПУЩЕНО ДО ЗАХИСТУ
Завідувач кафедри інженерії
програмного забезпечення
_____ Євген ДАВИДЕНКО
«___»_____ 2024 р.

КВАЛІФІКАЦІЙНА РОБОТА

НА ЗДОБУТТЯ ОСВІТНЬОГО СТУПЕНЯ МАГІСТРА

ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ РОЗПІЗНАВАННЯ ЖЕСТОВОЇ МОВИ НА ОСНОВІ АЛГОРИТМІВ МАШИННОГО НАВЧАННЯ

Спеціальність 121 Інженерія програмного забезпечення

Освітня програма «Інженерія програмного забезпечення»

Здобувач _____Микола ШУМАКОВ
«___»_____ 2024 р.

Керівник д-рка техн. наук, професорка _____ Альона ШВЕД
«___»_____ 2024 р.

Миколаїв – 2024

Чорноморський національний університет імені Петра Могили

(повне найменування закладу вищої освіти)

Факультет	Комп'ютерних наук
Кафедра	Інженерії програмного забезпечення
Рівень вищої освіти	Другий (магістерський)
Освітній ступень	Магістр
Спеціальність	Інженерія програмного забезпечення
Освітня програма	Інженерія програмного забезпечення

ЗАТВЕРДЖУЮ

Завідувач кафедри інженерії
програмного забезпечення
_____Євген ДАВИДЕНКО
«____»_____2024 р.

ЗАВДАННЯ
на кваліфікаційну роботу здобувача

_____Шумакову Миколі Віталійовичу_____
(прізвище, ім'я, по батькові здобувача)

1. Тема кваліфікаційної роботи
_____« Програмне забезпечення розпізнавання жестової мови на основі
_____алгоритмів машинного навчання»_____

Затверджена наказом ректора ЧНУ ім. Петра Могили № 220 від «4» вересня 2024 р.

2. Строк представлення кваліфікаційної роботи «20» грудня 2024 р.

3. Очікуваний результат роботи та початкові дані якщо такі потрібні
____Програмне забезпечення розпізнавання жестової мови. Відео жестової мови у режимі реального часу._____

4. Перелік питань, що підлягають розробці:
_____ Дослідження існуючих програмних рішень в галузі розпізнавання

жестів; дослідження можливості використання моделей на основі YOLOv5, проєктування концепції та архітектури програмного забезпечення розпізнавання, проєктування інтерфейсу користувача, програмна реалізація застосунку, формування датасетів, навчання моделі розпізнавання жестів, тестування та відлагодження.

5. Перелік графічних матеріалів
Презентація

6. Консультанти:

Консультант	Кафедра (організація)	Частина роботи

Керівник роботи _____ Альона ШВЕД
Особистий підпис *Власне ім'я ПРІЗВИЩЕ*

Здобувач _____ Микола ШУМАКОВ
Особистий підпис *Власне ім'я ПРІЗВИЩЕ*

Дата видачі завдання « ____»_____20_____р.

КАЛЕНДАРНИЙ ПЛАН

виконання кваліфікаційної роботи

Тема:_____«Програмне забезпечення розпізнавання жестової мови на основі алгоритмів машинного навчання»_____

№	Найменування роботи	Початок	Закінчення	Примітки
1.	Розробка та затвердження завдання на виконання КРМ	02.09.2024р.	06.09.2024р.	виконано
2.	Огляд літератури за темою роботи	09.09.2024р.	11.09.2024р.	виконано
3.	Складання календарного плану КРМ	12.09.2024р.	13.09.2024р.	виконано
4.	Аналіз предметної області	16.09.2024р.	18.09.2024р.	виконано
5.	Розробка проєктних рішень	19.09.2024р.	20.09.2024р.	виконано
6.	Моделювання та конструювання ПЗ	23.09.2024р.	24.09.2024р.	виконано
7.	Кодування, тестування та апробація розробленого ПЗ, аналіз результатів тестування, розробка керівництва користувача	25.09.2024р.	01.11.2024р.	виконано
8.	Відгук керівника КРМ	02.11.2024р.	05.11.2024р.	виконано
9.	Оформлення КРМ та презентації	06.11.2024р.	27.11.2024р.	виконано
10.	Попередній захист	28.11.2024р.	28.11.2024р.	виконано
11.	Рецензування	29.11.2024р.	30.11.2024р.	виконано
12.	Завершення оформлення КРМ та презентації	01.12.2024р.	13.12.2024р.	виконано
13.	Захист кваліфікаційної роботи	19.12.2024р.	19.12.2024р.	виконано

Розробив здобувач _____Шумаков М. В._____ _____
(прізвище, ім'я, по батькові) *(підпис)*
«__» _____ 20__ р.

Керівник роботи __д-рка. техн. наук, професорка Швед А. В. _____
(посада, прізвище, ім'я, по батькові) *(підпис)*
«__» _____ 20__ р.

АНОТАЦІЯ

до кваліфікаційної роботи магістра

«Програмне забезпечення розпізнавання жестової мови на основі алгоритмів машинного навчання»

Здобувач 608 гр.: Шумаков Микола Віталійович

Керівник: д-рка. техн. наук, професорка Швед А. В.

Жестова мова стає все більш популярною у сферах телебачення, освіти, комунікації та інклюзивних технологій, викликаючи потребу у вдосконаленні методів розпізнавання жестів. Зокрема, вона допомагає людям з вадами слуху покращувати спілкування. Спеціальне програмне забезпечення може покращити навчання та дослідження жестової мови, підвищуючи якість розпізнавання та сприяючи інклюзивності. Таке ПЗ дозволяє тестувати нові алгоритми машинного навчання, а також розвивати науково-технічну базу галузі. Інтерактивні додатки на основі машинного навчання допоможуть користувачам вдосконалювати знання жестової мови та сприятимуть її популяризації в суспільстві.

Об’єктом дослідження є процес розпізнавання мови жестів у режимі реального часу.

Предметом дослідження є сучасні технології машинного навчання і комп'ютерного зору для розпізнавання жестів у режимі реального часу.

Метою кваліфікаційної роботи є автоматизація процесу розпізнавання жестів, що позначають символи американського жестового алфавіту в режимі реального часу із застосуванням технологій машинного навчання.

Для досягнення даної мети необхідно вирішити наступні завдання:

– вивчити можливості та переваги алгоритмів машинного навчання для розпізнавання жестової мови;

– проаналізувати сучасні методи використання жестової мови та визначити ключові вимоги до програмного забезпечення;

– розробити концепцію та архітектуру програмного забезпечення;

– реалізувати основні компоненти застосунку, такі як моделювання жестів, розпізнавання, віртуальне середовище та інтерфейс користувача;

– провести тестування та відпрацювання розробленого програмного забезпечення з метою виявлення та усунення недоліків;

У першому розділі описується аналіз предметної області, а саме розпізнавання жестів і навчання розпізнаванню жестів за допомогою програмного забезпечення, а також аналіз аналогічного програмного забезпечення.

У другому розділі проводиться аналіз розробки системи розпізнавання жестової мови. Описано ключові функціональні можливості бібліотеки та їх застосування в обробці зображень. Наведено сценарії використання системи. Показано mockup інтерфейсу користувача та наведено діаграми, що демонструють архітектуру та потоки роботи системи.

У третьому розділі описується основна реалізація системи розпізнавання жестів, включаючи використання бібліотек та інструментів для обробки зображень, відео та даних з камери.

У четвертому розділі описується розробка системи для розпізнавання жестів американського жестового алфавіту за допомогою моделі YOLOv5. Процес включає підготовку даних (аугментацію), навчання моделі з попередньо натренованими вагами, використання CUDA для прискорення обчислень. Також розглядається оцінка точності моделі для досягнення високих результатів у реальних умовах.

У висновках описується аналіз роботи, яка проводилася для створення програмного застосунку та аналіз результатів які були отримані.

КМР викладена на 94 сторінки, вона містить 4 розділи, 31 ілюстрацій, 8 таблиці, 18 джерел в переліку посилань

Ключові слова: *розпізнавання жестів, Американська жестова мова(ASL), YOLOv5, модель, машинне навчання.*

ABSTRACT

of the Master's Thesis

" Sign Language Recognition Software Based on Machine Learning Algorithms "

Student: 608 group: Shumakov M. V.

Supervisor: Dr.Sc., Professor Shved M. T.

Sign language is becoming increasingly popular in the fields of television, education, communication, and inclusive technologies, creating a need for improved gesture recognition methods. Specifically, it helps individuals with hearing impairments enhance communication. Special software can improve the learning and research of sign language, increasing recognition quality and promoting inclusivity. Such software allows testing new machine learning algorithms and developing the scientific and technical foundation of the field. Interactive applications based on machine learning will help users improve their knowledge of sign language and contribute to its popularization in society.

The object of the research is the process of sign language recognition in real-time.

The subject of the research is modern machine learning and computer vision technologies for real-time gesture recognition.

The purpose of the qualification work is to automate the process of recognition of gestures denoting symbols of the American Sign Alphabet in real time using machine learning technologies.

To achieve this goal, the following tasks must be solved:

– to study the capabilities and advantages of machine learning algorithms for sign language recognition;

– to analyze modern methods of using sign language and determine key requirements for software;

– to develop the concept and architecture of the software;

– to implement the main components of the application, such as gesture modeling, recognition, virtual environment and user interface;

– to test and develop the developed software in order to identify and eliminate shortcomings;

The first section describes the analysis of the subject area, namely gesture recognition and training in gesture recognition using software, as well as the analysis of similar software.

The second section analyzes the development of a sign language recognition system. The key functional capabilities of the library and their application in image processing are described. The scenarios for using the system are presented. A mockup of the user interface is shown and diagrams demonstrating the architecture and workflows of the system are presented.

The third section describes the basic implementation of the gesture recognition system, including the use of libraries and tools for processing images, videos, and camera data.

This fourth section describes the development of a system for recognizing American Sign Language gestures using the YOLOv5 model. The process includes data preparation (augmentation), training the model with pre-trained weights, and using CUDA to accelerate calculations. It also considers the assessment of the accuracy of the model to achieve high results in real-world conditions.

The conclusions describe the analysis of the work that was done to create the software application and the analysis of the results that were obtained.

The MSD is set out on 94 pages, it contains 4 sections, 31 illustrations, 8 tables, 18 sources in the reference list

Keywords: gesture recognition, American Sign Language (ASL), YOLOv5, model, machine learning.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ

УЖМ – Українська жестова мова

ПЗ – Програмне забезпечення

ASL – American Sign Language

BSL – British Sign Language

LSF – Langue des Signes Française

JSL – Japanese Sign Language

UML – Unified Modeling Language

ВСТУП

Зростання популярності використання жестової мови в різних галузях, таких як телебачення, освіта, комунікація та інклюзивні технології, ставить питання розробки ефективних методів розпізнавання жестів. Однією з галузей застосування жестової мови є комунікація з людьми які мають вади слуху, де вона може використовуватись для полегшення спілкування та взаємодії. Програмне забезпечення дозволить створити ефективне середовище для навчання та досліджень, що підвищить якість розпізнавання жестів та сприятиме інклюзивності. Програмне забезпечення може допомогти у відпрацюванні різноманітних сценаріїв використання жестової мови та тестуванні нових алгоритмів машинного навчання, що сприяє розвитку науково-технічної бази цієї галузі та навчанню людей використовувати жестову мову. Також застосунок на базі машинного навчання може бути корисним для перевірки знань користувача жестової мови, та для створення інтерактивних навчальних програм, що допоможуть користувачам вдосконалювати свої навички жестової мови та сприятимуть поширенню її використання в суспільстві.

Об’єктом дослідження є процес розпізнавання мови жестів у режимі реального часу.

Предметом дослідження є сучасні технології машинного навчання і комп'ютерного зору для розпізнавання жестів у режимі реального часу.

Метою кваліфікаційної роботи є автоматизація процесу розпізнавання жестів, що позначають символи американського жестового алфавіту в режимі реального часу із застосуванням технологій машинного навчання.

Для досягнення даної мети необхідно вирішити наступні завдання:

– вивчити можливості та переваги алгоритмів машинного навчання для розпізнавання жестової мови;

– проаналізувати сучасні методи використання жестової мови та визначити ключові вимоги до програмного забезпечення;

– розробити концепцію та архітектуру програмного забезпечення;

– реалізувати основні компоненти застосунку, такі як моделювання жестів, розпізнавання, віртуальне середовище та інтерфейс користувача;

– провести тестування та відпрацювання розробленого програмного забезпечення з метою виявлення та усунення недоліків;

– розглянути можливості застосування розробленого застосунку в науково-технічних дослідженнях, таких як випробування нових алгоритмів розпізнавання жестів та відпрацювання стратегій поведінки в різних ситуаціях.

Апробація результатів КМР відбулась під час XXVII Всеукраїнської науково-практичної конференції «Могилянські читання 2024», Миколаїв, 06-10 листопада, 2024 р. (Додаток В).

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

Жестова мова – це складна і повноцінна система комунікації, яка використовується переважно глухими та слабочуючими людьми, але також може застосовуватись у ширшому колі для міжособистісного спілкування. На відміну від звукових мов, жестова мова передає значення за допомогою візуальних жестів, що охоплюють рухи рук, положення пальців, міміку, вираз обличчя та загальну поставу тіла. Жестова мова має власну граматику, лексичні структури і синтаксис, що робить її цілком самостійною мовною системою.

1.1 Актуальність жестової мови в сучасному суспільстві

Жестова мова відіграє важливу роль у забезпеченні рівноправної комунікації для людей з порушенням слуху, надаючи їм можливість ефективно взаємодіяти зі світом. У сучасному суспільстві важливість жестової мови суттєво зростає через декілька факторів:

- інклюзія та права людей з інвалідністю;
- розширення сфери застосування жестової мови;
- технологічний прогрес та жестова мова;
- підвищення обізнаності та популяризація жестової мови;
- виклики та перспективи.

У контексті зростаючої уваги до прав людей з інвалідністю, інклюзивність стала ключовим аспектом у всіх сферах життя, від освіти до ринку праці. Організації, уряди та міжнародні інституції сприяють розробці політик, які забезпечують рівний доступ до інформації та комунікації для всіх громадян, включаючи тих, хто використовує жестову мову.

Жестова мова є важливою складовою цієї інклюзії, оскільки вона дозволяє людям з порушенням слуху інтегруватися в суспільство, отримувати інформацію на рівні з іншими громадянами та брати участь у соціальних, культурних та професійних взаємодіях.

Зростає попит на переклад жестовою мовою в таких сферах, як телебачення, медіа, соціальні мережі та інтернет-платформи. Все більше програм і новинних випусків забезпечуються перекладами на жестову мову, що робить інформацію доступною для ширшої аудиторії. Це підсилює медіа-простір, надаючи людям з порушенням слуху можливість бути в курсі актуальних подій і новин.

Крім того, в освіті все частіше використовують жестову мову не тільки у спеціалізованих навчальних закладах, але й в інклюзивних школах та університетах, що сприяє рівним можливостям для студентів з порушенням слуху.

Жестова мова також активно впроваджується в громадських та державних установах, банківських та комерційних організаціях для забезпечення доступності послуг і консультацій.

Технології розпізнавання жестової мови на основі алгоритмів машинного навчання відкривають нові можливості для автоматизації процесів комунікації. Впровадження систем розпізнавання жестів у реальному часі дає змогу розробляти інноваційні рішення для взаємодії людей з порушенням слуху з навколишнім середовищем. Це включає розробку програмного забезпечення для жестового перекладу, мобільних застосунків для навчання жестової мови та інтерактивних платформ для навчальних і наукових досліджень.

Машинне навчання дозволяє створювати програми, здатні адаптуватися до індивідуальних особливостей жестів різних користувачів. Це підвищує ефективність використання жестової мови в різних сценаріях, від освітніх програм до професійного спілкування.

Останніми роками спостерігається зростання інтересу до вивчення жестової мови серед людей, які не мають порушень слуху. Цей процес стимулюється активною соціальною та культурною пропагандою інклюзії та прав глухих людей. Культурні події, фільми та інші медіапроєкти активно

популяризують використання жестової мови, роблячи її частиною загальнодоступного контенту.

Крім того, з'являється все більше онлайн-ресурсів, курсів і мобільних застосунків, які дозволяють вивчати жестову мову, що сприяє поширенню її серед різних верств населення.

Хоча жестова мова набуває все більшого поширення, існують значні виклики, пов'язані з її універсальністю. Кожна країна має свою власну національну жестову мову, що створює бар'єри для міжнародного спілкування та навчання. Однак, розвиток міжнародної жестової мови (International Sign) дає змогу долати ці бар'єри на глобальних подіях і зустрічах.

Важливим завданням є подальший розвиток технологій автоматичного розпізнавання жестової мови, які поки що стикаються з технічними обмеженнями, такими як точність розпізнавання у складних умовах освітлення, різні індивідуальні особливості жестів та зміни у швидкості виконання рухів.

Актуальність жестової мови зумовлена її незамінною роллю у забезпеченні прав на комунікацію для людей з порушенням слуху та інклюзивною функцією в сучасному суспільстві. Разом з технологічним прогресом, можливості автоматичного розпізнавання жестів і їх використання у різних сферах відкривають нові перспективи для інклюзії, навчання та розвитку технологій.

1.2 Види жестових мов

Жестові мови в усьому світі є самобутніми і відображають культуру, історію та унікальність глухих спільнот у кожній країні. Деякі з цих мов стали відомими на міжнародному рівні завдяки своїй поширеності та впливу на розвиток жестової комунікації в інших країнах.

1.2.1 Американська жестова мова (ASL)

Американська жестова мова (ASL) є однією з найбільш поширених і визнаних жестових мов у світі, особливо у Сполучених Штатах та англомовних частинах Канади. Її історія сягає початку 19-го століття, коли американський викладач Томас Галлодет запросив французького педагога Лорана Клерка допомогти створити першу школу для глухих у США. Через це розвиток ASL зазнав значного впливу французької жестової мови (LSF).

ASL має свою власну граматичну структуру, яка суттєво відрізняється від англійської мови. Наприклад, порядок слів у реченні в ASL може бути різним і зазвичай слідує структурі «тема-коментар», де спочатку повідомляється основна ідея, а потім до неї додаються уточнення. У цій мові використовується дворуковий пальцевий алфавіт, за допомогою якого можна передавати імена, назви місць та інші слова, для яких немає окремих жестів.

ASL багата на ідіоми та вирази, а також широко використовує міміку, рухи обличчя та тіла для передачі емоцій, питань, заперечення тощо. Наприклад, зміна форми або швидкості жесту може додати контекст, на кшталт напруженості чи емоційності, що надає мові динамічності та гнучкості. ASL також має багато діалектів і регіональних варіацій, які залежать від місцевих культурних впливів. Сьогодні ASL офіційно визнана в багатьох штатах США та використовується в освіті, медіа та навіть в інтернаціональних комунікаціях між глухими.

1.2.2 Британська жестова мова (BSL)

Британська жестова мова (BSL) є основною жестовою мовою у Великобританії та має значні відмінності від американської жестової мови (ASL), хоча обидві країни використовують англійську як офіційну усну мову. BSL розвинулася незалежно від ASL і базується на власній історії та культурі британської спільноти глухих.

У BSL використовується одноручний пальцевий алфавіт, який є своєрідним візуальним відображенням англійських букв. Особливістю цієї мови є широкий спектр використання просторової граматики. Наприклад, місце розташування рук у просторі, їх рухи та форма є основними компонентами, що змінюють значення жесту. BSL також відрізняється використанням різних граматичних конструкцій, порядок слів у реченнях не обов’язково відповідає англійському, оскільки важливішу роль відіграють контекст та інтонація жесту.

BSL має свої діалекти та регіональні відмінності, які пов'язані з культурною історією різних регіонів Великобританії. Знання BSL стало все більш поширеним серед чуючих людей, особливо завдяки зусиллям освітніх програм та телебачення. У Великобританії BSL офіційно визнана як окрема мова, що дозволяє глухим отримувати кращу підтримку та доступ до інформації.

1.2.3 Французька жестова мова (LSF)

Французька жестова мова (LSF) є однією з найстаріших жестових мов у світі та відіграла значну роль у становленні інших жестових мов, зокрема ASL. Її історія починається з XVIII століття, коли абат Шарль-Мішель де Л'Епе розробив систему жестів для навчання глухих дітей. Завдяки цьому LSF стала офіційною мовою навчання в школах для глухих у Франції.

LSF має унікальну граматичну структуру, яка суттєво відрізняється від французької усної мови. Основною її особливістю є використання просторової граматики та міміки для передачі значення, питань, емоцій та граматичних категорій, таких як час і кількість. Наприклад, LSF широко використовує простір для розміщення об’єктів або дій, щоб позначити зв’язки між ними. Пальцевий алфавіт LSF є дворуковим і слугує для передачі імен та спеціальних термінів.

У Франції LSF офіційно визнана як самостійна мова і використовується в освіті, культурі, медіа та на урядовому рівні. Як і інші жестові мови, LSF має свої діалекти, що залежить від регіону, в якому вона використовується. Сьогодні LSF залишається важливою частиною французької культури та ідентичності глухої спільноти.

1.2.4 Японська жестова мова (JSL)

Японська жестова мова (JSL) є офіційною мовою спілкування глухих у Японії. Її розвиток почався у XIX столітті, коли були створені перші школи для глухих у країні. З того часу JSL перетворилася на складну і самобутню мову, яка має свої унікальні риси.

JSL використовує систему жестів, що пов’язані з японською писемністю, а саме хіраганою та катаканою, для передачі звуків та літер. Однією з ключових особливостей JSL є акцент на рухах пальців, положенні рук та їх взаємодії з тілом. Граматика JSL значною мірою базується на японській культурі, і мова широко використовує міміку та зміни у виразах обличчя для передачі емоцій, питань та підтверджень. Це робить мову надзвичайно виразною та багатою на нюанси.

Японська жестова мова також має безліч діалектів, що відображають культурні відмінності різних регіонів Японії. У країні JSL визнана як самостійна мова, і існують активні програми підтримки та популяризації жестової комунікації, які включають навчання чуючих людей основам JSL.

1.2.5 Українська жестова мова (УЖМ)

Українська жестова мова (УЖМ) є основною мовою спілкування глухих в Україні. Її історія починається з моменту створення перших навчальних закладів для глухих у XIX столітті. Протягом багатьох років УЖМ розвивалася незалежно від інших жестових мов, набуваючи унікального словникового запасу, який тісно пов'язаний з українською культурою та історією.

Українська жестова мова має власну граматику, структуру речень та словниковий запас, що відрізняється від української усної мови. Вона використовує просторову граматику, де положення та рух рук у просторі відіграють вирішальну роль у зміні значення жесту. Міміка та вирази обличчя також є важливими елементами мови, передаючи емоції та граматичні відтінки. Порядок слів у реченнях УЖМ зазвичай слідує схемі «тема-коментар», яка дозволяє виділяти ключову інформацію на початку висловлювання.

Пальцевий алфавіт УЖМ є одноручним, дозволяючи передавати власні імена, назви міст, спеціальні терміни та інші слова, для яких немає окремого жесту. Цей алфавіт відрізняється від тих, що використовуються в інших жестових мовах, оскільки він тісно пов’язаний з українськими звуками. Щодо поширеності, українська жестова мова була офіційно визнана на законодавчому рівні у 2009 році. Це визнання забезпечило глухим людям право користуватися рідною мовою у всіх сферах суспільного життя. Сьогодні УЖМ активно використовується у повсякденному житті, освіті, медіа та навіть на державному рівні.

В Україні діє багато спеціалізованих навчальних закладів, де УЖМ є основним засобом навчання для дітей з вадами слуху. Існують також громадські організації, які займаються популяризацією та розвитком жестової мови, а також проводять курси для всіх, хто бажає її вивчити. Це особливо важливо для родичів глухих, перекладачів та інших зацікавлених осіб.

УЖМ має свої діалекти та регіональні особливості, які розвивалися в різних областях України. Наприклад, жести, які використовуються у східних і західних регіонах, можуть мати певні відмінності, хоча загалом носії різних діалектів можуть легко розуміти один одного. Сьогодні українська жестова мова є невід'ємною частиною суспільного життя. Вона широко використовується в медіа, особливо на телебаченні, де передбачено переклади для глухих і слабочуючих. У державних установах та на різноманітних заходах

все частіше присутні сурдоперекладачі, що сприяє більшій інтеграції глухих людей у суспільство.

1.2.6 Міжнародна жестова мова (International Sign, IS)

Міжнародна жестова мова (IS) є своєрідною спрощеною системою жестів, яка використовується під час міжнародних зустрічей глухих людей, таких як конференції, спортивні змагання та фестивалі. IS не є повноцінною мовою у класичному розумінні, оскільки не має складної граматики та структури, характерної для національних жестових мов.

IS базується на наборі найбільш зрозумілих та універсальних жестів, узятих з різних жестових мов. Вона створена для забезпечення основного спілкування між людьми, які не знають жестової мови одне одного. Для цього IS використовує спрощені жести, міміку та інтуїтивні рухи, які можна швидко зрозуміти. Хоча вона не може передати всю глибину думок і емоцій, яку можуть забезпечити національні жестові мови, IS виконує важливу роль у побудові комунікаційних мостів між представниками різних культур глухих.

Таким чином, ці жестові мови є не лише засобами спілкування, але й культурними явищами, що відображають унікальну історію та ідентичність спільнот, які їх використовують.

1.3 Аналіз аналогічного програмного забезпечення

Точність розпізнавання жестової мови значно залежить від використовуваних технологій (глибинні камери, сенсори руху, алгоритми комп'ютерного зору). Важливо враховувати середовище використання, оскільки умови освітлення, ракурси та швидкість жестів можуть впливати на ефективність системи.

Спеціалізоване обладнання дозволяє досягати високої точності, але робить системи менш доступними через їх високу вартість та складність у використанні.

Мобільні рішення мають переваги через доступність і простоту використання, але часто стикаються з обмеженнями у точності та наборі підтримуваних жестів.

Інтерактивність та можливість навчання користувачів є важливою складовою програмного забезпечення. Багато систем пропонують навчальні компоненти, що дозволяють користувачам покращувати свої навички жестової мови.

Виділення основних характеристики існуючого ПЗ:

Таблиця 1.1 – Аналіз HandTalk

Назва	HandTalk[1]
Розробник	Hand Talk Translator
Архітектура	iOS application та Android application
Мова реалізації	Java (Android), Swift (iOS)
Функції/Характеристики	Використання камери для розпізнавання жестів. Інтерактивні уроки для вивчення жестової мови. Підтримка текстового чату з перекладом жестової мови.
Переваги	Підтримка кількох мов жестів. Переклад в реальному часі. Інтуїтивно зрозумілий інтерфейс.
Недоліки	Може вимагати стабільного інтернет-з’єднання для роботи. Обмежена підтримка складних жестів. функціоналу за потреби користувача.

Таблиця 1.2 – Аналіз Pocket Sign

Назва	Pocket Sign[2]
Розробник	MobiReactor
Архітектура	iOS application та Android application

Кінець таблиці 1.2

Мова реалізації	Java (Android), Swift (iOS)
Функції/Характеристики	Сотні відеоуроків з американської жестової мови (ASL). Інтерактивні питання та словник. Підтримка навчання жестової мови для дітей. Відстеження прогресу користувача. Гейміфікація для мотивації користувачів. Підтримка офлайн режиму для доступу до уроків без інтернету.
Переваги	Велика кількість відеоуроків. Інтерактивні питання для перевірки знань. Підтримка навчання жестової мови для дітей.
Недоліки	Підтримка лише американської жестової мови (ASL). Може вимагати підписки для доступу до всіх функцій.

Таблиця 1.3 – Аналіз Lingvano

Назва	Lingvano[3]
Розробник	Lingvano GmbH
Архітектура	iOS application та Android application
Мова реалізації	JavaScript (web), Java (Android), Swift (iOS)
Функції/Характеристики	Інтерактивні уроки з американської жестової мови (ASL). Дзеркало для перевірки правильності виконання жестів. Словник жестової мови. Відстеження прогресу користувача. Інтерактивні вправи для закріплення знань. Підтримка офлайн режиму для доступу до уроків без інтернету.

Кінець таблиці 1.3

Переваги	Уроки створені глухими викладачами. Дзеркало для перевірки правильності жестів. Підтримка навчання в будь-який час і в будь-якому місці.
Недоліки	Підтримка лише американської жестової мови (ASL). Може вимагати підписки для доступу до всіх функцій.

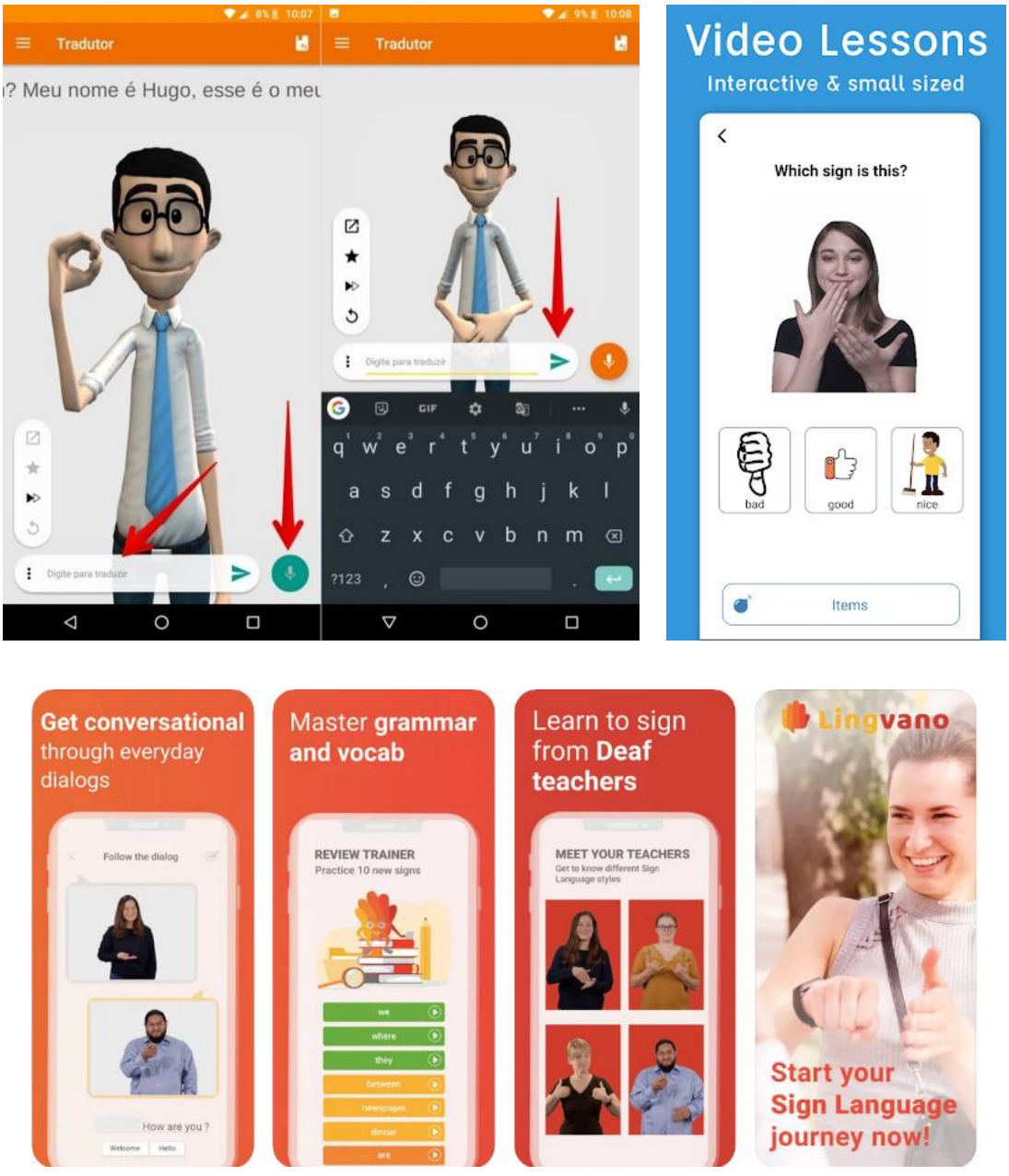


Рисунок 1.1 – Вигляд аналогічного ПЗ

Аналіз показує, що сучасні системи для розпізнавання жестової мови продовжують розвиватися, і технологічні інновації дозволяють удосконалювати точність та адаптивність таких рішень. Водночас, залишається відкритим питання доступності та підтримки широкого спектру жестових мов.

1.4 Специфікація вимог до програмного забезпечення

Призначення та межі проєкту:

– призначення системи (застосунку) – розробка мобільного застосунку для розпізнавання жестової мови за допомогою камери смартфона та перекладу її в текст у реальному часі. Застосунок призначений для полегшення спілкування між людьми, які використовують жестову мову, та тими, хто її не знає;

– погодження, що ухвалені в програмній документації – використання бібліотек машинного навчання та комп’ютерного зору для розпізнавання жестів. Основні компоненти застосунку будуть реалізовані на Python;

– межі проєкту ПЗ – крайня дата завершення роботи над ПЗ – 01.11.2024р.

Загальний опис:

– сфера застосування – застосунок націлений на використання користувачами для перекладу жестової мови в текст у реальному часі, що полегшить спілкування між людьми з порушенням слуху та іншими;

– характеристики користувачів – користувачі повинні мати базові навички роботи зі смартфоном. Застосунок буде корисним для людей з порушенням слуху, їхніх родичів, друзів та професіоналів, які працюють з такими людьми;

– загальна структура і склад системи – основні частини програмного забезпечення: модуль розпізнавання жестів, модуль перекладу жестів у текст, інтерфейс користувача;

– загальні обмеження – програмне забезпечення має працювати на операційних системах Android та iOS.

Функція системи «Розпізнавання жестової мови»:

– опис функції – функція розпізнавання жестової мови допомагає користувачу перекладати жести в текст у реальному часі за допомогою камери смартфона;

– функціональні вимоги – використання камери смартфона для захоплення жестів, розпізнавання жестів за допомогою алгоритмів машинного навчання, переклад розпізнаних жестів у текст, відображення перекладеного тексту на екрані смартфона.

Вимоги до інформаційного забезпечення:

– джерела і зміст вхідної інформації (даних) – основним джерелом вхідної інформації є камера смартфона, яка захоплює жести користувача;

– вимоги до способів організації, збереження та ведення інформації: усі дані зберігаються локально на пристрої користувача, можливість збереження історії перекладів.

Вимоги до технічного забезпечення:

– вимоги до технічного забезпечення – ОС: Android 8.0 або новіша, iOS 12.0 або новіша;

– процесор – ARM Cortex-A53 або новіший;

– оперативна пам’ять – 3 GB ОП або більше;

– камера – мінімальна роздільна здатність 720p.

Вимоги до програмного забезпечення:

– архітектура програмної системи – складається з клієнтської частини (мобільний застосунок);

– системне програмне забезпечення – використання бібліотек машинного навчання YOLOv5 та PyTorch;

– мова програмування – Python для основних алгоритмів.

Вимоги до зовнішніх інтерфейсів:

– інтерфейс користувача – має задовольняти усі вимоги дизайну, бути зручним у використанні, підтримувати кілька мов інтерфейсу;

– апаратний інтерфейс – смартфон користувача з операційною системою Android або iOS;

– програмний інтерфейс – API для взаємодії з камерою та бібліотеками машинного навчання.

Властивості програмного забезпечення:

– доступність – ПЗ доступне для будь-якого користувача, за умови наявності у користувача смартфона з відповідними технічними характеристиками;

– переносимість – програмне забезпечення може працювати на ОС Android та iOS;

– продуктивність – продуктивність роботи ПЗ залежить від характеристик смартфона.

Інші вимоги:

– усі вимоги сформовано та описано вище, доповнення не вимагається.

Висновки до розділу 1

У першому розділі проаналізовано роль жестової мови у забезпеченні комунікації для людей з порушенням слуху, підкреслено її значення в умовах зростаючої інклюзивності суспільства. Жестова мова є повноцінною системою з власною граматикою і синтаксисом, що дозволяє її користувачам активно брати участь у суспільному та професійному житті.

Розвиток технологій, таких як машинне навчання та комп'ютерний зір, відкриває нові можливості для автоматизації розпізнавання жестів. Це дозволяє створювати програмне забезпечення, яке здійснює переклад жестової мови в текст у реальному часі та навчання нових жестів через інтерактивні платформи, що спрощує комунікацію між людьми з порушенням слуху та тими, хто не володіє жестовою мовою.

Аналіз існуючого програмного забезпечення виявив як ефективні рішення, так і їх обмеження, такі як залежність від інтернету та обмежена підтримка мов жестів. Проблеми універсалізації жестової мови ускладнюють міжнародну комунікацію, однак міжнародна жестова мова допомагає подолати ці бар'єри. Розвиток мобільних систем розпізнавання жестової мови є важливим для підвищення доступності і соціальної інклюзії.

2 РОЗРОБКА АРХІТЕКТУРИ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Розробка архітектури ПЗ (програмного забезпечення) - це процес створення структурної організації системи, що включає визначення основних компонентів, їх функцій, взаємодії між ними та інші аспекти системи. Архітектура ПЗ є фундаментом для розробки програмного продукту, що відображає рішення про технічні та організаційні аспекти системи[4].

Основні цілі розробки архітектури ПЗ включають:

– визначення структури та організації програмного забезпечення, що включає розділення на компоненти, модулі та підсистеми;

– встановлення комунікації та взаємодії між різними компонентами системи, забезпечення інтеграції та співпраці між ними;

– опис принципів, шаблонів та практик, які будуть використані під час розробки програмного забезпечення;

– забезпечення масштабованості, гнучкості, розширюваності та стійкості системи, а також можливості адаптації до змін технологій та вимог замовника;

– оцінка технічних рішень та їх відповідності вимогам та обмеженням проєкту, врахування вимог щодо продуктивності, безпеки та надійності системи.

2.1 Сценарії використання системи

Таблиця 2.1 – Короткий Usecase

Usecase section	Comment
Use Case Name	Розпізнавання жестів у текст
Scope	System
Level	User-goal

Кінець таблиці 2.1

Primary Actor	Користувач
Stakeholders and interests	– користувач – переклад жестів у текст за допомогою камери смартфон; – система – коректно розпізнати жести та вивести текст.
Preconditions	Користувач має доступ до камери смартфона.
Success guarantee	Жести користувача коректно перекладені в текст.
Main Success Scenario	1) користувач виконує жест перед камерою; 2) система розпізнає жест; 3) система виводить текстовий переклад жесту на екран.
Extensions	Немає.
Special Requirements	Немає.
Technology and Data Variations List	Система використовує алгоритми машинного навчання для розпізнавання.
Frequency of Occurrence	Кожного разу, коли користувач виконує жест.
Miscellaneous	Немає.

Короткий Usecase описує ідеальний сценарій, коли користувач виконує жест, і система безпомилково розпізнає його та перекладає у текст. Ніяких помилок чи варіантів обробки помилок у цьому випадку не передбачається.

Таблиця 2.2 – Поверхневий Usecase

Usecase section	Comment
Use Case Name	Розпізнавання жестів у текст з обробкою помилок

Кінець таблиці 2.2

Scope	System
Level	User-goal
Primary Actor	Користувач
Stakeholders and interests	– користувач – переклад жестів у текст, навіть якщо виникають помилки; – система – розпізнати жести та обробити можливі помилки при введенні.
Preconditions	Користувач має доступ до камери смартфона.
Success guarantee	Жести користувача успішно перекладені у текст або користувач повторює жест до коректного розпізнавання.
Main Success Scenario	1) користувач виконує жест перед камерою; 2) система розпізнає жест; 3) система виводить текстовий переклад жесту на екран.
Extensions	2a) якщо жест не розпізнано, система просить користувача повторити жест; 2b) якщо система не може розпізнати жест, вона надає підказку для кращого виконання руху.
Special Requirements	Підтримка кількох мов жестової мови.
Technology and Data Variations List	Система використовує машинне навчання та базу даних для аналізу жестів.
Frequency of Occurrence	Кожного разу, коли користувач виконує жест.
Miscellaneous	Немає.

У поверхневому Usecase додано варіанти обробки помилок. Якщо жест не розпізнано, система надає можливість повторити жест або дає підказки для кращого виконання. Такий сценарій забезпечує більше варіантів взаємодії користувача із системою.

Таблиця 2.3 – Повний Usecase

Usecase section	Comment
Use Case Name	Повний процес розпізнавання жестів у текст
Scope	System
Level	User-goal
Primary Actor	Користувач
Stakeholders and interests	– користувач – переклад жестів у текст, з можливістю виправлення помилок і збереження результатів; – система – надання повного процесу розпізнавання жестів, з можливістю взаємодії та обробки помилок.
Preconditions	Користувач має доступ до камери смартфона і вибрав необхідну мову жестової мови.
Success guarantee	Жести користувача коректно перекладені у текст, користувач може зберегти результат.
Main Success Scenario	1) користувач вибирає мову жестової мови; 2) користувач виконує жест перед камерою; 3) система розпізнає жест; 4) система виводить текстовий переклад жесту на екран; 5) користувач може підтвердити або відхилити результат перекладу; 6) користувач може зберегти перекладений текст у файл або відправити його через месенджер.
Extensions	3a) якщо жест не розпізнано, система просить користувача повторити жест; 3b) якщо система не може розпізнати жест, вона пропонує інструкції для точнішого виконання; 5a) якщо користувач не згоден із перекладом, він може вибрати варіант із кількох можливих перекладів;

Кінець таблиці 2.3

	6а) чкщо користувач не зберіг результат, система нагадає про це.
Special Requirements	Можливість вибору кількох мов жестової мови, збереження результатів.
Technology and Data Variations List	Система використовує алгоритми машинного навчання, базу даних жестів та має можливість експортувати результати у різні формати.
Frequency of Occurrence	Кожного разу, коли користувач виконує жест.
Miscellaneous	Користувач має можливість експортувати результат перекладу або поділитися ним.

У повний Usecase представляє повний цикл розпізнавання жестів, включаючи варіанти обробки помилок, можливість збереження та експорту результатів. Користувач має контроль над процесом, може підтвердити чи відхилити результат, а також вибрати кілька мов для жестів.

2.2 Діаграма прецедентів

Діаграма використання (Use Case Diagram) – це тип діаграми в UML (Unified Modeling Language), яка використовується для опису функціональних вимог до системи з точки зору її взаємодії з користувачами та іншими системами. Вона ілюструє:

1) акторів – особи, системи або інші компоненти, які взаємодіють із системою. Актори можуть бути кінцевими користувачами (людьми) або іншими системами, які взаємодіють з досліджуваною системою. Актор (Actor) – представлений у вигляді фігури або значка (зазвичай людини або персонажа);

2) випадки використання (Use Cases) – описують специфічні функції або серії дій, які система виконує для задоволення потреб акторів. Випадки використання визначають, що система повинна робити, а не як це реалізується. Випадок використання (Use Case) – представлений у вигляді овалу, що містить назву функції або серії дій;

3) зв’язки – включають асоціації, залежності та включення між акторами та випадками використання. Зв'язки – лінії, що з'єднують акторів з випадками використання, або між випадками використання, що показують їхні взаємодії і залежності. Основні типи зв'язків:

– асоціація (Association) – вказує на взаємодію між актором і випадком використання;

– залежність (Dependency) – вказує, що один випадок використання залежить від іншого;

– включення (Include) – вказує, що один випадок використання завжди включає в себе інший;

– розширення (Extend) – вказує, що один випадок використання може бути розширений додатковим функціоналом в залежності від контексту.

Застосування:

– визначення функціональних вимог – визначення та документування потреб користувачів та можливостей системи;

– комунікація – допомагає розробникам, замовникам і користувачам зрозуміти функціональність системи і взаємодії;

– проєктування системи – планування архітектури системи на основі вимог, визначених у випадках використання.

Діаграма використання є важливим інструментом для візуалізації і аналізу функціональності системи з точки зору користувача, що допомагає забезпечити правильність розробки та відповідність вимогам.

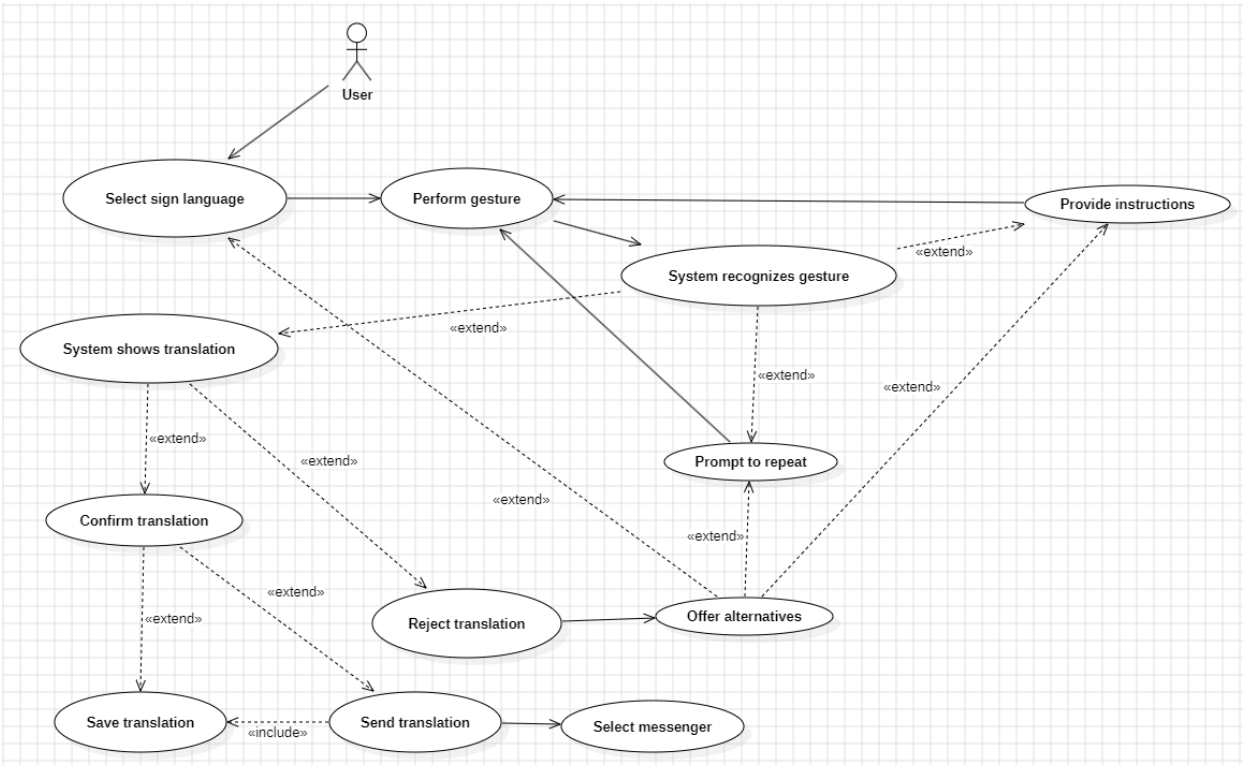


Рисунок 2.1 – Діаграма прецедентів (варіантів використання)

Діаграма використання демонструє, як користувач може взаємодіяти із системою розпізнавання жестової мови, а саме користувач вибирає мову жестів, після чого показує жест і система намагається визначити жест, який показує користувач. Якщо система не змогла визначити жест то вона пропонує показати його знову, якщо друга спроба невдала система пропонує інструкції для вирішення даної проблеми, коли система визначила усі жести, пропонується користувачу підтвердити або відмінити переклад і почати спочатку. Якщо користувач вибрав підтвердити, то система пропонує зберегти або поділитися перекладом у соцмережах, коли користувач вибирає поділитися, тоді система автоматично зберігає переклад у запропонованому форматі.

2.3 Діаграма послідовності

Діаграма послідовності (Sequence Diagram) – це тип діаграми в UML (Unified Modeling Language), який використовується для опису динаміки взаємодії між об'єктами в межах системи протягом певного сценарію. Вона ілюструє:

1) об'єкти – компоненти системи або інші учасники, які взаємодіють один з одним. Об'єкти можуть бути частинами системи або зовнішніми системами, що впливають на сценарій. Об'єкти (Objects) – представлені як прямокутники, що позначають різні компоненти системи або учасників;

2) повідомлення – комунікаційні сигнали, які об'єкти надсилають один одному, щоб досягти певної мети або виконати завдання. Повідомлення можуть включати запити, відповіді та інші дії. Повідомлення (Messages) – зображені у вигляді стрілок, що вказують напрямок комунікації між об'єктами і порядок їх виконання;

3) послідовність – порядок, у якому відбувається обмін повідомленнями між об'єктами, що визначає хід сценарію. Часові лінії (Lifelines) – вертикальні лінії, що представляють життєвий цикл об'єкта під час взаємодії.

Застосування:

– опис динаміки системи – допомагає зрозуміти, як об'єкти взаємодіють для досягнення конкретного результату або виконання завдання;

– дозволяє виявити можливі проблеми або вузькі місця в сценарії взаємодії;

– проєктування та документування – забезпечує чітке розуміння роботи системи, що допомагає в розробці і тестуванні.

Діаграма послідовності є важливим інструментом для візуалізації процесів взаємодії в системі, що допомагає забезпечити правильність реалізації та відповідність вимогам.

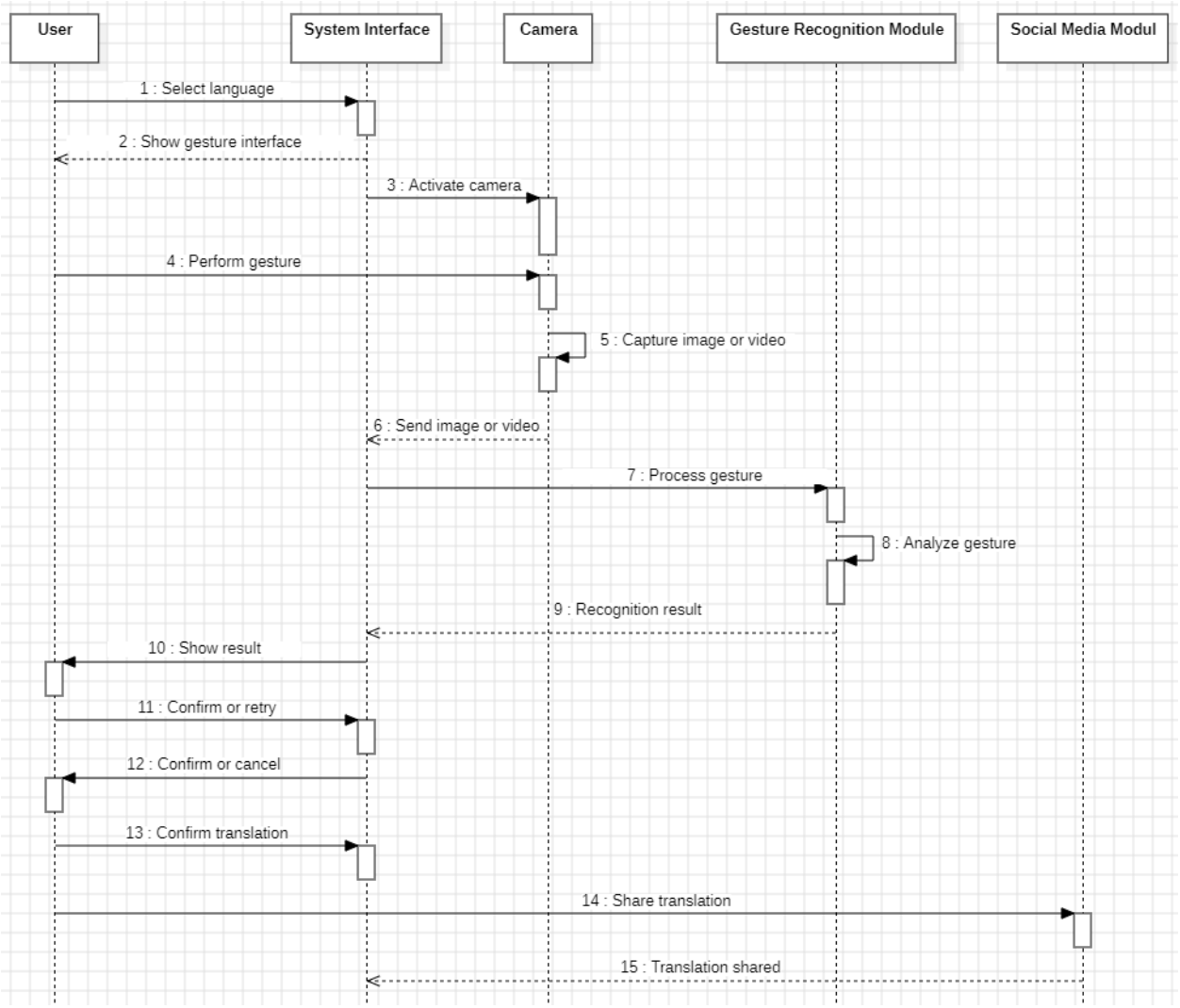


Рисунок 2.2 – Діаграма послідовності

Діаграма послідовності системи розпізнавання жестів демонструє, як користувач може взаємодіяти з системою для перекладу жестової мови. Спершу користувач вибирає бажану мову жестів, після чого показує жест через камеру. Система аналізує цей жест і намагається його розпізнати. Якщо жест не розпізнаний, система пропонує користувачу повторити його. У випадку, якщо друга спроба також невдала, система надає інструкції для правильного виконання жесту. Коли всі жести успішно розпізнані, система пропонує користувачу підтвердити або скасувати переклад. Якщо користувач підтверджує переклад, система пропонує зберегти його або поділитися результатом у соціальних мережах. У випадку, коли вибирається опція

'поділитися', система автоматично зберігає переклад у запропонованому форматі та публікує його в соцмережах."

2.4 Діаграма кооперації

Діаграма кооперації (Collaboration Diagram), також відома як діаграма співробітництва, є типом діаграми в UML (Unified Modeling Language), яка використовується для моделювання взаємодії між об'єктами в системі. Вона фокусується на структурних аспектах взаємодії, показуючи, як об'єкти співпрацюють між собою для виконання конкретного завдання або сценарію.

Основні елементи діаграми кооперації:

1) об'єкти – учасники взаємодії, представлені прямокутниками, що символізують компоненти системи або акторів, які взаємодіють для досягнення певної мети;

2) посилання між об'єктами – лінії, що з'єднують об'єкти і показують відносини або асоціації між ними;

3) повідомлення – нумеровані стрілки, які відображають обмін даними або сигналами між об'єктами, що вказують послідовність і характер взаємодії.

На відміну від діаграми послідовностей, яка акцентує увагу на часовій послідовності повідомлень, діаграма кооперації більше фокусується на організації об'єктів та їх зв'язках під час взаємодії.

Застосування:

– моделювання динамічної взаємодії – показує, як об'єкти співпрацюють, щоб виконати певний сценарій;

– аналіз і проєктування – допомагає зрозуміти структуру системи та взаємодії між її частинами;

– документування системи – використовується для пояснення, як різні компоненти системи взаємодіють один з одним на структурному рівні.

Діаграма кооперації є корисним інструментом для аналізу системи, допомагаючи зрозуміти взаємодію об'єктів та їхній вплив один на одного в процесі виконання завдань.

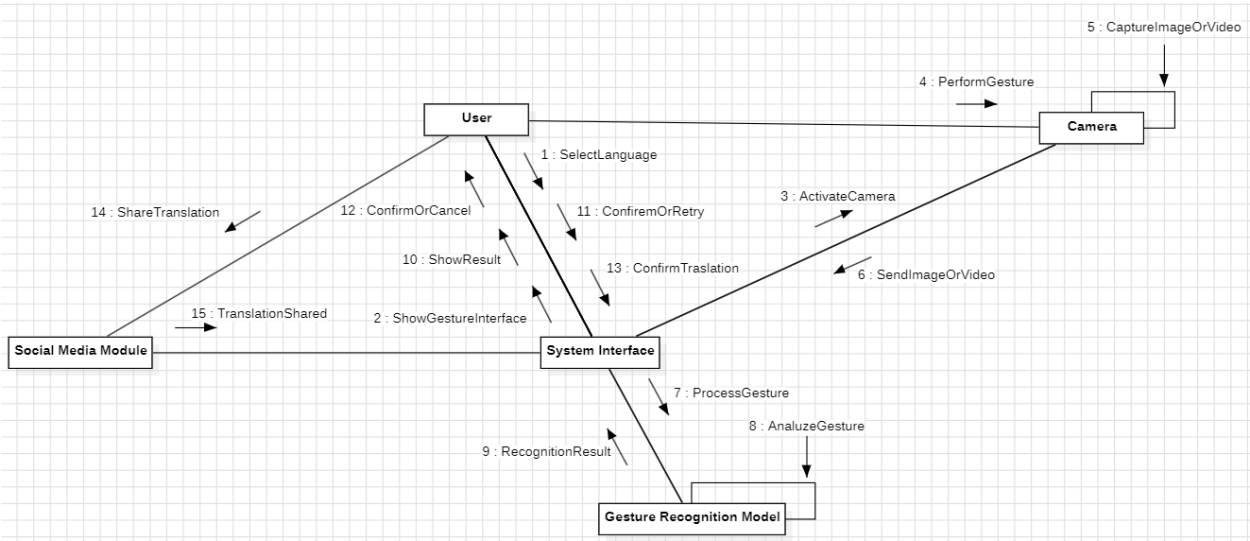


Рисунок 2.3 – Діаграма кооперації

Діаграма кооперації для системи розпізнавання жестів демонструє, як різні об'єкти системи співпрацюють для виконання завдання перекладу жестової мови. Спочатку користувач вибирає бажану мову жестів, що активує взаємодію між об'єктом користувача та інтерфейсом системи. Далі користувач показує жест через камеру, і камера надсилає відеодані до системи розпізнавання. Система обробляє ці дані та намагається розпізнати жест, співпрацюючи з модулем розпізнавання жестів. Якщо жест не розпізнано, система надсилає повідомлення користувачу з проханням повторити жест, а модуль інтерфейсу фіксує це повідомлення. У випадку невдалої другої спроби, модуль системи надає інструкції користувачу для правильного виконання жесту. Коли всі жести успішно розпізнані, система передає запит користувачу підтвердити або скасувати переклад. Якщо користувач підтверджує, система взаємодіє з модулем збереження та пропонує варіанти збереження або публікації перекладу в соціальних мережах. У разі вибору опції "поділитися", система автоматично зберігає переклад та співпрацює з модулем соціальних мереж для публікації результату.

2.5 Діаграма станів та переходів

Діаграма станів та переходів (State Transition Diagram) – це тип діаграми в UML (Unified Modeling Language), який використовується для опису динаміки системи, зокрема, того, як об'єкти або системи змінюють свій стан у відповідь на події. Вона ілюструє:

1) стан (State) – це певний момент або умова в життєвому циклі об'єкта чи системи, в якому об'єкт має певні властивості або поведінку. Стан може бути активним або неактивним залежно від умов. Стан (State) – представлений як прямокутник з заокругленими кутами, що позначає різні умови чи етапи в життєвому циклі об'єкта;

2) перехід (Transition) – це зміна з одного стану в інший, що відбувається у відповідь на певну подію або умову. Перехід визначає, як система реагує на зовнішні або внутрішні впливи, що призводять до зміни стану. Перехід (Transition) – зображений у вигляді стрілок, що з'єднують стани і вказують напрямок зміни;

3) події (Events) – це дії або умови, які ініціюють перехід між станами. Події можуть бути сигналами, запитами або змінами в середовищі системи. Події (Events) – текст біля стрілок, що описує умови або сигнали, які ініціюють перехід між станами.

Застосування:

– опис динаміки системи – допомагає зрозуміти, як система змінюється з часом і як реагує на різні події;

– аналіз процесів – дозволяє виявити можливі стани системи і визначити, як переходи між цими станами впливають на загальну функціональність;

– проєктування та документування – забезпечує чітке розуміння поведінки системи в різних умовах, що допомагає в розробці та тестуванні.

Діаграма станів та переходів є важливим інструментом для візуалізації поведінки системи в залежності від змін у її станах, що допомагає забезпечити коректність реалізації та відповідність вимогам.

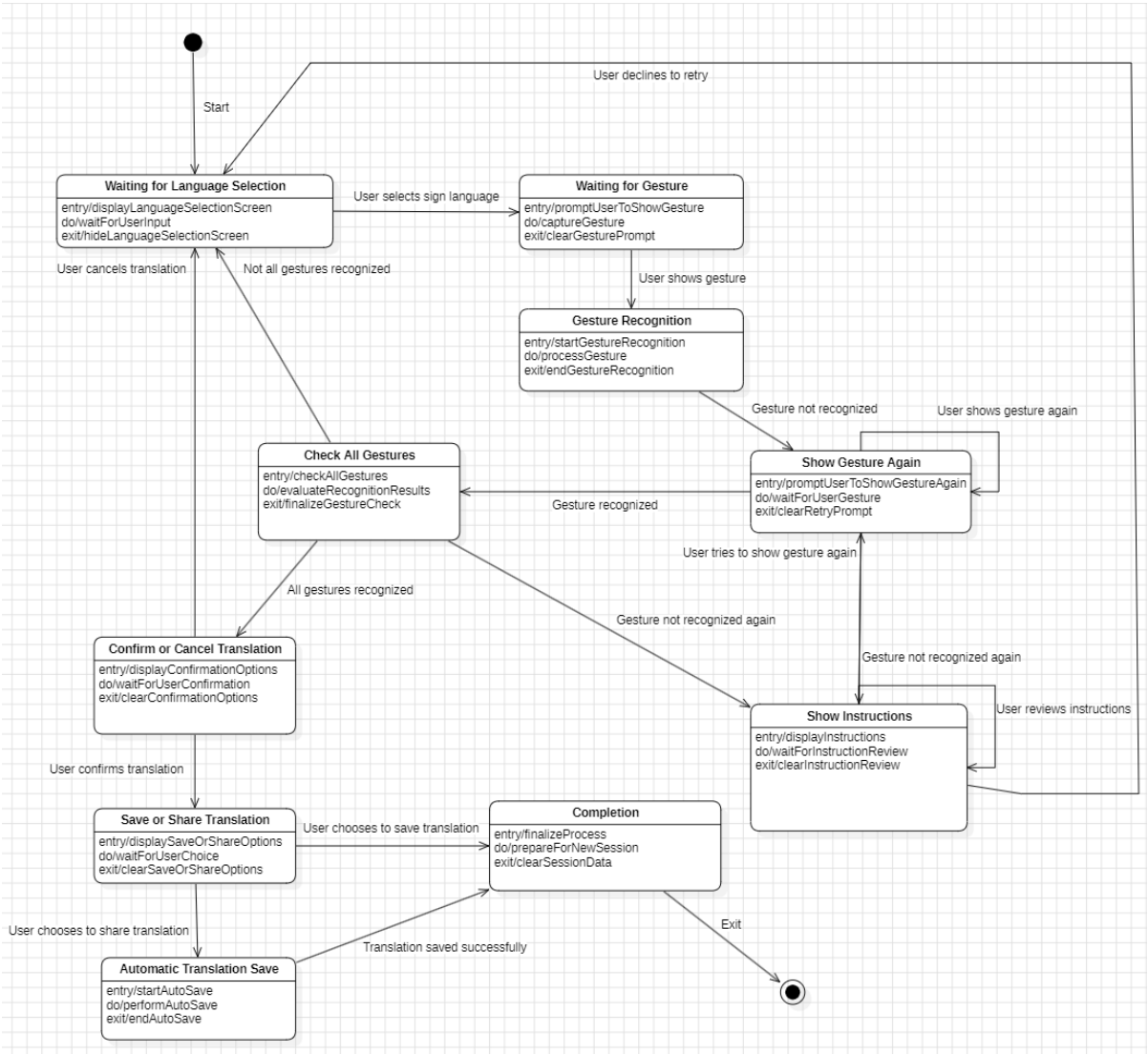


Рисунок 2.4 – Діаграма станів та переходів

Діаграма станів та переходів демонструє, як користувач взаємодіє із системою розпізнавання жестової мови. Процес починається з того, що користувач вибирає мову жестів. Після цього система переходить до стану очікування жесту, де користувач показує жест, і система намагається його розпізнати.

Якщо жест розпізнано, система переходить до перевірки всіх жестів. Якщо жест не розпізнано, система запрошує користувача показати жест

повторно. Якщо під час повторної спроби жест також не розпізнано, система пропонує переглянути інструкції для вирішення проблеми.

Після перегляду інструкцій користувач може спробувати показати жест знову або відмовитися від спроб. У випадку повторної спроби система повертається до етапу показу жесту знову. Якщо користувач відмовляється, система повертається до етапу вибору мови жестів.

Коли всі жести розпізнано, користувач має можливість підтвердити або скасувати переклад. Якщо користувач підтверджує переклад, система переходить до етапу збереження або поділу перекладу. Якщо користувач обирає поділитися перекладом, система автоматично його зберігає. Якщо користувач вибирає збереження, система завершує процес.

На завершення, система закінчує поточний сеанс і дозволяє користувачу вибрати нову мову жестів або завершити роботу із системою.

2.6 Діаграма діяльності

Діаграма діяльності (Activity Diagram) – це тип діаграми в UML (Unified Modeling Language), який використовується для опису послідовності дій або потоків роботи в системі, зокрема того, як виконуються процеси або дії. Вона ілюструє:

1) дія (Action) – це окремий крок або операція, яку виконує система або користувач. Дія може бути простим завданням або комплексною операцією, що відображає функціональність системи. Дія (Action) – представлена прямокутником із заокругленими кутами, що позначає операції, які виконуються системою або користувачем;

2) потік управління (Control Flow) – це стрілки, що з'єднують дії, і вказують на послідовність виконання операцій. Потік управління визначає порядок, в якому дії виконуються. Потік управління (Control Flow) – зображений у вигляді стрілок, що вказують напрямок і порядок виконання дій;

3) рішення (Decision) – це точка в процесі, де необхідно зробити вибір на основі певної умови. Використовується для моделювання альтернативних шляхів виконання дій. Рішення (Decision) – ромб, що представляє точку вибору з кількох альтернативних шляхів;

4) об'єднання (Merge) – це місце, де альтернативні потоки дій зливаються назад в один загальний потік;

5) розвилка (Fork) – використовується для одночасного виконання кількох потоків дій;

6) злиття (Join) – точка, де паралельні потоки з'єднуються, завершуючи паралельне виконання.

Застосування:

– моделювання процесів – допомагає зрозуміти послідовність виконання дій в системі або бізнес-процесі;

– аналіз алгоритмів – дозволяє виявити кроки процесу та визначити, як вони взаємодіють між собою;

– проєктування та документування – забезпечує чітке розуміння процесів і їх виконання, що допомагає в розробці, тестуванні та оптимізації системи.

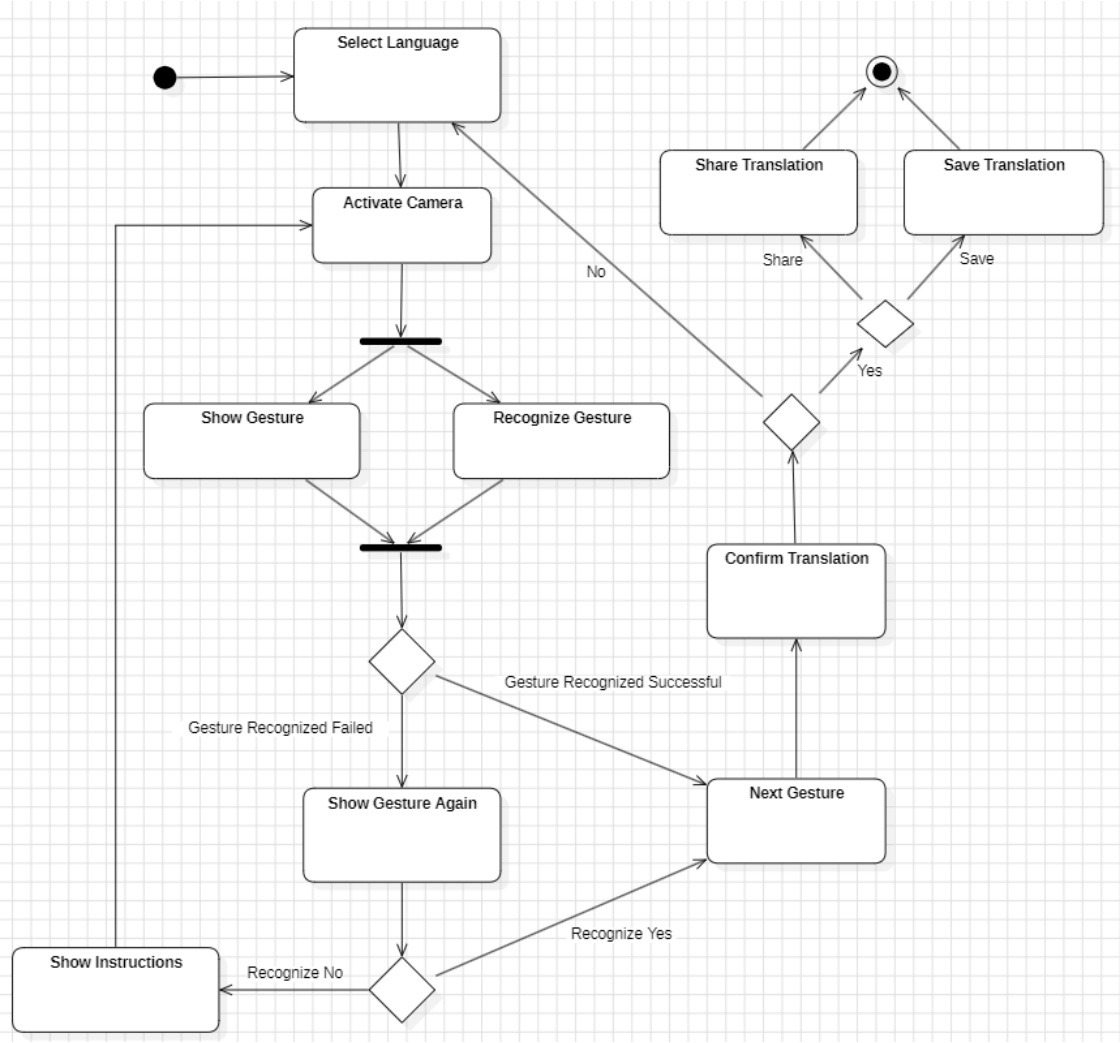


Рисунок 2.5 – Діаграма діяльності

Діаграма діяльності демонструє, як користувач може взаємодіяти із системою розпізнавання жестової мови. Спочатку користувач вибирає мову жестів, після чого система активує камеру для захоплення жестів. Користувач показує жест, і система намагається його розпізнати. Якщо система не змогла розпізнати жест, вона пропонує користувачу показати його знову. Якщо після другої спроби жест залишається нерозпізнаним, система пропонує інструкції для вирішення цієї проблеми. Після розпізнавання всіх жестів користувач може підтвердити або відмінити переклад і почати спочатку. Якщо користувач підтверджує переклад, система пропонує зберегти переклад або поділитися ним у соціальних мережах. Якщо користувач вибирає поділитися, система автоматично зберігає переклад у запропонованому форматі.

2.7 Mockup інтерфейсу користувача

Mockup інтерфейсу – це детальне графічне зображення, яке демонструє кінцевий вигляд майбутнього користувацького інтерфейсу в статичній формі. Він відображає, як будуть розташовані всі елементи інтерфейсу, такі як кнопки, меню, поля введення, іконки, текстові поля та інші інтерактивні компоненти. На відміну від прототипу, mockup не є інтерактивним або функціональним, але його головна мета – показати візуальну складову інтерфейсу в максимально наближеному до фінального вигляду[5].

Mockup дозволяє побачити кольорову палітру, шрифти, стилі, розміри елементів і їх розташування на екрані. Він використовується на етапі розробки для узгодження дизайну з командою та замовником, а також для внесення коректив до того, як буде створений функціональний продукт. Це інструмент, який дає можливість оцінити естетичні якості інтерфейсу та зручність його використання, а також попередньо протестувати з користувачами загальний візуальний стиль[6].

Деталізований mockup може включати не тільки основний екран програми, а й демонструвати всі важливі стани інтерфейсу, такі як активовані та деактивовані кнопки, вікна помилок, випадаючі меню тощо. Завдяки цьому, розробники можуть краще зрозуміти, як інтерфейс виглядатиме на різних етапах взаємодії з користувачем[7].

Mockup слугує основою для створення функціонального прототипу і дозволяє забезпечити узгодженість дизайну на етапі розробки продукту, що сприяє уникненню можливих непорозумінь і допомагає швидше досягти бажаного результату.

Рисунок 2.6 – Mockup початкової сторінки

При запуску застосунку користувач бачить логотип застосунку, іконку повідомлень, які можуть прийти з оновленням або від технічної підтримки. Найголовніше користувач має можливість вибрати мову жестів. Також у нижній панелі навігації користувач має можливість зайти у профіль користувача, словник жестів, за бажанням користувач може натиснути на іконку пошуку та знайти жести самостійно. Також користувач має можливість зайти у налаштування.

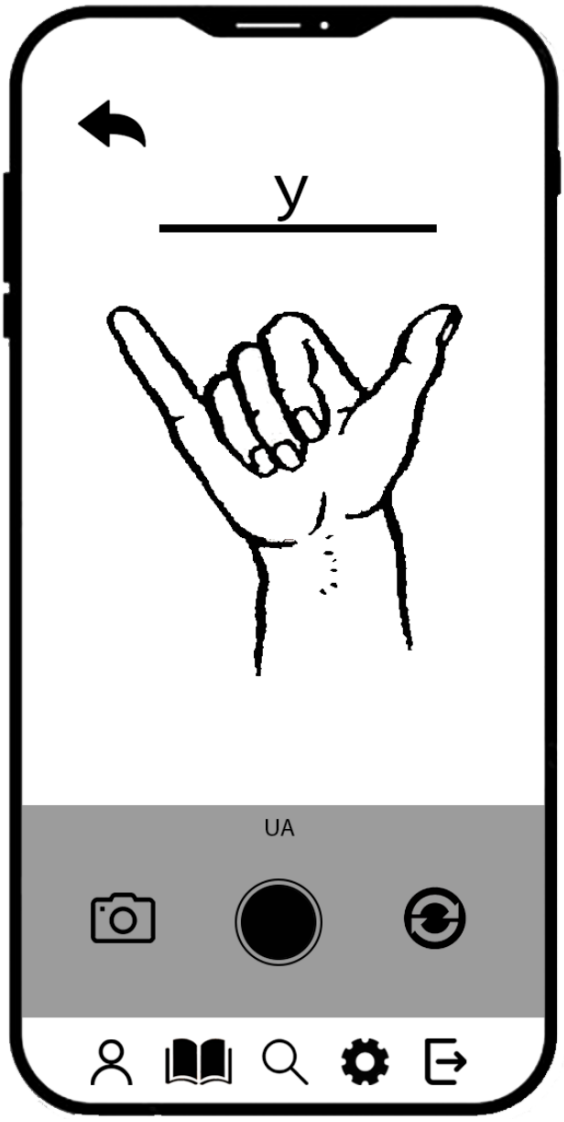


Рисунок 2.7 – Процес розпізнавання жесту

Після вибору мови, яку користувач хоче розпізнавати відкривається камера застосунку, де одразу у реальному часі відбувається розпізнавання жестів. Коли застосунок розпізнав жест він записує його значення у верхній рядок. Застосунок дає можливість переглянути жести які були розпізнані до цього і сфотографовані. Також є можливість зміни камери з основної на фронтальну.

Рисунок 2.8 – Підтвердження правильності розпізнавання

Дане вікно відкривається після того як користувач завершив розпізнавання жестів і застосунок питає підтвердження, правильно розпізнано або ні. Користувач повинен сам для себе вирішити чи підходить йому переклад або ні.

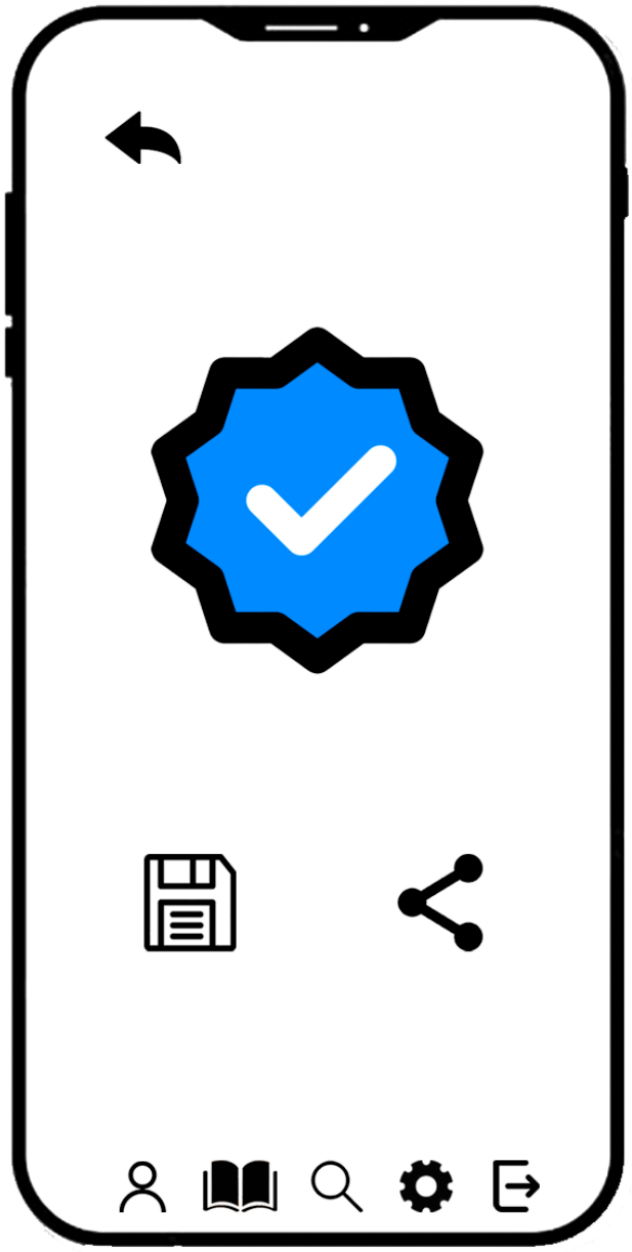


Рисунок 2.9 – Збереження або відправка даних

Якщо користувачу не підходить переклад, він має можливість показати жести знову. Але якщо користувач підтвердив правильність перекладу, тоді він потрапляє у вікно, де має можливість зберегти перекладений текст або поділитися ним у соціальних мережах.

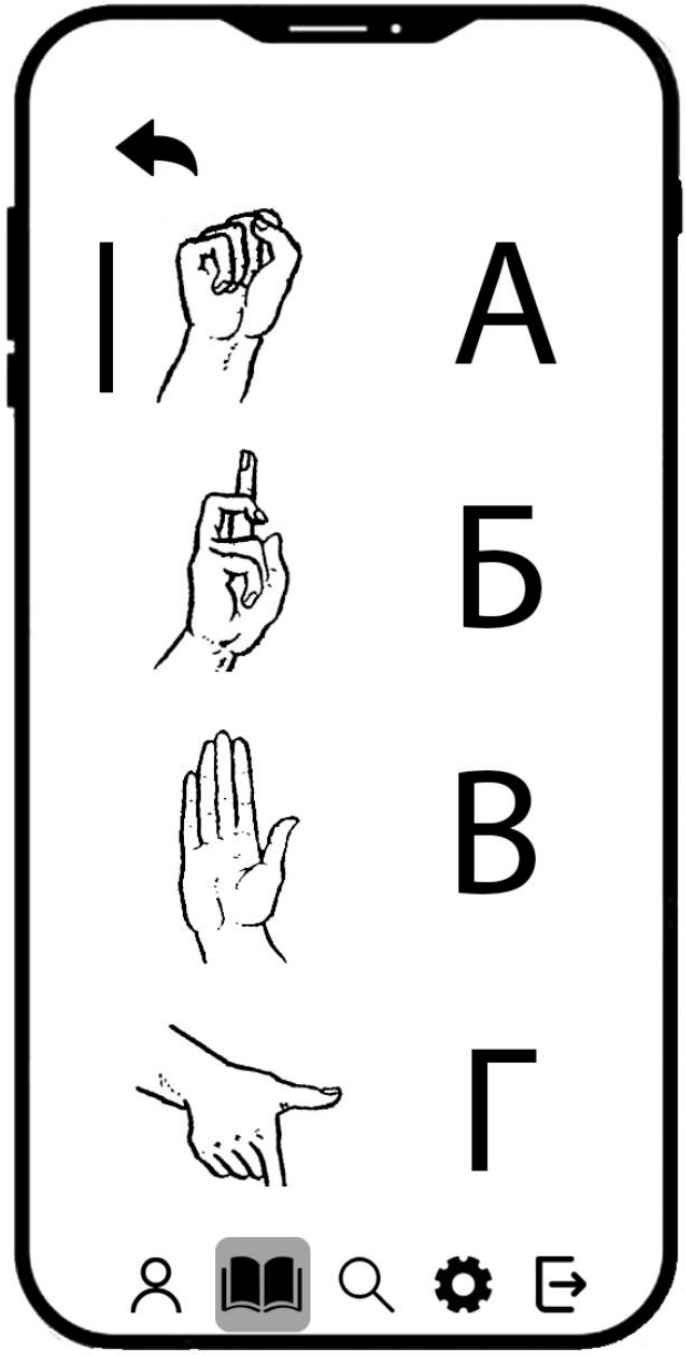


Рисунок 2.10 – Словник жестів

У тому випадку коли користувачу необхідно подивитися або дізнатися правильність виконання жесту, він має можливість зайти у словник і подивитися правильність показу жесту. У тому випадку якщо жест не знайдено то користувач має можливість ввести його у пошук натиснувши на іконку пошуку.

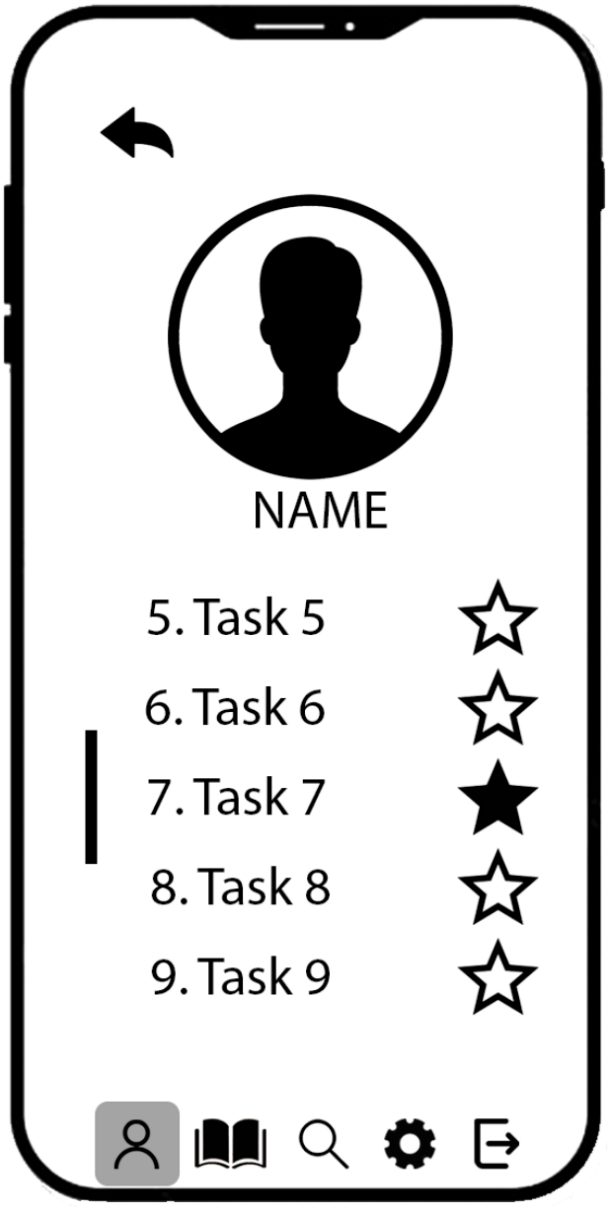


Рисунок 2.11 – Профіль користувача із завданнями

У користувача є можливість створити профіль, де буде відображатися іконка або фотографія. Під аватаркою написане ім’я користувача, яке було введено при реєстрації. Під ім’ям є можливість подивитися завдання які пропонує застосунок та їх прогрес виконання для кращого навчання жестової мови.

Висновки до розділу 2

У другому розділі детально розглянуто процес розробки системи розпізнавання жестів для мобільних пристроїв з акцентом на створення

зручного інтерфейсу користувача. Основою для розробки інтерфейсу стали діаграми варіантів використання, послідовності та станів, що дозволяють чітко визначити ключові етапи взаємодії користувача з системою.

Розроблено макет (mockup) інтерфейсу, який забезпечує користувача всіма необхідними функціями для ефективної роботи з системою розпізнавання жестів. На початковому екрані користувач має змогу обрати потрібну мову жестів, після чого йому пропонується виконати конкретний жест перед камерою мобільного пристрою. Система одразу надає зворотний зв’язок у вигляді текстового перекладу на основі розпізнаного жесту. Якщо жест не був розпізнаний коректно, користувач має можливість виконати його повторно, що значно підвищує точність перекладу.

Створені діаграми та mockups дозволяють забезпечити ефективну взаємодію між користувачем і системою, що сприяє точному і швидкому перекладу жестів у текст. Це значно полегшить спілкування для людей з порушеннями слуху та дозволить їм брати активну участь у різних соціальних процесах. Система може стати важливим інструментом для інтеграції людей із вадами слуху в суспільство, забезпечуючи зручний спосіб комунікації через мобільні пристрої.

3 МОДЕЛЮВАННЯ ТА КОНСТРУЮВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Архітектура ПЗ розробляється з акцентом на точність, масштабованість і реальну інтеграцію. Основним завданням є побудова системи, яка здатна розпізнавати жести алфавіту у реальному часі, використовуючи модель YOLOv5. В основі архітектури закладено модульний підхід, що забезпечує чітке розділення логіки обробки, навчання, валідації та роботи в реальному часі[7].

3.1 Розробка архітектури програмного забезпечення

Архітектура програмного забезпечення для системи розпізнавання жестів побудована на основі модульного підходу. Це забезпечує гнучкість і зручність у розвитку, тестуванні та модифікації різних частин системи. Кожен компонент має чітко визначену відповідальність, що дозволяє легко інтегрувати нові функції або модифікувати існуючі без серйозного впливу на інші частини програми:

- модуль підготовки даних;
- модуль навчання;
- модуль тестування та валідації;
- модуль роботи з реальним часом;
- графічний інтерфейс користувача.

Модуль підготовки даних займається збором, обробкою та підготовкою зображень, які будуть використовуватися для тренування та тестування моделі[8].

Нормалізація зображень – для правильного навчання моделі необхідно, щоб усі зображення мали однаковий розмір та відповідний діапазон значень пікселів (звичайно в діапазоні від 0 до 1 або від -1 до 1).

Зміна розміру зображень – оскільки зображення можуть бути різних розмірів, важливо привести їх до єдиного формату. Наприклад, кожне

зображення може бути змінено до стандартного розміру (наприклад, 224x224 пікселів).

Створення анотацій – анотації необхідні для навчання моделі, щоб вона могла правильно асоціювати кожне зображення з конкретним жестом. Це можуть бути координати розташування об'єктів (жестів) на зображенні у форматі bounding boxes, а також клас жесту (буква або команда).

Модуль також включає інструменти для анотації зображень, наприклад, LabelImg**,** яке дозволяє користувачам вручну позначати об'єкти на зображеннях.

Модуль навчання відповідає за налаштування та тренування моделі.

Підготовка навчальної вибірки – зібрані та попередньо оброблені зображення передаються на вхід моделі для тренування. Даний етап включає також поділ даних на тренувальну та валідаційну вибірки.

Налаштування моделі YOLOv5 – у цьому модулі здійснюється налаштування параметрів моделі YOLOv5, яка добре підходить для задач детекції об'єктів (жестів). Вибір цієї моделі обумовлений її високою швидкістю та точністю у реальному часі.

Процес навчання – під час навчання модель вивчає зв'язки між вхідними зображеннями та відповідними жестами, що дає змогу їй розпізнавати ці жести в нових зображеннях.

Перевірка ефективності – після кожної епохи навчання модель оцінюється на валідаційній вибірці. Різні метрики, такі як точність (accuracy), повнота (recall) та F1-міра, використовуються для оцінки ефективності.

Модуль тестування та валідації відповідає за перевірку навченої моделі на нових (не побачених раніше) даних.

Тестування моделі – після навчання модель перевіряється на тестових даних, щоб оцінити, наскільки добре вона працює в умовах, схожих на реальні.

Оцінка точності та повноти – використовуються метрики для визначення, як добре модель розпізнає жести. Точність вимірює частку

правильних прогнозів серед усіх передбачень, а повнота – частку правильних прогнозів серед усіх можливих правильних результатів.

Аналіз результатів – результати тестування аналізуються для визначення слабких місць моделі та можливих напрямків її покращення.

Модуль роботи з реальним часом дозволяє використовувати модель для розпізнавання жестів безпосередньо з відеопотоку, що дозволяє здійснювати взаємодію з користувачем у реальному часі;

Обробка відеопотоку – відео з камери передається в систему для подальшої обробки. Кадри з відеопотоку передаються моделі для розпізнавання жестів.

Детекція жестів – кожен кадр аналізується моделлю YOLOv5, яка визначає місцезнаходження жестів у межах кадру, виділяючи їх у вигляді прямокутних рамок (bounding boxes).

Виведення результатів – визначені жести виводяться на екран користувача разом з ідентифікованими буквами чи командами. Може бути також показана додаткова інформація, наприклад, точність виявлення.

Графічний інтерфейс користувача (GUI) є важливим елементом, що дозволяє користувачу взаємодіяти з програмним забезпеченням.

Перегляд відео – користувач може бачити відеопотік, на якому відображаються виявлені жести в реальному часі.

Налаштування параметрів – існує можливість налаштовувати різні параметри, наприклад, порогові значення для детекції жестів або швидкість обробки.

Кнопки управління – користувач може запускати або зупиняти процеси навчання, тестування, розпізнавання жестів, а також зберігати або ділитися результатами.

Виведення результатів – після розпізнавання жестів система відображає результат на екрані у вигляді тексту або графічних елементів.

Інтерфейс розроблений для забезпечення інтуїтивно зрозумілої взаємодії з користувачем, що дозволяє йому легко працювати з програмою та отримувати зворотний зв'язок у реальному часі.

3.2 Вибір технологій та мов програмування

Для розробки програмного забезпечення для системи розпізнавання жестів використані інструменти для розробки, які значно полегшують процес програмування, тестування та інтеграції коду.

Anaconda – це популярна платформа для роботи з Python, яка включає в себе менеджер пакетів, середовище для розробки та наукові бібліотеки для машинного навчання, обробки даних та аналізу. Anaconda надає зручний спосіб для керування бібліотеками та залежностями, що дозволяє легко створювати ізольовані середовища для різних проєктів. Вона підтримує важливі бібліотеки для наукових обчислень, такі як NumPy, Pandas, Matplotlib, і TensorFlow, що забезпечує гнучкість та можливість швидкого налаштування середовища для глибокого навчання та обробки зображень[9].

Рисунок 3.1 – Логотип Anaconda

Jupyter – це інтерактивне середовище для програмування, яке дозволяє створювати та запускати код по частинах, забезпечуючи можливість швидкої перевірки результатів і налаштування моделей. Jupyter Notebook є ідеальним інструментом для дослідницької роботи та тестування моделей машинного навчання, оскільки дозволяє зберігати код, документацію та графіки в одному

документі. Це зручний інструмент для експериментів з різними варіантами коду та для створення пояснень і візуалізацій під час роботи з даними[10].

Рисунок 3.2 – Логотип Jupyter

Visual Studio Code (VS Code) – це легкий, але потужний редактор коду, який є дуже популярним серед розробників Python завдяки своїй простоті використання, високій швидкості та потужним можливостям налаштування. VS Code підтримує інтеграцію з Python через плагіни та розширення, що дозволяє зручно писати та запускати Python-код, а також здійснювати налагодження та тестування. Крім того, VS Code підтримує роботу з контейнерами та віддаленими серверами, що дозволяє ефективно розробляти програми, а також інтегруватися з різними системами контролю версій, такими як Git[11].

Рисунок 3.3 – Логотип Visual Studio Code

Ці інструменти разом із бібліотеками Python забезпечують зручне та ефективне середовище для розробки та тестування програмного забезпечення для системи розпізнавання жестів.

Для розробки програмного забезпечення для системи розпізнавання жестів вибрано кілька ключових технологій, які забезпечують високу ефективність, масштабованість і зручність у реалізації задач машинного навчання, обробки зображень та створення інтерфейсу користувача[12].

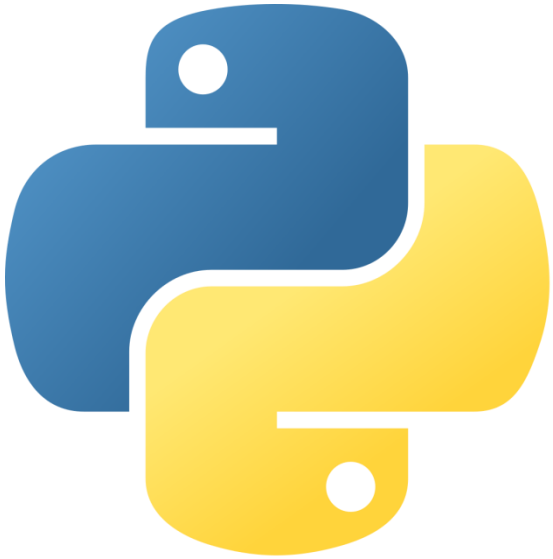


Рисунок 3.3 – Логотип Python

Python – це одна з найбільш популярних мов програмування в галузі машинного навчання та обробки зображень. Вибір Python обумовлений кількома важливими факторами:

1) популярність у науковому середовищі – Python широко використовується для розробки та експериментів з алгоритмами машинного навчання завдяки простоті синтаксису та великій кількості бібліотек;

2) численні бібліотеки для машинного навчання – Python має розвинуту екосистему бібліотек, таких як PyTorch, TensorFlow, scikit-learn, які дозволяють швидко розробляти та навчати моделі для розпізнавання зображень та інших задач;

3) інструменти для обробки зображень – для обробки та маніпуляцій з зображеннями в Python існують потужні бібліотеки, як-от OpenCV та Pillow, що забезпечують все необхідне для попередньої обробки та маніпуляції з графікою[13];

4) підтримка числових обчислень – Python активно використовує бібліотеки для числових обчислень, такі як NumPy та SciPy, що є важливими для обробки великих масивів даних і здійснення математичних операцій, необхідних для навчання моделей.

Фреймворк для глибокого навчання – PyTorch обраний як основний фреймворк для глибокого навчання, що забезпечує наступні переваги:

1) гнучкість і простота використання – PyTorch має простий і зрозумілий інтерфейс для розробки нейронних мереж. Це дозволяє швидко експериментувати з різними архітектурами моделей, включаючи YOLOv5, яка є основною моделлю для задачі розпізнавання жестів;

2) автоматичне диференціювання – PyTorch надає вбудовану систему автоматичного диференціювання, що дозволяє легко обчислювати градієнти та оптимізувати моделі;

3) робота з GPU – PyTorch має потужну підтримку обчислень на графічних процесорах (GPU), що значно пришвидшує процес навчання великих моделей, таких як YOLOv5;

4) широкий вибір попередньо навчених моделей – PyTorch надає доступ до численних попередньо навчених моделей для розпізнавання об'єктів, що дозволяє використовувати трансферне навчання та значно зменшити час на навчання.

Бібліотеки для обробки зображень: OpenCV та Pillow. OpenCV – це одна з найпоширеніших бібліотек для обробки зображень та комп'ютерного зору. Вона надає інструменти для роботи з відео, обробки зображень, детекції об'єктів, а також для реалізації різноманітних алгоритмів комп'ютерного зору:

1) зміна розміру зображень – OpenCV дозволяє зручно масштабувати зображення до необхідних розмірів для подальшого використання в моделі ;

2) обробка відео – бібліотека підтримує роботу з відео-файлами та реальним відеопотоком, що є необхідним для виведення результатів розпізнавання жестів в реальному часі;

3) фільтрація та нормалізація – OpenCV має потужні інструменти для фільтрації зображень і покращення їх якості перед передачею в модель.

Pillow – це бібліотека для обробки зображень в Python, яка є вдосконаленою версією Python Imaging Library (PIL). Вона використовується для базових операцій з зображеннями, таких як:

– обрізка та масштабування зображень;

– конвертація форматів – Pillow дозволяє працювати з різними форматами зображень, включаючи JPG, PNG, BMP, та інші.

Для розробки графічного інтерфейсу користувача обрано дві можливі технології: Tkinter та PyQt. Вибір між ними залежить від вимог до вигляду та функціональності інтерфейсу.

Tkinter – це стандартна бібліотека Python для створення простих графічних інтерфейсів. Вона має низькі вимоги до системи та є дуже легкою у використанні, що робить її чудовим вибором для додатків з базовими інтерфейсами.

– простота у використанні – Tkinter дозволяє швидко створювати вікна, кнопки, поля вводу та інші елементи інтерфейсу;

– обмежена функціональність – Tkinter має обмежені можливості для створення складних інтерфейсів порівняно з PyQt.

PyQt – це набір Python-обгорток для популярного фреймворку Qt, який надає потужні інструменти для створення професійних та функціональних інтерфейсів.

– багатий набір віджетів – PyQt дозволяє створювати складні інтерфейси з багатьма елементами управління, такими як діалогові вікна, меню, кнопки, текстові поля, графіки тощо;

– складність – для створення інтерфейсу за допомогою PyQt потрібно більше часу та зусиль, але він дає змогу реалізувати більш професійний вигляд та функціональність.

Вибір технологій для розробки програмного забезпечення базується на перевірених і ефективних рішеннях для задач машинного навчання та комп'ютерного зору. Python є оптимальним вибором завдяки своїй універсальності та підтримці великої кількості бібліотек, таких як PyTorch для глибокого навчання та OpenCV для обробки зображень. Для розробки графічного інтерфейсу були вибрані Tkinter та PyQt в залежності від складності інтерфейсу. Ці технології разом забезпечують ефективну реалізацію задачі розпізнавання жестів за допомогою зображень.

3.3 Вибір компонентів

Для розробки системи розпізнавання жестів вибрано кілька компонентів та інструментів, які сприяють ефективності та зручності реалізації ключових задач, таких як обробка зображень, навчання моделі, анотація даних та створення графічного інтерфейсу користувача.

Для детекції жестів обрана модель YOLOv5 (You Only Look Once Version 5). Ця потужна модель для виявлення об'єктів є однією з найшвидших і найбільш точних у своєму класі, що дозволяє ефективно виявляти і класифікувати об'єкти, зокрема жести, на зображеннях. YOLOv5 підтримує налаштування для специфічних задач, таких як розпізнавання жестів, і має високу адаптивність, завдяки чому можна налаштувати її під конкретні потреби проєкту. Легкість інтеграції з іншими компонентами програмного забезпечення через PyTorch робить її особливо привабливою для використання в таких системах[14].

Для анотації зображень використано інструмент LabelImg. Цей простий у використанні інструмент дозволяє позначати області жестів на зображеннях, створюючи текстові файли з координатами bounding boxes, які потім використовуються для тренування моделі. LabelImg підтримує популярні формати, такі як XML для PASCAL VOC та TXT для YOLO, що спрощує підготовку даних для навчання.

OpenCV – одна з найпопулярніших бібліотек для обробки зображень та відео в реальному часі, активно використовувалася для виконання важливих задач, таких як обробка зображень та відео з камери. OpenCV забезпечує зручні інструменти для підготовки зображень до обробки, зокрема для зміни їх розміру, нормалізації та фільтрації. Бібліотека також використовується для отримання відеопотоку з веб-камери та візуалізації результатів розпізнавання жестів в реальному часі, що є ключовою частиною системи. Для накладення межових прямокутників на розпізнані жести OpenCV надає зручні функції, що дозволяє користувачам бачити результат розпізнавання безпосередньо на екрані.

PyTorch – фреймворк для глибокого навчання, вибраний для тренування та валідації моделі YOLOv5. PyTorch забезпечує гнучкість і простоту використання завдяки інтуїтивно зрозумілому API, що дозволяє швидко створювати і тестувати нейронні мережі. Підтримка GPU значно пришвидшує процес тренування, особливо при роботі з великими моделями, такими як YOLOv5. Крім того, PyTorch підтримує трансферне навчання, що дозволяє використовувати попередньо навчені моделі і адаптувати їх під специфічну задачу розпізнавання жестів.

Для створення графічного інтерфейсу користувача використані дві бібліотеки: Tkinter та PyQt. Tkinter є простим і зручним інструментом для створення базових інтерфейсів з мінімальними вимогами до ресурсів, що дозволяє швидко реалізувати основні елементи управління, такі як кнопки та поля вводу. PyQt, в свою чергу, є більш потужним інструментом для створення

складних інтерфейсів, з підтримкою мультимедіа та більш широким набором віджетів. PyQt дозволяє створювати професійні інтерфейси, але вимагає більше ресурсів та часу для освоєння.

Вибір цих компонентів обумовлений їх здатністю ефективно вирішувати завдання, необхідні для розпізнавання жестів. YOLOv5 забезпечує точну детекцію жестів, LabelImg – швидке створення анотацій, OpenCV – обробку зображень і відео в реальному часі, PyTorch – тренування та тестування моделей. Tkinter і PyQt дозволяють створити зручний інтерфейс користувача, що покращує взаємодію з програмним забезпеченням. Цей набір інструментів забезпечує високу ефективність розробки і зручність для кінцевого користувача.

3.4 Діаграма класів

Діаграма класів – це один із основних типів діаграм в UML (Unified Modeling Language), який описує структуру системи шляхом показу класів, їх атрибутів, методів та зв'язків між класами. Вона визначає, як об'єкти в системі взаємодіють між собою та як організовано зберігання і обробка даних.

Класи на діаграмі представлено прямокутниками, які поділяються на три частини: верхня частина містить ім'я класу, середня – атрибути класу (дані, що зберігаються в ньому), а нижня – методи класу (функціональність, яку клас надає).

Зв'язки між класами можуть бути різними: асоціації, агрегації, композиції, залежності, наслідування чи реалізація інтерфейсів. Вони визначають, як класи взаємодіють один з одним або як один клас є частиною іншого. Наприклад, асоціація може вказувати на те, що один клас використовує інший для виконання своєї функції, а агрегація або композиція вказують на те, що один клас є частиною іншого. Наслідування використовується, коли один клас є розширенням іншого, а реалізація інтерфейсів – коли клас реалізує методи, визначені інтерфейсом.

Діаграма класів є важливим інструментом для розробки програмного забезпечення, оскільки вона дозволяє зрозуміти структуру та архітектуру системи, а також допомагає визначити, як різні частини програми взаємодіють між собою. Вона є основою для подальшого кодування, тестування та підтримки програм.

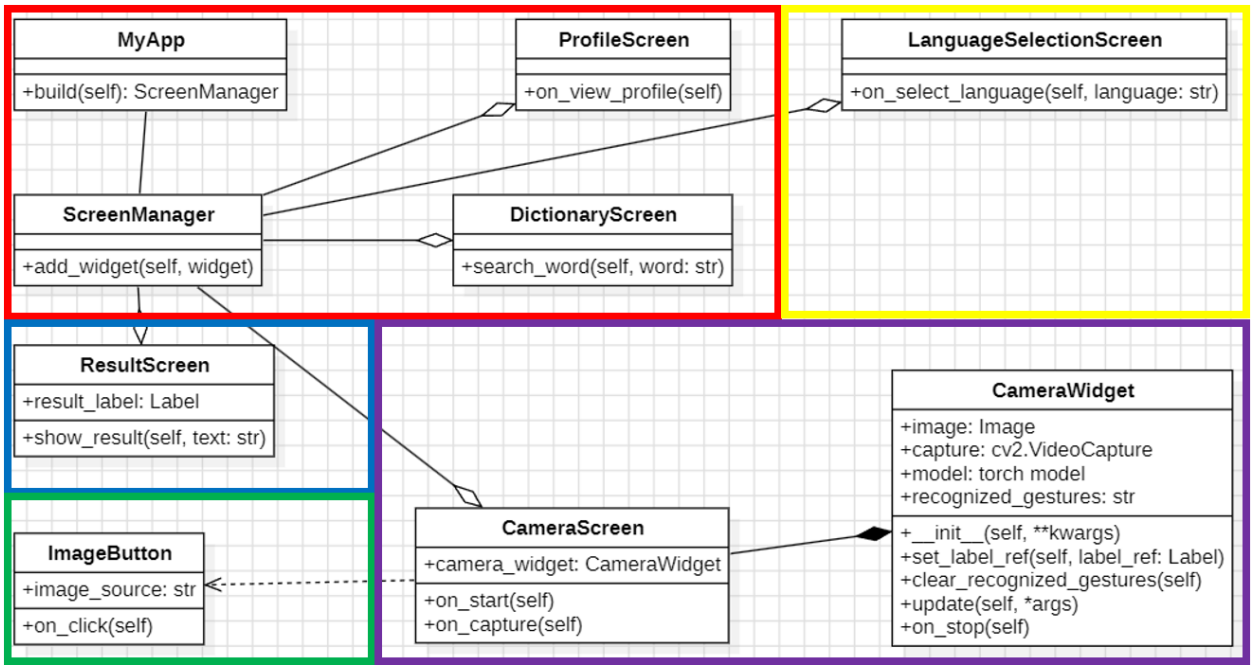


Рисунок 3.4 – Діаграма класів

Діаграма класів зображує структуру системи для розпізнавання жестів, яка складається з кількох ключових компонентів і зв'язків між ними. Центральним елементом є клас MyApp, який виконує роль головного контролера, що ініціалізує та управляє всіма екранними компонентами. Цей клас взаємодіє з класом ScreenManager, який управляє навігацією між різними екранами застосунка, такими як LanguageSelectionScreen, CameraScreen, ResultScreen, ProfileScreen та DictionaryScreen.

Клас CameraScreen є відповідальним за відображення відеопотоку з камери, а також за захоплення та обробку відео за допомогою класу CameraWidget, який відповідає за взаємодію з камерою та здійснює розпізнавання жестів. CameraWidget зберігає атрибути, такі як відео

захоплення, модель для розпізнавання жестів (через torch_model) та відображення результатів (через label).

Клас ResultScreen відповідає за відображення результатів розпізнавання жестів на екрані, надаючи користувачеві текстовий результат у вигляді перекладу. Цей клас отримує дані від CameraScreen та CameraWidget.

Клас ProfileScreen дозволяє користувачеві переглядати та редагувати свій профіль, а DictionaryScreen забезпечує доступ до словника жестів, де користувач може переглядати значення різних жестів.

Клас ImageButton є частиною інтерфейсу користувача, що дозволяє взаємодіяти з іншими екранами, надаючи можливість переходу між екранами або виконання інших дій.

Усі ці класи взаємодіють між собою через асоціації, агрегації та композиції, що дозволяє побудувати модульну та чітко організовану систему для розпізнавання жестів. Кожен клас має свою специфічну роль у забезпеченні функціональності програми, при цьому забезпечується чіткий розподіл обов'язків між компонентами.

3.5 Діаграма компонентів та впровадження

Діаграма відображає зв’язки між класами: асоціації для постійної взаємодії між компонентами, залежності для зовнішніх інструментів, як-от камера, та агрегацію для менш тісно пов’язаних сервісів, таких як соцмережі.

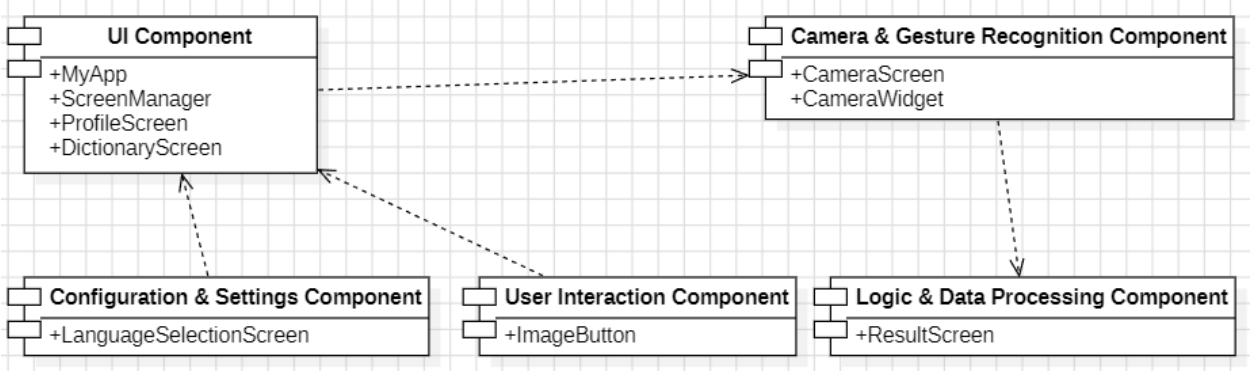


Рисунок 3.5 – Діаграма компонентів програмного комплексу

Діаграма компонентів представляє систему, розділену на логічні частини, де кожен компонент відповідає за певну функціональність і включає відповідні класи. UI Component взаємодіє з користувачем, збираючи дані через класи MyApp, ScreenManager, ProfileScreen та DictionaryScreen. Цей компонент має залежність від Camera and Gesture Recognition Component, який займається обробкою відео з камери та розпізнаванням жестів через класи CameraScreen і CameraWidget.

Camera and Gesture Recognition Component відповідає за захоплення відео та аналіз жестів. Він має залежність від Logic and Data Processing Component, який відображає результати розпізнавання на екрані через клас ResultScreen. Клас CameraWidget обробляє відео, використовуючи зовнішні бібліотеки, такі як cv2.VideoCapture, а також модель для розпізнавання жестів через torch_model.

Всі компоненти з'єднані через чітко визначені залежності, що забезпечують обмін даними і запитами між компонентами. UI Component керує взаємодією з користувачем і переходом між екранами, Camera and Gesture Recognition Component обробляє захоплене відео і розпізнає жести, в той час як Logic and Data Processing Component відповідає за відображення результатів та обробку даних.

Ця структура забезпечує модульність і чіткий розподіл обов'язків, де кожен компонент виконує свою специфічну роль у процесі обробки жестів і взаємодії з користувачем.

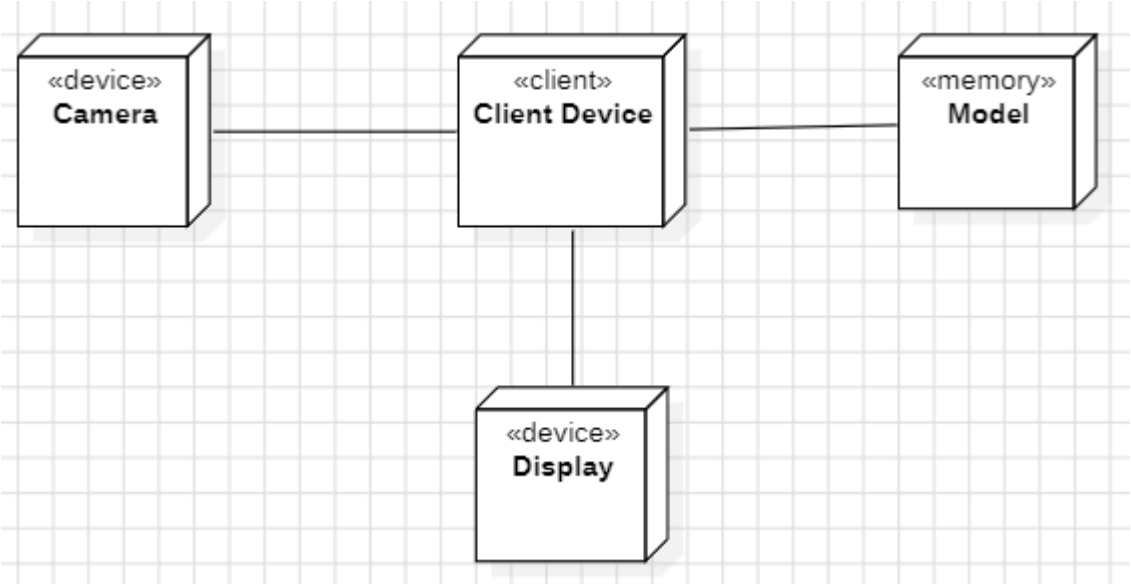


Рисунок 3.6 – Графічне зображення процесорів та пристроїв на діаграмі впровадження

Діаграма впровадження описує фізичне розташування та зв'язки між компонентами в системі, які забезпечують її функціональність. Вона зображує, як різні частини програми будуть реалізовані на різних пристроях або системах.

Client Device – цей компонент є головним елементом системи. Він містить усі основні елементи програми, які обробляють вхідні дані від користувача, управляють відеопотоком та здійснюють комунікацію з іншими компонентами. Це може бути комп'ютер, мобільний телефон або інший пристрій, який запускає програму. Відповідає за взаємодію з користувачем, запуск моделі для розпізнавання жестів, обробку результатів і передачу їх на дисплей.

Model – це компонент, що містить алгоритми для розпізнавання жестів, включаючи модель машинного навчання або штучного інтелекту. Цей компонент займається обробкою вхідних даних (зображень або відео) і прогнозуванням жестів, що дозволяє системі розпізнавати рухи руки чи інші жести, пов'язані з певними символами або командами. Модель знаходиться в залежності від Client Device, оскільки саме на цьому пристрої запускається її виконання.

Camera – компонент, що відповідає за захоплення відео або зображень у реальному часі. Камера надає потік зображень, які необхідні для розпізнавання жестів. Вона підключена до Client Device та надає йому необхідні дані для подальшої обробки.

Display – компонент, що відповідає за виведення результатів роботи системи користувачеві. Це може бути екран комп'ютера або мобільного пристрою, на якому відображаються результати розпізнавання жесту, інструкції для користувача чи повідомлення про помилки. Client Device керує Display для виведення відповідної інформації.

Усі компоненти взаємодіють через залежності. Client Device залежить від Camera для отримання відеопотоку, від Model для обробки даних і прогнозування жестів, а також від Display для виведення результатів. Тому Client Device є основним компонентом, який інтегрує та координує роботу інших частин системи.

3.6 Опис інтерфейсів програмного забезпечення

При запуску застосунку користувач бачить логотип програми, що займає центральне місце на екрані. У верхньому правому кутку розміщена іконка повідомлень, яка інформує про нові оновлення або важливі сповіщення від технічної підтримки. Основним елементом інтерфейсу є можливість вибору мови жестів, яку користувач хоче розпізнавати. Цю опцію можна вибрати через меню на головній сторінці. У нижній частині екрана знаходиться панель навігації з кількома кнопками, які дозволяють перейти до профілю, словника жестів, пошуку або налаштувань. Користувач може легко вибрати потрібний розділ для виконання своїх дій.

Після вибору мови жестів користувач переходить на екран з камерою, де здійснюється реальне розпізнавання жестів. Камера займає основну частину екрану, а програма відображає розпізнаний жест у верхньому рядку з його значенням. У процесі розпізнавання жестів є можливість перемикати камеру з

основної на фронтальну за допомогою спеціальної кнопки. Користувач також може зберігати розпізнані жести для подальшого перегляду чи аналізу.

Після розпізнавання жесту програма запитує користувача, чи вірно виконано переклад. Якщо результат не влаштовує користувача, він може показати жест ще раз. Якщо ж переклад є правильним, програма надає опцію зберегти результат або поділитися ним у соціальних мережах, відкриваючи вікно для вибору подальших дій.

У застосунку є доступ до словника жестів, де користувач може перевірити правильність виконання конкретного жесту. Якщо потрібного жесту немає в словнику, доступна функція пошуку. Крім того, програма дозволяє створити профіль користувача, де відображаються його аватарка, ім’я та інформація про навчальний прогрес. Під ім’ям можна побачити завдання, які пропонує застосунок для поліпшення навичок у жестовій мові.

Інтерфейс програми розроблений таким чином, щоб бути простим і зрозумілим для користувача, з легким доступом до всіх основних функцій. Дизайн передбачає мінімалістичний підхід, що дозволяє користувачеві зручно орієнтуватися в програмі та ефективно використовувати її для навчання жестовій мові.

Висновки до розділу 3

У результаті розробки програмного забезпечення для системи розпізнавання жестів створено гнучку, модульну архітектуру, яка дозволяє ефективно обробляти дані, тренувати та тестувати модель, а також забезпечувати інтеграцію з реальним часом для роботи з відеопотоком. Архітектура включає кілька ключових компонентів: модуль підготовки даних, модуль навчання, тестування та валідації, а також модуль роботи з реальним часом, що дозволяє користувачеві взаємодіяти з системою безпосередньо через відео.

Для реалізації проєкту обрані потужні технології, зокрема Python, PyTorch для глибокого навчання, а також популярні бібліотеки для обробки зображень, як OpenCV і Pillow. Використання платформи Anaconda, інтерактивного середовища Jupyter, а також редактора Visual Studio Code забезпечують зручне налаштування та ефективну розробку програмного забезпечення.

Основна модель для розпізнавання жестів – YOLOv5, що обрана завдяки своїй швидкості і точності в реальному часі, дозволяє ефективно виконувати детекцію жестів в умовах відеопотоку. Цей вибір технологій і підходів забезпечує високі результати в задачах розпізнавання жестів, оптимізуючи систему для реальних умов використання.

4 РОЗРОБКА ТА ТЕСТУВАННЯ ПРОГРМНОГО ЗАБЕЗПЕЧЕННЯ

Розробка програмного забезпечення є ключовим етапом реалізації будь-якого програмного проекту, що передбачає створення ефективного та надійного продукту, здатного вирішувати поставлені завдання. Особливу увагу приділено тестуванню, яке дозволяє оцінити коректність функціонування програми, її стабільність та відповідність початковим вимогам. Цей етап забезпечує високу якість кінцевого продукту, а також формує основу для його подальшої підтримки та розвитку.

4.1 Аугментація даних

Для навчання та тестування моделі обрано датасет жестової мови американського алфавіту, доступний на платформі Roboflow. Датасет включає високоякісні зображення жестів, що відповідають літерам англійського алфавіту (A–Z)[15].

Основними критеріями вибору цього датасету стали:

– різноманітність даних – зображення жестів виконані різними людьми, що сприяє підвищенню узагальнюючої здатності моделі;

– структура – датасет розділений на навчальну, валідаційну та тестову вибірки, що дозволяє ефективно тренувати і перевіряти модель, 720 фотографій;

– якість зображень – роздільна здатність у 1526x2047 пікселів забезпечує достатню деталізацію жестів для точної класифікації;

– доступність та готовність – дані легко інтегруються в проєкт, що скорочує час на підготовку власного датасету.

Цей набір даних є оптимальним вибором для задачі класифікації жестів завдяки своїй збалансованості, багатогранності та придатності для використання з сучасними алгоритмами комп’ютерного зору.

Для вихідних зображень вручну були створені обмежувальні рамки за допомогою програми labelImg.

Після цього зображення разом із координатами рамок піддавалися обробці в конвеєрі альбументації, який змінював їх розмір до квадратів 1024×1024 пікселів і застосовував різноманітні трансформації.

Серед таких змін були повороти на задані кути, зміщення положення, розмиття, горизонтальне віддзеркалення, випадкове видалення областей і модифікація кольорів.

Для початку відобразимо зображення з нанесеними обмежувальними рамками, використовуючи координати рамок із відповідних текстових файлів (додаток А).

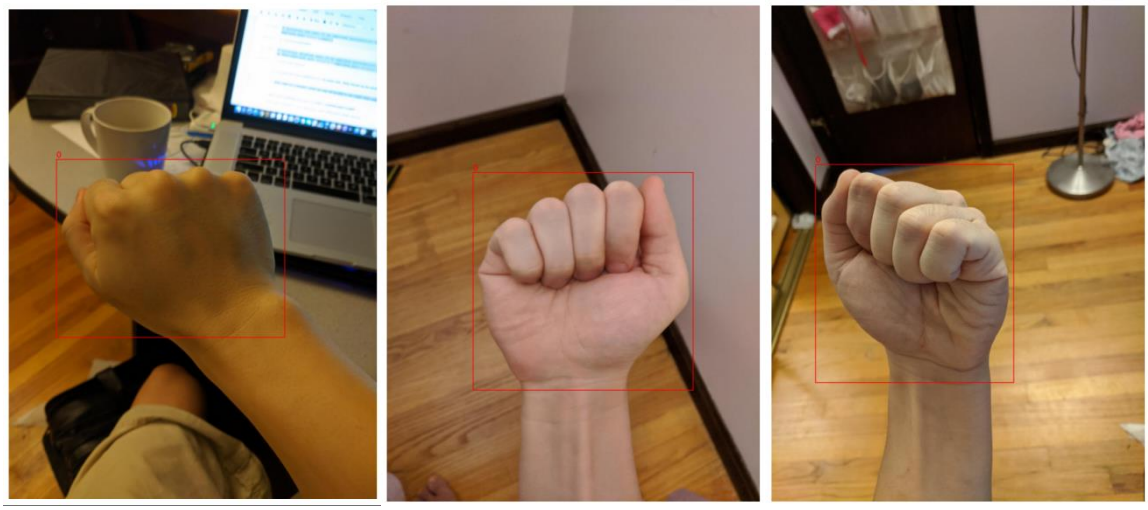


Рисунок 4.1 – Обмежувальні рамки

Імпорт бібліотек: завантажуються необхідні бібліотеки для роботи із зображеннями (cv2), побудови графіків (matplotlib) та роботи з масивами (numpy).

Функція plot_image_with_bboxes:

– завантажує зображення;
– читає координати обмежувальних рамок із файлу;
– перетворює нормалізовані координати рамок у абсолютні;
– наносить рамки та підпис класу на зображення;
– відображає оброблене зображення.

Функція visualize_train_images:

– обробляє всі зображення у папці train/images;

– для кожного зображення шукає відповідний файл із координатами рамок у папці train/labels;

– викликає функцію для нанесення рамок.

Запуск – вказується шлях до базової папки проєкту, і скрипт обробляє всі зображення з папки train.

Після чого обробляємо одне зображення із застосуванням аугментації (перетворень) для підготовки до моделювання машинного навчання.

Завантаження та обробка анотацій:

– завантажуються текстові анотації в YOLO-форматі, які містять інформацію про класи та координати обмежувальних рамок;

– конвертуються індекси класів у відповідні букви алфавіту на основі мапування класів.

Пробна аугментація зображень(додаток Б): застосовується трансформації до зображень і відповідних обмежувальних рамок за допомогою бібліотеки Albumentations. Включає:

– зміна розміру до 1024x1024;

– обертання, зсуви, масштабування;

– розмиття, горизонтальні перевороти, зміна яскравості, контрасту, насиченості;

– додавання довільного стирання (CoarseDropout).

Візуалізація обмежувальних рамок: будується та показується зображення з обмежувальними рамками та класами для перевірки результатів.

Збереження результатів:

– зберігаються аугментовані зображення у вигляді нових файлів;

– зберігаються відповідні анотації в YOLO-форматі.

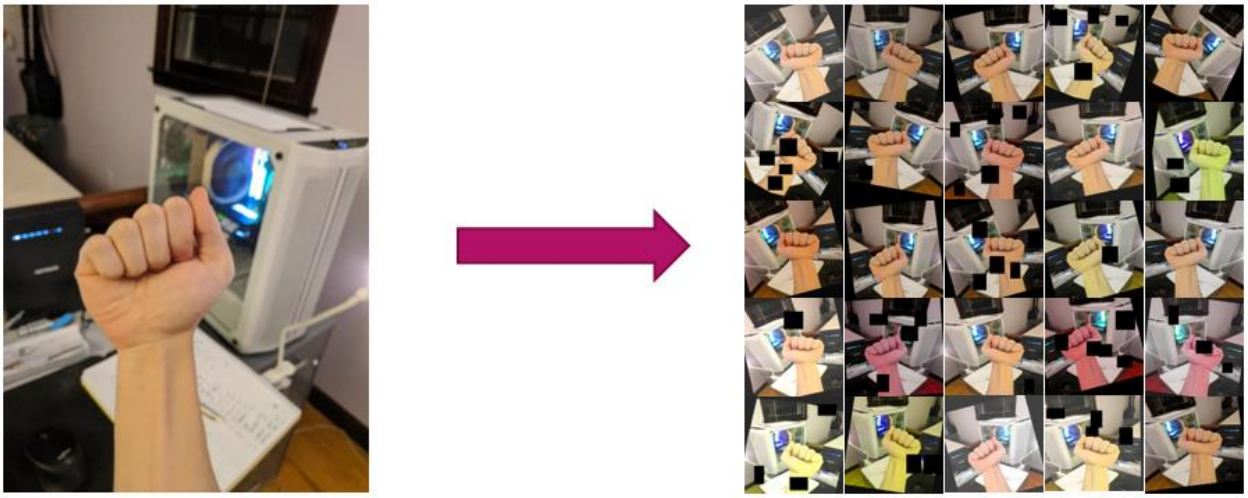


Рисунок 4.2 – Приклад аугментації

Для кожного зображення створено 25 доповнених зображень, у результаті чого для моделювання використовувався набір із 18 000 зображень.

4.2 Навчання моделі

Навчання моделі YOLOv5 із використанням попередньо підготовлених вагів базується на концепції перенесення навчання, що дозволяє значно зменшити час тренування та підвищити точність моделі. Попередньо підготовлені ваги були натреновані на великому датасеті, наприклад, COCO, який включає понад 80 класів об'єктів. Ці ваги містять інформацію про загальні ознаки, такі як текстури, контури та форми, що є базовими для розпізнавання об'єктів.

Навчання починається з підготовки даних. Зображення та їхні анотації мають бути у форматі YOLO, де координати обмежувальних рамок нормалізовані у відносних значеннях. Файл конфігурації також налаштовується для зазначення шляхів до тренувальних і валідаційних наборів даних, кількості класів і їхніх назв.

Модель використовує ваги як стартову точку, адаптуючи їх до нового набору даних. На початкових етапах модель використовує ці ваги для виділення базових ознак, а під час навчання фінальні шари

переналаштовуються для специфічного завдання. Цей процес дозволяє зменшити потребу в великих обсягах даних і знизити ризик перенавчання.

Після завершення навчання оцінюється точність моделі, зокрема, за показниками точності (precision), повноти (recall) та середнього значення точності (mAP). Модель генерує найкращі ваги, які можна використовувати для подальшого прогнозування на нових даних. Це дозволяє створювати ефективні рішення для розпізнавання об'єктів навіть за обмежених ресурсів і часу.

Для моделі, що розробляється, була використана архітектура YOLOv5m, яка є однією з варіацій YOLOv5, орієнтованою на досягнення балансу між точністю розпізнавання та швидкістю обробки. YOLOv5m відноситься до середнього рівня складності серед інших моделей сімейства YOLOv5, таких як YOLOv5s (швидка, але менш точна) та YOLOv5x (повільна, але з найвищою точністю). Ця версія є оптимальною для випадків, коли потрібна висока точність розпізнавання за збереження прийнятної продуктивності на сучасному обладнанні.

Архітектура YOLOv5m базується на модульному підході, який включає три основні компоненти: backbone, neck і head. Backbone відповідає за витягнення ознак із зображення, використовуючи механізм CSPNet для зменшення кількості параметрів і підвищення ефективності. Neck забезпечує створення багаторівневих ознак за допомогою PANet, що покращує здатність моделі розпізнавати об'єкти різних розмірів. Head виконує фінальне прогнозування обмежувальних рамок, класів і ймовірностей для кожного об'єкта.

Використання YOLOv5m дозволяє обробляти як дрібні, так і великі об'єкти, що робить цю архітектуру придатною для задач із різноманітними типами даних. У розроблюваній моделі YOLOv5m було адаптовано до конкретного датасету жестової мови шляхом перенавчання попередньо

підготовлених вагів. Це дало змогу швидко адаптувати модель до нових класів об'єктів і забезпечити високу точність розпізнавання жестів за прийнятного часу обробки в реальному часі.

Навчання моделі було запущено з використанням GPU з підтримкою технології CUDA, що суттєво прискорило процес обчислень. CUDA (Compute Unified Device Architecture) – це платформа паралельних обчислень, розроблена компанією NVIDIA, яка дозволяє виконувати ресурсоємні задачі, такі як навчання нейронних мереж, на графічному процесорі[16].

Використання GPU забезпечило значно вищу швидкість виконання матричних операцій, які є основою алгоритмів глибокого навчання, порівняно з CPU. Це особливо актуально для моделі YOLOv5s, яка обробляє велику кількість даних і виконує складні обчислення під час навчання.

Під час навчання забезпечено оптимальне використання апаратних ресурсів за рахунок обробки великих міні-батчів і паралельного виконання обчислень, що дозволило зменшити час, необхідний для кожної епохи. Це також дало змогу експериментувати з гіперпараметрами та швидше отримувати результати, що вплинули на точність моделі.

Обчислювальне середовище з підтримкою CUDA продемонструвало високу ефективність, дозволяючи досягти необхідної продуктивності для обробки й аналізу великого обсягу даних у стислі строки.

4.3 Аналіз навчання моделі

Після тренування моделі YOLOv5 у директорії exp зберігаються різні файли, що відображають результати та характеристики навчання моделі. Їх значення полягає в аналізі роботи моделі, оцінці її продуктивності та повторному відтворенні умов експерименту.

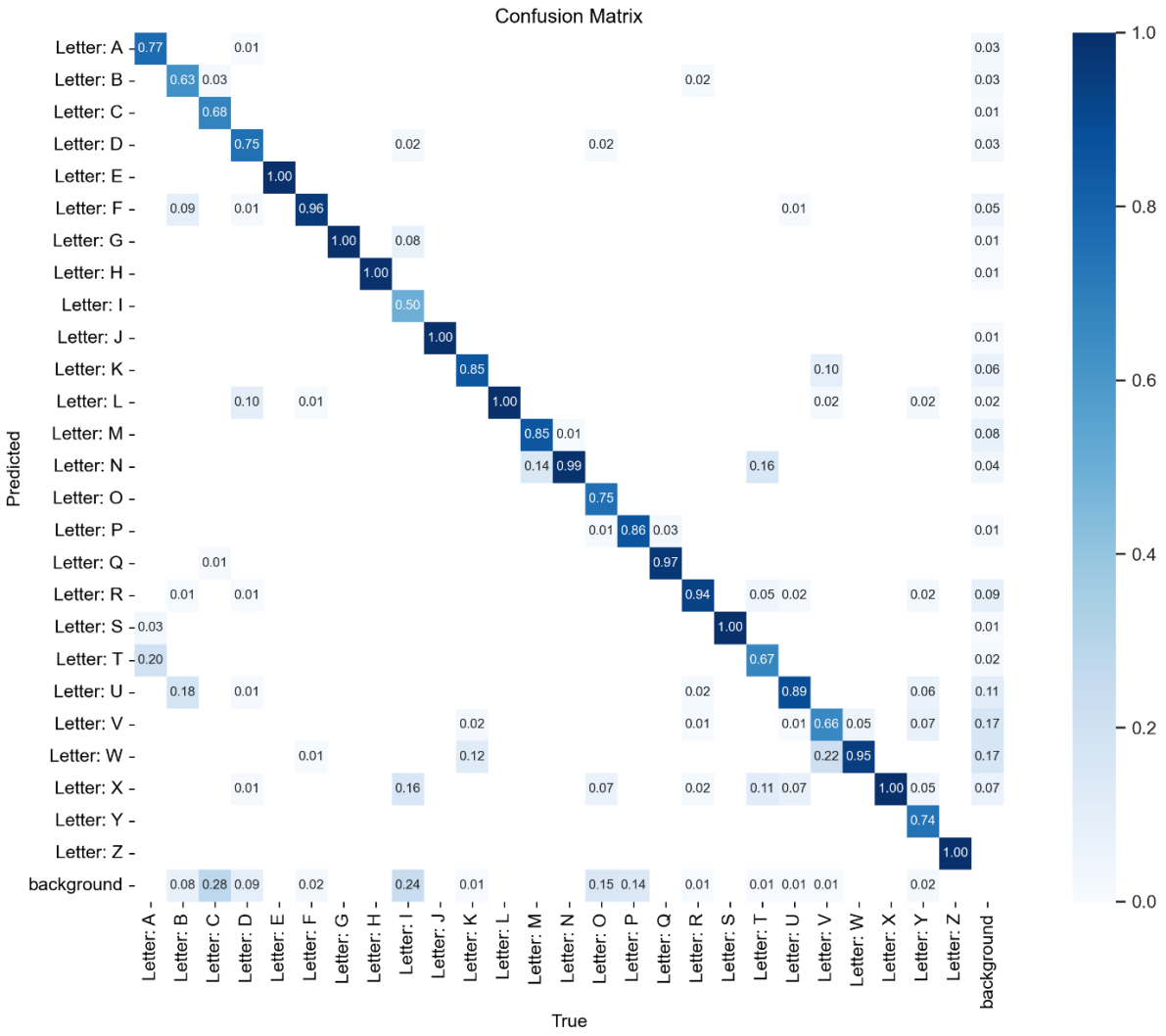


Рисунок 4.3 – Confusion matrix

Графічне представлення матриці плутанини (confusion matrix). Вона показує, як добре модель класифікує об'єкти, відображаючи співвідношення між правильними та хибними передбаченнями для кожного класу[17].

Аналізуючи надану вами матрицю плутанини, можна виділити найбільш успішні та проблемні частини розпізнавання:

Найкращі результати:

– letter E (істинна та передбачена мітка) – значення 1.0. Це означає, що модель правильно класифікувала всі зображення літери E;

– letter G – також має значення 1.0 в діагоналі, що свідчить про чудову точність у класифікації цієї літери;

– letter H – знову-таки, модель правильно класифікувала всі зображення, отримавши значення 1.0;

– letter J – значення 1.0 також вказує на відмінну класифікацію;

– letter L – значення 1.0 вказує на відмінну класифікацію.

Ці літери демонструють відмінне розпізнавання.

Найгірші результати:

– letter B – тут спостерігаємо низьке значення для передбачення літери "B", зі значенням 0.63. для правильного класу та високими значеннями для інших літер (наприклад, "Letter U" – 0.18), що вказує на плутанину між буквами.

– letter I – аналогічна ситуація: значення 0.50 для правильної літери I, що свідчить про плутанину між літерою I та іншими класами а особливо background зі значенням 0.24.

– letter V – маємо низьке значення 0.66 для передбаченої літери V, що означає певну плутанину, особливо з "Letter W" (0.22).

Плутанина між класами:

Літери Letter A та Letter T мають помітну плутанину з точністю 0.20, що свідчить про те, що модель іноді плутає ці літери.

Також спостерігається певна плутанина між літерами Letter U та Letter B. Це означає, що модель іноді неправильно класифікує ці літери.

Загальний огляд:

Велика частина літер (наприклад, Letter E, Letter G, Letter H) має непогані показники точності, хоча є й певні помилки.

Проблеми з класифікацією більше стосуються літер з подібною формою, як I, B, V, де модель не завжди здатна чітко відрізнити один знак від іншого.

Ця матриця дає цінну інформацію про слабкі та сильні сторони моделі, що дозволяє вам зрозуміти, які саме літери потребують додаткового навчання або корекції для поліпшення результатів.

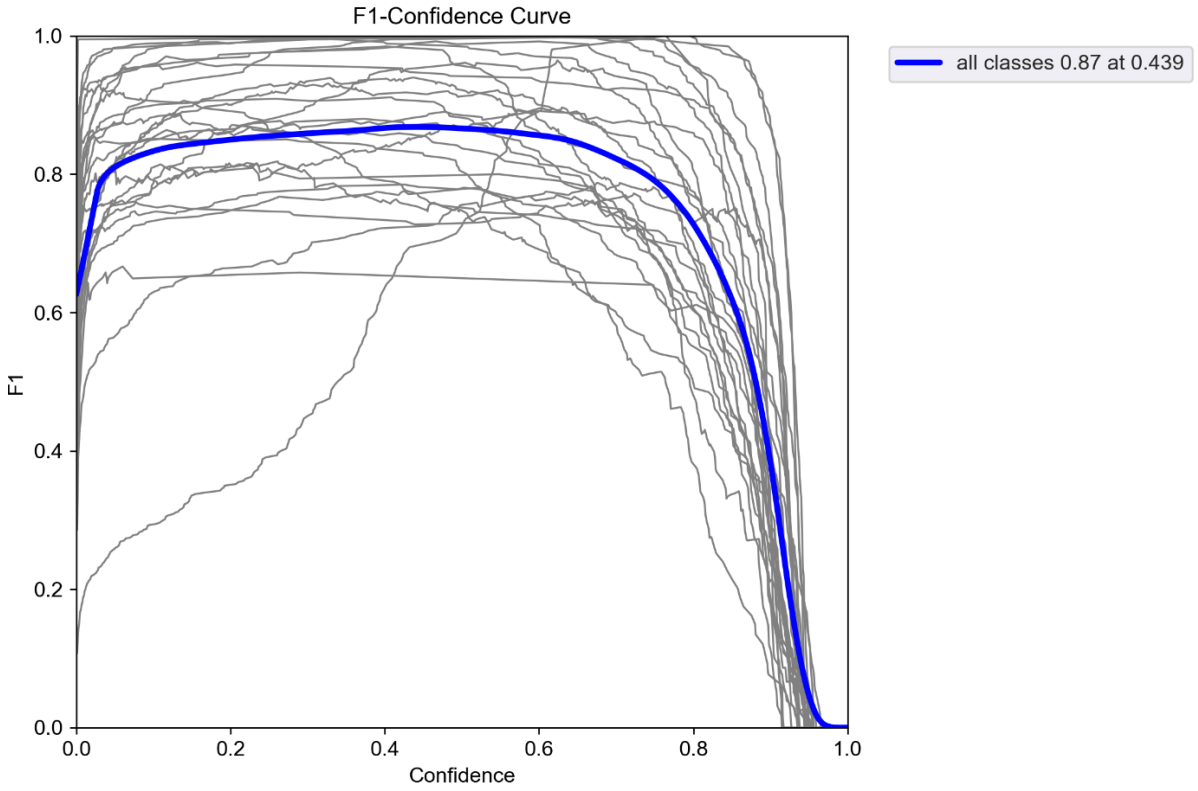


Рисунок 4.4 – Графік кривої F1-метрики

Графік показує криву F1-оцінки для різних рівнів довіри (confidence), зокрема для кожного класу (позначено сірими лініями) та загальну криву для всіх класів (позначену синьою лінією).

Крива F1-конфіденції відображає, як змінюється значення метрики F1 в залежності від порогу довіри, що використовується для прийняття рішень. F1-метрика є середнім гармонійним між точністю (precision) та повнотою (recall) й використовується для оцінки якості класифікаційної моделі.

Синя лінія – показує загальний результат F1 для всіх класів при певному порозі довіри 0.439, що дає значення F1 = 0.87. Це досить хороший результат, що вказує на збалансовану роботу моделі.

Сірі лінії – кожна сірка лінія представляє F1-оцінку для окремого класу при різних рівнях довіри. Вони варіюються, але більшість з них стабільно тримають високі значення при порогах довіри близьких до 0.4-0.5.

Загалом, графік демонструє, що модель має хороший рівень узгодженості між точністю та повнотою для більшості класів, особливо якщо вибраний оптимальний поріг довіри.

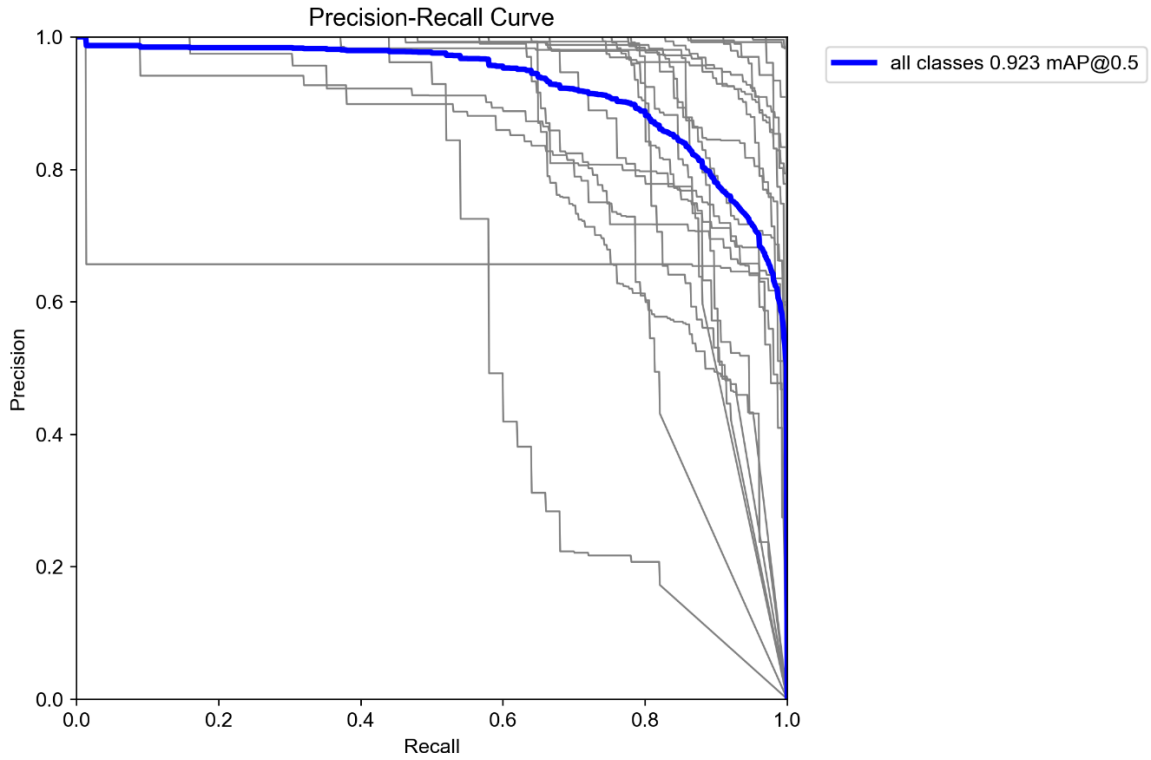


Рисунок 4.5 – Крива точність-відгук

На графіку представлена Precision-Recall Curve (Крива точність-відгук) для моделі класифікації з усіма класами, разом із середнім показником mAP@0.5 (mean Average Precision).

Висока середня точність (mAP):

– значення 0.923 вказує на те, що модель може ефективно розпізнавати об'єкти більшості класів;

– це підходить для багатьох застосувань, включаючи розпізнавання жестів.

Широкий діапазон кривих для окремих класів (сірі лінії):

– деякі класи мають Precision-Recall криві ближче до лівого нижнього кута, що свідчить про гіршу продуктивність для цих класів;

– варто перевірити, які саме класи мають нижчу продуктивність, і можливо, доопрацювати дані (наприклад, додати приклади для цих класів або провести аугментацію).

Чітке узагальнення – синя крива показує, що навіть з урахуванням варіацій між класами, модель в цілому добре справляється.

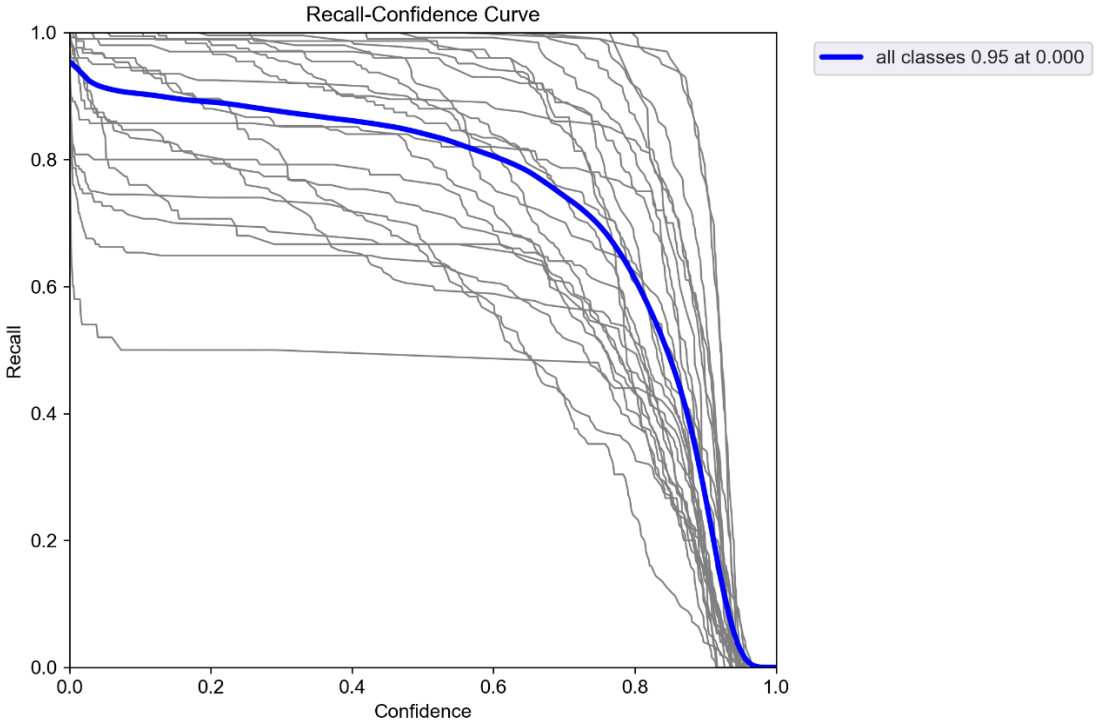


Рисунок 4.6 – Крива відгук-впевненість

На графіку представлена Recall-Confidence Curve (Крива відгук-впевненість) для моделі класифікації, яка показує залежність відгуку (recall) від рівня впевненості (confidence) для усіх класів.

Високий початковий відгук (0.95), при низькому порозі впевненості модель виявляє більшість правильних передбачень, що є хорошим показником.

Поступове зниження відгуку – синя лінія показує, що навіть при високих рівнях впевненості (до 0.8) модель зберігає досить високий відгук. Це означає, що модель стабільна при різних порогах.

Варіації між класами:

– деякі сірі лінії падають швидше, що свідчить про труднощі моделі в ідентифікації певних класів з високою впевненістю;

– варто перевірити ці класи та покращити їхню представленість у навчальних даних.

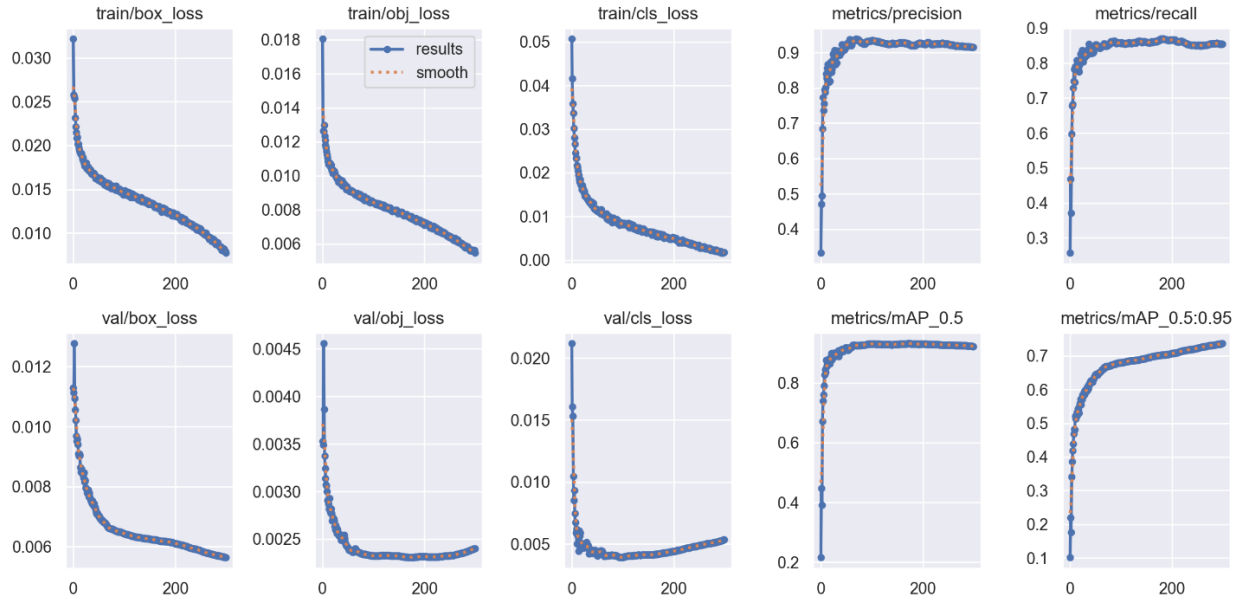


Рисунок 4.7 – Результати навчання та валідації моделі

Train/box_loss:

– втрати для координат об'єктів (bounding boxes) поступово знижуються, що свідчить про покращення точності локалізації об’єктів;

– зниження стабільне, без значних стрибків.

train/obj_loss:

– втрати для виявлення об'єктів також знижуються, досягаючи низьких значень (<0.006);

– це свідчить про те, що модель стає впевненою у виявленні присутності об'єктів.

train/cls_loss – втрати класифікації знижуються майже до нуля, що означає високу точність у розпізнаванні класів.

Метрики під час навчання:

– metrics/precision – точність стабільно висока (>0.9), що показує здатність моделі правильно класифікувати виявлені об'єкти без помилкових позитивних результатів;

– metrics/recall – відгук поступово зростає і стабілізується близько 0.9, що вказує на здатність моделі знаходити всі наявні об’єкти.

Втрати під час валідації (нижній ряд) – val/box_loss, val/obj_loss, val/cls_loss, усі втрати на валідаційному наборі демонструють схожу динаміку до навчальних втрат. Це свідчить про те, що модель не перенавчвється, а результати добре узагальнюються.

Метрики на валідації:

– metrics/mAP_0.5 – Mean Average Precision (mAP) при IoU = 0.5 швидко зростає і стабілізується більше 0.9, що свідчить про високу точність детектування;

– metrics/mAP_0.5:0.95 – mAP для різних порогів IoU також стабільно зростає, досягаючи значення приблизно 0.7, що є хорошим результатом для більш строгих умов оцінки.

Таблиця 4.1 – Початок тренування моделі

epoch	train/box_loss	train/obj_loss	train/cls_loss	metrics/precision	metrics/recall	metrics/mAP_0.5	metrics/mAP	val/box_loss	al/obj_loss	val/cls_loss	x/lr0	/lr1
0	0,032	0,018	0,051	0,333	0,257	0,216	0,102	0,011	0,004	0,021	0,070	0,003
1	0,026	0,013	0,042	0,471	0,468	0,448	0,220	0,011	0,003	0,016	0,040	0,007
2	0,026	0,013	0,036	0,495	0,371	0,392	0,177	0,013	0,005	0,015	0,010	0,010
3	0,025	0,013	0,034	0,684	0,597	0,671	0,341	0,011	0,004	0,010	0,010	0,010
4	0,023	0,012	0,030	0,773	0,678	0,741	0,387	0,011	0,003	0,009	0,010	0,010

Втрати під час навчання:

– train/box_loss (втрата координат) – починається з 0.032198 і поступово зменшується до 0.02316. Це показує, що модель швидко навчається точніше визначати координати об'єктів;

– train/obj_loss (втрата виявлення об'єктів) – зменшується з 0.018038 до 0.012322, свідчачи про прогрес у визначенні наявності об’єктів;

– train/cls_loss (втрата класифікації) – падає з 0.050665 до 0.030213, що вказує на покращення точності розпізнавання класів.

Метрики на навчанні:

– metrics/precision (точність) – стартує з 0.33316 і стрімко зростає до 0.77284, показуючи зменшення помилкових позитивних результатів;

– metrics/recall (відгук) – початкове значення 0.25674 зростає до 0.67801, що демонструє здатність моделі знаходити більше об'єктів;

– metrics/mAP_0.5 – починається з 0.21589 і доходить до 0.7408, свідчачи про значне покращення в детектуванні об'єктів;

– metrics/mAP_0.5:0.95 – зростає з 0.10178 до 0.38656, що підтверджує прогрес для більш складних умов оцінювання.

Втрати під час валідації:

– val/box_loss – легке коливання, але в цілому зменшується з 0.011301 до 0.010561;

– val/obj_loss – теж поступово зменшується, з 0.0035265 до 0.003375;

– val/cls_loss – падає з 0.021174 до 0.0085081, вказуючи на покращення розпізнавання класів на валідації.

Таблиця 4.2 – Кінець тренування моделі

epoch	train/box_loss	train/obj_loss	train/cls_loss	metrics/precision	metrics/recall	metrics/mAP_0.5	metrics/mAP	val/box_loss	al/obj_loss	val/cls_loss	x/lr0	/lr1
295	0,008	0,006	0,002	0,915	0,855	0,924	0,735	0,006	0,002	0,005	0,000	0,000
296	0,008	0,006	0,002	0,915	0,855	0,923	0,735	0,006	0,002	0,005	0,000	0,000
297	0,008	0,006	0,002	0,915	0,854	0,923	0,736	0,006	0,002	0,005	0,000	0,000
298	0,008	0,006	0,002	0,915	0,854	0,923	0,736	0,006	0,002	0,005	0,000	0,000
299	0,008	0,005	0,002	0,915	0,854	0,923	0,736	0,006	0,002	0,005	0,000	0,000

Втрати під час навчання:

– train/box_loss – знизилася до 0.0077158, що показує високу точність у локалізації об'єктів;

– train/obj_loss – досягла 0.0054721, вказуючи на впевнене виявлення об’єктів;

– train/cls_loss – вже на рівні 0.0016987, що демонструє практично бездоганну класифікацію.

Метрики на навчанні:

– metrics/precision – залишилася стабільно високою, ~0.915;

– metrics/recall – тримається на рівні 0.854;

– metrics/mAP_0.5 – досягла 0.923, що є чудовим результатом;

– metrics/mAP_0.5:0.95 – стабільно на 0.736, що вказує на хорошу роботу при строгих умовах оцінювання.

Втрати під час валідації:

– val/box_loss – досягла низького значення 0.0056328;

– val/obj_loss – зменшилася до 0.0024032;

– val/cls_loss – стабільно низька, ~0.0053446;

Значення x/lr0, x/lr1, x/lr2 (навчальна швидкість):

– на початку епохи 0: 0.07, 0.0033, 0.0033;

– поступово знижується до 0.000166 на епосі 299, що свідчить про використання політики зменшення швидкості навчання, коли втрати стабілізуються.

На початку навчання модель демонструє швидкий прогрес, втрати значно зменшуються, а метрики помітно зростають. Уже до епохи 4 значення mAP перевищує 0.74, що свідчить про ефективне засвоєння даних.

На завершальних етапах навчання показники стабілізуються: всі втрати досягли мінімуму, а значення метрик залишаються стабільно високими. Значення mAP_0.5 перевищує 0.92, а mAP_0.5:0.95 становить більше 0.73, що підтверджує готовність моделі до використання в реальних умовах.

Навчання можна завершувати, оскільки модель досягла плато у метриках і втрат, а подальше навчання не принесе суттєвих покращень.

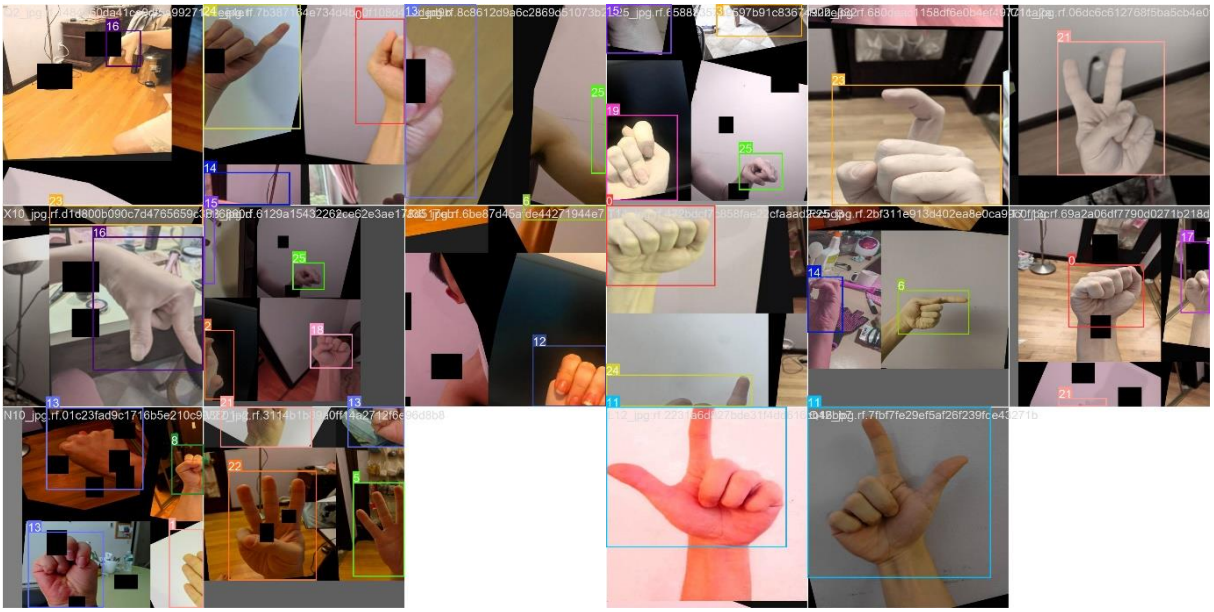


Рисунок 4.8 – Візуалізація зображень на етапі навчання

Візуалізація зображень із тренувального набору разом із передбаченнями моделі, що показує роботу детектора на етапі навчання.

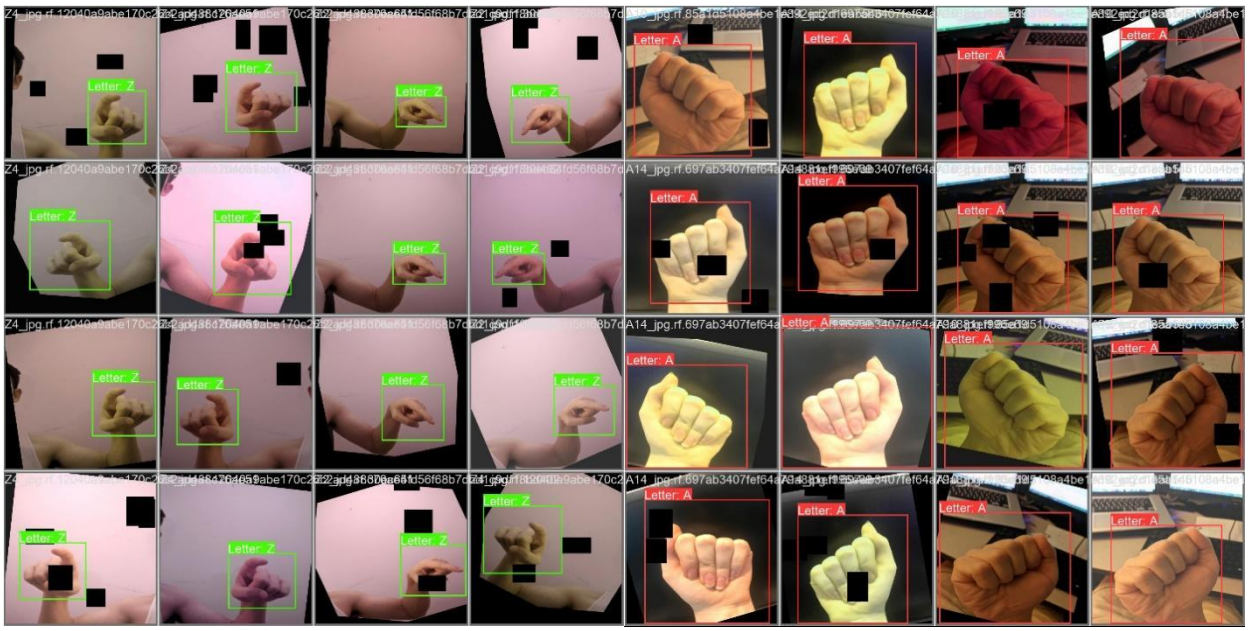


Рисунок 4.9 – Оцінка правильності анотацій

Візуалізація зображень із валідаційного набору з нанесеними мітками (ground truth), що допомагає оцінити правильність анотацій.

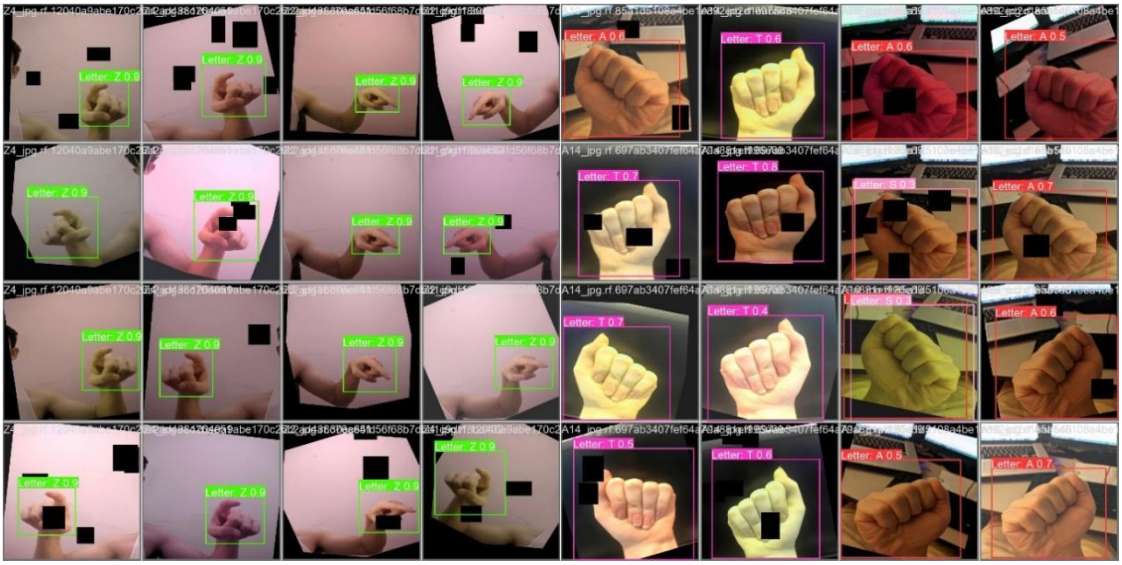


Рисунок 4.10 – Передбаченнями моделі

Візуалізація зображень із валідаційного набору з передбаченнями моделі, що дозволяє оцінити її роботу на валідаційних даних.

Як можна побачити що при передбачені схожі жести передбачаються не так добре як хотілося.

4.4 Створення інтерфейсу користувача

Інтерфейс користувача (UI, User Interface) – це система, яка дозволяє користувачам взаємодіяти з комп'ютерними програмами або пристроями. Він включає в себе всі елементи, через які людина може керувати програмним забезпеченням або пристроєм, такі як кнопки, меню, поля введення, текстові поля, вікна та інші графічні компоненти[18].

Інтерфейс користувача може бути графічним (GUI) або текстовим (CLI). У графічному інтерфейсі використовуються візуальні елементи, які дозволяють користувачеві здійснювати взаємодію з програмою через маніпуляції мишкою або сенсорним екраном. Текстовий інтерфейс, натомість, працює через введення команд через клавіатуру.

Основна мета інтерфейсу користувача – зробити взаємодію з системою максимально зручною, зрозумілою та ефективною для користувача.

Інтерфейс користувача, розроблений за допомогою Kivy, включає кілька важливих екранів для взаємодії з користувачем, що дозволяють зручно працювати з жестами та їх перекладом.

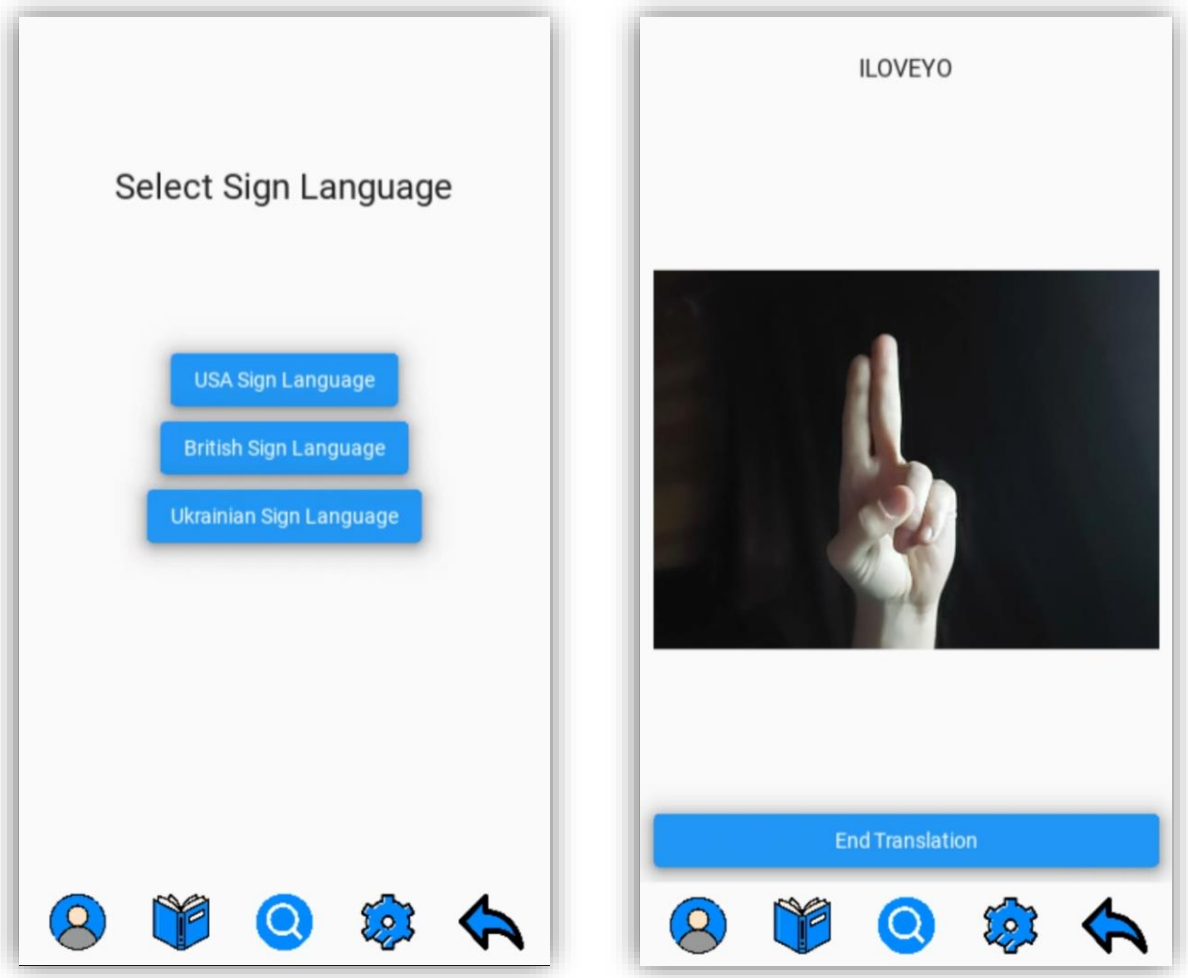


Рисунок 4.11 – Екрани вибору мови та камери

Екран вибору мови дозволяє користувачу обрати мову, на яку будуть перекладатися жести. Інтерфейс включає кілька кнопок із доступними мовами. Після вибору мови користувач переходить до основного функціоналу програми.

Екран камери відображає відео з камери в реальному часі. Система аналізує жести та відображає текстовий результат розпізнавання прямо на відео. Після закінчення перекладу користувач має можливість закінчити переклад.

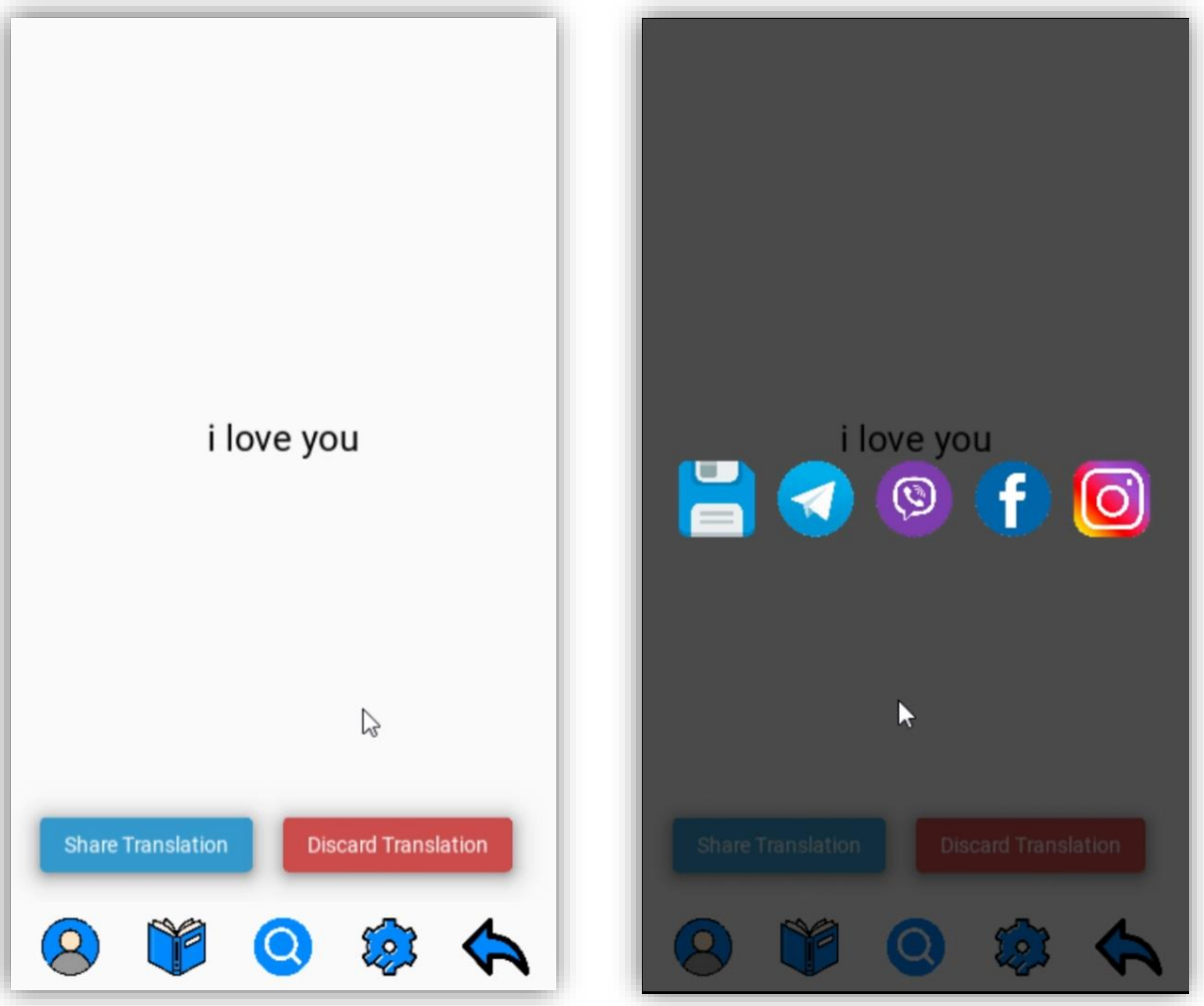


Рисунок 4.12 – Екран результату

На екрані результату перекладу відображається переклад жесту або серії жестів, які програма успішно розпізнала. Відбувається валідація тексту, а саме автоматична розтановка пробілів між словами, та вправлення помилок в залежності від контексту. У разі потреби є можливість скинути результат і повернутися до екрана камери для нової спроби або поділитися кінцевим результатом у соціальних мережах.

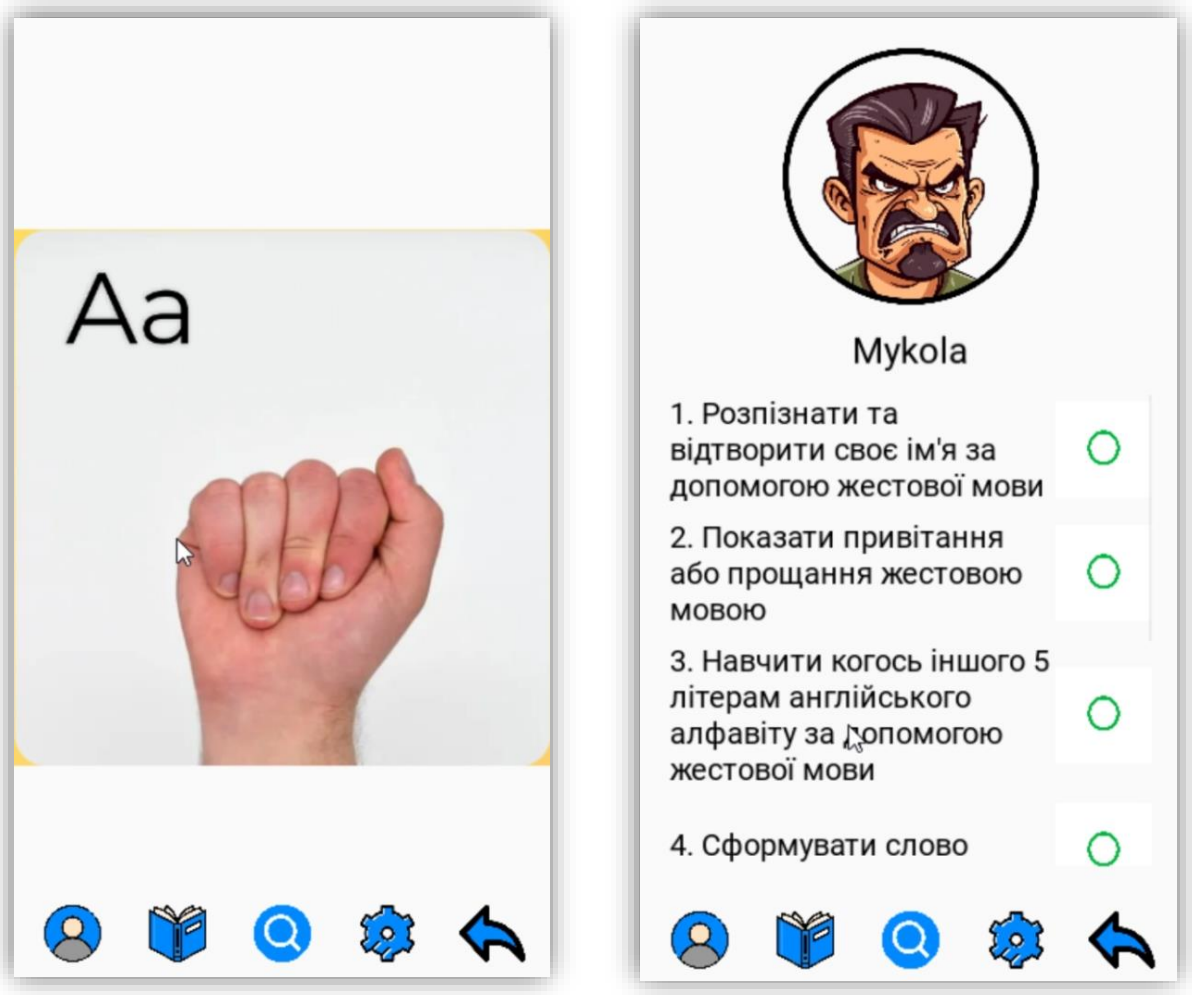


Рисунок 4.13 – Екрани словнику та профілю користувача

Словник дозволяє користувачу переглядати доступні жести та їх значення. Крім того, користувач може звертатися до словника для уточнення значення жесту, якщо система не змогла розпізнати його одразу.

Екран профілю надає можливість користувачу налаштувати персональну інформацію, таку як ім’я або аватар профілю. Можна відмітити виконані завдання та передивитися доступні для виконання.

Висновки до розділу 4

У розділі детально описано етапи розробки та тестування програмного забезпечення для класифікації жестів американського алфавіту. Спочатку була здійснена підготовка та аугментація даних із використанням датасету жестової

мови, що дозволило значно збільшити кількість тренувальних зображень, покращивши здатність моделі до узагальнення.

Наступним етапом стало навчання моделі YOLOv5m за допомогою попередньо підготовлених вагів, що дозволило зменшити час тренування та покращити точність розпізнавання. Враховуючи використання технології CUDA, навчання моделі було значно прискорене, що дало можливість швидше отримати результати й оптимізувати гіперпараметри. Результати навчання були оцінені за допомогою метрик точності, повноти та середнього значення точності (mAP), що дозволило визначити сильні та слабкі сторони моделі.

ВИСНОВКИ

У ході виконання кваліфікаційної роботи магістра здійснено розробку та впровадження програмного забезпечення для автоматизації процесу розпізнавання жестів, що позначають символи американського жестового алфавіту, у режимі реального часу із застосуванням технологій машинного навчання.

Для досягнення поставленої мети виконано такі завдання:

– досліджено алгоритми машинного навчання для розпізнавання жестової мови;

– проаналізовано сучасні методи використання жестової мови та визначено ключові вимоги до програмного забезпечення;

– розроблено концепцію та архітектуру застосунку;

– реалізовано основні компоненти, зокрема моделювання жестів, розпізнавання, віртуальне середовище та інтерфейс користувача;

– проведено тестування розробленого застосунку з метою виявлення та усунення недоліків;

– розглянуто можливості застосування програмного забезпечення для науково-технічних досліджень і тестування нових алгоритмів.

Практичне значення отриманих результатів полягає у створенні інструменту, що сприяє вдосконаленню процесу навчання жестовій мові, підвищенню якості розпізнавання жестів та розширенню можливостей інклюзивних технологій. Розроблене програмне забезпечення може використовуватись у дослідженнях і навчанні, забезпечуючи інтерактивний досвід та сприяючи популяризації жестової мови у суспільстві.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Hand Talk Translator. Hand Talk. Hand Talk. URL: https://www.handtalk.me/en/ (Last accessed: 15.09.2024).

2. Lingvano GmbH. Lingvano. Lingvano. URL: https://www.lingvano.com/asl/ (Last accessed: 23.09.2024).

3. MobiReactor. Pocket Sign. Google Play. URL: https://play.google.com/store/apps/details?id=com.mobireactor.signlanguage&hl=en_US (Last accessed: 16.09.2024).

4. Charles T. F. Python Programming Crash Course. Charlie Creative Lab, 2020.

5. Mueller J. P., Massaron L. Machine Learning for Dummies. Wiley & Sons, Incorporated, John, 2021. 432 p.

6. Що таке мокап? Як його використовують графічні та вебдизайнери - Koval Web. Koval Web. URL: https://kovalweb.com/uk/what-is-mockup/ (дата звернення: 11.09.2024).

7. StarUML. StarUML. URL: https://staruml.io/download (Last accessed: 09.09.2024).

8. Волошко С., Поночовний Ю. Навчальний посібник з дисципліни “технології розробки програмного забезпечення ”. Полтава : Полт. нац. техн. ун-т ім. Юрія Кондратюка, 2017. 218 с.

9. Download Anaconda Distribution | Anaconda. Anaconda. URL: https://www.anaconda.com/download (Last accessed: 08.12.2024).

10. Project Jupyter. Project Jupyter | Home. URL: https://jupyter.org/ (Last accessed: 08.12.2024).

11. Microsoft. Visual Studio Code - Code Editing. Redefined. Visual Studio Code - Code Editing. Redefined. URL: https://code.visualstudio.com/ (Last accessed: 08.12.2024).

12. Welcome to Python.org. Python.org. URL: https://www.python.org (Last accessed: 08.12.2024).

13. Brahmbhatt S. Introduction to Computer Vision and OpenCV. Practical OpenCV. Berkeley, CA, 2013. P. 3–5. URL: https://doi.org/10.1007/978-1-4302-6080-6_1 (Last accessed: 09.10.2024).

14. Zhao L., Tohti T., Hamdulla A. BDC-YOLOv5: a helmet detection model employs improved YOLOv5. Signal, Image and Video Processing. 2023. URL: https://doi.org/10.1007/s11760-023-02677-x (Last accessed: 08.12.2024).

15. Roboflow: Computer vision tools for developers and enterprises. Roboflow: Computer vision tools for developers and enterprises. URL: https://roboflow.com/ (Last accessed: 08.12.2024).

16. CUDA Toolkit 12.1 Downloads. NVIDIA Developer. URL: https://developer.nvidia.com/cuda-downloads (Last accessed: 08.12.2024).

17. Confusion_matrix. scikit-learn. URL: https://scikit-learn.org/1.5/modules/generated/sklearn.metrics.confusion_matrix.html (Last of accessed: 08.12.2024).

18. Курс «UX/UI. Проєктування та дизайн інтерфейсів» від Креативної Практики. Creative Practice. URL: https://cases.media/creativepractice/course/ux-ui-design-basics/about (дата звернення: 13.09.2024).

ДОДАТОК А

Візуалізація рамок

```
import os
import cv2
import matplotlib.pyplot as plt
import numpy as np  # Необхідний імпорт бібліотеки numpy
def plot_image_with_bboxes(image_path, label_path):
    # Завантаження зображення
    image = cv2.imread(image_path)
    image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
    # Читання анотацій
    with open(label_path, 'r') as file:
        lines = file.readlines()
    # Додавання обмежувальних рамок
    for line in lines:
        values = line.strip().split()
        if len(values) == 5:  # перевірка кількості значень
            class_id = values[0]
            x_center, y_center, width, height = map(float,
values[1:5])

            # Перетворення координат з відносних (нормалізованих)
до абсолютних
            h, w = image.shape[:2]
            x_center *= w
            y_center *= h
            width *= w
            height *= h

            x1 = int(x_center - width / 2)
            y1 = int(y_center - height / 2)
            x2 = int(x_center + width / 2)
            y2 = int(y_center + height / 2)
```

```
                # Додавання обмежувальних рамок
                cv2.rectangle(image, (x1, y1), (x2, y2), (255, 0, 0),
2)
                # Відображення класу на обмежувальній рамці
                cv2.putText(image,   class_id,   (x1,   y1   -   10),
cv2.FONT_HERSHEY_SIMPLEX, 1, (255, 0, 0), 2, cv2.LINE_AA)
            else:
                print(f"Warning:   Unexpected   number   of   values
({len(values)}) in line: {line.strip()}")
        # Відображення зображення з обмежувальними рамками
        plt.figure(figsize=(10, 10))
        plt.imshow(image)
        plt.axis('off')
        plt.show()
    def visualize_train_images(base_path):
        images_path = os.path.join(base_path, 'train', 'images')
        labels_path = os.path.join(base_path, 'train', 'labels')
        for filename in os.listdir(images_path):
            if filename.endswith('.jpg'):  # Або будь-який інший формат
ваших зображень
                image_path = os.path.join(images_path, filename)
                label_path        =        os.path.join(labels_path,
filename.replace('.jpg', '.txt'))
                plot_image_with_bboxes(image_path, label_path)
    # Вкажіть шлях до вашої папки з проектом
    base_path = 'C:/Users/shuma/VERSION7'  # Використовуйте прямі слеші
або подвійні зворотні слеші
    # Візуалізація всіх зображень з обмежувальними рамками в папці
train
    visualize_train_images(base_path)
```

ДОДАТОК Б

Аугментація даних

```
import os
import cv2
import numpy as np
import albumentations as A
import matplotlib.pyplot as plt
# Повне мапування класів
class_mapping = { '0': 'A', '1': 'B', '2': 'C', '3': 'D', '4': 'E', '5': 'F', '6': 'G', '7': 'H', '8': 'I', '9': 'J', '10': 'K', '11': 'L', '12': 'M', '13': 'N', '14': 'O', '15': 'P', '16': 'Q', '17': 'R', '18': 'S', '19': 'T', '20': 'U', '21': 'V', '22': 'W', '23': 'X', '24': 'Y', '25': 'Z'}
def load_annotations(label_path):
    with open(label_path, 'r') as file:
        lines = file.readlines()
    annotations = []
    for line in lines:
        values = line.strip().split()
        if len(values) == 5:  # перевірка кількості значень
            class_id = values[0]
            x_center, y_center, width, height = map(float, values[1:])
            annotations.append({
                'class_id': class_mapping.get(class_id, 'unknown'),  # отримуємо назву класу
                'bbox': [x_center, y_center, width, height]
            })
    return annotations
def save_annotations(label_path, annotations):
    reverse_class_mapping = {v: k for k, v in class_mapping.items()}
    with open(label_path, 'w') as file:
```

```
            for ann in annotations:
                class_id = reverse_class_mapping.get(ann['class_id'],
'0')
                bbox = ' '.join(map(str, ann['bbox']))
                file.write(f"{class_id} {bbox}\n")
    def augment_image(image, annotations, transform):
        # Збереження bbox у форматі YOLO
        bboxes = [ann['bbox'] for ann in annotations]
        class_labels = [ann['class_id'] for ann in annotations]
        # Застосування трансформацій до зображення та обмежувальних
рамок
        transformed = transform(image=image, bboxes=bboxes,
class_labels=class_labels)
        # Повернення перетвореного зображення і оновлених анотацій у
форматі YOLO
        transformed_image = transformed['image']
        transformed_annotations = []
        for cl, bbox in zip(transformed['class_labels'],
transformed['bboxes']):
            transformed_annotations.append({
                'class_id': cl,
                'bbox': bbox
            })
        return transformed_image, transformed_annotations
    def plot_image_with_bboxes(image, annotations):
        # Додавання обмежувальних рамок
        reverse_class_mapping = {v: k for k, v in
class_mapping.items()}
        for ann in annotations:
            bbox = ann['bbox']
            class_id = reverse_class_mapping.get(ann['class_id'], '0')
            x_center, y_center, width, height = bbox
            h, w = image.shape[:2]
            x_center *= w
```

```
            y_center *= h
            width *= w
            height *= h
            x_min = int(x_center - width / 2)
            y_min = int(y_center - height / 2)
            x_max = int(x_center + width / 2)
            y_max = int(y_center + height / 2)
            cv2.rectangle(image, (x_min, y_min), (x_max, y_max), (255,
0, 0), 2)
            cv2.putText(image, class_mapping[class_id], (x_min, y_min
- 10), cv2.FONT_HERSHEY_SIMPLEX, 1, (255, 0, 0), 2, cv2.LINE_AA)
        # Відображення зображення з обмежувальними рамками
        plt.figure(figsize=(10, 10))
        plt.imshow(image)
        plt.axis('off')
        plt.show()
    def process_single_image(image_filename, base_path):
        transform = A.Compose([
            A.Resize(1024, 1024),
            A.Rotate(limit=30,       border_mode=cv2.BORDER_CONSTANT,
value=0),
            A.ShiftScaleRotate(shift_limit=0.1,      scale_limit=0.1,
rotate_limit=15, p=0.7),
            A.Blur(blur_limit=7, p=0.2),
            A.HorizontalFlip(p=0.5),
            A.RandomBrightnessContrast(p=0.5),
            A.HueSaturationValue(hue_shift_limit=20,
sat_shift_limit=30, val_shift_limit=20, p=0.5),
            A.CoarseDropout(max_holes=5,     max_height=int(0.2*1024),
max_width=int(0.2*1024),     min_holes=1,     min_height=int(0.1*1024),
min_width=int(0.1*1024), p=0.5)
        ],                    bbox_params=A.BboxParams(format='yolo',
label_fields=['class_labels']))
        images_path = os.path.join(base_path, 'train', 'images')
```

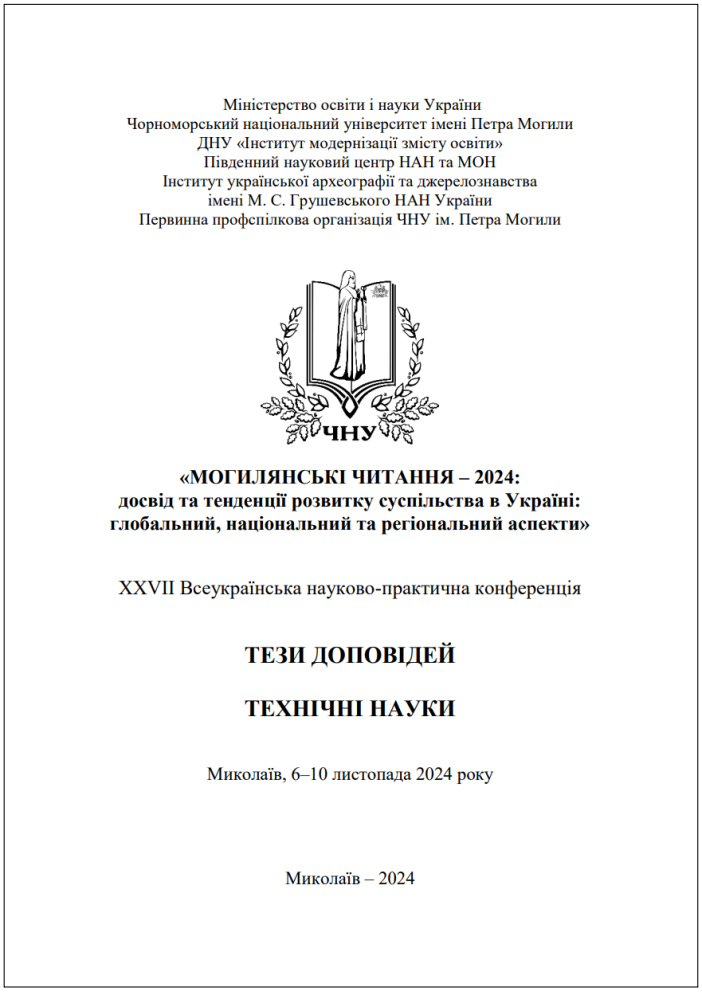
```
        labels_path = os.path.join(base_path, 'train', 'labels')
        output_images_path = os.path.join(base_path,
'single_image_aug', 'images')
        output_labels_path = os.path.join(base_path,
'single_image_aug', 'labels')
        os.makedirs(output_images_path, exist_ok=True)
        os.makedirs(output_labels_path, exist_ok=True)
        image_path = os.path.join(images_path, image_filename)
        label_path = os.path.join(labels_path,
image_filename.replace('.jpg', '.txt'))
        image = cv2.imread(image_path)
        annotations = load_annotations(label_path)
        for i in range(25):
            transformed_image, transformed_annotations =
augment_image(image, annotations, transform)
            output_image_path = os.path.join(output_images_path,
f"{os.path.splitext(image_filename)[0]}_aug_{i}.jpg")
            output_label_path = os.path.join(output_labels_path,
f"{os.path.splitext(image_filename)[0]}_aug_{i}.txt")
            cv2.imwrite(output_image_path, transformed_image)
            save_annotations(output_label_path,
transformed_annotations)
            print(f"Зображення та анотації збережено:
{output_image_path}")
            print(f"Збережені анотації: {output_label_path}")
            plot_image_with_bboxes(transformed_image.copy(),
transformed_annotations)
    # Вкажіть шлях до вашої папки з проектом
    base_path = 'C:/Users/shuma/VERSION7'
    # Вкажіть назву файлу для перевірки
    image_filename = 'O25_jpg.rf.5a6cbb32da5741dc371a84fce3cbe86f.jpg'
# Змініть на фактичну назву файлу
    # Обробка одного зображення з використанням конвеєра альбументацій
    process_single_image(image_filename, base_path)
```

ДОДАТОК В

Апробація кваліфікаційної роботи

Результати досліджень були представлені на конференції:

Могилянські читання – 2024 : досвід та тенденції розвитку суспільства в Україні : глобальний, національний та регіональний аспекти. Технічні науки : XXVII Всеукр. наук.-практ. конф. : 6–10 листоп. 2024 р., м. Миколаїв : тези / М-во освіти і науки України ; ЧНУ ім. Петра Могили ; ДНУ «Ін-т модернізації змісту освіти» ; Півд. наук. центр НАН та МОН України ; Ін-т укр. археографії та джерелоз-навства ім. М. С. Грушевського НАН України; Первинна профспілкова орг. ЧНУ ім. Петра Могили. – Миколаїв : Вид-во ЧНУ ім. Петра Могили, 2024.



Міністерство освіти і науки України
Чорноморський національний університет імені Петра Могили
ДНУ «Інститут модернізації змісту освіти»
Південний науковий центр НАН та МОН
Інститут української археографії та джерелознавства
імені М. С. Грушевського НАН України
Первинна профспілкова організація ЧНУ ім. Петра Могили

ЧНУ

**«МОГИЛЯНСЬКІ ЧИТАННЯ – 2024:
досвід та тенденції розвитку суспільства в Україні:
глобальний, національний та регіональний аспекти»**

XXVII Всеукраїнська науково-практична конференція

ТЕЗИ ДОПОВІДЕЙ

ТЕХНІЧНІ НАУКИ

Миколаїв, 6–10 листопада 2024 року

Миколаїв – 2024

Рисунок В.1 – Обкладинка збірника тез конференції Могилянські читання – 2024