МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Національний аерокосмічний університет ім. М.Є. Жуковського
«Харківський авіаційний інститут»

Факультет програмної інженерії та бізнесу

Кафедра інженерії програмного забезпечення

Пояснювальна записка до дипломного проекту

магістра
(освітній ступінь)

на тему«Програмне забезпечення для навчання дактилології»

ХАІ.603.667п1.121.186381.20В

Виконав: студент 6 курсу групи №667п1
Спеціальність 121 – Інженерія програмного забезпечення
(код та найменування)
Освітня програма Хмарні обчислення та Інтернет речей
(найменування)
Бєлов Д.В
(прізвище й ініціали студента)
Керівник: Туркін І.Б.
(прізвище й ініціали)
Рецензент: Лященко О.С.
(прізвище й ініціали)

Харків – 2020

Міністерство світи і науки України
Національний аерокосмічний університет ім. М. Є. Жуковського
«Харківський авіаційний інститут»

Факультет програмної інженерії та бізнесу
(повне найменування)
Кафедра інженерії програмного забезпечення
(повне найменування)
Рівень вищої освіти другий (магістерський)
Спеціальність 121 – інженерія програмного забезпечення
(код та найменування)
Освітня програма хмарні обчислення та Інтернет речей
(найменування)

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ І.Б. Туркін
(підпис) (ініціали та прізвище)
“_____” _____ 2020 року

З А В Д А Н Н Я
НА ДИПЛОМНИЙ ПРОЕКТ СТУДЕНТУ

Бєлову Денису Віталійовичу
(прізвище, ім’я, по батькові)

1. Тема дипломного проекту Програмне забезпечення для навчання дактилології

керівник дипломного проекту Туркін Ігор Борисович, д.т.н., професор
(прізвище, ім’я, по батькові, науковий ступінь, вчене звання)
затверджені наказом Університету № _____ від “____” _____ 2020 року
2. Термін подання студентом проекту 24.11.2020
3. Вихідні дані до проекту виконати аналіз проблем соціальної адаптації людей з вадами слуху, розробити прототип програмного забезпечення для вирішення покращення процесу навчання дактилогії
4. Зміст пояснювальної записки (перелік питань, які потрібно розробити Провести аналіз існуючих проблем людей з вадами слуху. Провести аналіз існуючих методів та рішень для покращення комунікації та соціальної адаптації для людей з порушеннями слуху.
5. Перелік графічного матеріалу Об’єкт, предмет та методи досліджень – 1 слайд, Мета та задачі роботи – 1 слайд, аналіз предметної області та існуючих рішень – 5 слайдів, моделі і методи розпізнавання жестів – 4 слайд, розроблення прототипу програмного забеспечення для навчання дактилології – 8 слайдів, висновки – 1 слайд

6. Консультанти розділів проекту

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1÷3	Туркін І.Б., Зав. Каф. 603 д.т.н. професор		

Нормоконтроль _____ В.А. Постернакова____ «___»_____20__р.
(підпис) (ініціали та прізвище)

7. Дата видачі завдання «02» 09 2020 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проекту	Строк виконання етапів проекту	Примітка
1	Огляд і аналіз сервісів для допомоги в соціальній адаптації людей з вадами слуху	02.09.20-08.09.20	
2	Аналіз методів навчання дактилогії	09.09.20-15.09.20	
3	Аналіз методів розпізнавання жестової мови	19.09.20-29.09.20	
4	Розробка методу і алгоритмів розпізнавання жестової мови в реальному часі	01.10.20-	
5	Розробка прототипу ПЗ	09.10.20	
6	Оформлювання пояснювальної записки до дипломного проекту	10.10.20-14.10.20	
7	Підготовка доповіді.	15.10.20	
8	Підготовка презентації	16.10.20-17.10.20	

Студент _____ **Бєлов Д.В.**
(підпис) (прізвище та ініціали)

Керівник проекту ____ **Туркін І. Б**
(підпис) (прізвище та ініціали)

РЕФЕРАТ

Пояснювальна записка на дипломний проект: 77 с., 49 рис., 43 джерела.

Рівень сучасних технологій дозволяє людям з різними порушеннями легше асимілюватися з суспільством. Основною проблемою залишається проблема комунікації людей з вадами зору або слуху з іншими людьми. У той час як люди з вадами зору можуть використовувати звичайну звукову мову, людям з вадами слуху потрібні наявність навичок мови жестів у осіб з якими вони спілкуються. До того ж, коммукаціонний бар'єр значно заважає або навіть не дозволяє людям з проблемами слуху брати участь в конференціях в якості доповідача.

Метою дипломного проекту є створення прототипу програмного забезпечення, який дозволить навчатись дактилогії з перекладом жестів на основі відеоряду в реальному часі. Це дозволить частково вирішити проблему комунікації з людьми з порушеннями слуху і дасть їм можливість легше асимілюватися з суспільством.

Для досягнення даної мети в дипломному проекті були виконані наступні завдання:

- проведено аналіз проблеми асиміляції і комунікації з суспільством для людей з обмеженнями слуху;
- проаналізовані існуючі системи та сервіси для допомоги людям з обмеженнями слуху, а також пов'язані з ними методи і підходи;
- обрана проблема комунікації для її подальшого аналізу і пошуку можливих рішень;
- проведено аналіз сучасних рішень для детекції та класифікації жестів на основі відео потоку;
- проведено архітектурне і функціональне планування прототипу додатка для трансляції жестів в текст;
- реалізований прототип додатка;

Актуальність поставленого завдання обумовлена відсутністю відповідного рішення проблеми комунікації для людей з порушенням слуху.

Практичним результатом проекту є розробка прототипу додатка, яке дозволити ефективніше навчати людей мові жестів, що в свою чергу дозволить скоротити розрив в комунікації між чуючими людьми і людьми з порушеннями слуху.

МАШИННЕ НАВЧАННЯ, РОЗПІЗНАВАННЯ ЖЕСТІВ, ПОРУШЕННЯ СЛУХУ, МОВА ЖЕСТІВ, АНАЛІЗ ВІДЕОПОТОКІВ, РОЗПІЗНАВАННЯ І КЛАСИФІКАЦІЯ

РЕФЕРАТ

Пояснительная записка на дипломный проект: 77 с., 49 ил., 43 источника.

Уровень современных технологий позволяет людям с различными нарушениями легче ассимилироваться с обществом. Основной проблемой остается проблема коммуникации людей с недостатками зрения или слуха с другими людьми. В то время как люди с недостатками зрения могут использовать обычные звуковые язык, людям с недостатками слуха требуются наличие навыков языка жестов у лиц с которыми они общаются. К тому же, коммукационный барьер значительно мешает или даже не позволяет людям с проблемами слуха брать участие в конференциях в качестве докладчика.

Целью дипломного проекта является создание прототипа программного обеспечения, который даст возможность более эфективного обучения дактилогии и языку жестов. Это позволит частично решить проблему коммуникации с людьми с нарушениями слуха и даст им возможность легче ассимилироваться с обществом.

Для достижения данной цели в дипломном проекте были выполнены следующие задания:

- проведен анализ проблемы ассимиляции и коммуникации с обществом для людей с ограничениями слуха;
- проанализированы существующие системы и сервисы для помощи людям с ограничениями слуха, а также связанные с ними методы и подходы;
- выбрана проблема коммуникации для её последующего анализа и поиска возможных решений;
- проведен анализ современных решений для детекции и классификации жестов на основе видео потока;
- проведено архитектурное и функциональное планирование прототипа приложения для трансляции жестов в текст;
- реализован прототип приложения;

Актуальность поставленного задания обусловлена отсутствием подходящего решения проблемы коммуникации для людей с нарушением слуха.

Практичным результатом проекта есть разработка прототипа приложения, которое даст возможность более эфективного обучения дактилогии и языку жестов, что в свою очередь позволит сократить разрыв в коммуникации между слышащими людьми и людьми с нарушениями слуха.

МАШИННОЕ ОБУЧЕНИЕ, РАСПОЗНОВАНИЕ ЖЕСТОВ, НАРУШЕНИЯ СЛУХА, ЯЗЫК ЖЕСТОВ, АНАЛИЗ ВИДЕОПОТОКА, РАСПОЗНОВАНИЕ И КЛАСИФИКАЦИЯ

ABSTRACT

Master’s thesis: 77 pages, 49 figures, 43 references.

The level of modern technology makes it easier for people with various disabilities to assimilate into society. The main problem remains the problem of communication of people with visual or hearing impairments with other people. While visually impaired people can use common sound language, hearing impaired people require sign language skills from the people with whom they communicate. In addition, the communication barrier significantly hinders or even prevents people with hearing impairments from taking part in conferences as a speaker.

The aim of the thesis is to create a software prototype that will allow people to learn sign language more efficiently. This will partially solve the problem of communication with people with hearing impairmants and will make it easier for them to assimilate with society.

To achieve this goal, the following tasks were completed in the thesis:

- the analysis of the problem of assimilation and communication with society for people with hearing impairments was carried out;
- analyzed existing systems and services for helping people with hearing impairments, as well as related methods and approaches;
- the problem of communication was selected for its subsequent analysis and search for possible solutions;
- analysis of modern solutions for the detection and classification of gestures based on the video stream;
- carried out the architectural and functional planning of the prototype of the application for translating gestures into text;
- implemented a prototype of the application;

The relevance of the task is due to the lack of a suitable solution to the problem of communication for people with hearing impairment.

The practical result of the work is the development of a prototype application that will allow to learn sign language more efficiently, which in turn will reduce the gap in communication between hearing people and people with hearing impairments.

MACHINE LEARNING, GESTURE RECOGNITION, HEARING IMPAIRMENT, GESTURE LANGUAGE, VIDEO STREAM ANALYSIS, RECOGNITION AND CLASSIFICATION

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

НН – нейрона мережа.

ПЗ – програмне забезпечення.

Дактилогія (дактилологія) – система пальцевих знаків, що використовується як засіб комунікації, своєрідна форма мовлення, що базується на використанні пальців рук.

СКБД – система контролю баз даних

UI – user interface

UX – user experience

ВСТУП

Актуальність теми. Люди, які мають вади слуху належать до соціально незахищеного шару суспільства. Їх кількість у світі, згідно ВОЗ, оцінюється в 466 мільйонів, що більше ніж 5% від світової популяції. При цьому основні проблеми через вади слуху відносяться до проблем в комунікації між людьми. Існують різні рішення для вирішення данних проблем, але більшість із них потребує наявності людини-сурдоперекладача, або додаткового обладнання.

Роботи провідних науковців та світових організацій (перелік береться з посилань підрозділу 1.1.1) сконцентровані на питаннях:

- доступності додатків для людей з вадами слуху;
- створення додатків для покращення соціальної адаптації людей з вадами слуху
- створення та оцінки дизайну додатків та мобільних додатків у частності з урахуванням особливостей їх використання людьми з вадами слуху.

Між тим завдання точного розпізнавання жестів в реальному часі для навчання мові жестів залишається досі не вирішеним, саме це й зумовлює актуальність дипломного проекту.

Мета роботи. Підвищити точність розпізнавання жестів в реальному часі при навчанні дактилогії шляхом застосування штучних нейроних мерех та ефективних методів їх навчання.

Завдання дослідження наступні:

1 Виконати аналіз проблеми соціалізації людей з вадами слуху, розглянути існуючі програмні рішення та інструментальні засоби, моделі та методи для навчання дактилогії.

2 Розробити моделі штучних нейроних мереж та методи аналізу зображення для класифікації жестів.

3 Розробити прототип програмного забезпечення для класифікації жестів на основі відеоряду.

4 З використанням розробленого прототипу ПЗ виконати аналіз швидкодіїї та точності класифікації жестів за різних умов оточуючої середи.

Об’єкт дослідження. Інформаційна технологія допомоги в навчанні дактилогії.

Предмет дослідження. Моделі та методи розпізнавання та класифікування жестів в реальному часі на основі відеоряду

Методи дослідження.

Для аналізу проблем соціалізації людей з вадами слуху та існуючих засобів для сурдоперекладу використано методи системного аналізу та класифікації. Для розпізнавання характеристик жестів на основі відоряду буде використана модель машинного навчання на основі згорткової нейронной мережі. Для розробки моделі штучних нейронних мереж використано

архітектуру побудовану на основі моделі MobileNetV2. В якості основного інструменту для побудови та тренування моделі використовується фреймворк TensorFlow. Розроблення прототипу ПЗ виконано з використанням ООП-парадигми та фреймворку Windows Forms. Для оцінки ефективності розробленого методу застосовано методи математичної статистики в завданнях оцінки точності та вірогідності розпізнавання жестів в реальному часі.

Наукова новизна. Вдосконалено інформаційну технологію автоматичного сурдоперекладу у текст за рахунок використання штучних нейроних мереж для обробки відеоряду, що дозволяє підвищити точність розпізнавання жестів в реальному часі.

Практична значимість. Надання можливості більш єфективного навчання мові жестів з автоматичним перекладанням без допоміжного обладнання або допомоги сурдоперекладача.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ІСНУЮЧИХ РІШЕНЬ

1.1 Аналіз проблеми соціальної адаптації людей з порушенням слуху

Згідно з дослідженями Всесвітньої Організації Здоров'я [38], станом на 2018 рік у світі нараховується 466 мільйонів людей з порушеннями слуху. За попередніми розрахунками, до 2030-го року ця кількість збільшиться до 630 мільйонів, а до 2050-го року – до 900 мільйонів.

Важко переоцінити вплив часткової чи повної втрати слуху на соціальну складову людини. Так як слуховий апарат людини є одним із найінформативніших джерел інформації на рівні з зоровим апаратом, то зниження якості слуху або повна його втрата дуже негативно впливає на соціальну складову людського життя.

Втрата слуху несе за собою багато проблем серед яких:

- погіршення орієнтації через відсутність дублюючих знаків для голосових повідомлень;
- відсутність можливості вести конференції або презентації без перекладача;
- погана соціальна комунікація;
- проблеми з отриманням освіти.

Для деяких проблем існують рішення та створюються спеціальні державні програми. Проте в Україні відсутні гранти на навчання сурдоперекладачів та, згідно з статистикою [37], оплата праці для них знаходиться найнижчому рівні у Європі. Через це в Україні більшість працівників держслужб не володіють жестовою мовою ні на якому рівні.

Важливою проблемою також є недоступність освіти для людей з вадами слуху. Більшість вузів України не мають належного устаткування щоб надавати людям з вадами слуху потрібну інформацію.

Як ми бачимо, основна проблема пов'язана з комунікацією і відсутністю технічного забезпечення для комунікації в специфічних ситуаціях.

Висновок: на даний момент відсутнє рішення для перекладу з мови жестів яке б могло бути використане у більшості ситуацій, таких як спілкування з оточуючими, проведення презентацій та публічних виступів та інші. Обмеження в можливості спілкування з оточуючими значно погіршує якість життя людей з вадами слуху, тому наявність програмного рішення могла б значно покращити певні аспекти взаємодії із соціумом.

1.1.1 **Огляд програмно-апаратних рішень для допомоги у соціальній адаптації людей з порушенням слуху**

1.1.1.1 Microsoft Kinect Sign Language Translator

Рисунок 1.1 – Головне вікно програми

Опис: Система на основі Kinect може фіксувати знаки, зроблені людиною; розпізнати значення цього знака, використовуючи сам знак, а також положення тіла та поставу; а потім вона переводить знаки на розмовну мову чи письмові слова. У системі Kinect є аватар, який буде переводити розмовні слова слухової особи на знаки для людини з порушеннями слуху.

Данний проект знаходиться на стадії прототипу. Більшість досліджень проводиться у Китаї.

Переваги:

- переклад як з мови жестів в звичайну мову так и навпаки;
- підтримка розмовного перекладу.

Недоліки:

- необхідність у використанні пристроя Microsoft Kinnect;
- непортативне рішення.

1.1.1.2 Motionsavvy

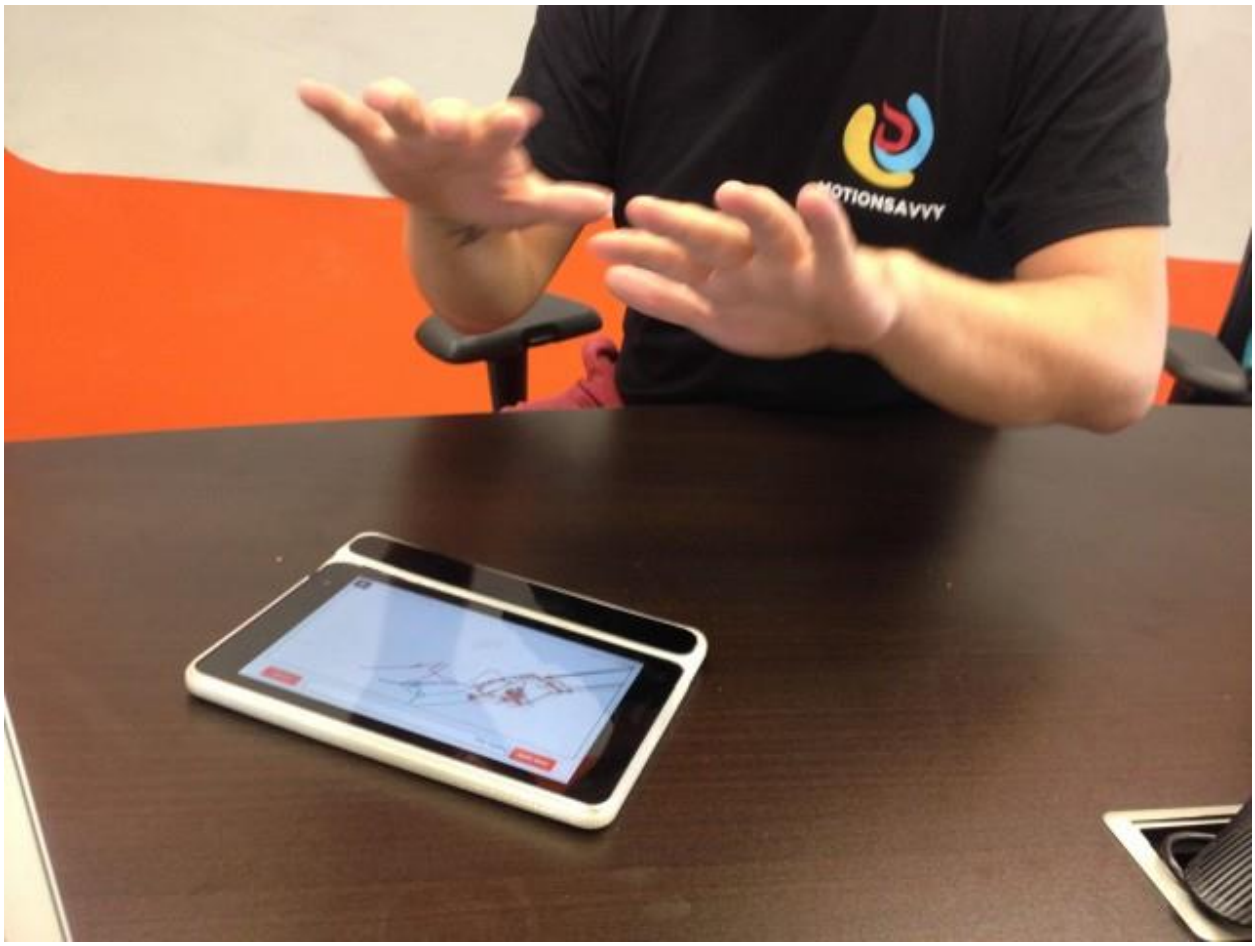

Рисунок 1.2 – Motionsavvy

Опис: Компанія створила програмне забезпечення, котре дозволяє розпізнавати жести за допомогою комбінації планшету та ІК-пристрою MotionLeap.

Переваги:

- переклад як з МЖ в звичайну мову так и навпаки;
- портативність.

Недоліки:

- необхідність у використанні пристрою MotionLeap;
- MotionLeap має відомі проблеми зі зчитуванням рук, що перетинаються. Можливо ця проблема існує у даному програмному забезпеченні.

1.1.1.3 LeapMotion

Leap Motion - це технологія, що розробляється, заснована на захопленні руху, для людино-комп'ютерного взаємодії.

The Leap - це невеликий USB-пристрій, розроблений для столу користувачів, робочою частиною розташовується вгору, тим самим створюючи 3D-область взаємодії об'ємом близько 227 дециметрів кубічних (тобто уявний куб зі стороною 61 см).

Leap Motion розповсюдила тисячі пристроїв безкоштовно розробникам, які зацікавлені в розробці додатків під нього.

Рисунок 1.3 – Приклад роботи

1.1.1.4 BeWarned

BeWarned – це технологічний стартап, націлений на створення низки сервісів з метою полегчення життя для людей з вадами слуху.

До списку сервісів стартапу входять:

– BeWarned Connect – сервіс, який допомогає комунікації людей шляхом конвертації тексту у голос та голосу у текст.
– BeWarned Dance – сервіс, який дозволяє людям з вадами слуху відчувати музику шляхом її візуалізації через мобільний пристрій.

– BeWarned Emergency Call – сервіс, за допомогою якого слабочуючий користувач може викликати чуючих людей на допомогу, якщо потрібно.
– BeWarned Sound Monitor – сервіс, який вирішує проблему безпеки, попереджуючи користувача про наявність небезпечних звуків (звук сигналу автомобіля, баркіт собаки, крик людини, сирена) [39].

Рисунок 1.4 – BeWarned

Висновки: існує багато рішень, які допомагають людям з вадами слуху соціалізуватись. Більшість із них вирішує різні проблемі, починаючи зі спілкування між людьми і закінчуючи відтворення музикальних витворів. Але немає зручного рішення, яке б дозволяло людям з вадами слуху спілкуватись з чуючими людьми за допомогою мови жестів, за умови, що чуюча людина не знає мови жестів. Тому серед можливих та актуальних завдань для подальшого вивчення в роботі магістра обрано методи розпізнавання та класифікації жестів на основі відеоряду для її подальшого вікористання в задачі сурдоперекладу.

1.1.2 **Аналіз наукових публікацій з вирішення проблеми соціальної адаптації людей з порушенням слуху**

У вільному доступі знаходиться велика кількість публікацій на тему створення додатків для допомогу людям з порушеннями слуху. У їх числі публікації на тему допомоги соціалізації людям з порушенням слуху. Далі розглянемо деякі з них:

Design and Implementation of Application for the Hearing Impaired People [40]

У цій публікації розглядається створення мобільного додатку для допомоги людям з порушенням слуху. Також розгладаються різні підходи та практики створення мобільних додатків для людей з різними типами фізичних порушень. Приводиться статистика щодо розповсюдження різних типів порушень серед населення, статистика серед користувачів різних девайсів. Також показані результати опитання щодо знань мови жестів серед населення та щодо загальних знаннь шодо порушень слуху.

Mobile Device Apps for People with Hearing Loss. Expanding the Horizons of Hearing Access [41]

Ця стаття розгладає вбудовані можливості смартфонів для допомоги людям з порушенням слуху. Також вона описує деякі доступні додатки які надають функціональність яка полегшує життя для людей з порушеннями слуху. До списку розгланутих технологій входять: визуальна нотіфікація та нотіфікація вібрацією, налаштування гучності, субтитри, трансформація мови у текст та інші.

Assistive Android Application For Hearing Impaired People Using Sign Language [42]

У цій публікації розглядається створення мобільного додатку для допомоги у спілкуванні між людьми з порушенням слуху та чуючими людьми. Додаток працює в оба боки – може перекладати з тексту у жести та навпаки, але технологія лише співставляє зображення жесту та символ вербальної мови, через що не підтримується переклад у реальному часі за допомогою зображення користувача.

Dimensions for hearing-impaired mobile application usability model [43]

Дана публікація розглядає виміри оцінки якості використання мобільних додатків для використання людьми з вадами слуху.

В публікації виділяють наступні виміри:

- ефективність (Efficiency) - швидкість виконання користувачами завдання та досягнення цілей продукту. Також забезпечення продуктивності роботи користувача під час використаня програми.
- задоволення (Satisfaction) - успіх програми та наскільки приємно користуватися нею.
- навчальність (Learnability) – кількфіість зусиль потрібних для легкого використання додатку.

- ефективність (Effectiveness) – Міра якості інтерфейсу та точності і повноти досягнених користувачем цілей.
- зрозумілість (Understandability) – Міра розуміння додатку користувачем.
- доступність (Accessibility) – Міра повноти та зручності використання додатку не зважаючи на фізичні вади.

1.2 **Аналіз існуючих методів та моделей розпізнання жестів**

Методи розпізнавання в системах комп'ютерного зору можна розділити на два види – методи на основі створення тривимірної моделі руки й методи на основі виділення ознак. Методи на основі створення тривимірної моделі руки базуються на побудові кінематичної моделі, яка враховує всі можливі ступені свободи. Для цього потрібно оцінити жести руки за допомогою порівняння положення руки на вхідному зображенні та двовимірної проекції моделі жесту з бази даних. Такі методи потенційно дозволяють розпізнавати значну кількість жестів.

Основні пристрої зчитування даних для розпізнавання жестів:

1 Електронні рукавички. Даний пристрій може зчитати інформацію про позицію та кут повороту рук за допомогою магнітних та інерціальних датчиків.
2 Depth-aware cameras. Певні види камер дають змогу отримати карту глибин об'єкту спостереження. За допомогою цієї карти є можливість відтворити його тривимірну репрезентацію.
3 Стереокамери. Використовуючи дві камери, положення яких одна відносно іншої відомо заздалегідь, можна отримати тривимірне зображення об'єкту спостереження.
4 ІК-пристрої. Деякі пристрої можуть відтворити 3д модель пристрою за допомогою інфочервоних датчиків.

Одиночна камера. Звичайна 2д камера може бути використана для розпізнавання жестів. Вважається, що даний метод отримання інформації не являється настільки ж ефективним як інші, але він являється найбільш доступним для людей.

Останній спосіб є найбільш відповідним, тому що він являється найдоступнішим для кінцевого користувача. Так, в процесі перекладу може бути використана звичайна веб-камера або камера смартфону. Разом з цим з'являється ряд проблем адже якість зображення камери залежить від її технічних характеристик, а також від умов навколишнього середовища.

1.2.1 **Методи аналізу даних**

1.2.1.1 Фільтрація

До цієї групи входять методи, які дають змогу виділити на зображеннях необхідні області без їх аналізу. Більша часина цих методів виконує якесь єдине перетворення до всіх точок зображення. На моменті фільтрації аналіз зображення не робиться, але точки, котрі проходять фільтрацію, можна розглядати як області із особливими характерностями.

1.2.1.2 Вейвлети

Вейвлет-аналізом зображення називається пошук довільного патерна на зображенні за допомогою згортки з моделлю цього патерна. Існує набір класичних функцій, які використовуються у вейвлет-аналізі. До них відносяться вейвлет Хаара, вейвлет Морле, вейвлет мексиканська шляпа, тощо. Типи вейвлетів надано на рисунку 1.5.

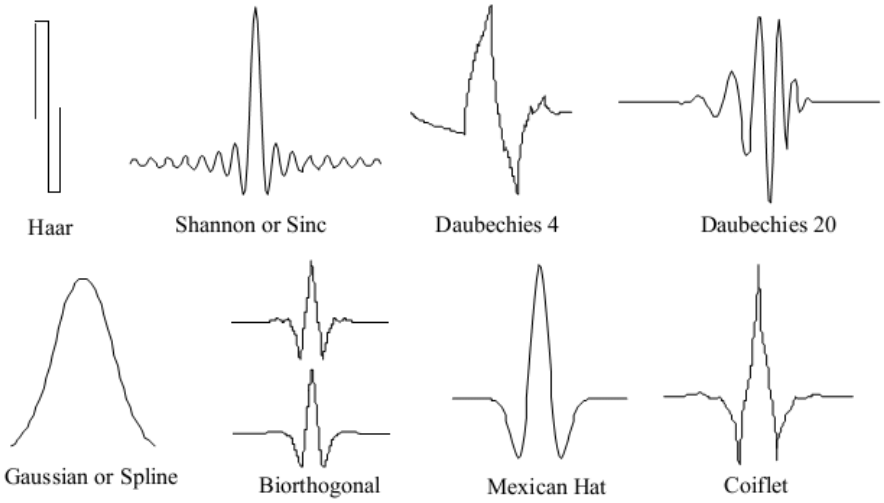

Рисунок 1.5 – Типи вейвлетів

1.2.1.3 Бінарізація за порогом

Найбільш просте та розпоширине перетворення – це бінарізація зображення за порогом. Для RGB зображення та зображення у відтінках сірого порогом є значення кольору.

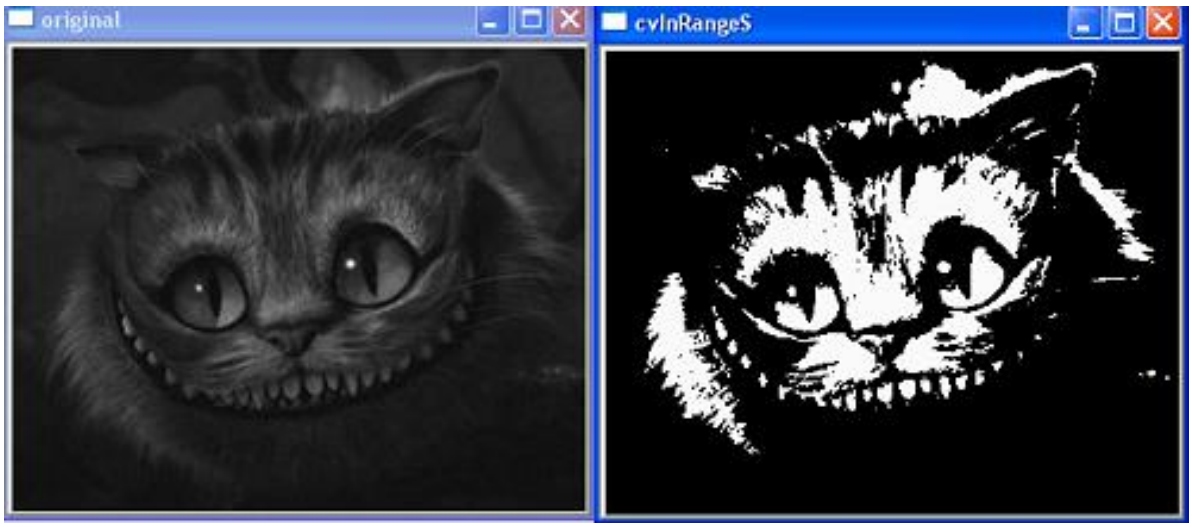

Рисунок 1.5 – Приклад бінарізації за порогом

1.2.1.4 Контури

Знаходження контурів об’єктів є ефективним методом переходу від роботи над зображенням до роботи над об’єктом зображення. У випадку, коли об’єкт має складну геометричну форму, але гарно виділяється на фоні, то, частіш за все, виділення контурів повністю вирішує задачу фільтрації.

Існують наступні методи, які вирішують задачу фільтрації контурів:

- Оператор Кенні;
- Оператор Соблея;
- Оператор Лапласа;
- Оператор Прюітт;
- Оператор Робертса.

У більшості випадків задачу розпізнавання контурів можна вирішити саме оператором Кенні.

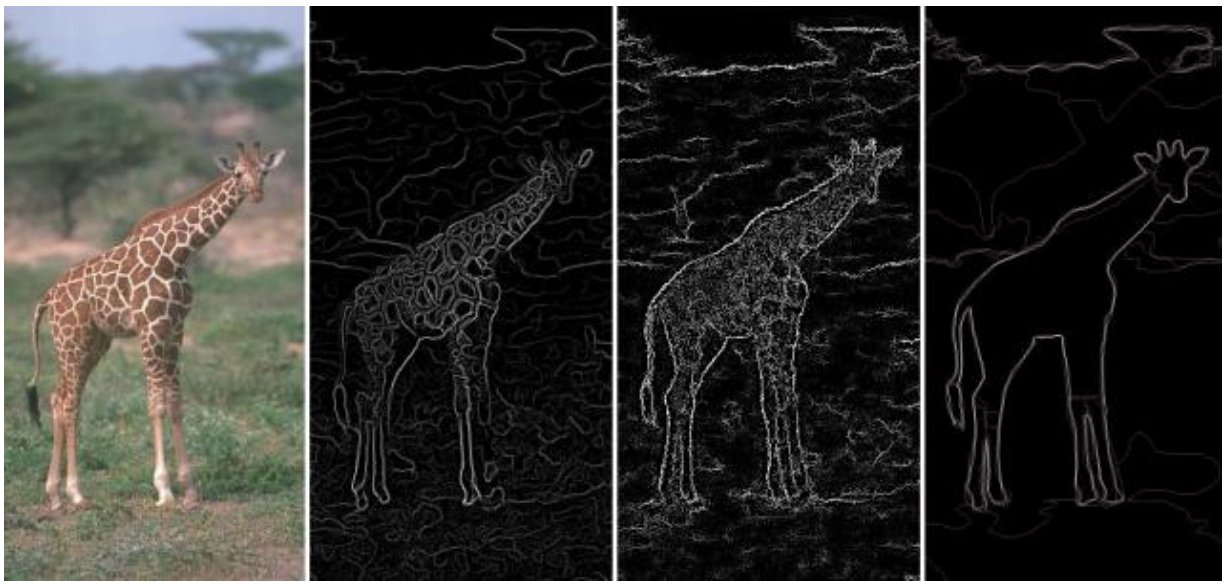

Рисунок 1.6 – Приклад алгоритмів розпізнавання контурів

Контурний аналіз дозволяє описувати, зберігати, порівнювати і робити пошук об'єктів, представлених у вигляді своїх зовнішніх обрисів - контурів. Передбачається, що контур містить всю необхідну інформацію про форму об'єкта. Внутрішні точки об'єкта до уваги не приймаються. Це обмежує область застосування алгоритмів контурного аналізу, але розгляд контурів дозволяє перейти від двовимірного простору зображення до простору контурів і тим самим знизити обчислювальну і алгоритмічну складність. Ефективним методом аналізу і порівняння обмежують областей (контурів) є аналіз їх моментів.

Момент - це сумарна характеристика контуру, отримана інтегруванням (або підсумовуванням) всіх пікселів контуру.

1.2.1.5 Кореляція

В обробці зображень, фазова кореляція - це метод співставлення зображень, який використовує швидкий підхід з переходом у частотну область для визначення взаємного паралельного зсуву двох однакових зображень.

Рисунок 1.7 – Приклад кореляції зображень

1.2.2 **Логічна обробка результатів фільтрації**

Результатом фільтрації є набір придатних до обробки даних. Але часто їх важко використовувати без попередньої обробки.

1.2.2.1 Контурний аналіз

Задачею контурного аналізу є отримання контуру зображення із його границь. Контур є унікальною характеристикою об’єкта, що дозволяє вирішити задачу класифікації.

1.2.2.2 Особливі точки

Особливі точки – це унікальні характеристики об’єкта, які дозволяють співставити об’єкт із самим собою або з об’єктами схожого класу. Існують багато способів, які дозволяють виділити такі точки. Деякі з них виділяють ці точки через сусідні кадри, а деякі через великий проміжок часу та зміну освітлення. Деякі алгоритми дозволяють знайти особливі точки, які залишаються такими навіть при поворотах об’єкта.

Види особливих точок:

− точки, які залишаються стабільними протягом секунд: дані точки слугують для супроводження об’єкта у сусідніх кадрах відео, або для

склеювання зображення з декількох камер. До таких точок можна віднести локальні максимуми зображення, кути, точки з максимумом дисперсії, визначені градієнти тощо;
- точки, які залишаються стабільними при зміні освітлення та невеликих рухах об’єкта: дані точки слугують для навчання з подальшою класифікацією типів об’єктів. Наприклад класифікатор обличчя – це продукт системи, створеної саме на таких точках;
- стабільні точки: дані точки залишаються стабільними навіть при повороті зображення. Алгоритми знаходження таких точок набагато складніші, їх менше, та зачасту, вони запатентовані.

1.2.3 **Навчання**

Після локалізації об’єкта дослідження та виділення стабільних точок або контурів використовуються методи, які дозволяють приймати рішення. Описати цей процес можна наступним чином: Знаходиться тестова вибірка, яка має декілька классів об’єктів. Для кожного зображення є набір ознак, котрі були виділені за допомогою деякого алгоритму. Алгоритм навчання повинен створити таку модель, за допомогою яких він зможе проаналізувати нове зображення та зробити висновок, який із об’єктів знаходиться на зображенні. До алгоритмів навчання відносяться такі алгоритми, як нейроні мережі, регресії, k-means кластеризатори, AdaBoost класифікатори, тощо.

1.2.4 **Метод активних контурів без попереднього виділення кордонів**

На відміну від традиційного способу активних контурів цей метод не вимагає попереднього виділення кордонів об'єкта зображення, а вихідне зображення не обов'язково згладжувати. Крива, або змійка (замкнутої округлої форми), рухається з довільної точки зображення. При перетині кордону вона починає деформуватися і приймати форму об'єкта на зображенні, як би заповнюючи внутрішню його частину.

1.2.5 **Детектор кордонів Кенні**

Дж. Кенні вивчив математичну проблему отримання фільтра, оптимального за критеріями виділення, локалізації та мінімізації декількох відгуків одного краю. Це означає, що детектор повинен реагувати на межі, але при цьому ігнорувати помилкові, точно визначати лінію кордону і реагувати

на кожен кордон один раз, що дозволяє уникнути сприйняття широких смуг зміни яскравості як сукупності кордонів.

Алгоритм включає в себе:

- згладжування - розмиття зображення для видалення шуму;
- пошук градієнтів - межі відзначаються там, де градієнт
- зображення набуває максимальне значення;
- придушення немаксімумов - тільки локальні максимуми відзначаються як кордони;
- подвійну порогову фільтрацію - потенційні кордону
- визначаються порогами;
- трасування області неоднозначності - підсумкові кордони;
- встановлюються шляхом придушення всіх країв, не пов'язаних з певними (сильними) межами.

Для зменшення чутливості алгоритму до шуму застосовується перша похідна від Гауссіана. Після застосування фільтра, зображення стає злегка розмитим (рисунок 1.8).

Рисунок 1.8 – Приклад роботи контурного детектора Кенні

1.2.6 **Відстеження контурів**

Метод полягає в послідовному кресленні кордону між об'єктом і фоном. Простежує точка у вигляді «жука» повзає по зображенню до тих пір, поки не доходить до темної області (об'єкт). Тоді «жук» повертається ліворуч і рухається по кривій, поки не досягне меж об'єкта, після цього повертається направо і повторює процес, поки не досягне околиці початкової точки (рисунок 1.9).

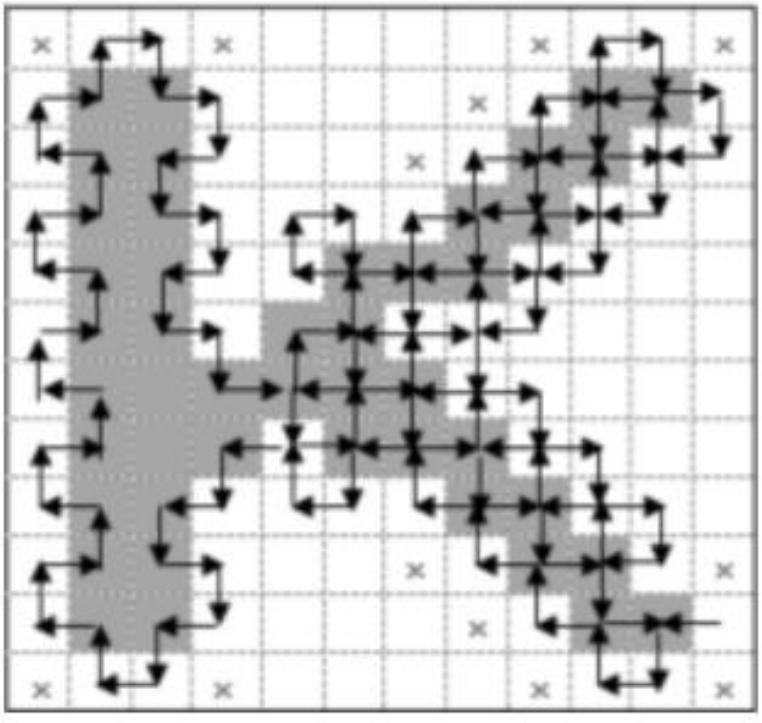

Рисунок 1.9 – Приклад алгоритму «Жука»

1.2.7 **Існуючі алгоритмми**

1.2.7.1 Локалізація жестів

Першим алгоритмом виділення рук на зображенні було обрано фільтрацію по кольору шкіри. Даний метод полягає в тому, щоб виділити пікселі кольору шкіри із зображення. Дані про колір шкіри можна отримати із кольору обличчя. Результатами дослідження були незадовільні, так як усі тестові данні були у схожих до шкіри тонах. До стандартних каскадів Хаара також входять класифікатори долоні, кулака, руки тощо, але вони працюють набагато гірше аніж відповідний класифікатор обличчя. Каскади Хаара дуже чуттєві до кута нахилу зображення, а отже, за допомогою них неможливо зробити класифікатор рук у всіх можливих положеннях.

1.2.7.2 Виділення рук на основі аналізу рухів

Для подальших досліджень було зроблено припущення, що мова жестів за своєю природою є рухом, та саме області руху є найбільш інформативними для аналізу відеоряду.

1.2.7.3 Оптичний потік та лінії Фарнебеку

Є багато методів аналізу рухів на зображенні. Найбільш інформативний, та складний з точки зору обчислення є підрахунок оптичного потоку кадрів. Основною ідеєю цього методу є обчислення зсуву пікселей у сусідніх зображеннях.

Рисунок 1.10 – Приклад знаходження оптичного потоку за допомогою алгоритму Фарнебека

1.2.8 **Розпізнавання жестів**

1.2.8.1 Метод гістограм кольорів

Ідея методу гістограм кольорів для порівняння зображень зводиться до наступного. Уся множина кольорів розбивається на набір непересічних, повністю охоплюючих зображення підмножин. Для зображення формується гістограма, що відображає частку кожної підмножини кольорів в кольоровій гамі зображення. Для порівняння гістограм вводиться поняття відстані між ними.

При розбитті RGB-кольорів за яскравістю обчислюється інтенсивність кожного кольору. Отримане значення, вкладене в проміжок між 0 і 255, потрапляє в один з інтервалів, на які розбивається діапазон можливих значень. Як відстань між гістограмами використовується сума модулів різниці відповідних елементів гістограм; деякий удосконалення методу досягається при обчислюванні відстані на підставі поелементного порівняння гістограм з

урахуванням сусідніх елементів. Цей метод найбільш ефективний для чорнобілих зображень.

Для кольорових RGB-зображень кращі результати дає інший спосіб - розбиття RGB-кольорів на прямокутні паралелепіпеди. Кольорове RGB-простір розглядається як тривимірний куб, кожна вісь якого відповідає одному з трьох основних кольорів (червоний, зелений або синій. При такому розгляді будь-який колір RGB-зображення може бути представлений точкою куба. Для побудови гістограми кольорів кожна сторона ділиться на 4 рівних інтервалу, відповідно RGB-куб ділиться на 64 прямокутних паралелепіпеда. Гістограма зображення відображає розподіл точок RGB-простору, відповідних кольорам пікселів зображення. Як відстань між гістограмами використовується покомпонентна сума модулів різниці між ними. Незважаючи на граничну простоту підходу, він показує досить стабільні результати.

Основним недоліком методу колірних гістограм є те, що він втрачає інформацію про просторове розташування об'єктів. Абсолютно різні картинки можуть мати подібні гістограми кольорів. Наприклад, зображення осіннього листя може містити багато невеликих плям червоного кольору. Це дасть подібну колірну гістограму із зображенням великого червоного об'єкта.

1.2.8.2 Метод особливих точок

Не існує універсального чи точного визначення, що собою являє ознака, і точне визначення часто залежить від задачі або типу застосування. Враховуючи це, ознака визначається як «цікава» частина зображення, і ознаки використовуються як відправні точки для багатьох алгоритмів комп'ютерного зору. Оскільки ознаки використовуються як відправні точки та основні примітиви для наступних алгоритмів, загальний алгоритм часто буде лише настільки добрим, наскільки добрим є його детектор ознак. Отже, бажаною властивістю детектора ознак є повторюваність: чи буде одну й ту ж ознаку виявлено на двох або більше різних зображеннях однієї й тієї ж сцени, чи ні.

Виявлення ознак є низькорівневою операцією обробки зображень. А саме, вона зазвичай виконується як перша операція на зображенні, і перевіряє кожен піксель, щоби побачити, чи присутня ознака в цьому пікселі. Якщо це є частиною більшого алгоритму, то цей алгоритм зазвичай перевірятиме зображення лише в областях ознак. Як вбудована передумова для виявлення ознак, вхідне зображення зазвичай згладжується ґаусовим ядром у масштабно-просторовому представленні, й обчислюється одне або декілька зображень ознак, часто виражених в термінах операцій локальних похідних зображень.

Іноді, коли виявлення ознак є обчислювально витратним, і присутні часові обмеження, може застосовуватися алгоритм вищого рівня для скеровування етапу виявлення ознак, так що пошук ознак здійснюватиметься лише деякими частинами зображення.

Багато алгоритмів комп'ютерного зору використовують виявлення ознак в якості першого кроку, так що в результаті було розроблено дуже велику кількість детекторів ознак. Вони сильно різняться за типами ознак, що виявляють, за обчислювальною складністю та повторюваністю.

1.2.8.2.1 Особливі точки

Терміни кути та особливі точки використовуються часом як взаємозамінні, й відносяться до точкових ознак зображення, які мають локальну двовимірну структуру. Термін «кут» виник тому, що перші алгоритми спочатку виконували виявлення контурів, а потім аналізували контури для знаходження швидких змін у напрямку (кутів). Ці алгоритми згодом розвинулися до того, що явне виявлення контурів стало більше не потрібним, наприклад, при аналізі високих значень кривини градієнту зображення. Потім було відмічено, що так звані кути також виявлялися на тих частинах зображення, що не є кутами в традиційному розумінні (наприклад, можуть виявлятися невеликі яскраві плями на темному тлі). Ці точки часто називаються особливими точками, але термін «кут» може використовуватись як традиція.

1.2.8.2.2 SIFT

SIFT(Scale Invariant Feature Transform) – є одним з найпоширеніших та найточніших алгоритмів знаходження та опису особливих точок. Особлива точка у SIFT є ділянка зображення (ключова точка) з пов’язаним до неї описом. Ключові точки знаходяться за допомогою SIFT-детектора. Їх опис розраховує SIFT-дескриптор.

1.2.8.2.3 SIFT-детектор

SIFT-ключова точка – це кругла область зображення з орієнтацією. Вона описується чотирьома параметрами: координатами центра точки x та y, масштабом (радіусом точки) та орієнтацією. Основною перевагою SIFT ключивих точок є їх стійкість до геометричних перетворень, а саме перетворень зсуву, повороту чи масштабуваню.

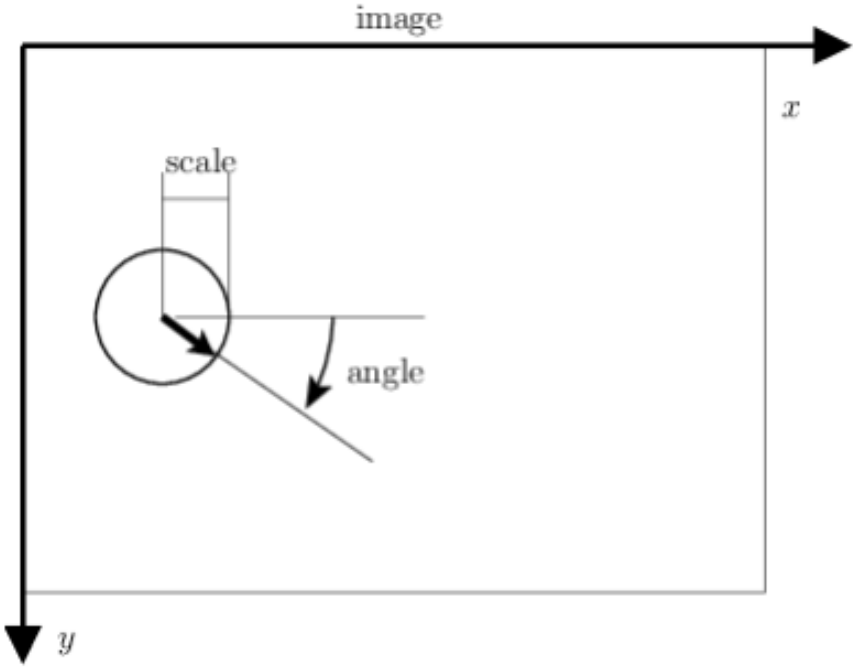

Рисунок 1.11 – Графічне зображення опису SIFT-ключової точки

1.2.8.2.4 SIFT-дескриптор

Дескриптор SIFT є 3-D просторова гістограма градієнтів. Градієнт в кожному пікселі розглядається як зразок тривимірного елементарного вектора ознак, утвореного положенням піксела та орієнтацією градієнта. Зразки нормуються за допомогою норм градієнта, що формує SIFT дескриптор регіону.

1.2.8.3 Розпізнавання знаків за допомогою нейронних мереж та навчання з доглядом.

Насправді, для розпізнавання та інтерпретації жестів, які висловлюють якийсь сенс нам не потрібно мати модель жесту або його обьємне зображення.

Використовуючи згортовані нейронні мережі ми можемо вирішувати задачу класифікації і цього буде достатньо для інтерпретації окремих жестів.

Для того, щоби побудувати модель нейромережі та навчити її розпізнаванню жестів нам необхідно мати доволі великий та різноманітний набір навчальних данних. Також навчальні данні повинні бути заздалегіть прокласифіковані.

Для класифікації візуальних данних зазвичай використовуються згортовані нейромережі. Нижче зображенна їх типічна архітектура

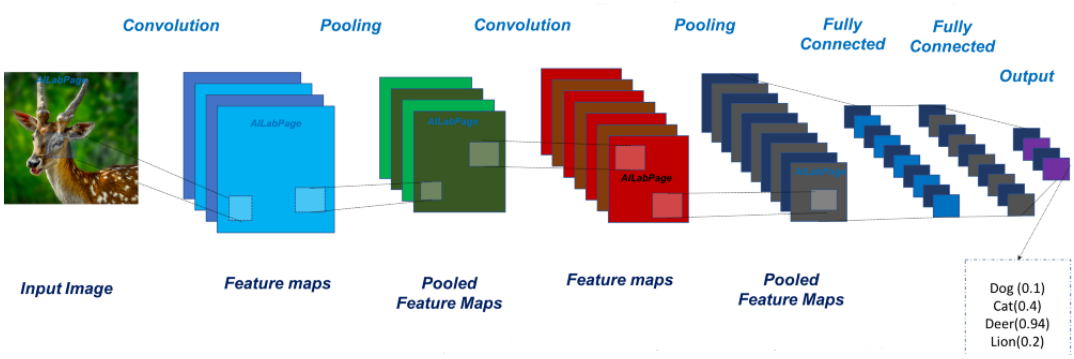

Рисунок 1.12 – Типічна архітектура згортованої нейромережі

Згортовані нейромережі потребують доволі великий набір данних для навчання, проте показують хороші результати в задачі класифікації.

1.3 Огляд інструментальних засобів для обробки відеоряду та машиного навчання

1.3.1 Matlab

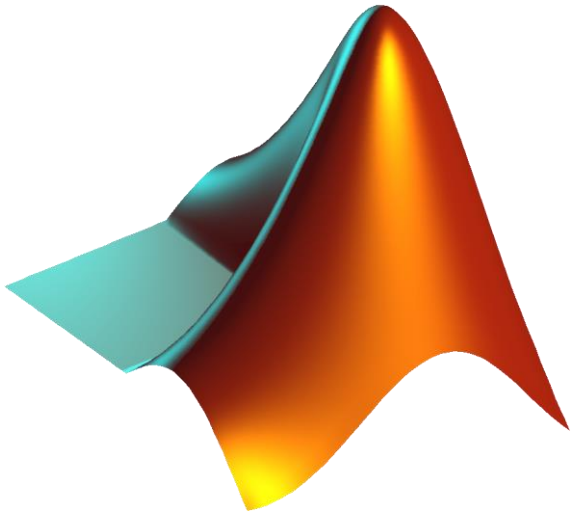

Рисунок 1.13 – Логотип Matlab

Matlab - пакет прикладних програм для числового аналізу, а також мова програмування, що використовується в даному пакеті. Система створена компанією The MathWorks і є зручним засобом для роботи з математичними матрицями, малювання функцій, роботи з алгоритмами, створення робочих оболонок (user interfaces) з програмами в інших мовах програмування. Хоча цей продукт спеціалізується на чисельному обчисленні, спеціальні інструментальні засоби працюють з програмним забезпеченням Maple, що робить його повноцінною системою для роботи з алгеброю.

Matlab має більше, ніж мільйон користувачів на виробництвах і науковців. Ціна базової комерційної версії без інструментів близько 2000 дол. США і лише 100 дол. США для навчальних закладів з мінімальним набором інструментів.

Застосування:

- Matlab надає користувачеві велику кількість функцій для аналізу даних, які покривають майже всі області математики, зокрема:
- Матриці та лінійна алгебра - алгебра матриць, лінійні рівняння, власні значення і вектори, сингулярності, факторизація матриць та інше;
- многочлени та інтерполяція - корені многочленів, операції над многочленами та їх диференціювання, інтерполяція та екстраполяція кривих;
- математична статистика та аналіз даних - статистичні функції, статистична регресія, цифрова фільтрація, швидке перетворення Фур’є та інші;
- обробка даних - набір спеціальних функцій, включаючи побудову графіків, оптимізацію, пошук нулів, чисельне інтегрування та інше;
- диференційні рівняння - вирішення диференційних і диференційно-алгебраїчних рівнянь, диференційних рівнянь із запізнюванням, рівнянь з обмеженнями, рівнянь в часткових похідних та інше;
- розріджені матриці - спеціальний клас даних пакету Matlab, що використовується у спеціалізованих додатках;
- цілочисельна арифметика - виконання операцій цілочисельної арифметики в середовищі Matlab.

1.3.2 **TensorFlow**

Рисунок 1.14 – Логотип 1.6.2 TensorFlow

TensorFlow - відкрита програмна бібліотека для машинного навчання цілій низці задач, розроблена компанією Google для задоволення її потреб у системах, здатних будувати та тренувати нейронні мережі для виявлення та розшифровування образів та кореляцій, аналогічно до навчання й розуміння, які застосовують люди. Її наразі застосовують як для досліджень, так і для розробки продуктів Google, часто замінюючи на його ролі її закритого попередника, DistBelief. TensorFlow було початково розроблено командою Google Brain для внутрішнього використання в Google, поки її не було випущено під відкритою ліцензією Apache 2.0 9 листопада 2015 року. [21]

TensorFlow є системою машинного навчання Google Brain другого покоління, випущеною як відкрите програмне забезпечення 9 листопада 2015 року. В той час як еталонна реалізація працює на одиничних пристроях, TensorFlow може працювати на декількох центральних та графічних процесорах (включно з додатковими розширеннями CUDA для обчислень загального призначення на графічних процесорах). TensorFlow доступна для 64-розрядних Linux, macOS, Windows, та для мобільних обчислювальних платформ, включно з Android та iOS.

Обчислення TensorFlow виражаються як станові графи потоків даних. Назва TensorFlow походить від операцій, що такі нейронні мережі виконують над багатовимірними масивами даних. Ці багатовимірні масиви називають «тензорами». В червні 2016 року Джефф Дін з Google заявив, що TensorFlow згадували 1 500 репозиторіїв на GitHub, лише 5 з яких були від Google.

Данна бабліотека дуже добре підходить для створення та налаштування моделей нейромереж для широкого кола застосувань.

1.3.3 **OpenCV**

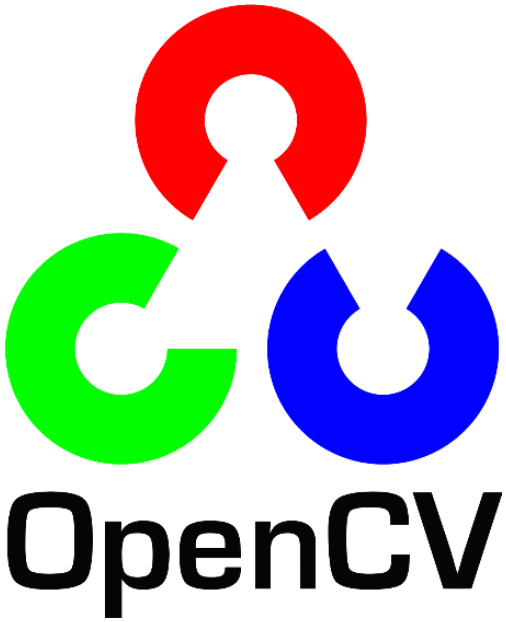

Рисунок 1.15 – Логотип OpenCV

1

OpenCV (англ. Open Source Computer Vision Library, бібліотека комп'ютерного зору з відкритим кодом) - бібліотека функцій та алгоритмів комп'ютерного зору, обробки зображень і чисельних алгоритмів загального призначення з відкритим кодом.

Бібліотека надає засоби для обробки і аналізу вмісту зображень, у тому числі розпізнавання об'єктів на фотографіях (наприклад, осіб і фігур людей, тексту тощо), відстежування руху об'єктів, перетворення зображень, застосування методів машинного навчання і виявлення загальних елементів на різних зображеннях.

Бібліотека розроблена Intel і нині підтримується Willow Garage та Itseez. Серцевий код бібліотеки написаний мовою C++ і поширюється під ліцензією BSD. Біндинги підготовлені для різних мов програмування, таких як Python, Java, Ruby, Matlab, Lua та інших. Може вільно використовуватися в академічних та комерційних цілях.

Бібліотека містить понад 2500 оптимізованих алгоритмів, серед яких повний набір як класичних так і практичних алгоритмів машинного навчання і комп'ютерного зору. Алгоритми OpenCV застосовують у таких сферах:

- аналіз та обробка зображень;
- системи з розпізнавання обличчя;
- ідентифікації об'єктів;

- розпізнавання жестів на відео;
- відстежування переміщення камери;
- побудова 3D моделей об'єктів;
- створення 3D хмар точок зі стерео камер;
- склеювання зображень між собою, для створення зображень всієї сцени з високою роздільною здатністю;
- система взаємодії людини з комп'ютером;
- пошуку схожих зображень із бази даних;
- усування ефекту червоних очей при фотозйомці зі спалахом;
- стеження за рухом очей;
- аналіз руху;
- ідентифікація об'єктів;
- сегментація зображення;
- трекінґ відео;
- розпізнавання елементів сцени і додавання маркерів для створення доповненої реальності;
- та інші.

1.4 **Висновки з розділу 1**

У цьому розділі було розглянуто:

- сучасні рішення для считування жестів;
- методи зчитування;
- методи аналізу даних;
- існуючі алгоритми обробки зображень;
- можливі засоби реалізації.

Наведено переваги та недоліки сучасних рішень, зображено приклади роботи алгоритмів локалізації та класифікації.

2 МОДЕЛІ І МЕТОДИ РОЗПІЗНАВАННЯ ЖЕСТІВ

2.1 Нейронні мережі

2.1.1 Базові поняття штучної нейронної мережі

Нейронна мережа це взаємопов’язана збірка елементів, «юнітів» або «вузлів», чия функціональність побудована на основі тваринного нейрона. [31] Іншими словами, це матиматична модель біологічних процесів мозгу живих істот. Здатність штучних нейронних мереж до обробки даних згідно поставленої задачі отримується за допомогою коефіцієнту з’єднань між різними шарами нейронної мережі, більш відомим як «вага» з’єднання. Цей коєфіцієнт отримується в процессі «навчання» мережі. Під час цього процессу коєфіцієнти з’єднань мережі коригуються на основі паттернів та характеристик отриманиз із вхідних данних.

Щоб краще зрозуміти принцип роботи нейромережі потрібно звернутися до нейробіології.

Нейрон - структурна і функціональна одиниця нервової системи, пристосована до здійснення прийому, обробки, збереження, передачі і інтеграції інформації., формує поведінку, вищу нервову і психічну діяльність живих істот [31]. Вигляд нейрону можно побачити на рисунку нижче. (Рисунок 2.1)

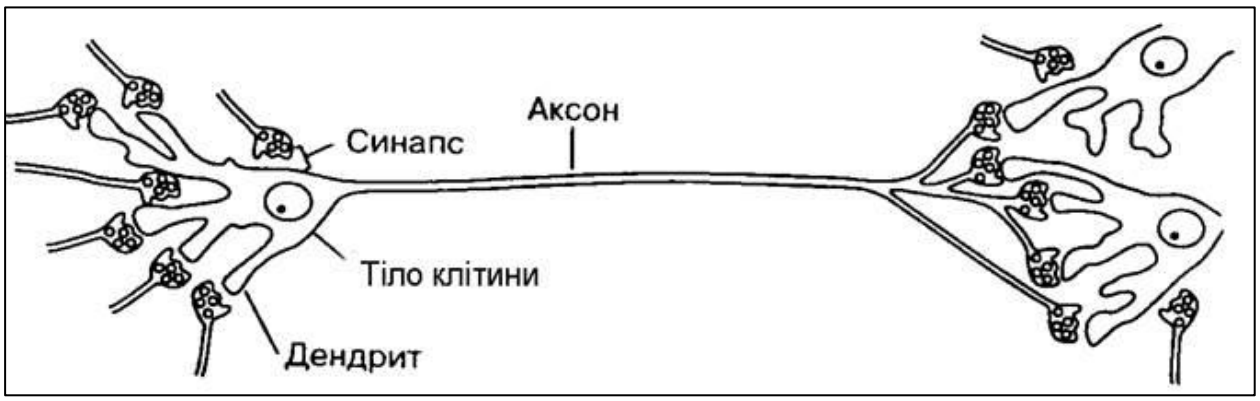

Рисунок 2.1 – Біологічний нейрон

Як видно на рисунку, нейрон має декілька основних частин:

- Тіло клітини. Воно тримає в собі спадкову інформацію про будову нейрона та так звану «функцію активації» яка буде розланута пізніше.
- Аксон. Частина нейрона яка передає нервовий імпульс далі до інших нервових клітин.

– Дендрид та синапс. Це частини клітини які забезпечують з’єднання з іншими нервовими клітинами.

Схожу структуру має і математична модель нейрону. Зображення моделі нейрону можна побачити на рисунку нижче. (Рисунок 2.2)

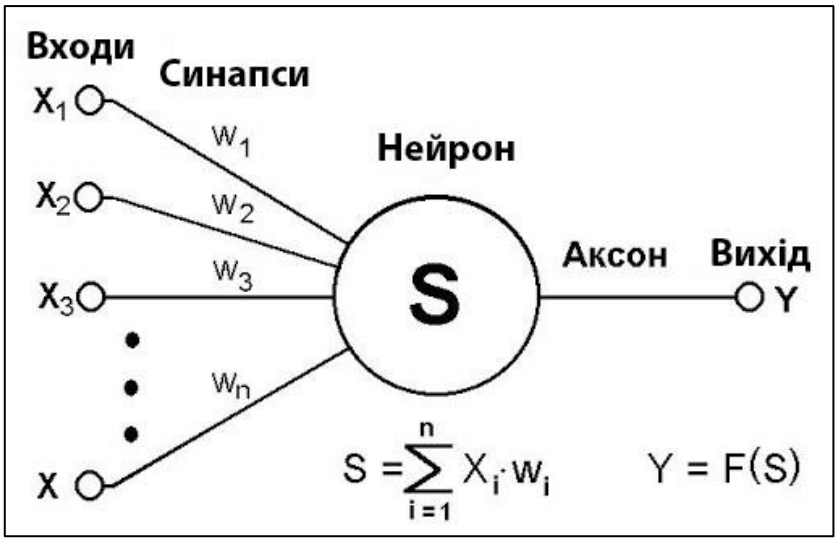

Рисунок 2.2 – Математична модель нейрону

Але нейрон наодинці не містить ніякої цінності в плані розрахунків. Все що він родить це отримує значення, накладує на нього функцію активації, отримує значення, та передає новий імпульс далі в залежності від значення отриманого від функції. Справжну цінність несуть різні комбінації з’єднань нейронів та різні комбінації ваг їх з’єднань. Якраз комбінацію цих з’єднань і називають «нейрона мережа».

На рисунку нижче можно побачити просту структуру нейронної мережі. (Рисунок 2.3)

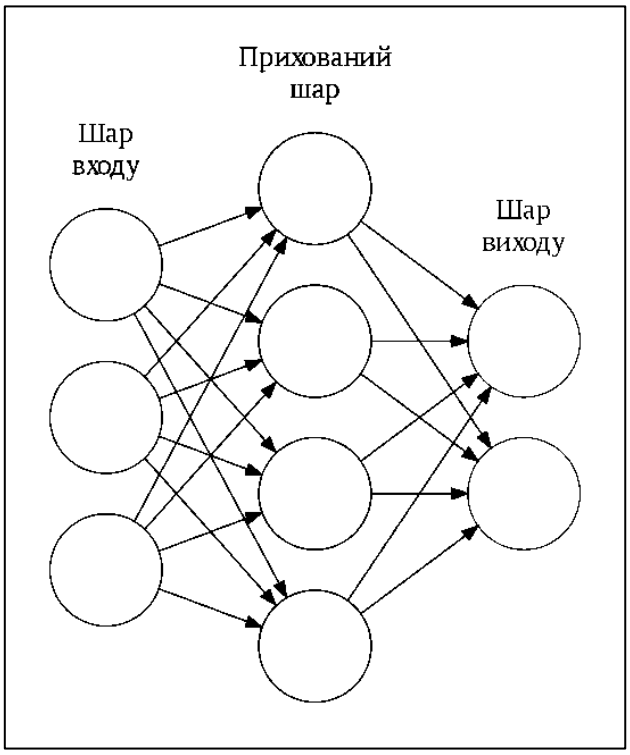

Рисунок 2.3 – Проста структура нейронної мережі

В цій мережі можно виділити окремі «шари»:

- Шар входу. Це масив нейронів якій приймає інформацію у її «сирому» вигляді. Тобто в такому, в якому вона поступає із зовнішнього світу. Його можно порівняти з сітківкою ока.
- Прихований шар. Це шар який знаходиться між шаром входу і шаром виходу. За рахунок цього массиву нейронів і відбувається основний процес обробки інформації.
- Шар виходу. Це шар, за допомогою якого система реагує на інформацію з навколишнього світу. В живому світі це може бути нейрони рухової системи.

Також прихованих шарів могу бути набагато більше, якщо точніше, то їх кількість не обмежена, як показано на рисунку нижче. (Рисунок 2.4)

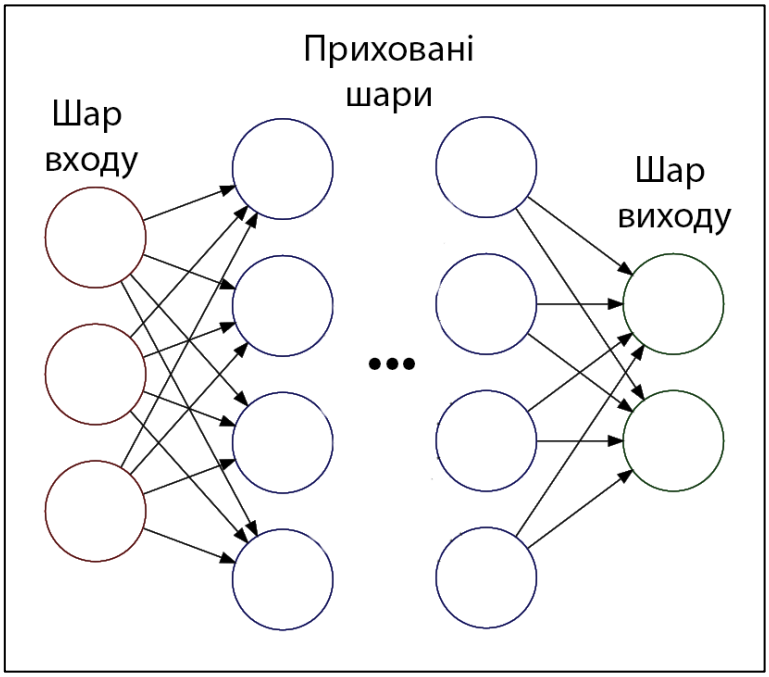

Рисунок 2.4 – Багатошарова «глибока» нейрона мережа

В такому випадку нейрону мережу називають «глибокою» (DNN – deep neural network). И для роботи з нею використовують принципи глубокого навчання (Deep Learning). В залежності від структури прихованих шарів, типу їх зв’язків, кількості нейронів в шарі та інших прикмет нейронні мережі розділяють на багато різних типів, кожний з яких добре підходить під окрему задачу. Різні архітектури нейронних мереж будут розглянуті далі.

В мережі, нервовий імпульс подорожує від одного нейрону до інших. Кожен нейрон з’єднаний з тисячами інших нейронів, тому він одночасно аккумулює та обробляє тисячі імпульсів.

Нейрон має деяке значення сили імпульсу, яке називається «порогом активації». Попадаючи до нейрону, імпульс повинен мати силу більшу або рівну порогу щоб нейрон активувався та передав новий імпульс далі по мережі. Оцінка сили імпульсу та перехід у стан «активний»-«неактивний» здійснюється за допомогою функції активації.

2.1.2 **Функція активації**

Як розглядалось вище, нейрон має багато вхідних сигналів, які називають «синапси» та один вихідний сигнал який передає аксон. Як показано на рисунку вище (Рисунок 2.2) вхідні сигнали можна позначити як x_1, x_2…x_n. Тоді сукупність цих вхідних сигналів можна позначити як X. Кожен синапс (тобто вхідний сигнал) характерезується силою зв’язку, або його вагою. Для кожного

синапсу його вага позначена як w_i. Для того, щоб забеспечити нелінійність виходу з нейрону використовується функція активації.

Функція активації це функція, яка приймає вектор як параметр, а її результатом є скалярна величина. Залежно від виду функції, ця величина може бути цілим або дійсним числом. Існує багато разних функцій активації, кожна з яких використовується в залежноості від задачі та архітектурних особливостей мережі. Далі розглянемо список найвідоміших. Деякі з них є застарілими та використовуються як приклад.

2.1.2.1 Функція Гевісайда

Функція Гевісайда - це розривна функція дійсної змінної значення якої рівне 0 для від'ємних значень аргумента і рівне 1 для додатніх значень аргумента. В більшості випадків значення функції в точці нуль H(0) не є важливим. Функція названа на честь англійського математика Олівера Гевісайда і широко використовується в теорії керування і обробці сигналів. В теорії ймовірності функція Гевісайда з 'H(0)=1 є функцією розподілу випадкової змінної, що майже напевно рівна нулю. [33]

Ця функція використовується в персептроні – найпростішій нейроній мережі. Вона використовується лише як приклад під час навчання курсу нейронних мереж. Графік фунції можна побачити на малюнку нижче. (Рисунок 2.5)

Як видно з графіку, сигнал на виході нейрону буде дорівнювати 1 якщо сума вхідних сигналів є позитивним числом і 0 якщо сума – негативне число.

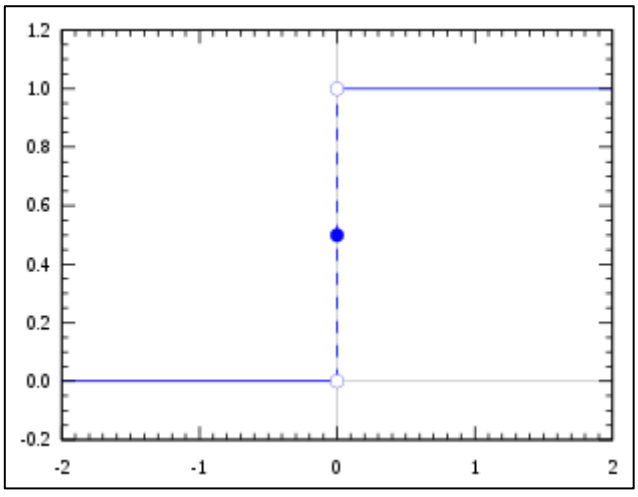

Рисунок 2.5 – Графік функції Гевісайда

2.1.2.2 Лінійна функція

Лінійна функція – це функція при використанні якої, вихідний сигнал пов’язаний лінійно зі сумою сигналів на вході. Сигнал помножується на коєфіціент функції та сумується з вільним членом рівняння. Графіком лінійної функції є пряма нахилина на коєфіціент функції та зміщена на вільний член.

В сучасних рішеннях лінійна функція майже не використовується.

Графік функції можно побачити на зображені нижче. (Рисунок 2.6)

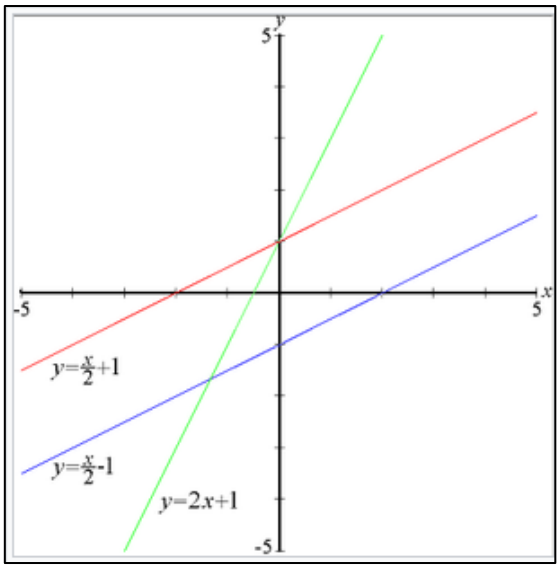

Рисунок 2.6 – Графіки лінійних функцій

2.1.2.3 Сигмоїдальна функція (логістична функція)

Сигмоїда — це неперервно диференційована монотонна нелінійна S-подібна функція, яка часто застосовується для «згладжування» значень деякої величини. [34] Тобто функція посилює слабкі сигнали та запобігає посиленню сильних.

Ця функція часто використовується в нейроних мережах. Вона надає їй деяку нелінійність при цьому не сильно змінюючи результат роботи. Вона набула такої популярності через те, що її похідну дуже легко виразити через саму функцію, що істотно скорочує обчислювальну складність методу зворотного поширення помилки, та робить його придатним на практиці.

Графік сигмоїди можна побачити на рисунку нижче. (Рисунок 2.7)

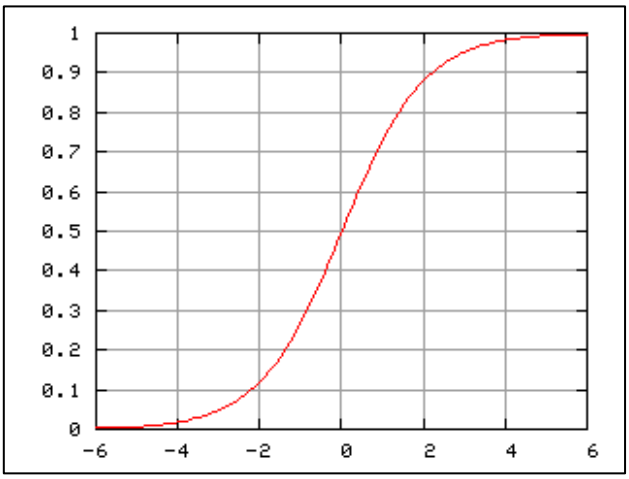

Рисунок 2.7 – Сигмоїда

2.1.2.4 Гіперболічні функції

До цього класу функцій відносяться: гіперболічний синус, гіперболічний косинус, гіперболічний тангенс, гіперболічний котангенс, гіперболічні секанс і косеканс.

Найчастіше в нейроніх мережах використовується гіперболічний тангенс, через його схожіть з сигмоїдною функцією. Різниця в тому, що гіперболічний тангенс лежить в межах значень -1:1 що спрощую процедуру навчання мережі.

Графіки функцій можна побачити на ррисунку нижче. (Рисунок 2.8)

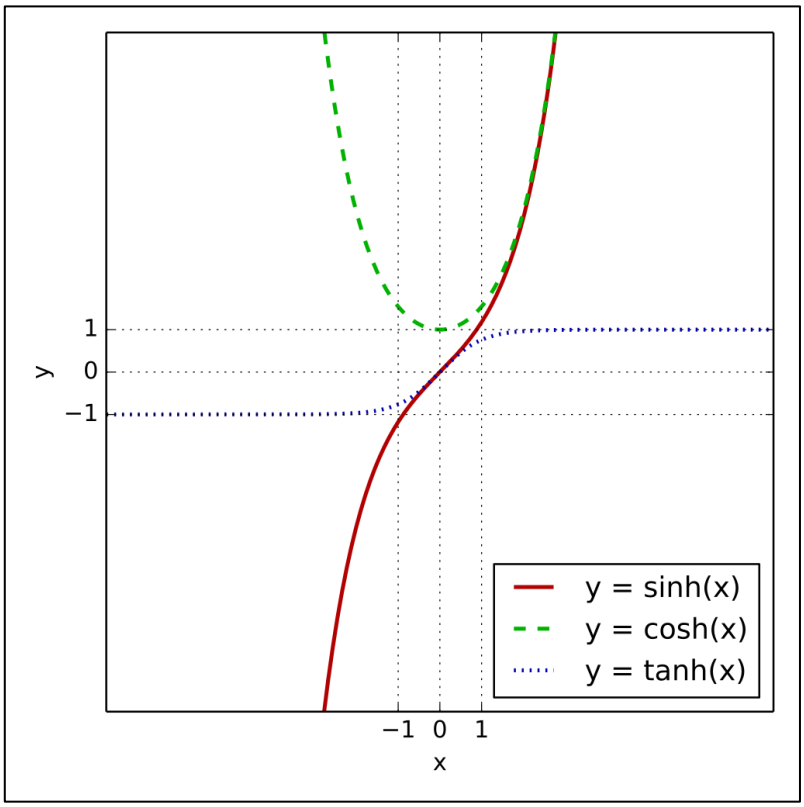

Рисунок 2.8 – Гіперболічні функції

2.1.2.5 ReLU (Reclified Linear Unit)

ReLU - або випрямляч (англ. rectifier) у контексті штучних нейронних мереж є передавальною функцією. [35]

Вона визначена наступною формулою:

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

де x – вхідне значення нейрона.

Ця функція була представлена у 2000-у році, а в 2011 році було показано як з її допомогою забезпечити краще навчання глибинних нейронних мереж. Вона показує себе краще ніж логістична функція та гіперболічний тангенс. Ще існує гладке наближення цієї функції яке називається softplus-функція.

Графіки обох функцій можна побачити на зображенні нижче. (Рисунок 2.9)

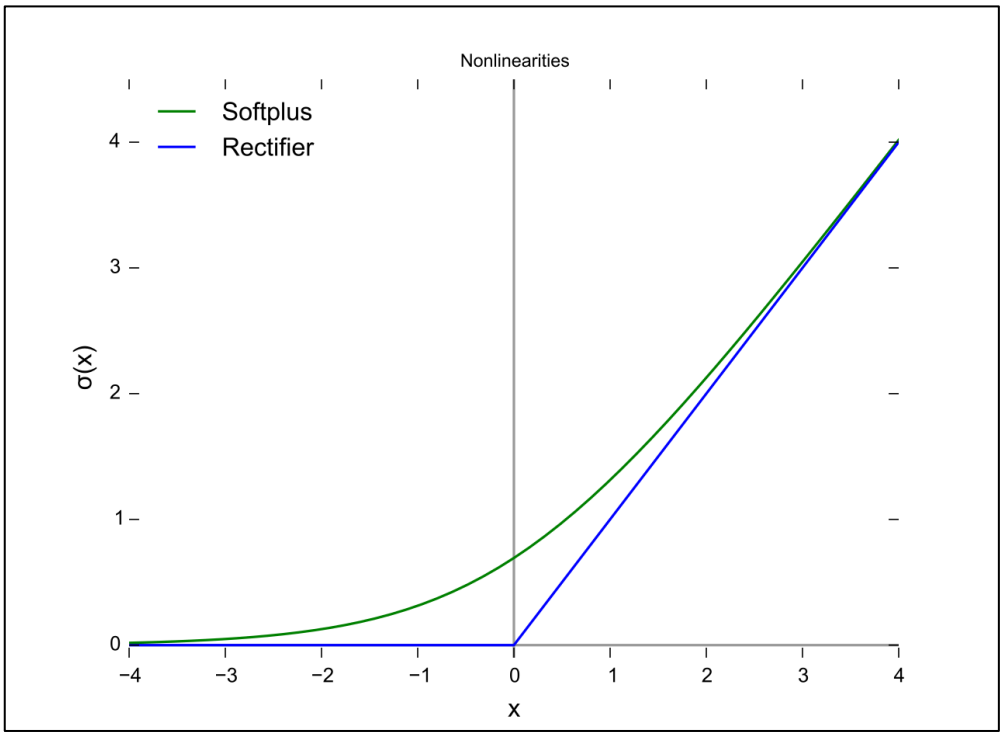

Рисунок 2.9 – Графіки ReLU та softplus функцій

Через те, що похідна данної функції дорівнює 0 або 1, її використання приводить до зменшення ваг, ща позитивно позначається на обчислювальній здатності мережі. Також ця функція запобігає розростанню і загасанню градієнтів.

На сьогоднішній день існує велика кількість різних модифиікацій цієї функції, які вирішують різні проблеми надійності функціі при проходженні вкликих графієнтів через нейрон.

2.1.3 **Архітектури нейронних мереж**

Більшість архітектур нейронніх мереж можна поділити на два класси:

- Нейромережі прямого розповсюдження (FNN, Feedforward Neural Network). Це тип нейромереж у яких розповсюдження сигналу відбувається прямолінійно, від вхідних нейронів, через приховані шари нейронів і виходять у шар вихідних нейронів. Частіше такі мережі використолвуються для задач кластеризації та апроксимації;
- Рекурентні нейромережі (RNN, Recurent Neural Network). це клас штучних нейронних мереж, у якому з'єднання між вузлами утворюють граф орієнтований у часі. Це створює внутрішній стан мережі, що дозволяє їй проявляти динамічну поведінку в часі. На відміну від нейронних мереж прямого поширення, РНМ можуть використовувати свою внутрішню пам'ять для обробки довільних послідовностей входів.

Це робить їх застосовними до таких задач, як розпізнавання несегментованого неперервного рукописного тексту та розпізнавання мовлення; [36]

Давайте розглянемо приклади архітектур нейроних мереж обох типів.

2.1.3.1 Перцептрон Розенблатта

Перцептрон – це нейромережа прямого розповсюдження, яка складається з трьох типів шарів нейронів:

- Рецептори. Шар рецепторів це масив нейронів, які можуть перебувати у стані «збудження» або стані «спокою». Лише у стані збудження вони передають сигнал далі, до асоціативних нейронів. У реальному світі вони відповідають, наприклад, світлочутливим клітинам сітківки ока;
- Асоціативні нейрони. Це шар нейронів які приймають сигнал від набору (асоціації) рецепторів. Ці нейрони також можуть перебувати в одному з двух станів. Як тільки сума сигналів з рецепторів перевищує певнгу величину, нейрон активізується та передає сигнал до суматора. Кожний сигнал помножується на певний коєфіціент який називають «вагою» зв’язку;
- Реагуючі нейрони. Це нейрони, які отримують сигнал від асоциативних нейронів, накладають на їх сумму певну функцію та віддають результат;

Елементарний персептрон складається з одного шару кожного виду нейтронів.

Можливості цього типу нейромереж в основному полягають у здатності до класифікації та апроксимації.

Основні ж обмеженя полягають у нездатності до узагальнення своїх характеристик на нові стимули та ситуації, а також відсутність можливості аналізувати складні ситуації шляхом розкладення їх на більш прості.

На основі елементарного персептрона було стрворено багато його різновідностей, але основна ідея залишається однаковою: пряме поширення сигналу та корегування характеристик шляхом підбору з’єднань та їх ваг.

Зображення елементарного персептрону можна побачити на рисунку нижче. (Рисунок 2.10)

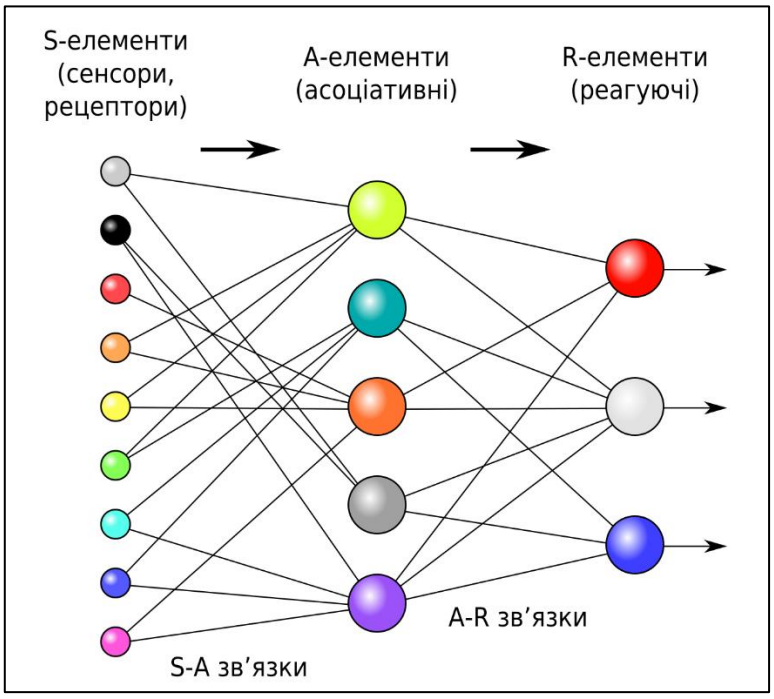

Рисунок 2.10 – Елементарний персептрон

2.1.3.2 Повнорекурентна мережа

Повнорекурентна мережа – це основна архітектура в якій кожен нейрон має оріентоване з’єднання з усіма іншими нейронами. Кожен з вузлів має змінну в часі дійснозначну активацію. Кожне з'єднання має змінювану дійснозначну вагу. Деякі з вузлів називаються входовими вузлами, деякі — виходовими, а решта — прихованими вузлами.

Простими словами: вихідний сигнал нейрона є вхідним сигналом для іншого нейрона який згаходиться в шарі з меншим індексом, тобто знаходиться у мережі раніше. За допомогою цього підходу ми маємо можливість зберігання тимчасової інформації в мережі. Саме через це рекурентні нейронні мережі підходять для задач обробки послідовностей.

На основі цієї архітектури побудовано багато інших архітектур. Основна сфера їх використання – випадки коли важлива послідовність елементів:

- аналіз тексту;
- автоматичний переклад;
- автоматичне розпізнавання мови з аудіо;
- прогнозування наступного кадру відео на основі попередніх;
- розпізнавання емоцій на основі відеоряду;
- прогнозування наступного пікселю зображення на основі оточующих;
- генерація опису зображень;

Нижче можно побачити зображення повнорекурентної мережі та її окремого слою. (Рисунок 2.11-Рисунок 2.12)

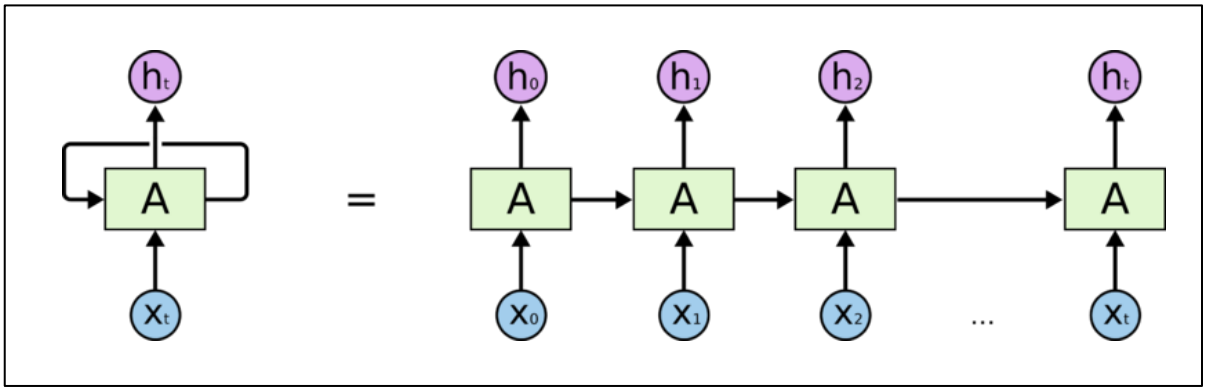

Рисунок 2.11 – Рекурентна нейронна мережа

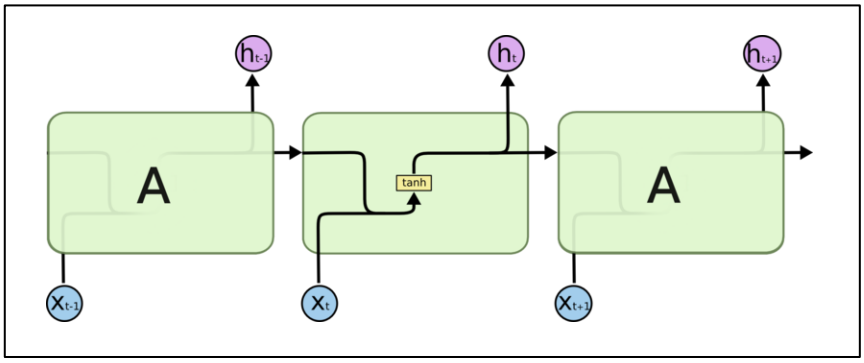

Рисунок 2.12 – Шар рекурентної нейронної мережі

На рисунках x – вектор сигналів вхідного слою, A – нейрон прихованого шару, h – вектор сигналів вихідного шару.

Особливості

2.1.4 **Навчання нейронних мереж**

Навчання нейроних мереж – це процесс підбору архітектури та ваг зв’язків між нейронами для ефективного виконання поставлених перед мережею задач. Існують багато способів навчання, які ефективні у різних випадках.

2.2 **Аналіз зображень за допомогою нейронних мереж**

Розглянемо елесментарний персептрон, як модель розпізнавання образів та характеристик. Зображення персептрону для аналізу зображення показано на рисунку нижче. (Рисунок 2.13)

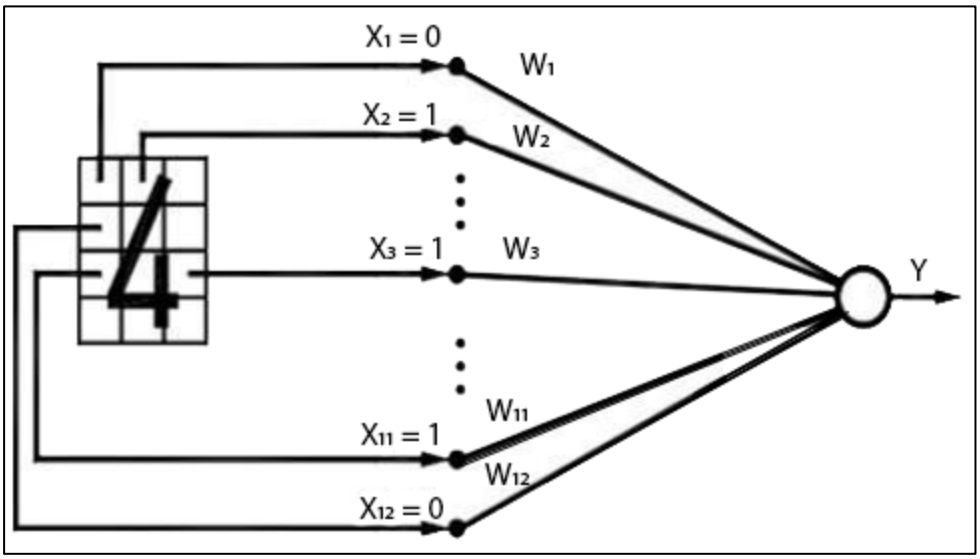

Рисунок 2.13 – Персептрон для аналізу зображення

2.2.1 **Згорткова нейронна мережа**

Згорткові нейронні мережі мають ефективну архітектуру для розпізнавання образів на зображеннях. Особливість цієї архітектури в тому, що в мережі чередуються згортові шари та шари агрегування.

Зображеня згортокової нейроної мережі можно побачити на рисунку нижче. (Рисунок 2.14)

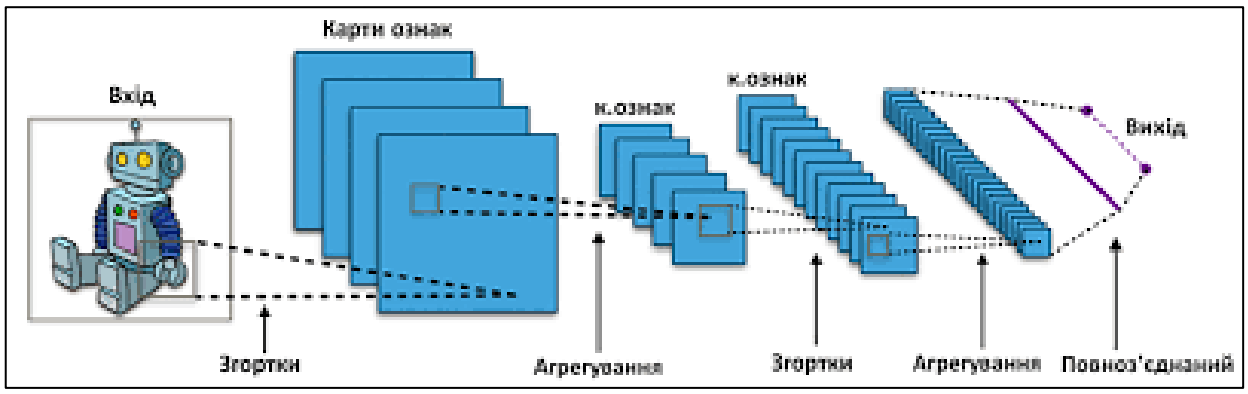

Рисунок 2.14 – Згорткова нейрона мережа

2.3 **Аналіз відеоряду за допомогою згорткової нейронної мережі**

Згорткові нейронні мережі добре підходять для аналізу зображень. Зображення – це двухвимірний масив данних. Відео – це посуті набір зображень які розташовані послідовно і змінюються з часом. Тому для аналізу

відео двовимірні згорткові нейроні мережі не підходять, тому потрібно використовувати тривимірні.

2.3.1 **Тривимірна згорткова мережа**

Принцип роботи трьохвимірної згорткової нейронної мережі можна побачити на рисунку нижче. (Рисунок 2.15)

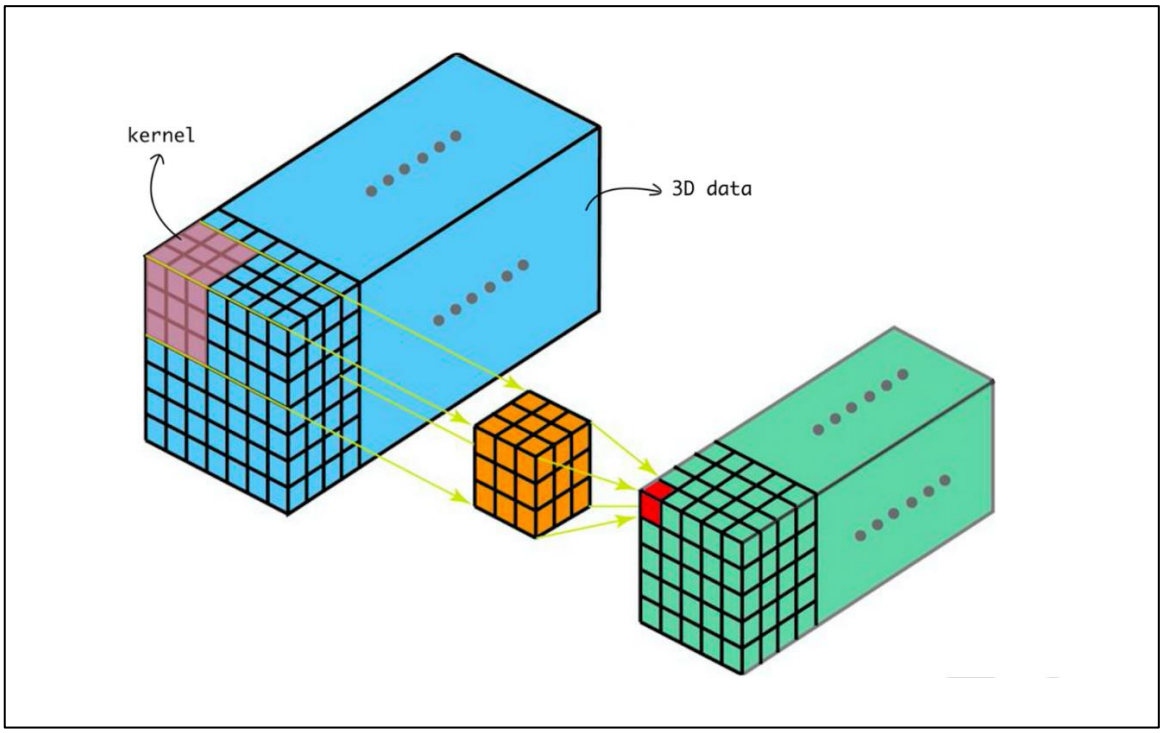

Рисунок 2.15 – Робота трьохвимірної згорткової нейронної мережі

2.4 **Інстременти та данні для навчання нейронної мережі**

Для створення та навчання моделі машинного навчання будемо використовувати TensorFlow.

Для навчання нейронної мережі використовуються так звані датасети, які містять в собі велику кількість данних.

3 РОЗРОБЛЕННЯ ПРОТОТИПУ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ НАВЧАННЯ ДАКТИЛОЛОГІЇ

3.1 Ціль та задачі розроблення прототипу програмного забезпечення

Першим кроком в проектуванні програмного забезпечення є визначення мети та цілей які він повинен досягти. Прототип програмного забезпечення, створенного в рамках данного проекту повинен надати можливість для користувача навчитися використовувати засоби дактилогії для комунікації з іншими людьми. Також засіб повинен надати можливість покращити навички дактилогії, швидкість та точність спілкування. Процесс використання програмного забеспечення повинен бути інтерактивним, автономним та надавати систему «рівнів» та досягнень.

Для того, щоб досягти поставленних цілей потрібно реалізувати компьютерний додаток, який буде підтримувати розпізнавання жестів користувача з відеопотоку (вебкамери), надавати систему рівнів та досягнень. Також додаток повинен надавати можливість користувачу створити аккаунт, щоб зробити процесс навчання більш персоналізованим та надати можливість зберігання прогрессу навчання.

Далі потрібно розбити процесс розробки системи на декілька компонентів та декілька етапів.

Серед компонентів можно виділити:

– Сервер розпізнання. Це компонент, який буде відповідати за розпізнання жестів на зображеннях.
– Клієнт. Десктопний додаток, який буде надавати кінцевому користувачу доступ до системи. Компонент який матиме інтерфейс користувача.
– База данних. Компонент, який відповідає за збереження та взаємодію з данними потрібними для додатку.

Розробку цих компонентів можно розділити на декілька етапів:

– створення методів використання, робота над сценаріями взаємодії з користувачем.
– проектування і розробка дизайну користувальницького інтерфейсу;
– розробка серверу розпізнавання.
– розробка клієнтської частини додатку;
– проектування структури бази даних.
– налаштування взаємодії між створенними компонентами.

Між усіма компонентами потрібно налаштувати взаємодію, яка дозволить без затримок отримувати потрібні данні та робити потрібні розрахунки. Усі взаємодії можно розділити на три основні:

– клієнт-користувач

– клієнт-сервер розпізнання
– клієнт-база данних

Для кожної взаємодії важливо створити прозорий інтерфейс з обох боків взаємодії.

3.2 **Аналіз вимог до прототипу ПЗ**

3.2.1 **Початкові дані**

Кожен із виділених компонентів має свій набір відповідальностей, не знає нічого про реалізацію інших компонентів та предоставляє прозорий інтерфейс для взаємодії.

Почнемо з компоненту клієнту. Клієнт буде реалізований на базі технології Windows Forms. За допомогою існуючих компонентів технології та її гнучкості буде дуже легко керувати вхідними данними користувача – зображення з веб-камери, текстовий ввід та інші. Також розробка пришвидшується наявністю великої кількості бібліотек та фреймворків, які спрощують роботу з низкорівневими сутностями, такими як сокети.

Процесс роботи клієнту можно розбити на наступні елементи:

– робота з базою данних. Для роботи з сховищем данних буде використовуватись засоби Entety Framework. Це потужний інструмент для роботи з сутностями з БД, їх контролю та маніпуляціями з данними.
– отримання данних з веб-камери користувача. Для цього буде використовуватися бібліотека AForge, яка дозволяє працювати на високому рівні з низькорівневими модулями, такими як веб-камера. Бібліотека була створенна для аналізу та обробки зображень, тому вона проваджує достатній рівень швидкості та зручності взаємодії.
– обробка зображення та розпізнання жестів. Для розпізнання жестів буде реалізовано окремий компонент. Для взаємодії з компонентом на боці клієнту буде реалізовано сервіси для роботи з сокетами. Уся взаємодія клієнт-сервер розпізнання буде проводитися через мережеві сокети, щоб зменшити відставання між отриманним зображенням та його опрацьованням.

Наступним буде компонент серверу розпізнавання зображень. Він побудований на основі фреймворку для роботи з машинним навчанням Tensorflow.

Роботу серверу розпізнавання зображень можно розділити на декілька частин:

– отримання зображення. Для отримання зображення ззовні сервер має відкритий сокет. Сокет отримує інформацію та декодує її. Для

корретної обробки вхідної інформації, вона повинна відповідати вхідному формату, а саме – знаходитись у правильному JSON форматі.

– класифікація зображення. Класифікація зображення відбувається за допомогою штучної нейронної мережі побудованої, натренованої та збереженої за допомогою фреймворку Tensorflow

Останнім компонентом є база данних. В додатку буде використовуватись SQL Server як СУБД та Entity Framework як фреймворк для роботи з сущностями БД.

3.2.2 **Побудова діаграм варіантів використання**

Діаграма використання

Діаграма варіантів використання є вихідним концептуальним поданням або концептуальною моделю системи в процесі її проектування і розробки.

Розробка діаграми варіантів використання переслідує наступні мети:

- визначити спільні кордони і контекст модельованої предметної області на початкових етапах проектування системи;

- сформулювати загальні вимоги до функціональної поведінки проектованої системи;

- розробити вихідну концептуальну модель системи для її подальшої деталізації у формі логічних і фізичних моделей;

- підготувати вихідну документацію для взаємодії розробників системи.

Суть даної діаграми полягає в наступному: проектована система представляється у вигляді множини сутностей або акторів, що взаємодіють з системою за допомогою так званих варіантів використання. При цьому актором (actor) або дійовою особою називається будь-яка сутність, що взаємодіє з системою ззовні. Це може бути людина, технічний пристрій, програма або будь-яка інша система, яка може служити джерелом впливу на систему так, як визначить сам розробник. У свою чергу, варіант використання (use case) служить для опису сервісів, які система надає актору. Іншими словами, кожен варіант використання визначає деякий набір дій, які впливають на систему при взаємодії з актором. При цьому нічого не говориться про те, яким чином буде реалізовано взаємодію акторів з системою.

Діаграма варіантів використання для системи містить декілька акторів, серед яких: користувач, клієнт, сервер розпізнавання, база даних. Кожен актор має свої функції.

Користувач додатку має функції (рис. 3.1):

- розпізнавання жестів;
- відстеження прогресу;
- вибір рівня;
- реєстрація;
- отримання винагород;

Клієнт має такі функції (рис. 3.2):
- отримання зображення з веб-камери
- відображення інтерфейсу
- комунікація з базою даних
- комунікація з сервером

Сервер має такі функції (рис 3.3)
- комунікація з клієнтом
- отримання зображень
- тренування моделі НН
- збереження моделі НН
- класифікація жестів

База данних має такі функції (рис. 3.3)
- збереження данних

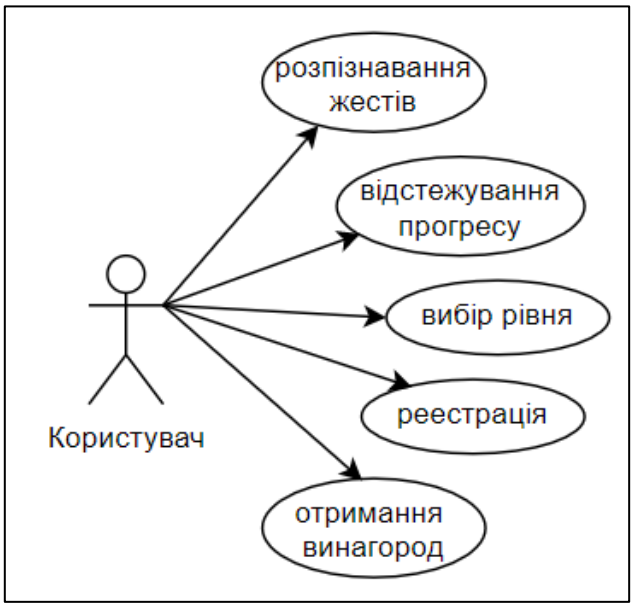

Рисунок 3.1 – Діаграма варіантів використання користувача додатку

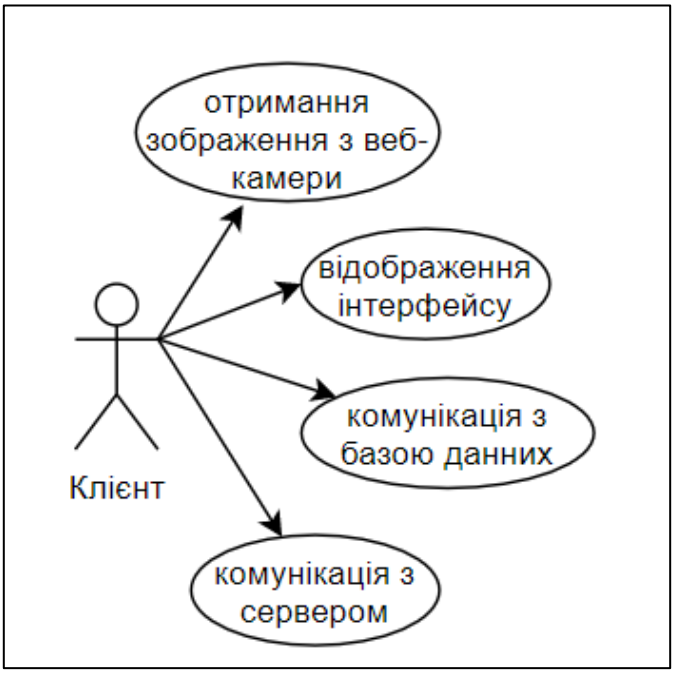

Рисунок 3.2 – Діаграма варіантів використання для клієнту

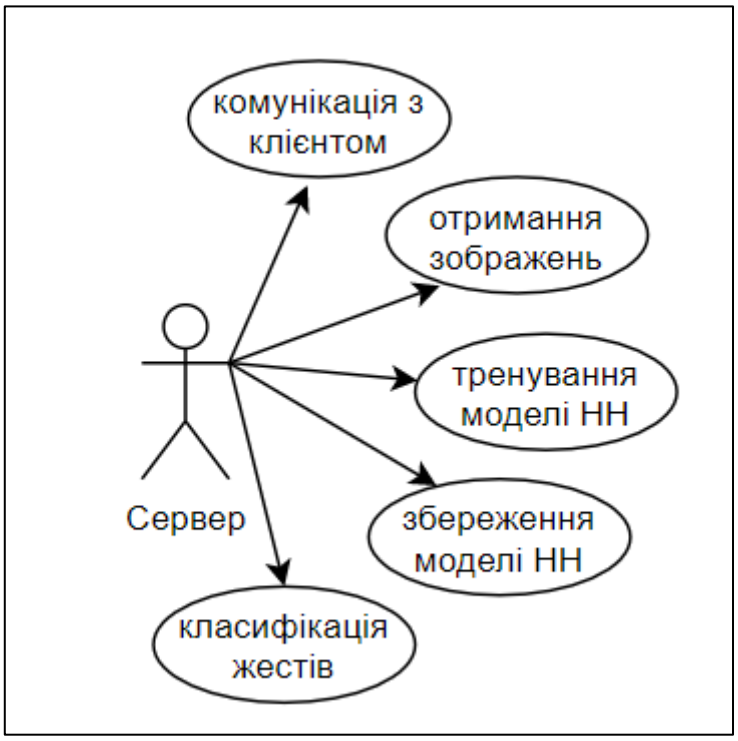

Рисунок 3.3 – Діаграма варіантів використання для серверу

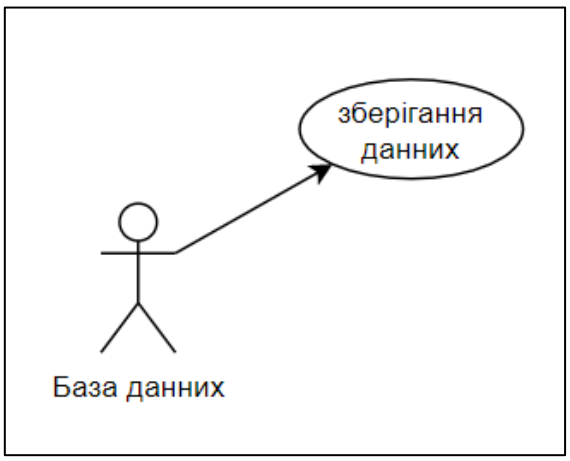

Рисунок 3.4 – Діаграма варіантів використання бази даних

3.2.3 **Вимоги до ПЗ**

3.2.4 **Функціональні вимоги**

Функціональні вимоги до ПЗ наступні:

1 Користувач системи має наступні можливості:

– інформація про жести, які використовує користувач

– отримання справки

– відстеження власного прогресу
– отримання винагород за досягнення
– можливість зареєструватися
– можливість обрати рівень
– можливість повторити пройдені рівні

2 Сервер має наступні можливості
– тренування моделі нейронної мережі
– збереження моделі нейронної мережі
– завантаження моделі нейронної мережі
– використання моделі для отримання передбачень класифікації жестів
– комунікація з клієнтом

3 Клієнт повинент мати наступні можливості
– відображення інтерфейсу користувача
– отримання відео-інформації від веб-камери
– комунікація з базою даних
– комунікація з сервером розпізнання

3.2.5 **Нефункціональні вимоги**

До нефункціональних вимог відносяться такі:

Мінімальні системні вимоги для роботи програми на стороні клієнта наведені в таблиці 3.1;

Таблиця 3.1 – Конфігурація ПК для роботи програми на стороні клієнта

Операційна система	Microsoft Windows 7,10
Процессор	Intel Core I3, AMD64 або краще
Об'єм оперативної пам'яті	1 ГБ
Додакове обладнання	Веб-камера, або інший пристрій захвату зображення

Мінімальні системні вимоги для роботи клієнта наведені в таблиці 3.2.

Таблиця 3.2 – Конфігурація ПК для роботи програми на стороні серверу

Операційна система	Microsoft Windows Server, Linux, Ubuntu
Процессор	Intel Core I3, AMD64 або краще
Об'єм оперативної пам'яті	16 ГБ
Додакове обладнання	Бажана присутність графічного прискорювача GeForce RTX 2060 або краще

Мінімальні системні вимоги для роботи програми та бази даних на стороні сервера наведені в таблиці 3.3;

Таблиця 3.3 – Конфігурація ПК для роботи програми на стороні сервера

Операційна система	Ubuntu 16.04 xenial
Інтернет канал	Не нижче 100 мегабит
Процессор	Intel Xeon E5-2676 v3 або краще
Об'єм оперативної пам'яті	1 ГБ

Необхідно постійне підключення до інтернету;
Форма побудови меню веб додатку на екрані:

- на формі додатку повинно бути вікно з відео отриманного з веб-камери, щоб користувач розумів як програма бачить його жести;
- на формі повинен бути список рівнів з яких можно обрати бажаний.
- повинна бути можливість закрити додаток, згорнути його або зробити на весь екран.

3.2.6 **Специфікації варінтів використання**

Сценарії використання, варіанти використання або прецеденти - специфікація послідовностей дій (варіанти послідовностей і помилкові послідовності) які може здійснювати система, підсистема або клас, взаємодіючи з зовнішніми дійовими особами.

Специфікація варіантів використання виконана у вигляді таблиць з описом прецедентів і сценаріїв і наведена в таблицях 3.4 – 3.11.

Таблиця 3.4 – Сценарій реєстрації

Ім'я	registration.
Назва	Реєстрація.
Опис	Реєстрація користувача у системі.
Передумова	Додаток запущено.
Постумова	Створено запис користувача у базі даних.
Основний потік	1 Користувач відкриває додаток. 2 Користувач вводить потрібні дані. 3 Користувач натискає кнопку «зареєструватися»

Продовження таблиці 3.4

Альтернативний потік	1 Користувач відкриває додаток. 2 Користувач вводить потрібні дані. 3 Користувач отримує повідомлення про те, що введені некоректні дані.
Альтернативний потік	1 Користувач відкриває додаток. 2 Користувач вводить потрібні дані. 3 Користувач отримує повідомлення про те, що користувач з такими данними вже існує.

Таблиця 3.5 – Сценарій вибору рівня

Ім'я	Select_level.
Назва	Вибір рівня.
Опис	Вибір рівня для проходження.
Передумова	Додаток відкрито, здійснено вхід в систему, рівень розблоковано.
Постумова	1 Потрібний рівень завантажено. 2 Рівень відображено на екрані
Основний потік	1 Користувач натискає на назву рівня серед списку рівней.

Таблиця 3.6 – Сценарій розпізнавання жестів

Ім'я	Gesture_recognition
Назва	Розпізнаваня жестів.
Опис	Користувач показує жести до веб камери.
Передумова	1 Відкритий потрібний рівень. 2 Веб-камера ввімкнена.
Постумова	Клієнт відображує передбачення щодо відображенних жестів.
Основний потік	1 Користувач вибирає рівень. 2 Користувач вмикає камеру. 3 Користувач показує жести до веб-камери.
Альтернативний потік	1 Користувач вибирає рівень. 2 Користувач отримує повідомлення що рівень не розблоковано.
Альтернативний потік	1 Користувач вибирає рівень. 2 Користувач вмикає веб-камеру. 3 Користувач отримує повідомлення, ща веб-камера не доступна

Таблиця 3.7 – Сценарій розблокування рівня

Ім'я	Level_unblocking
Назва	Розблокування рівня.
Опис	Користувач розблоковує рівень.
Передумова	Користувач розблокував попередні рівні.
Постумова	Рівень розблоковано.
Основний потік	1 Користувач проходить попередній рівень. 2 Користувач переходить на наступний рівень.

Таблиця 3.8 – Сценарій тренування моделі

Ім'я	Model_training
Назва	Тренування моделі
Опис	Користувач тренує нову модель на основі тренувального датасету
Передумова	1 Не існує інших моделей у каталозі моделей з ім’ям поточної моделі. 2 Користувач запустив сервер.
Потсумова	1 Модель натренована та готова до використання. 2 Модель збережена у каталог моделей з ім’ям поточної моделі.
Основний потік	1 Користувач переконується, що поточної моделі не існує та видаляє її якщо вона є. 2 Користувач запускає сервер.

Таблиця 3.9 – Сценарій завантаження моделі

Ім'я	Load_model
Назва	Завантаження моделі для використаня сервером.
Опис	Користувач запускає сервер і сервер завантажує створену модель із каталога моделей.
Передумова	1 Користувач має збережену модель в каталозі моделей з ім’ям поточної моделі. 2 Користувач запустив сервер.
Потсумова	Модель завантажена на сервер та готова до використання.
Основний потік	1 Користувач поміщає модель до каталогу моделей. 2 Користувач запускає сервер.

3.2.7 **Розробка макетів екранних форм**

Базуючись на вимогах до ПЗ, створемо діаграму переходів між сторінками інтерфейсу клієнту. Діаграма зображена на рисунку 3.5

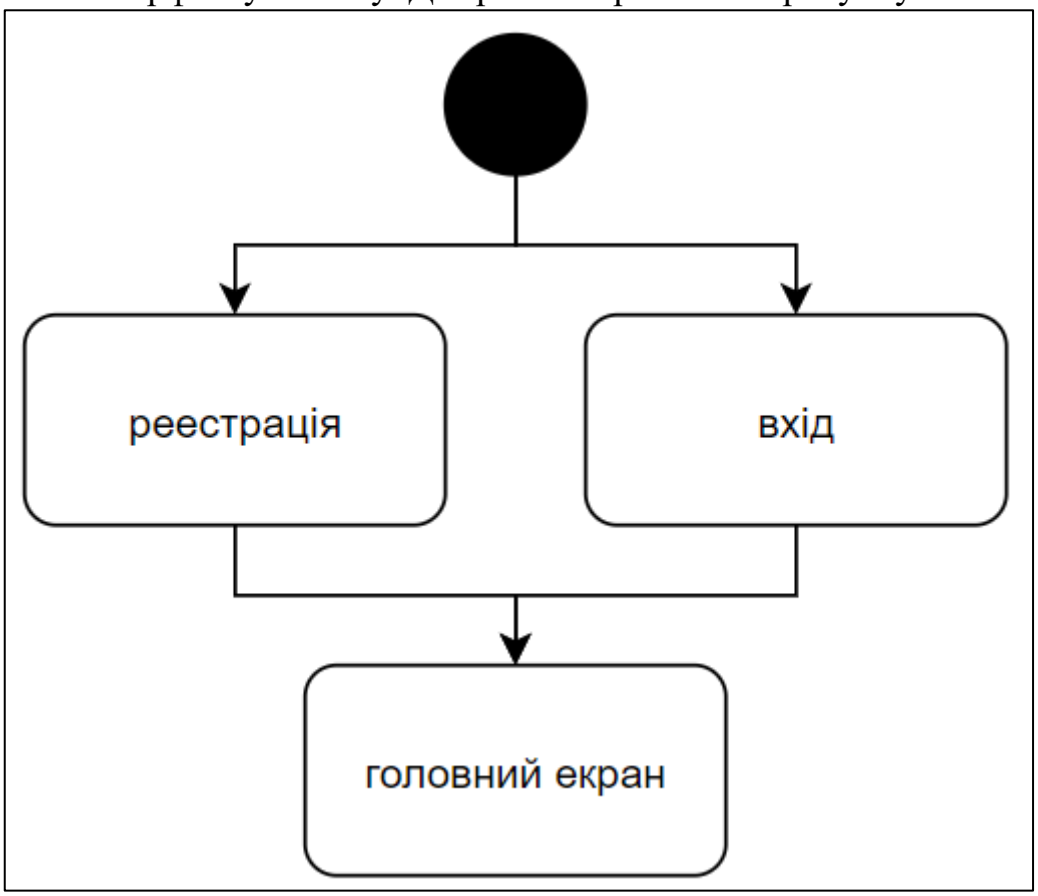

Рисунок 3.5 – Діаграма переходів між сторінками

Макет сторінки логіну зображений на рисунку 3.6

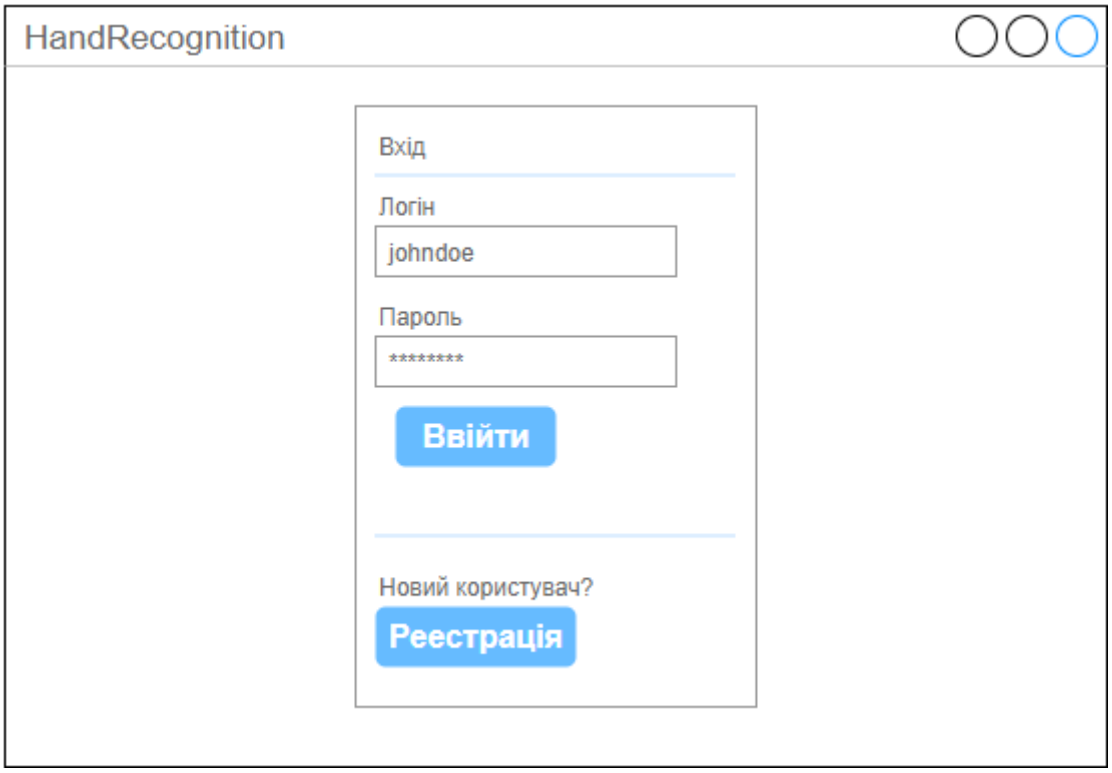

Рисунок 3.6 – Макет сторінки логіну

Макет сторінки реєстрації зображено на рисунку 3.7

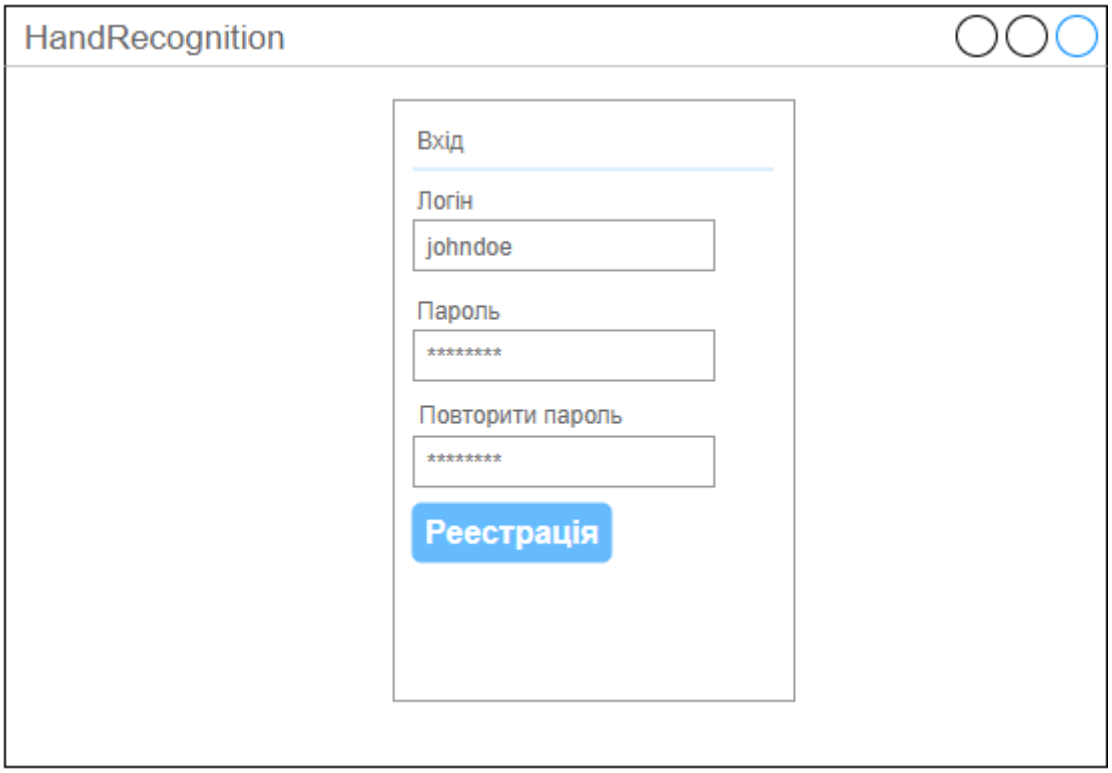

Рисунок 3.7 – Макет сторінки реєстрації

Макет головної сторінки зображено на рисунку 3.8

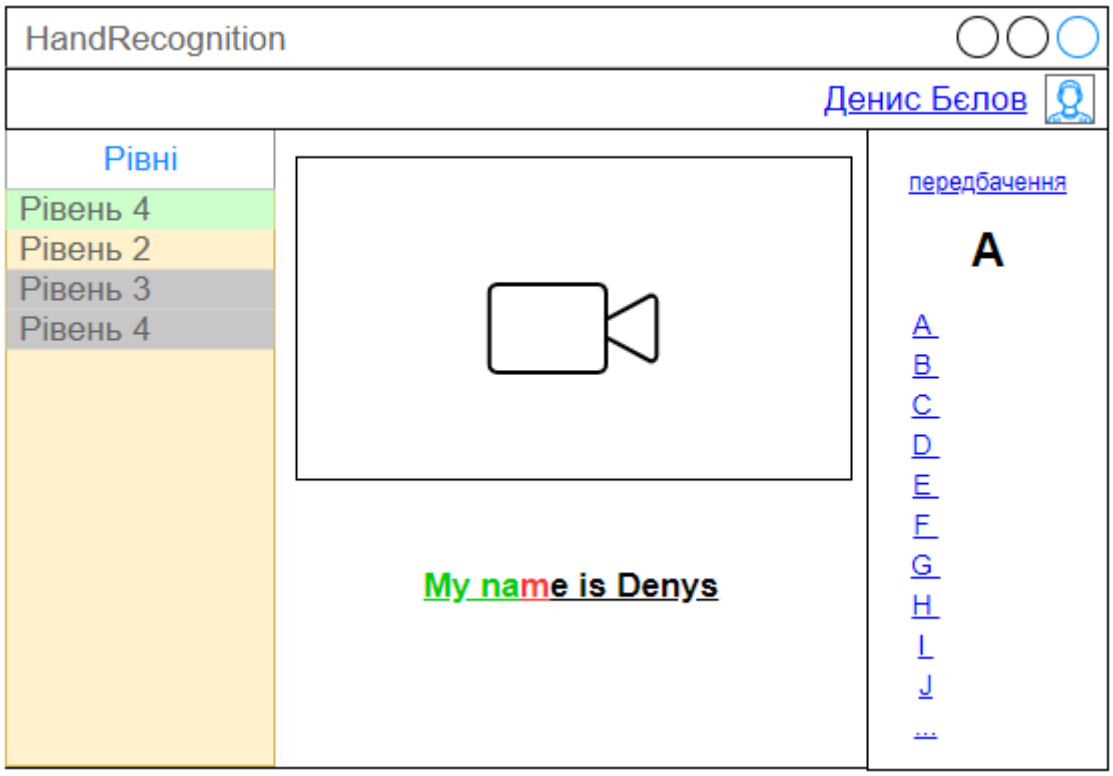

Рисунок 3.8 – Макет головної сторінки

3.3 **Архітектурне проектування додатку**

Додаток складається з двух основних частин:

– Клієнт. Додаток який відповідає за комунікацію з користувачем. Він показує інтерфейс користувачу, отримує від нього вхідні дані та комуніцює з сервером розпізнання та базою даних
– Сервер розпізнання. Розпізнавання образів на зображеннях потребує доволі високих вичислювальних потужностей. Також стек технологій який використовується для класифікації зображень не підходить для створення додатків з інтерфейсом користувача, тому було вирішено розділити функціональність на дві частини.

При такому розділенні можливе створення додаткових клієнтів, наприклад, на мобільних платформах, які будуть використовувати один сервер для розпізнання. А також можливе використання іншого стеку технологій розпізнання. Таке співвідношення клієнт-сервер допомагає зробити систему більш гнучкою, адже усе що знають компоненти один про одного – це інтерфейс, через який вони спілкуються.

Так як, розпізнання потрібно проводити зі швидкістю близькою до реального часу, потрібно обрати шлях комунікації між сервером та клієнтом який допоможе зберегти максимальну швидкість комунікації. Ідеальним варіантом є розробка власного протоколу комунікаціїї, та це сильно збільшить складність розробки. Для прототипу додатку було вирішено обрати технологію веб-сокетів та протокол TCP/IP.

Архітектура програмної системи зображена на рисунку

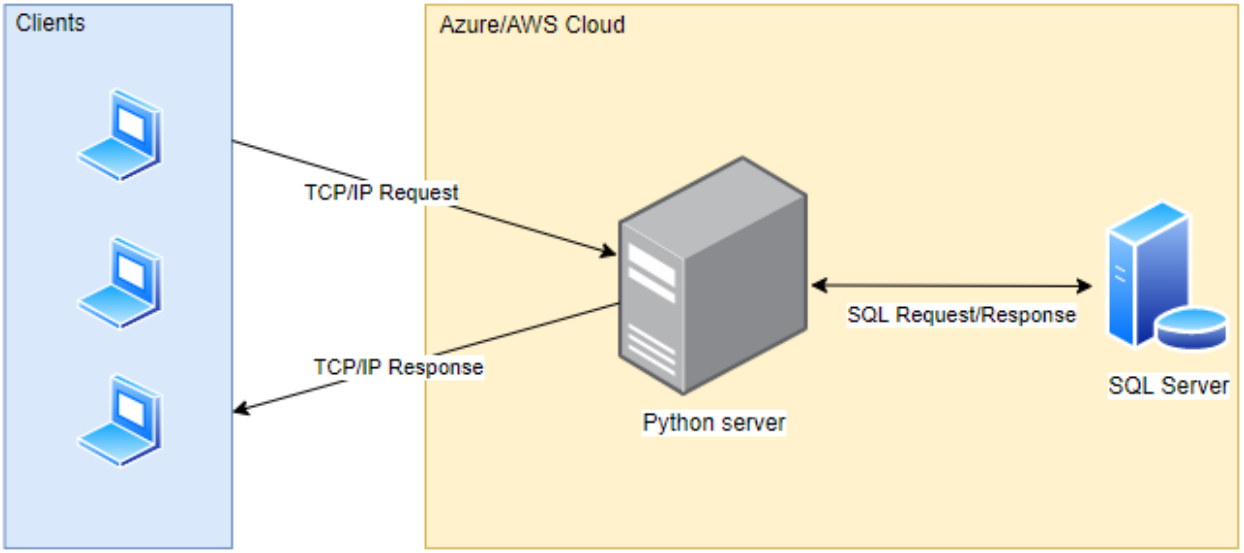

Рисунок 3.9 – Архітектура системи

3.3.1 **Вибір і обґрунтування вибору стеку технологій**

Вибір стеку технології це важливий етап розробки проекту адже від цього буде залежати майбутня вартість підтримки та розширення системи.

Також стек технологій впливає на масштабованість, функціональність та можливість підтримки іншими розробниками.

Обираючи стек технологій також слід звернути увагу на доступні можливості хмарного розміщення розроблених компонентів, на вартість та на наявність різних варіантів хмарного розміщення.

Також при виборі інструментів, бібліотек та фреймворків потрібно звернути увагу на якість їх підтримки та наявності документації. Також важливою ознакою є наявність попиту серед розробників на ринку, що зумовлює наявність співтовариства, яке також може допомогти при виникненні питань щодо технології.

Беручі до уваги вище перераховані факти, а також опираючись на власний досвід було вирішено обрати наступний стек технологій:

– TensorFlow, як інструмент для роботи зі штучними нейронними мережами.

- Python як мову програмування зі зв’язкою з TensorFlow.
- Pip як пакетний менеджер для Python
- Windows Forms як фреймворк для створення додатку клієнта.
- Nuget як пакетний менеджер для Windows Forms
- База даних – SQL Server

3.3.1.1 Засоби розробки

Для кожного з розроблюємих компонентів існує набір інструментів, які значно полегчують процесс розробки. Далі будуть перераховані обрані інструменти:

- Visual Studio 2019. Ця IDE була обрана для створення додатку клієнта. Вона підтримується компанією Microsoft та має набір вбудуваних інструментів для компіляції, збірки та компоновки додатків. Вона має технологію інтелектуальної підсвітки коду Intellisens яка підсвічує помилки у коді та можливі покращення. Також вона допомагає закінчити фразу базуючусь на вже написаній частині коду. Це все дуже присшвидшує процесс розробки додатку. Також середа розробки підтримує режим Debug який дозволяє втручатися в процесс виконання додатку та знаходити можливі помилки, які важко виявити під час статичного аналізу коду. Підтримка інтерактивної компоновкки форм інтерфейсу користувача допомагає швидше та наглядніше створювати інтерфейс користувача.
- Visual Stusio Code. Ця IDE є текстовим редактором з підтримкою багатьох інструментів для полегчення розробки. Вона була обрана для написання серверу розпізнання. Важливою частиною VS Code є наявніть каталогу плагінів, які дозволяють розширити її функціональніть. Для більш зручної розробки було встановлено плагін Python IDE який додає можливість інтерактивної підсвітки коду написаного на мові Python.
- Microsoft SQL Server Management Studio. Ця IDE найкраще підходить для розробки та підтримки баз даних SQL Server. Вона має набір інструментів для створення, налаштування та підтримки серверів баз даних. Також вона допомогає скомпонувати структуру бази данних.

3.3.1.2 TensorFlow

Цей інструмент створено і підтримується Google, що зумовлює високу якість підтримки та популярність серед розробників. TensorFlow містить у собі велику кількість інструментів для пре- та пост-обробки даних, їх візуалізації, завантаження та збереження. Також він підтримує роботу з безліччю

архітектур штучних нейронних мереж і навіть має власну бібліотеку моделей натренованих на великій кількості вхідних даних та готових до використання.

TensorFlow поставляється в двух варіаціях – варіація для роботи на процессорі (CPU) та варіація для роботи на графічному прискорювачі (GPU). Варіація з графічним прискорювачем значно випереджає варіацію з процессором так як використовує архітектуру CUDA (коли це можливо). Також графічний прискорювач часто набагато швидше процессора в задачах машиного навчання через свою архітектуру. На рисунку 3.10 нижче можно побачити порівняння швидкості тренування моделі штучної нейронної мережі при використанні процессора на графічного прискорювача.

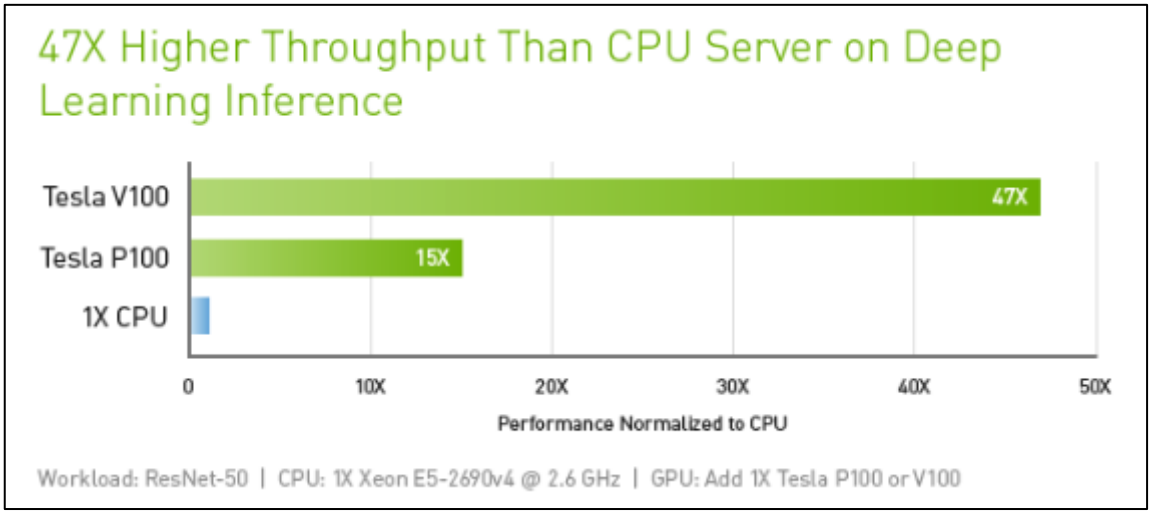

Рисунок 3.10 – Відношення потужностей процессора та графічного прискорювача

Частиною TensorFlow є Keras – відкрита нейромережева бібліотека. Вона є своєрідним інтерфейсом для роботи з мережами глубокого машиного навчання. Keras надає можливість роботи з штучними нейронними мережами на більш високому рівні не залежно від модуля який проводить математичні обчислення. Також Keras містить велику кількість інструментів для обробки зображень, які будуть використанні в нашому додатку.

3.3.1.3 Windows Forms

Windows Froms це перевірений часом фреймворк, який допомагає створювати подіє-орієнтовані додатки з насиченим інтерактивним інтерфейсом користувача. Він є частиною фреймворку Microsoft .NET Framework та підтримується компанією Microsoft.

Прямим конкурентом данного фреймворку є фреймворк WPF також створенний компанією Microsoft. Але він частіше використовується коли

додаток потребує більш інтерактивного та гнучкого інтерфейсу. Для нашого додатку ми будемо використовувати Windows Forms, тому що це пришвидшить розробку прототипу, та задовольняє усім потребам додатку.

3.3.2 **Розробка клієнту**

Для створення клієнту було обрано трьохшарову архітектуру з наступними шарами:

- Шар представлення. Це шар який містить в собі класи які відповідають за формування інтерфейсу користувача та комунікацію з користувачем. Цей шар має залежності від абстракцій нижчого шару – шару бізнес логіки. Сам шар є найвищим, що означає, що ніякий інший шар не матиме залужностей від нього.
- Шар бізнес логіки. Це шар на якому зібрані сервіси, провайдери та помічники, які відповідають за обробку та маніпуляції данними. Вони готують дані перед передачею їх до шару представлення. Також сервіси містять у собі логіку по трансформації даних перед передачею їх до шару доступу до даних.
- Шар доступу до данних. Він аккумулює у собі сервіси для роботи зі сховищем даних. Ці сервіси допомогають зберігати, змінювати, видаляти та отримувати дані зі сховища данних. Є найнижчим шаром тому майже не має залежностей, окрім залежностей від шару моделей, якщо такий існує.

На рисунку 3.11 можно побачити структуру архітектури клієнта

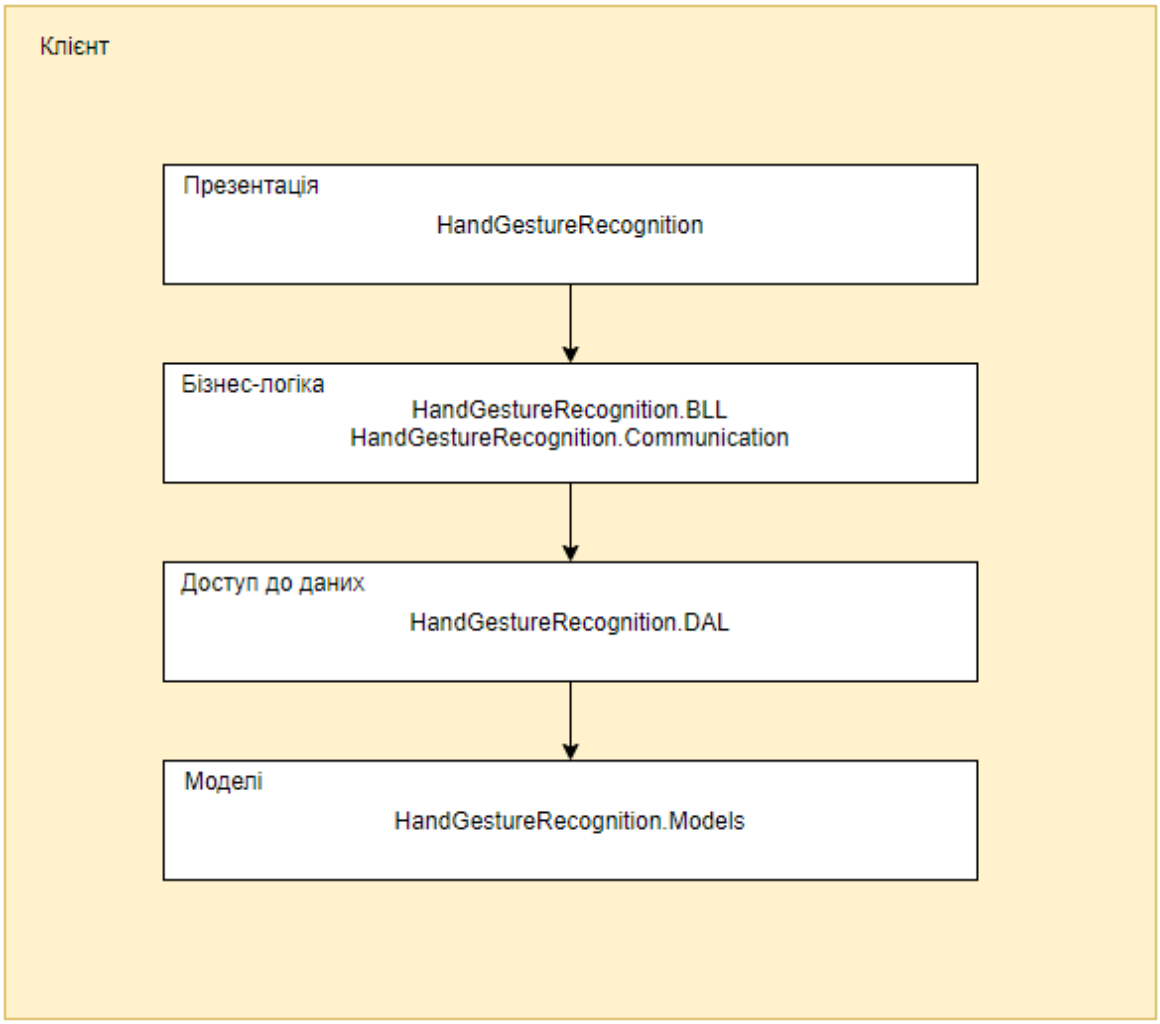

Рисунок 3.11 – Архітектура клієнту

3.3.3 **Розробка серверу розпізнавання**

Розробка серверу розпізнавання включає в себе реалізацію двух частин: частину комунікації з клієнтом, та частину для роботи з моделлю нейронної мережі. Перша частина буде отримувати дані від клієнта та відправляти результати роботи нейронної мережі, друга частина буде обробляти дані і за допомогою завантаженої моделі створювати передбачення щодо жестів показаних на зображені.

3.3.4 **Розробка бази данних**

В якості СКБД використовується SQL Server. Такий вибір обумовлюється декількома факторами:

– Серед сущностей присутні відношення, тому була обрана реляційна база данних.

3.3.5 **Розробка архітектури штучної нейронної мережі**

Модель штучної нейронної мережі приймає зображення як вхідний параметер та віддає масив значень від 0 до 1 для кожного класу. Значення є вірогідністю надлежності зображення до одного з класів, тому сума усіх значень масиву сумується до 1.

Так як вхідні та вихідни дані визначені, залишається визначити внутрішню будову мережі. Від внутрішньої будови моделі буде залежати точність передбачень моделі, швидкість її тренування та швидкість розрахунку передбачень.

Для прототипу нейронної мережі було вирішено використати наступну структуру:

– Вхідний шар – шар нормалізації. Цей шар отримує на вході тензор 200x200x3 (120000 нейронів) на виході такий же тензор. Цей шар отримує зображення розміром 200 на 200 пікселей з трьома каналами (червоний, зелений, синій). Значення кожного пікселя в кожному з каналів знаходиться між значеннями 0 та 255, та це не дуже зручно для обробки нейроною мережею, тому всі значення приводяться до значень між 0 та 1
– Шар розпізнання. Цей шар представляє з себе повноцінну натренеровану модель під назвою MobileNetV2. Ця модель має власну архітектуру на натренована на одному з найбільших датасетів COCO (Common Objects in Context). Модель розробляється та підтримується Google. Також у моделі відсутній класифікатор, для того щоб мати можливість натренувати його для розпізнання своїх класів. Тому основна роль шару – виділення характеристик, які далі будуть використовані для класифікації. На вході шар отримує тензор розміром 7x7x1280 (62720 нейронов) та на виході віддає такий же тензор.
– Шар вирівнювання. Цей шар приймає на вхід тензор 7x7x1280 (62720 нейронов), та на виході має тензор розміром 62720. Роль шару – вирівнювання багатомірного тензору в одномірний масив для того щоб передати його далі на шар класифікації.
– Шар класифікації – шар, який отримує тензор розміром 62720 та ввідає масив з значеннями які визначають вірогідність кожного класу.

3.4 Детальне проектування класів (опис методів класів) для реалізації підсистем

Згідно з функціональними вимогами до ПЗ доцільною є декомпозиція додатку на декілька класів тим самим дотримуючись принципу єдиної відповідальності та покращуючи загальниу архітектуру додатку.

В результаті декомпозиції були розроблені наступні компоненти:

- Рівень презентації
  - MainForm
  - LoginForm
  - RegistrationForm
  - Program
- Рівень бізнес логіки
  - AuthorizationService
  - ConfigurationManager
  - Encoder
  - LevelHandler
  - LeverService
  - UserService
- Рівень доступу до даних
  - DatabaseInitializer
  - DataContext
  - Repository
  - UnitOfWork
- Рівень комунікації
  - Client
- Моделі
  - Level
  - User
  - VideoFrame

На основі варінатів використання було спроектовано властивості і методи класів додатку. Результати проектування описані в таблицях 3.13-3.25, діаграма класів відображена на малюнку 3.11.

Таблиця 3.13 – Опис класу − MainForm

Назва класу	MainForm
Властивості класу	_tokens _authorizationService _levelService _levelHandler _filterInfoCollection _videoCaptureDevice _rectangle LetterChanged _configurationManager _receivedPredictions TrainFolderPath
Методи класу	OnLetterChanged OnFinished UpdateExcercise MainForm_Load PrepareLevels PrepareCamera PrepareLabels LoginUser FinalFrame_NewFrame UpdateLabel MainForm_FormClosing SelectCamera SelectResolution comboBoxCamera_SelectedIndexChanged resolutionComboBox_SelectedIndexChanged MainForm_KeyDown buttonSavePicture_Click buttonLogOut_Click listBoxLevels_SelectedIndexChanged UpdateRecord

Таблиця 3.14 – Опис класу − LoginForm

Назва класу	LoginFrom
Властивості класу	Authorized _authorizationService
Методи класу	buttonLogin_Click buttonRegister_Click

Таблиця 3.15 – Опис класу RegistrationForm

Назва класу	RegistrationForm
Властивості класу	_userService _encoder
Методи класу	buttonRegister_Click

Таблиця 3.16 – Опис класу Program

Назва класу	Program
Властивості класу	Kernel
Методи класу	Main

Таблиця 3.17 – Опис класу AuthorizationService

Назва класу	AuthorizationService
Властивості класу	_userService _configurationManager UserFilePath User
Методи класу	LogIn CreateUserFile Logout

Таблиця 3.18 – Опис класу ConfigurationManager

Назва класу	ConfigurationManager
Методи класу	GetValue GetValueOrDefault GetConnectionString GetSection

Таблиця 3.19 – Опис класу Encoder

Назва класу	Encoder
Методи класу	Encode

Таблиця 3.20 – Опис класу LevelHandler

Назва класу	LevelHandler
Властивості класу	_currentIndex IsFinished Started Timer Level
Методи класу	Start Turn ParseInput TranslateCurrentCharacter

Таблиця 3.21 – Опис класу LevelService

Назва класу	LevelService
Властивості класу	_unitOfWork
Методи класу	Create Update Delete Get GetAll

Таблиця 3.22 – Опис класу UserService

Назва класу	UserService
Властивості класу	_unitOfWork _encoder
Методи класу	Create Update Delete Get GetAll ValidPassword

Таблиця 3.23 – Опис класу DatabaseInitialiser

Назва класу	DatabaseInitialiser
Методи класу	Seed

Таблиця 3.24 – Опис класу DataContext

Назва класу	DataContext
Властивості класу	Users Levels
Методи класу	OnModelCreating

Таблиця 3.25 – Опис класу Repository

Назва класу	Repository
Властивості класу	_dataContext _entities
Методи класу	Create Get Remove Update

Таблиця 3.25 – Опис класу UnitOfWork

Назва класу	UnitOfWork
Властивості класу	_dataContext _users _levels
Методи класу	Commit

Таблиця 3.25 – Опис класу Level

Назва класу	Level
Властивості класу	Id Name Exercise Record
Методи класу	ToString

Таблиця 3.25 – Опис класу User

Назва класу	User
Властивості класу	Email Name Surname EncodedPassword

Таблиця 3.25 – Опис класу VideoFrame

Назва класу	VideoFrame
Властивості класу	Height Width Data

3.5 **Розробка інтерфейсу**

Інтерфейс додатку був розроблений на основі функціональних вимог, а також дотримуючись основних правил створення інтерфейсів користувача.

Нижче, на рисунку 3.12, можно побачити форму логіну. Вона з’являється в самому початку роботи з додатком. Форма використовується для входу до додатку. Вхід потрібен для отримання інформації щодо користувача, його прогресу та інших даних.

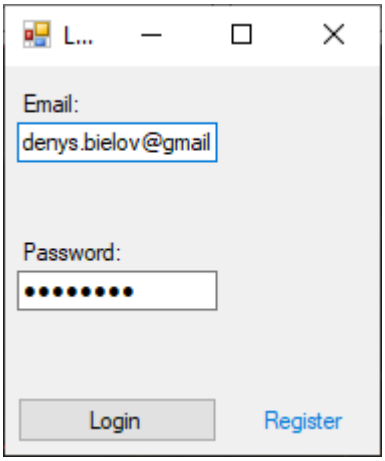

Рисунок 3.12 – форма логіну

Якщо користувач не має створеного облікового запису у системі, то він має можливість створити її, перейшовши на форму реєстрації. Її можно побачити на рисунку 3.13.

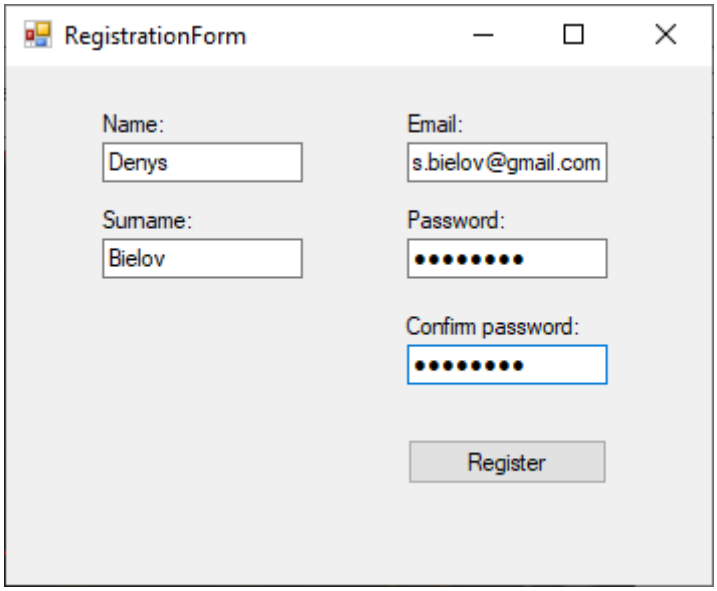

Рисунок 3.13 – форма реєстрації

Після того як обліковий запис створено, та виконано вхід в систему користувач матиме змогу побачити основне вікно додатку. Воно зображене на малюнках 3.14-3.16.

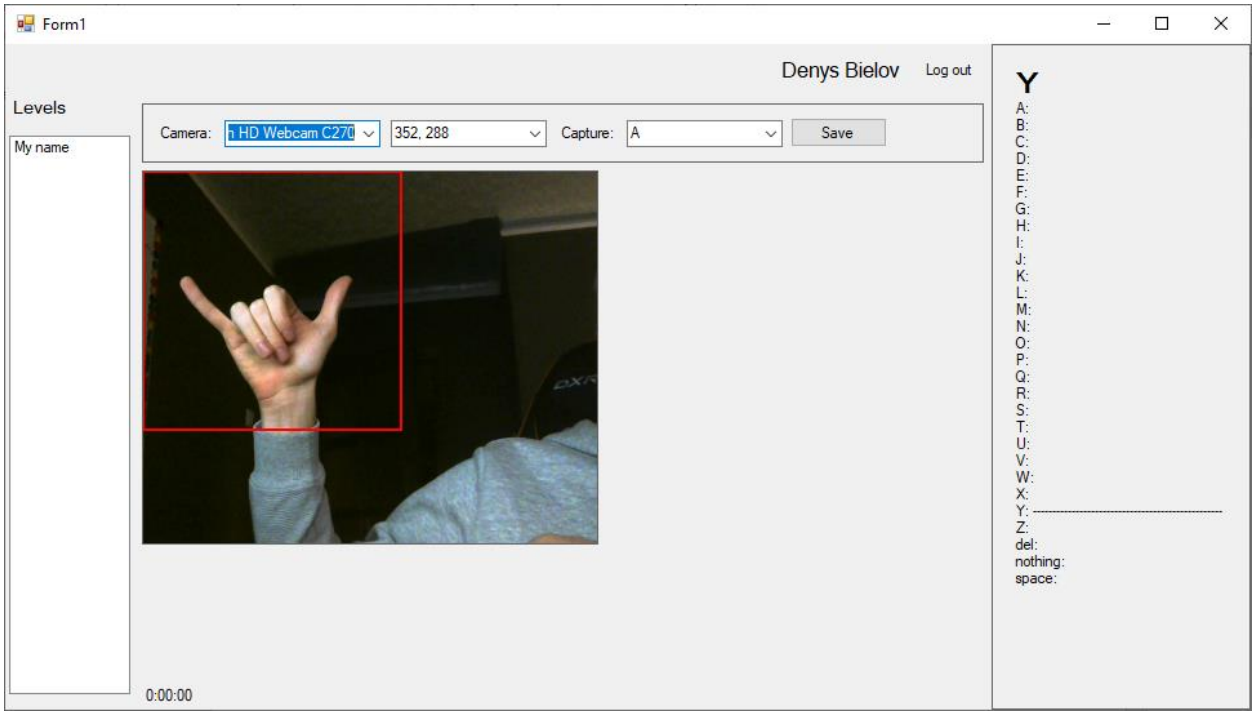

Рисунок 3.14 – основне вікно додатка з жестом Y

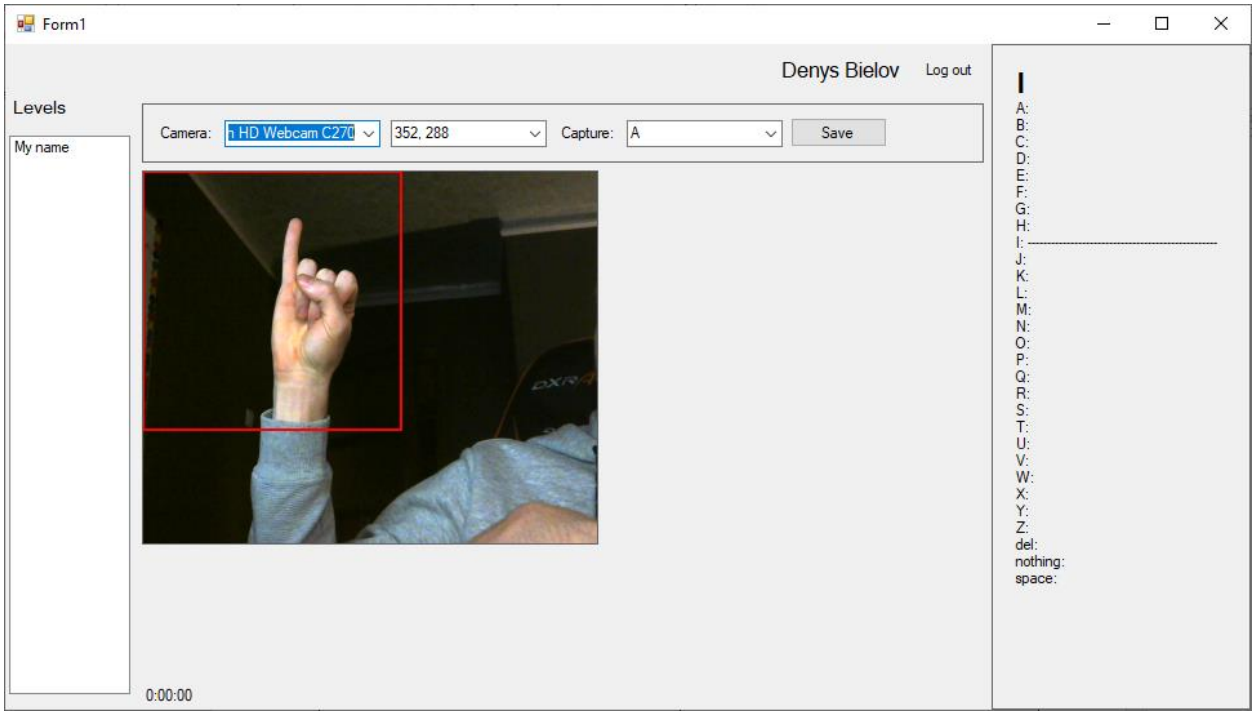

Рисунок 3.15 – вікно додатка з жестом I

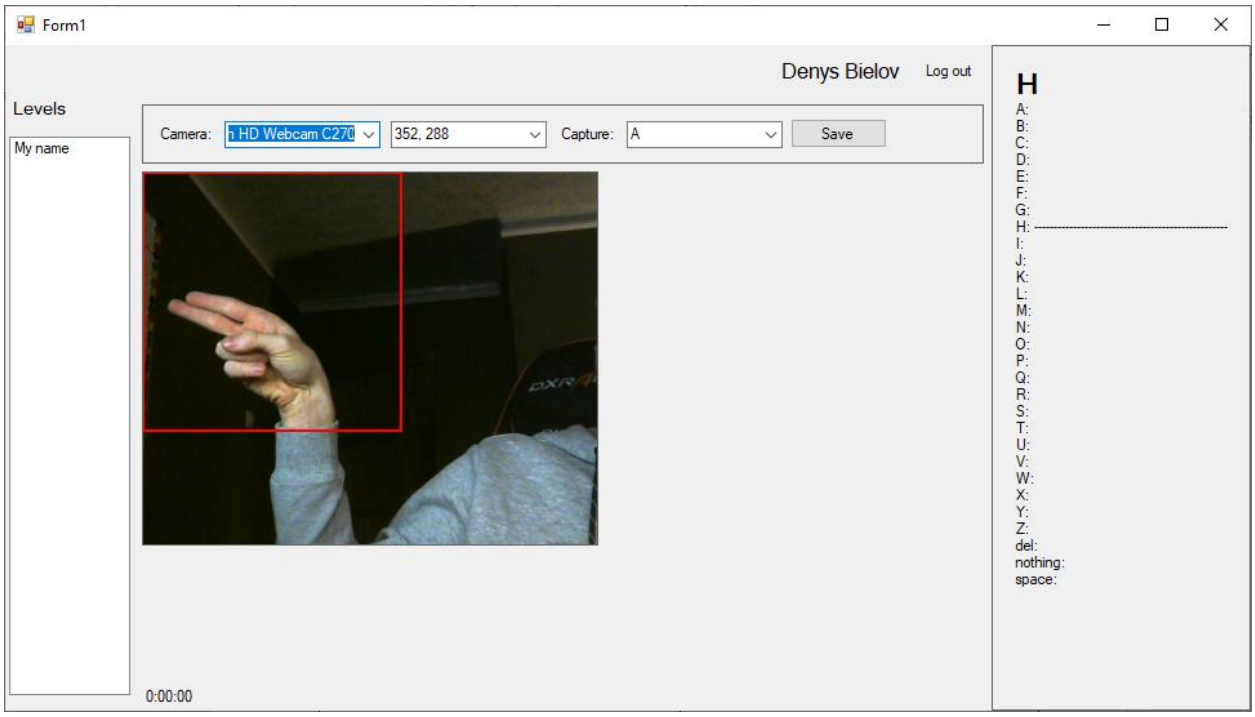

Рисунок 3.16 – вікно додатка з жестом H

Також користувач має можливість обрати один з наявних рівнів та перейти у вікно проходження рівня. Вікно проходження рівня зображено на рисунку 3.17.

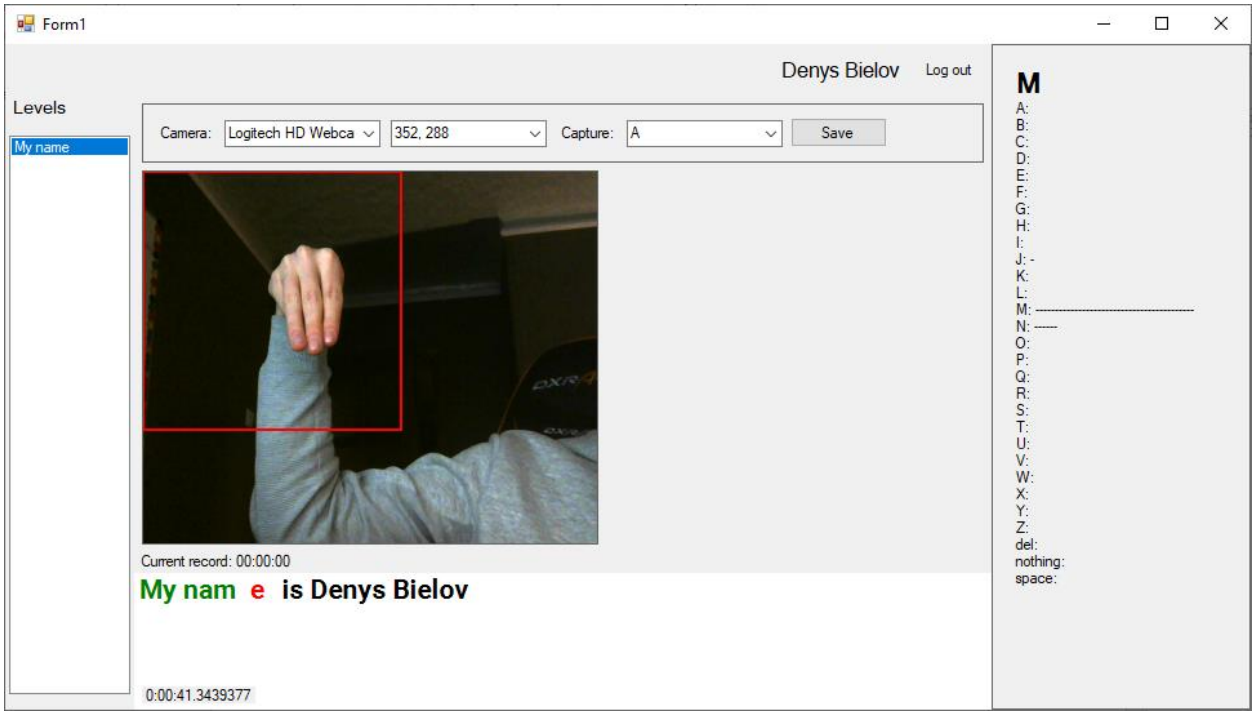

Рисунок 3.17 – вікно додатка з обраним рівнем

Після завершення рівня користувач отримає повідомлення з результатами виконання рівня. Зображення результатів можно побачити на рисунку 3.18.

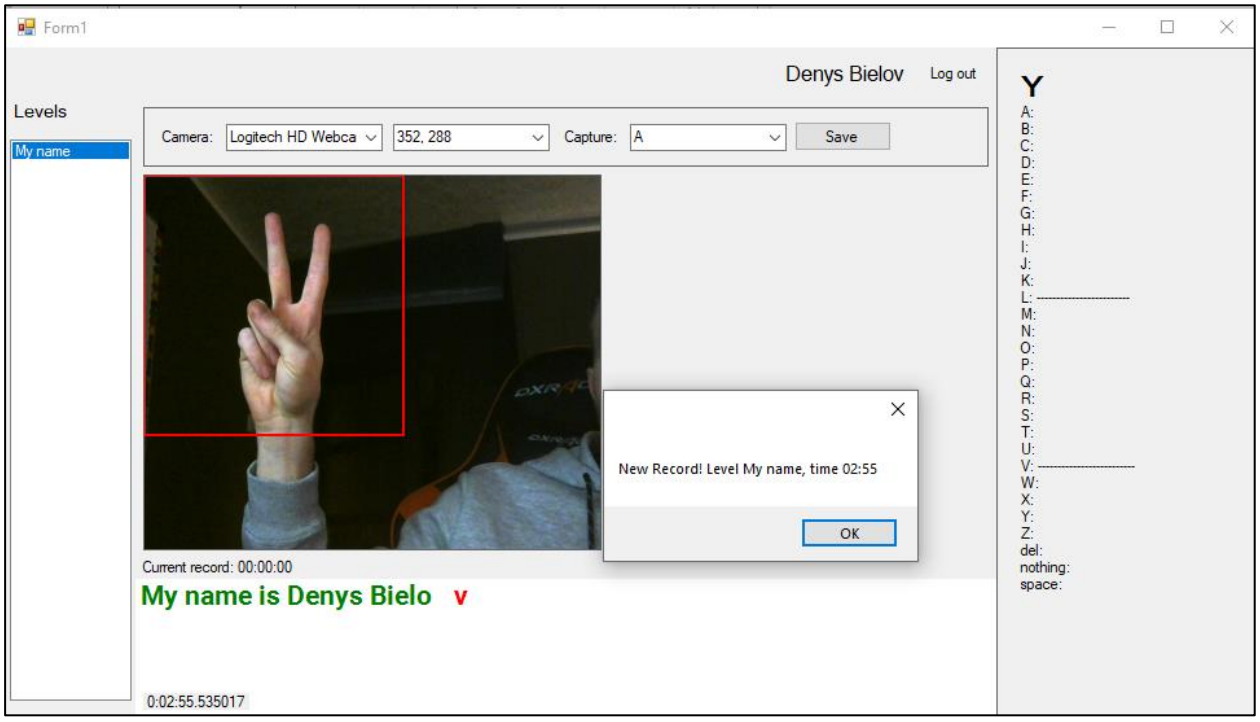

Рисунок 3.18 – вікно додатка з завершеним рівнем

3.6 **Перевірка працездатності розроблених моделей та методів**

Для перевірки працездатності розробленого прототипу ПЗ було проведено валідацію моделі штучної нейронної мережі на тестовому датасеті який не був використаний у процессі навчання. При цьому була проведено різна кількість циклів навчання. Нижче на рис. 3.19 можно побачити графік залежності точності класифікації зображень від кількості циклів навчання.

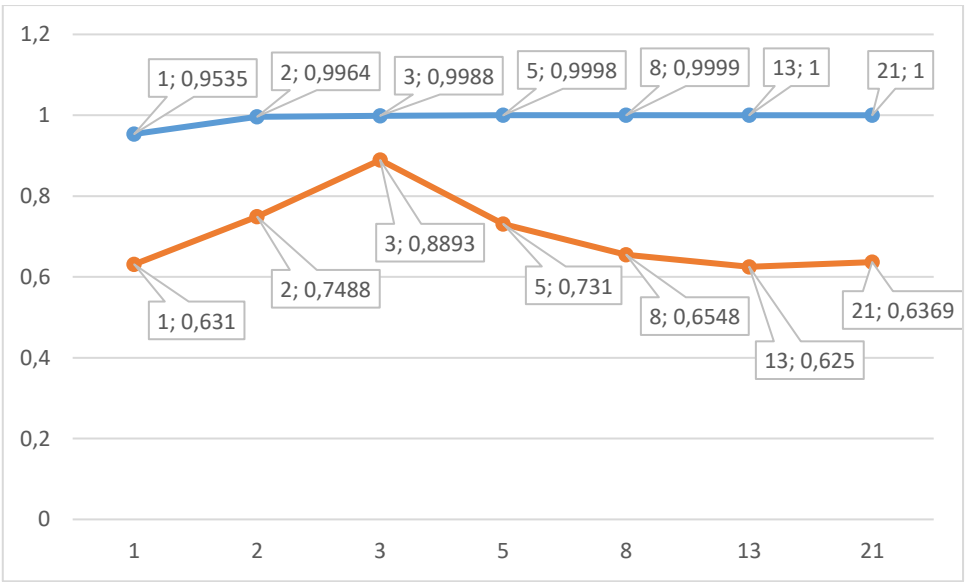

Рисунок 3.19 – Графік залежності точності класифікації від кількості циклів навчання

Як бачимо з графіку, найкращої точності на тестовому сеті вдалося досягти при 3-х циклах навчання. Далі точність на тестовому сеті йде на спад, при цьому точність на валідаційному сеті йде в гору досягаючи значення 1, що означає 100% точність. Частіше за все це говорить про перетренованість нейронної мережі, тобто мережа занадто звикла до навчальних даних і починає гірше працювати з даними яких вона ще не бачила.

3.7 **Висновки з розділу 3**

В даному розділу було сформовано цілі, а також задачі для розробки прототипу програмного засобу:

- проектування і розробка дизайну користувальницького інтерфейсу, робота над досвідом взаємодії;
- розробка користувальницького інтерфейсу і клієнтської частини додатку;

– проектування структури бази даних, розробка серверу розпізнання, інтеграція з базою даних.

На основі вимог та специфікацій варіантів використання були спроектовані макети екранних форм.

Після вибору архітектури програмного засобу для клієнту, серверу та їх взаємодії, було проведено вибір стеку технологій які знадобляться для реалізації описаної раніше логіки з зручним та інтуїтивно зрозумілим інтерфейсом.

За основу користувацького прототипу програмного засобу була обрана бібліотека Windows Forms яка надає функціонал для отримання та обробки даних від користувача. Для серверу розпізнавання було обрано бібліотеку Tensorflow на мові Python. Була розроблена архітектура штучної нейроної мережі. Був підготовлений датасет. Датасет був використаний для навчання та валідації моделі штучної нейроної мережі.

Мовою програмуваня для реалізації клієнту було обрано C#. Для реалізації серверу розпізнання було обрано мову Python. В якості СКБД було обрано Microsoft SQL Server.

Лістинг коду розробленого прототипу представлено у додатку А.

ВИСНОВКИ

В результаті написання дипломного проекту магістра на тему «Програмне забезпечення для навчання дактилології» було виконано аналіз проблем соціалізації людей з вадами слуху також були проаналізовані методи і засоби допомоги соціалізації людям з вадами слуху та оглянуті готові програмні комплекси.

Далі було проведено аналіз моделей і методів технології розпізнавання жестів на основі відеоряду в результаті чого було вибрано алгоритми агрегації та обробки данних для розроблюваного програмного засобу.

Виконано аналіз вимог до програмного забезпечення. Побудована концептуальна модель клієнтської і серверної частин програмного засобу, описані специфікації варіантів використання, розроблені макети екранних форм додатку.

Виконано проектування програмного забезпечення. Розроблено модель архітектури клієнту, спроектовані класи, побудована діаграма компонентів і діаграма розгортання. Обґрунтовано вибір середовища розробки та мови програмування, обґрунтований вибір СКБД для реалізації програмного продукту. Виконано декомпозицію проекту, описані методи класів і розроблені алгоритми для реалізації.

Проведено тестування працездатності розроблених моделей. Проведено дослідження залежності ефективності моделі штучної нейронної мережі від кількості циклів навчання.

ПЕРЕЛІК ПОСИЛАНЬ

1. Абакумов В.Г. Интерпретация движений рук расширяет возможности интерактивного управления в интеллектуальных системах / В.Г. Абакумов, Е.Ю. Ломакина // Природные и интеллектуальные ресурсы Сибири. – 2009. – С.199-202
2. Lomakina O.Y. Gestures Recognition as a New Information Input Device for Automatic System Control / O.Y. Lomakina // «Modern Problems of Radio Engineering, Telecommunications, and Computer Science» : Proceedings of X-th International Conference TCSET'2010. – Lviv-Slavske, 2010. – P. 100-103.
3. Hand tracking and gesture recognition for human-computer interaction / [C. Manresa et al.] // Electronic Letters on Computer Vision and Image Analysis. – 2005. – № 5(3). – P. 96-104.
4. Абакумов В.Г. Застосування жестів рук при людино-машинному інтерфейсі / В.Г. Абакумов, О.Ю. Ломакіна, О.Б. Яровенко // Электроника и связь. – Тематический выпуск : Электроника и нанотехнологии. – 2011.
5. Aggarwal J.K. Human Motion Analysis: A Review / Aggarwal J.K. and Q Cai. // Computer Vision and Image Understanding 73. –1999. – № 3. – P. 428-440.
6. Stenger B. (2001) Model-based 3D tracking of an articulated hand / B. Stenger, P.R.S. Mendonca, R. Cipolla // In: The 20th IEEE International Conference on Computer Vision and Pattern Recognition (CVPR'01). – December 2001. – Kauai, HI, US.
7. Kato M. Articulated Hand Tracking by PCA-ICA approach / M. Kato, Y.W. Chen and G. Xu // Proceedings of the IEEE International Conference on Automatic Face and Gesture Recognition. – Southampton, 2006. – P. 329-333.
8. Sign Language Translator – Режим доступа: http://research.microsoft.com/enus/collaboration/stories/kinect-sign-lang_screen.jpg. Дата доступа: 06.04.2016.
9. Motion Savvy: Рекламний постер – Режим доступа: https://images.indiegogo.com/file_attachments/931256/files/20141014180025 - features.png?1413334825. − Дата доступа: 25.04.2016.
10. Приклад фільтрації за порогом. – Режим доступа: https://habrastorage.org/getpro/habr/post_images/b22/e12/c6d/b22e12c6d33e b898e34b373475be35d8.jpg. Дата доступа: 10.04.2016.
11. Continuous Wavelet Transform – Режим доступа: http://www.satmagazine.com/cgi-bin/display_image.cgi?1114554895. Дата доступа: 10.04.2016.
12. Matlab: Корреляція зображень. – Режим доступа: http://www.mathworks.com/help/images/ref/imshowpair_ex1_diff.png. Дата доступу: 10.04.2016.

13. Boundary Detection with Sketch Tokens - Режим доступа http://cs.brown.edu/courses/cs143/proj5/teaser.png. – Дата доступа: 13.04.2016.
14. Farneback optical flow – Режим доступа: https://1.bp.blogspot.com/-SYVmZsXoYuA/Vs6-nFguzBI/AAAAAAAABr4/T1Iwd8Bq3o/s1600/opticalflow.png - Дата доступа: 18.03.2016.
15. T.J. Keating "An Improved Method of Digital Image Correlation" Photogrammetric Engineering and Remote Sensing \ T.J. Keating, P.R. Wolf, F.L. Scarpace, 1975 . – С. 993-1002.
16. SIFT: Офіційний сайт – Режим доступа: http://sift.jcvi.org/- Дата доступа: 20.03.2016.
17. SIFT: Опис ключової точки. – Режим доступа: http://www.vlfeat.org/api/sift-frame.png. – Дата доступа: 03.03.2016. 63
18. Байгарова Н.С. Методы индексирования и поиска изображений. – Режим доступа: http://web.ihep.su/library/pubs/aconf00/dconf00/ps/030.pdf. \ Байгарова Н.С., Бухштаб Ю.А, Горный А.А. Дата доступа: 10.05.2016.
19. Башков Е.А. Поиск изображений в больших БД с использованием корреляции цветовых гистограм.\ Башков Е.А., Шозда Н.С. – Режим доступа: http://graphicon.ru/html/2002/pdf/Bashkov_Zhozda_Re.pdf. Дата доступа: 10.05.2016.
20. Броневич А.Г. Анализ неопределенности выделения информативных признаков и представление изображений \ АГ Броневич, 2013 . – С. 65-69.
21. TensorFlow [Електронний ресурс] // Wikipedia. – 2019. – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/TensorFlow.
22. Сурдопереводчик [Електронний ресурс] // Wikipedia – Режим доступу до ресурсу: https://ru.wikipedia.org/wiki/Сурдопереводчик
23. Нейронные сети для начинающих [Електронний ресурс] // Habr. – 2019. – Режим доступу до ресурсу: https://habr.com/ru/post/312450/.
24. A Comprehensive Guide to Convolutional Neural Networks [Електронний ресурс] // towards data science. – 2019. – Режим доступу до ресурсу: https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53.
25. Sign Language Translator [Електронний ресурс] // spreadthesign. – 2019. – Режим доступу до ресурсу: https://www.spreadthesign.com/en.us/search/.
26. How smartphones handle huge Neural Networks [Електронний ресурс] // Heartbeat. – 2019. – Режим доступу до ресурсу: https://heartbeat.fritz.ai/how-smartphones-manage-to-handle-huge-neural-networks-269debcb243d.
27. Convolutional Neural Network (CNN) [Електронний ресурс] // TensorFlow. – 2019. – Режим доступу до ресурсу: https://www.tensorflow.org/tutorials/images/cnn.

28. A Comprehensive Guide to Types of Neural Networks [Електронний ресурс] // Digital Vidya. – 2019. – Режим доступу до ресурсу: https://www.digitalvidya.com/blog/types-of-neural-networks/.
29. Action classification with convolutional neural networks [Електронний ресурс] // Coursera. – 2019. – Режим доступу до ресурсу: https://ru.coursera.org/lecture/deep-learning-in-computer-vision/action-classification-with-convolutional-neural-networks-0R0ds.
30. Action Recognition from Videos using Deep Neural Networks [Електронний ресурс] // escholarship. – 2019. – Режим доступу до ресурсу: https://escholarship.org/uc/item/2mr798mn.
31. Kevin G. An introduction to neural networks / Gurney Kevin. – London and New York: University of Sheffield, 2004. – 317 с. – (e Taylor & Francis e-Library).
32. Asd
33. Функція Гевісайда [Електронний ресурс] // Wikipedia. – 2020. – Режим доступу до ресурсу: uk.wikipedia.org/wiki/Функція_Гевісайда.
34. Сигмоїда [Електронний ресурс] // Wikipedia. – 2020. – Режим доступу до ресурсу: uk.wikipedia.org/wiki/Сигмоїда.
35. ReLU [Електронний ресурс] // Wikipedia. – 2020. – Режим доступу до ресурсу: uk.wikipedia.org/wiki/ReLU.
36. Рекурентна нейронна мережа [Електронний ресурс] // Wikipedia. – 2020. – Режим доступу до ресурсу: uk.wikipedia.org/wiki/Рекурентна_нейронна_мережа.
37. Найвагоміші проблеми людей з вадами слуху в Україні [Електронний ресурс] // Громадський простір. – 2014. – Режим доступу до ресурсу: https://ua.interfax.com.ua/news/press-release/225754.html.
38. Deafness [Електронний ресурс] // World Health Organization. – 2018. – Режим доступу до ресурсу: https://www.who.int/news-room/facts-in-pictures/detail/deafness.
39. Потапчук В. DOU Проектор: BeWarned — приложения для глухих и слабослышащих людей [Електронний ресурс] / Виталий Потапчук // DOU. – 2017. – Режим доступу до ресурсу: https://dou.ua/lenta/articles/dou-projector-bewarned/.
40. Design and Implementation of Application for the Hearing Impaired People [Електронний ресурс] / [K. ChoulWoo, K. Dongyun, L. Hyundo та ін.] // MATEC Web of Conferences. – 2017. – Режим доступу до ресурсу: https://www.matec-conferences.org/articles/matecconf/pdf/2017/39/matecconf_cscc2017_04007.pdf.
41. Larry M. Mobile Device Apps for People with Hearing Loss [Електронний ресурс] / Medwetsky Larry // Hearing Loss Magazine. – 2015. – Режим доступу до ресурсу: https://www.hearingloss.org/wp-content/uploads/HLM_NovDec2015_Medwetsky.pdf?pdf=2015-hlm-nd-lmedwetsky.

42. Joshi A. Assistive Android Application For Hearing Impaired People Using Sign Language [Електронний ресурс] / A. Joshi, M. Homeminitha, P. Pavithra // AENSI Publication. – 2017. – Режим доступу до ресурсу: http://www.aensiweb.net/AENSIWEB/anas/anas/2017/May/166-171.pdf.
43. Dimensions for hearing-impaired mobile application usability model [Електронний ресурс] / [N. Shelena Soosay, H. Azham, H. Nor Laily та ін.] // AIP Publishing. – 2017. – Режим доступу до ресурсу: https://aip.scitation.org/doi/pdf/10.1063/1.5005441.

ДОДАТОК А

Лістинг програмного коду

MainForm.cs

```
using System;
using System.Collections.Generic;
using AForge.Video;
using AForge.Video.DirectShow;
using System.Drawing;
using System.Drawing.Imaging;
using System.IO;
using System.Linq;
using System.Windows.Forms;
using HandGestureRecognition.BLL.Interfaces;
using HandGestureRecognition.BLL.Services;
using HandGestureRecognition.Communication;
using HandGestureRecognition.Extensions;
using HandGestureRecognition.Model;
using Newtonsoft.Json;

namespace HandGestureRecognition
{
    public partial class MainForm : Form
    {
        private Dictionary<string, string> _tokens;
        private readonly IAuthorizationService _authorizationService;
        private readonly ILevelService _levelService;
        private readonly ILevelHandler _levelHandler;
        private FilterInfoCollection _filterInfoCollection;
        private VideoCaptureDevice _videoCaptureDevice;
        private Rectangle _rectangle = new Rectangle(0, 0, 200, 200);
        private event EventHandler<string> LetterChanged;

        private readonly ConfigurationManager _configurationManager;
        private bool? _receivedPredictions;
        private string TrainFolderPath =>
_configurationManager.GetValue(Constants.TrainFolderPath);
        private string TestFolderPath =>
_configurationManager.GetValue(Constants.TestFolderPath);

        public MainForm(IAuthorizationService authorizationService,
ILevelService levelService, ILevelHandler levelHandler)
        {
            _configurationManager = new ConfigurationManager();
            _authorizationService = authorizationService;
            _tokens = new Dictionary<string, string>();
            _levelService = levelService;
            _levelHandler = levelHandler;
            _levelHandler.Finished += OnFinished;

            LetterChanged += OnLetterChanged;

            InitializeComponent();
        }

        private void OnLetterChanged(object sender, string e)
        {
            if (_levelHandler.Started && !_levelHandler.IsFinished)
            {
```

```
                var result = _levelHandler.Turn(e);

                UpdateExcercise(result);
            }
        }

        private void OnFinished(object sender, TimeSpan time)
        {
            if (time < _levelHandler.Level.Record || _levelHandler.Level.Record.Ticks == 0)
            {
                MessageBox.Show(string.Format("New Record! Level {0}, time {1:D2}:{2:D2}", _levelHandler.Level,
                    _levelHandler.Timer.Elapsed.Minutes, _levelHandler.Timer.Elapsed.Seconds));
                _levelHandler.Level.Record = _levelHandler.Timer.Elapsed;
                UpdateRecord(_levelHandler.Level.Record);
                _levelService.Update(_levelHandler.Level);
            }
        }

        private void UpdateExcercise(LevelHandler.TurnResult result)
        {
            var textChunks = new List<PictureBoxExtensions.ColoredText>();

            if (result.CurrentIndex > 0)
            {
                textChunks.Add(new PictureBoxExtensions.ColoredText
                {
                    Color = Color.Green,
                    Text = result.Level.Exercise.Substring(0, result.CurrentIndex)
                });
            }

            if (result.CurrentIndex < result.Level.Exercise.Length)
            {
                textChunks.Add(new PictureBoxExtensions.ColoredText
                {
                    Color = Color.Red,
                    Text = result.Level.Exercise.Substring(result.CurrentIndex, 1)
                });
            }

            if (result.CurrentIndex < result.Level.Exercise.Length - 1)
            {
                textChunks.Add(new PictureBoxExtensions.ColoredText
                {
                    Color = Color.Black,
                    Text = result.Level.Exercise.Substring(result.CurrentIndex + 1,
                        result.Level.Exercise.Length - result.CurrentIndex - 1)
                });
            }

            pictureBoxExercise.RenderText(textChunks.ToArray());
        }

        private void MainForm_Load(object sender, EventArgs e)
```

```
        {
            LoginUser();

            PrepareCamera();

            PrepareLabels();

            PrepareLevels();

            Client.Instance.StartClient();
        }

        private void PrepareLevels()
        {
            Level[] levels = _levelService.GetAll().ToArray();

            listBoxLevels.Items.AddRange(levels);
        }

        private void PrepareCamera()
        {
            _filterInfoCollection                    =                    new
FilterInfoCollection(FilterCategory.VideoInputDevice);
            if (_filterInfoCollection.Count > 0)
            {
                foreach (FilterInfo device in _filterInfoCollection)
                {
                    comboBoxCamera.Items.Add(device.Name);
                }

                comboBoxCamera.SelectedIndex = 0;
                SelectCamera(0);
            }
        }

        private void PrepareLabels()
        {
            var                          directories                          =
Directory.EnumerateDirectories(TrainFolderPath).Select(Path.GetFileName);

            foreach (var directory in directories)
            {
                comboBoxLabels.Items.Add(directory);
            }

            comboBoxLabels.SelectedIndex = 0;
        }

        private void LoginUser()
        {
            if (_authorizationService.User == null)
            {
                var loginForm = new LoginFrom(_authorizationService);

                loginForm.ShowDialog(this);

                if (!loginForm.Authorized)
                {
                    Close();

                    return;
```

```
                }

            }
            labelUserName.Text      =      $"{_authorizationService.User?.Name}
{_authorizationService.User?.Surname}";
            if (_tokens.ContainsKey("{name}"))
            {
                _tokens["{name}"] = labelUserName.Text;
            }
            else
            {
                _tokens.Add("{name}", labelUserName.Text);
            }
        }

        private async void FinalFrame_NewFrame(object sender, NewFrameEventArgs
eventArgs)
        {
            var image = (Bitmap)eventArgs.Frame.Clone();

            var imageToSend = image;

            if     (_videoCaptureDevice.VideoResolution.FrameSize.Height     >
_rectangle.Size.Height &&
                _videoCaptureDevice.VideoResolution.FrameSize.Width          >
_rectangle.Size.Width)
            {
                imageToSend = image.Clone(_rectangle, image.PixelFormat);

                using (Graphics graphics = Graphics.FromImage(image))
                {
                    using (Pen myBrush = new Pen(Color.Red, 2))
                    {
                        graphics.DrawRectangle(myBrush, _rectangle);
                    }
                }
            }

            void  ChangeMainCameraPicture()  =>  pictureBoxMainCamera.Image  =
image;

            pictureBoxMainCamera.Invoke((Action)ChangeMainCameraPicture);

            if (_receivedPredictions == null || _receivedPredictions == false)
            {
                _receivedPredictions = true;
                await Client.Instance.SendToServer(new VideoFrame
                {
                    Data = ImageToByte(imageToSend)
                }).ContinueWith(async task =>
                {
                    var response = await Client.Instance.Receive<string>();
                    if (!string.IsNullOrWhiteSpace(response))
                    {
                        UpdateLabel(response);
                    }

                    _receivedPredictions = false;
                });
            }
        }
```

```
        private void UpdateLabel(string predictions)
        {
            var dict = JsonConvert.DeserializeObject<Dictionary<string,
double>>(predictions);
            var max = dict.FirstOrDefault(x => Math.Abs(x.Value -
dict.Values.Max()) < 0.00001).Key;
            var str = string.Join("\n", dict.Select(p => $"{p.Key}:
{String.Concat(Enumerable.Repeat("-", (int)(p.Value * 50)))}" /*p.Key == max ?
$"----> {p.Key} - {p.Value}" : $"{p.Key} - {p.Value}"*/));
            void Action1() => labelPredictions.Text = str;
            labelPredictions.Invoke((Action)Action1);
            void Action2() => labelCurrentLetter.Text = max;
            if (labelCurrentLetter.Text != max)
            {
                LetterChanged(this, max);
            }
            void Action3() => labelTimer.Text =
_levelHandler.Timer.Elapsed.ToString("g");
            labelTimer.Invoke((Action)Action3);
            labelPredictions.Invoke((Action)Action2);
        }

        public static byte[] ImageToByte(Image img)
        {
            ImageConverter converter = new ImageConverter();
            return (byte[])converter.ConvertTo(img, typeof(byte[]));
        }

        private void MainForm_FormClosing(object sender, FormClosingEventArgs
e)
        {
            if (_videoCaptureDevice?.IsRunning ?? false)
            {
                _videoCaptureDevice.Stop();
            }
        }

        private void SelectCamera(int cameraIndex)
        {
            _videoCaptureDevice = new VideoCaptureDevice();
            _videoCaptureDevice = new
VideoCaptureDevice(_filterInfoCollection[cameraIndex].MonikerString);
            _videoCaptureDevice.NewFrame += FinalFrame_NewFrame;
            comboBoxResolution.Items.Clear();

            foreach (var videoCapability in
_videoCaptureDevice.VideoCapabilities)
            {
                comboBoxResolution.Items.Add(videoCapability.FrameSize);
            }

            comboBoxResolution.SelectedIndex = 5;
            _videoCaptureDevice.VideoResolution =
_videoCaptureDevice.VideoCapabilities[comboBoxResolution.SelectedIndex];
            _videoCaptureDevice.Start();
        }

        private void SelectResolution(int resolutionIndex)
        {
            if (_videoCaptureDevice.IsRunning)
```

```
            {
                _videoCaptureDevice.Stop();
            }
            _videoCaptureDevice = new VideoCaptureDevice();
            _videoCaptureDevice                    =                    new
VideoCaptureDevice(_filterInfoCollection[comboBoxCamera.SelectedIndex].Monike
rString);
            _videoCaptureDevice.NewFrame += FinalFrame_NewFrame;
            _videoCaptureDevice.VideoResolution                              =
_videoCaptureDevice.VideoCapabilities[resolutionIndex];
            _videoCaptureDevice.Start();
        }

        private   void   comboBoxCamera_SelectedIndexChanged(object   sender,
EventArgs e)
        {
            SelectCamera(comboBoxCamera.SelectedIndex);
        }

        private  void  resolutionComboBox_SelectedIndexChanged(object  sender,
EventArgs e)
        {
            SelectResolution(comboBoxResolution.SelectedIndex);
        }

        private void MainForm_KeyDown(object sender, KeyEventArgs e)
        {
            if (e.KeyCode == Keys.Space)
            {
                buttonSavePicture_Click(null, null);
            }
        }

        private void buttonSavePicture_Click(object sender, EventArgs e)
        {
            var                            folderName                            =
comboBoxLabels.GetItemText(comboBoxLabels.SelectedItem);
            var path = Path.Combine(TestFolderPath, folderName);

            var                            fileNumber                            =
Directory.EnumerateFiles(path).Select(Path.GetFileNameWithoutExtension).Max(n
=>
            {
                var currentFileNumber = n.Replace(folderName, string.Empty);
                if (int.TryParse(currentFileNumber, out var number))
                {
                    return number;
                }

                return 0;
            });

            var fileName = $"{folderName}{fileNumber + 1}.jpg";

            var filePath = Path.Combine(path, fileName);

            ((Bitmap)pictureBoxMainCamera.Image).Clone(_rectangle,
pictureBoxMainCamera.Image.PixelFormat)
                .Save(filePath, ImageFormat.Jpeg);

            labelFileCreated.Text = $"{fileName} created";
```

```
        }

        private void buttonLogOut_Click(object sender, EventArgs e)
        {
            _authorizationService.Logout();
            LoginUser();
        }

        private    void    listBoxLevels_SelectedIndexChanged(object    sender,
EventArgs e)
        {
            if (listBoxLevels.SelectedIndex == -1)
            {
                return;
            }

            var                                level                                =
(Level)listBoxLevels.Items[listBoxLevels.SelectedIndex];

            UpdateRecord(level.Record);

            level.Exercise = ReplaceTokens(level.Exercise);

            _levelHandler.Start(level);

            pictureBoxExercise.RenderText(new PictureBoxExtensions.ColoredText
            {
                Color = Color.Black,
                Text = level.Exercise
            });
        }

        private void UpdateRecord(TimeSpan record)
        {
            labelRecord.Text = $"Current record: {record}";
        }

        private string ReplaceTokens(string pattern)
        {
            foreach (var token in _tokens)
            {
                pattern = pattern.Replace(token.Key, token.Value);
            }

            return pattern;
        }
    }
}
```

Client.cs

```
using System;
using System.Net;
using System.Net.Sockets;
using System.Text;
using System.Threading;
using System.Threading.Tasks;
using HandGestureRecognition.Communication.Extensions;
using Newtonsoft.Json;

namespace HandGestureRecognition.Communication
```

```
{
    public class Client
    {
        private static readonly ManualResetEvent ConnectDone =
            new ManualResetEvent(false);
        private static readonly ManualResetEvent SendDone =
            new ManualResetEvent(false);
        private static readonly ManualResetEvent ReceiveDone =
            new ManualResetEvent(false);

        private const string IpAddress = "127.0.0.1";
        private const int Port = 44778;
        private static Client _instance;
        private Socket _server;
        public static Client Instance => _instance = _instance ?? new Client();

        // The response from the remote device.
        private static readonly string Response = string.Empty;

        public void StartClient()
        {
            var thread = new Thread(Receive);
            thread.Start();
        }

        private void Receive()
        {
            try
            {
                // Establish the remote endpoint for the socket.
                // The name of the
                IPAddress ipAddress = IPAddress.Parse(IpAddress);
                IPEndPoint endPoint = new IPEndPoint(ipAddress, Port);

                // Create a TCP/IP socket.
                Socket client = new Socket(ipAddress.AddressFamily,
                    SocketType.Stream, ProtocolType.Tcp);

                // Connect to the remote endpoint.
                client.BeginConnect(endPoint, ConnectCallback, client);
                ConnectDone.WaitOne();
            }
            catch (Exception e)
            {
                Console.WriteLine(e.ToString());
            }

        }

        private void ConnectCallback(IAsyncResult asyncResult)
        {
            try
            {
                // Retrieve the socket from the state object.
                Socket client = (Socket)asyncResult.AsyncState;

                // Complete the connection.
                client.EndConnect(asyncResult);

                Console.WriteLine("Socket connected to {0}",
                    client.RemoteEndPoint);
```

```
                _server = client;

                // Signal that the connection has been made.
                ConnectDone.Set();
            }
            catch (Exception e)
            {
                Console.WriteLine(e.ToString());
            }
        }

        public Task<T> Receive<T>()
        {
            return Task.Run(() =>
            {
                if (_server != null && _server.Connected)
                {
                    var state = new StateObject
                    {
                        WorkSocket = _server
                    };

                    try
                    {
                        ReceiveDone.Reset();

                        _server.BeginReceive(state.Buffer,                    0,
StateObject.BufferSize, 0, ReceiveCallback, state);

                        ReceiveDone.WaitOne();
                    }
                    catch(Exception exception)
                    {
                        Console.WriteLine(exception.Message);
                    }

                    if (!string.IsNullOrWhiteSpace(state.Response))
                    {
                        if (typeof(T) == typeof(String))
                        {
                            return       (T)Convert.ChangeType(state.Response,
typeof(T));
                        }

                        return
JsonConvert.DeserializeObject<T>(state.Response);
                    }
                }

                return default;
            });
        }

        public Task SendToServer<T>(T data)
        {
            return Task.Run(() =>
                   {
                       if (_server != null && _server.Connected)
                       {
                           Send(data);
```

```
                    }
                });

            //throw new ApplicationException("Server is not connected");
        }


        private void ReceiveCallback(IAsyncResult asyncResult)
        {
            // Retrieve the state object and the handler socket
            // from the asynchronous state object.
            try
            {
                // Retrieve the state object and the client socket
                // from the asynchronous state object.
                StateObject state = (StateObject)asyncResult.AsyncState;
                Socket client = state.WorkSocket;

                // Read data from the remote device.
                int bytesRead = client.EndReceive(asyncResult);

                if (client.Available > 0)
                {
                    // There might be more data, so store the data received so
far.

state.StringBuilder.Append(Encoding.ASCII.GetString(state.Buffer,           0,
bytesRead));

                    // Get the rest of the data.
                    client.BeginReceive(state.Buffer,                       0,
StateObject.BufferSize, 0, ReceiveCallback, state);
                }
                else
                {

state.StringBuilder.Append(Encoding.ASCII.GetString(state.Buffer,           0,
bytesRead));

                    // All the data has arrived; put it in response.
                    if (state.StringBuilder.Length > 1)
                    {
                        state.Response = state.StringBuilder.ToString();
                    }
                    // Signal that all bytes have been received.
                    ReceiveDone.Set();
                }
            }
            catch (Exception e)
            {
                Console.WriteLine(e.ToString());
            }

            ReceiveDone.Set();
        }

        private void Send<T>(T data)
        {
            try
            {
                if (!_server.IsDead())
```

```
                {
                    string dataString = JsonConvert.SerializeObject(data);

                    // Convert the string data to byte data using ASCII
encoding.
                    byte[] byteData = Encoding.ASCII.GetBytes(dataString);

                    SendDone.Reset();
                    // Begin sending the data to the remote device.
                    _server.BeginSend(byteData,    0,    byteData.Length,    0,
SendCallback, _server);

                    SendDone.WaitOne();
                }
                else
                {
                    _server = null;
                }
            }
            catch (SocketException exception)
            {
                Console.WriteLine(exception.Message);
                _server?.Shutdown(SocketShutdown.Both);
                _server?.Close();
            }
        }

        private void SendCallback(IAsyncResult asyncResult)
        {
            try
            {
                // Retrieve the socket from the state object.
                Socket handler = (Socket)asyncResult.AsyncState;

                // Complete sending the data to the remote device.
                int bytesSent = handler.EndSend(asyncResult);
                Console.WriteLine("Sent {0} bytes to client.", bytesSent);
                SendDone.Set();
            }
            catch (Exception e)
            {
                Console.WriteLine(e.ToString());
            }
        }
    }
}
```

Server.cs

```
using System;
using System.Net;
using System.Net.Sockets;
using System.Text;
using System.Threading;
using System.Threading.Tasks;
using HandGestureRecognition.Communication.Extensions;
using Newtonsoft.Json;

namespace HandGestureRecognition.Communication
{
    public class Server
    {
```

```
        private bool _isStarted;
        private static Server _instance;

        public static Server Instance => _instance = _instance ?? new Server();

        private Socket _client;
        private    readonly    ManualResetEvent    _acceptingDone    =    new
ManualResetEvent(false);
        private     readonly     ManualResetEvent     _sendDone     =     new
ManualResetEvent(false);
        private    readonly    ManualResetEvent    _receiveDone    =    new
ManualResetEvent(false);
        private Socket _serverSocket;
        private readonly IPEndPoint _endPoint;
        private Thread _serverThread;

        private Server()
        {
            _client = null;
            _endPoint = new IPEndPoint(IPAddress.Any, 44778);
        }

        private void Listen(object state)
        {
            try
            {
                _serverSocket    =    new    Socket(AddressFamily.InterNetwork,
SocketType.Stream, ProtocolType.Tcp);
                _serverSocket.Bind(_endPoint);
                _serverSocket.Listen(1);

                while (true)
                {
                    // Set the event to nonsignaled state.
                    _receiveDone.Reset();

                    // Start an asynchronous socket to listen for connections.
                    Console.WriteLine("Waiting for a connection...");
                    _serverSocket.BeginAccept(AcceptCallback, _serverSocket);

                    // Wait until a connection is made before continuing.
                    _receiveDone.WaitOne();
                }

            }
            catch (Exception exception)
            {
                Console.WriteLine(exception.Message);
                _serverSocket.Close();
            }
        }

        private void AcceptCallback(IAsyncResult asyncResult)
        {
            // Signal the main thread to continue.
            _receiveDone.Set();

            // Get the socket that handles the client request.
            Socket listener = (Socket)asyncResult.AsyncState;
            Socket handler = listener.EndAccept(asyncResult);
```

```
            _client = handler;

            //handler.BeginReceive(state.Buffer, 0, StateObject.BufferSize, 0,
ReadCallback, state);
        }

        private void ReadCallback(IAsyncResult asyncResult)
        {
            // Retrieve the state object and the handler socket
            // from the asynchronous state object.
            try
            {
                // Retrieve the state object and the client socket
                // from the asynchronous state object.
                StateObject state = (StateObject)asyncResult.AsyncState;
                Socket client = state.WorkSocket;

                // Read data from the remote device.
                int bytesRead = client.EndReceive(asyncResult);

                if (client.Available > 0)
                {
                    // There might be more data, so store the data received so
far.

state.StringBuilder.Append(Encoding.ASCII.GetString(state.Buffer,           0,
bytesRead));

                    // Get the rest of the data.
                    client.BeginReceive(state.Buffer,                       0,
StateObject.BufferSize, 0, ReadCallback, state);
                }
                else
                {

state.StringBuilder.Append(Encoding.ASCII.GetString(state.Buffer,           0,
bytesRead));

                    // All the data has arrived; put it in response.
                    if (state.StringBuilder.Length > 1)
                    {
                        state.Response = state.StringBuilder.ToString();
                    }
                    // Signal that all bytes have been received.
                    _receiveDone.Set();
                }
            }
            catch (Exception e)
            {
                Console.WriteLine(e.ToString());
            }

            _receiveDone.Set();
        }

        private void Send<T>(Socket handler, T data)
        {
            try
            {
                if (!handler.IsDead())
                {
```

```
                    string dataString = JsonConvert.SerializeObject(data);

                    // Convert the string data to byte data using ASCII
encoding.
                    byte[] byteData = Encoding.ASCII.GetBytes(dataString);

                    _sendDone.Reset();
                    // Begin sending the data to the remote device.
                    handler.BeginSend(byteData,    0,    byteData.Length,    0,
SendCallback, handler);

                    _sendDone.WaitOne();
                }
                else
                {
                    _client = null;
                }
            }
            catch (SocketException exception)
            {
                Console.WriteLine(exception.Message);
                handler.Shutdown(SocketShutdown.Both);
                handler.Close();
            }
        }

        private void SendCallback(IAsyncResult asyncResult)
        {
            try
            {
                // Retrieve the socket from the state object.
                Socket handler = (Socket)asyncResult.AsyncState;

                // Complete sending the data to the remote device.
                int bytesSent = handler.EndSend(asyncResult);
                Console.WriteLine("Sent {0} bytes to client.", bytesSent);
                _sendDone.Set();
            }
            catch (Exception e)
            {
                Console.WriteLine(e.ToString());
            }
        }

        public void StartServer()
        {
            if (!_isStarted)
            {
                _serverThread = new Thread(Listen);
                _serverThread.Start();
                _isStarted = true;
            }
        }

        public void StopServer()
        {
            _serverThread.Abort();
        }

        public Task SendToClient<T>(T data)
```

```
        {
            if (_isStarted)
            {
                return Task.Run(() =>
                {
                    if (_client != null)
                    {
                        Send(_client, data);
                    }
                });
            }

            throw new ApplicationException("Server is not started");
        }

        public Task<T> Receive<T>()
        {
            return Task.Run(() =>
            {
                if (_client != null)
                {
                    var state = new StateObject
                    {
                        WorkSocket = _client
                    };

                    try
                    {
                        _receiveDone.Reset();

                        _client.BeginReceive(state.Buffer,                    0,
StateObject.BufferSize, 0, ReadCallback, state);

                        _receiveDone.WaitOne();
                    }
                    catch
                    {
                        //ignore
                    }

                    if (!string.IsNullOrWhiteSpace(state.Response))
                    {
                        if (typeof(T) == typeof(String))
                        {
                            return       (T)Convert.ChangeType(state.Response,
typeof(T));
                        }

                        return
JsonConvert.DeserializeObject<T>(state.Response);
                    }
                }

                return default(T);
            });
        }
    }
}
```

UserService.cs

```
using System;
```

```
using System.Collections.Generic;
using System.Linq;
using HandGestureRecognition.BLL.Interfaces;
using HandGestureRecognition.DAL;
using HandGestureRecognition.Model;

namespace HandGestureRecognition.BLL.Services
{
    public class UserService : IUserService
    {
        private readonly IUnitOfWork _unitOfWork;
        private readonly IEncoder _encoder;

        public UserService(IUnitOfWork unitOfWork, IEncoder encoder)
        {
            _unitOfWork = unitOfWork;
            _encoder = encoder;
        }

        public void Create(User entity)
        {
            _unitOfWork.Users.Create(entity);
            _unitOfWork.Commit();
        }

        public void Update(User entity)
        {
            throw new NotImplementedException();
        }

        public void Delete(User entity)
        {
            throw new NotImplementedException();
        }

        public User Get(string key)
        {
            return       _unitOfWork.Users.Get(u      =>      u.Email      ==
key).SingleOrDefault();
        }

        public IEnumerable<User> GetAll()
        {
            throw new NotImplementedException();
        }

        public bool ValidPassword(string email, string password)
        {
            var user = Get(email);

            if (user == null)
            {
                return false;
            }

            var encodedPassword = _encoder.Encode(password);

            return user.EncodedPassword == encodedPassword;
        }
    }
}
```

LevelService.cs

```
using System.Collections.Generic;
using HandGestureRecognition.BLL.Interfaces;
using HandGestureRecognition.DAL;
using HandGestureRecognition.Model;

namespace HandGestureRecognition.BLL.Services
{
    public class LevelService : ILevelService
    {
        private IUnitOfWork _unitOfWork;

        public LevelService(IUnitOfWork unitOfWork)
        {
            _unitOfWork = unitOfWork;
        }

        public void Create(Level entity)
        {
            throw new System.NotImplementedException();
        }

        public void Update(Level entity)
        {
            _unitOfWork.Levels.Update(entity);
            _unitOfWork.Commit();
        }

        public void Delete(Level entity)
        {
            throw new System.NotImplementedException();
        }

        public Level Get(int key)
        {
            throw new System.NotImplementedException();
        }

        public IEnumerable<Level> GetAll()
        {
            return _unitOfWork.Levels.Get();
        }
    }
}
```

LevelHandler.cs

```
using System;
using System.Diagnostics;
using HandGestureRecognition.BLL.Interfaces;
using HandGestureRecognition.Model;

namespace HandGestureRecognition.BLL.Services
{
    public class LevelHandler : ILevelHandler
    {
        private int _currentIndex;

        public LevelHandler(Stopwatch timer)
        {
            Timer = timer;
```

```
}

private bool _isFinished;

public bool IsFinished
{
    get => _isFinished;
    private set
    {
        if (value)
        {
            Timer.Stop();
            Finished(this, Timer.Elapsed);
        }

        _isFinished = value;
    }
}

public bool Started { get; private set; }
public Stopwatch Timer { get; }
public Level Level { get; private set; }
public event EventHandler<TimeSpan> Finished;

public void Start(Level level)
{
    Level = level;
    _currentIndex = 0;
    Started = true;
}

public TurnResult Turn(string input)
{
    if (_currentIndex == 0)
    {
        Timer.Restart();
    }

    var type = ParseInput(input);

    switch (type)
    {
        case CharacterType.Character:
        case CharacterType.Space:
            if (!IsFinished)
            {
                if (TranslateCurrentCharacter() == input.ToLower())
                {
                    _currentIndex++;
                }

                if (_currentIndex >= Level.Exercise.Length)
                {
                    IsFinished = true;
                }
            }
            break;
        case CharacterType.Delete:
            _currentIndex--;
            IsFinished = false;
            if (!Timer.IsRunning)
```

```
                {
                    Timer.Start();
                }
                break;
            case CharacterType.Nothing:
                break;
            default:
                throw new ArgumentOutOfRangeException();
        }

        return new TurnResult
        {
            CurrentIndex = _currentIndex,
            Level = Level
        };
    }

    private CharacterType ParseInput(string input)
    {
        switch (input)
        {
            case "space":
                return CharacterType.Space;
            case "del":
                return CharacterType.Delete;
            case "nothing":
                return CharacterType.Nothing;
            default:
                return CharacterType.Character;
        }
    }

    private string TranslateCurrentCharacter()
    {
        var currentLetter = Level.Exercise[_currentIndex];
        switch (currentLetter)
        {
            case ' ':
                return "space";
            default:
                return $"{currentLetter}".ToLower();
        }
    }

    private enum CharacterType
    {
        Character,
        Space,
        Delete,
        Nothing
    }

    public class TurnResult
    {
        public int CurrentIndex { get; set; }
        public Level Level { get; set; }
    }
  }
}
```

Encoder.cs

```
using System.Security.Cryptography;
using System.Text;
using HandGestureRecognition.BLL.Interfaces;

namespace HandGestureRecognition.BLL.Services
{
    public class Encoder : IEncoder
    {
        public string Encode(string value)
        {
            byte[] data = Encoding.ASCII.GetBytes(value);
            data = new SHA256Managed().ComputeHash(data);
            return Encoding.ASCII.GetString(data);
        }
    }
}
```

ConfigurationManager.cs

```
using System;
using System.Configuration;
using HandGestureRecognition.BLL.Interfaces;

namespace HandGestureRecognition.BLL.Services
{
    public class ConfigurationManager : IConfigurationManager
    {
        public T GetValue<T>(string id)
        {
            var                                value                                =
System.Configuration.ConfigurationManager.AppSettings[id];

            return (T)Convert.ChangeType(value, typeof(T));
        }

        public T GetValueOrDefault<T>(string id, T defaultValue)
        {
            var                                value                                =
System.Configuration.ConfigurationManager.AppSettings[id];

            return value == null ? defaultValue : (T)Convert.ChangeType(value,
typeof(T));
        }

        public string GetValueOrDefault(string id, string defaultValue)
        {
            var                                value                                =
System.Configuration.ConfigurationManager.AppSettings[id];

            return value ?? defaultValue;
        }

        public string GetValue(string id)
        {
            var                                value                                =
System.Configuration.ConfigurationManager.AppSettings[id];

            return value;
        }

        public string GetConnectionString(string name)
        {
```

```
            var                                value                                =
System.Configuration.ConfigurationManager.ConnectionStrings[name].ConnectionS
tring;

            return value;
        }

        public T GetSection<T>(string name) where T : ConfigurationSection
        {
            return
(T)System.Configuration.ConfigurationManager.GetSection(name);
        }
    }
}
```

AuthorizationService.cs

```
using System.IO;
using HandGestureRecognition.BLL.Interfaces;
using HandGestureRecognition.Model;
using Newtonsoft.Json;

namespace HandGestureRecognition.BLL.Services
{
    public class AuthorizationService : IAuthorizationService
    {
        private readonly IUserService _userService;
        private readonly IConfigurationManager _configurationManager;
        private              string              UserFilePath              =>
_configurationManager.GetValue(Constants.UserFilePath);

        public User User
        {
            get
            {
                try
                {
                    return string.IsNullOrWhiteSpace(UserFilePath)
                        ? null
                        :
JsonConvert.DeserializeObject<User>(File.ReadAllText(UserFilePath));
                }
                catch
                {
                    return null;
                }
            }
        }

        public         AuthorizationService(IUserService         userService,
IConfigurationManager configurationManager)
        {
            _userService = userService;
            _configurationManager = configurationManager;
        }

        public User LogIn(string email, string password)
        {
            var user = _userService.Get(email);

            if (user == null)
            {
```

```
                throw new UserDoesNotExistException($"User with email {email}
does not exist");
            }

            var passwordIsValid = _userService.ValidPassword(email, password);

            if (!passwordIsValid)
            {
                throw new PasswordIsIncorrectException($"Credentials are not
correct");
            }

            CreateUserFile(user);

            return user;
        }

        private void CreateUserFile(User user)
        {
            using (var writer = new StreamWriter(File.Create(UserFilePath)))
            {
                writer.Write(JsonConvert.SerializeObject(user));
            }
        }

        public void LogIn(User user)
        {
            CreateUserFile(user);
        }

        public void Logout()
        {
            File.Delete(UserFilePath);
        }
    }
}
```

app.py

```
import os
import io
import time
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import tensorflow as tf
import pathlib
import socket
import json
import base64
from tensorflow.keras.preprocessing import image as kerim
from PIL import Image
from sklearn.model_selection import train_test_split
from tensorflow.keras import layers
keras = tf.keras
AUTOTUNE = tf.data.experimental.AUTOTUNE

class bcolors:
    HEADER = '\033[95m'
    OKBLUE = '\033[94m'
    OKCYAN = '\033[96m'
    OKGREEN = '\033[92m'
```

```
    WARNING = '\033[93m'
    FAIL = '\033[91m'
    ENDC = '\033[0m'
    BOLD = '\033[1m'
    UNDERLINE = '\033[4m'

#Constants
HOST = '127.0.0.1'
PORT = 44778
IMG_SIZE = 200
BATCH_SIZE = 16
IMG_SHAPE = (IMG_SIZE, IMG_SIZE, 3)
MODEL_FOLDER = "./model"
TEST_EPOCHS = [1,2,3,5,8,13,21]

def recv_basic(clientSocket):
  total_data=[]
  while True:
    data = clientSocket.recv(8192)
    string = data.decode()
    if string.find("}") > -1:
      total_data.append(string)
      break
    total_data.append(string)
  return ''.join(total_data)

def camera_stream(model):
  while True:
    with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as s:
      s.bind((HOST, PORT))
      s.listen()
      print("start to listen on " + HOST + ":" + str(PORT))
      conn, address = s.accept()
      with conn:
        print("accepted connection with: ", address)
        try:
          conn.settimeout(1000)
          while True:
            data = recv_basic(conn)
            image = json_to_image(data)
            array = image_to_array(image)
            predictions = get_prediction(array, model)
            dictionary = dict(zip(class_names, predictions))
            result = json.dumps(dictionary)
            conn.send(result.encode())
        except:
          print("error occured")
          pass

def json_to_image(string):
  frame = json.loads(string)
  imageData = base64.b64decode(frame['Data'])
  image = Image.open(io.BytesIO(imageData))
  return image

def image_to_array(image):
  image = image.convert('RGB')
  image = image.resize((IMG_SIZE, IMG_SIZE), Image.NEAREST)
  image = keras.preprocessing.image.img_to_array(image)
  input_arr = np.array([image])
  return input_arr
```

```
def get_prediction(input_arr, model):
  predictions = model.predict(input_arr)
  score = tf.nn.softmax(predictions[0]).numpy().tolist()
  return score

def format_example(image, label):
  """
  returns an image that is reshaped to IMG_SIZE
  """
  image = tf.cast(image, tf.float32)
  image = (image/127.5) - 1
  image = tf.image.resize(image, (IMG_SIZE, IMG_SIZE))
  return image, label

def setup_base_model():
  print(IMG_SHAPE)
  # Create the base model from the pre-trained model MobileNet V2
  base_model = tf.keras.applications.MobileNetV2(input_shape=IMG_SHAPE,
                                                 include_top=False,
                                                 weights='imagenet')
  base_model.trainable = False
  return base_model

def setup_model(base_model):
  #normalization layer to normalize values in range 0..255 to 0..1
  normalization_layer                                                    =
tf.keras.layers.experimental.preprocessing.Rescaling(1./255)
  #global_average_layer = tf.keras.layers.GlobalAveragePooling2D()
  flatten_layer = keras.layers.Flatten()
  prediction_layer = keras.layers.Dense(29)
  input = keras.Input(shape=IMG_SHAPE)

  model = tf.keras.Sequential([
    input,
    normalization_layer,
    base_model,
    flatten_layer,
    prediction_layer
    ])

  model.summary()
  return model

def train(model, train_ds, val_ds, epochs):
  #Training the Model
  base_learning_rate = 0.0001
  model.compile(optimizer=tf.keras.optimizers.RMSprop(lr=base_learning_rate),

loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True),
              metrics=['accuracy'])
  initial_epochs = epochs #default 3
  validation_steps=20

  loss0,accuracy0 = model.evaluate(val_ds, steps = validation_steps)

  # Now we can train it on our images
  history = model.fit(train_ds,
                      epochs=initial_epochs,
                      validation_data=val_ds)
  acc = history.history['accuracy']
```

```
  print(acc)

def save_model(model):
  model.save(MODEL_FOLDER)

def prepare_dataset(data_dir):
  data_dir = pathlib.Path(data_dir)

  train_ds = tf.keras.preprocessing.image_dataset_from_directory(
    data_dir,
    validation_split=0.2,
    subset="training",
    seed=123,
    image_size=(IMG_SIZE, IMG_SIZE),
    batch_size=BATCH_SIZE)
  val_ds = tf.keras.preprocessing.image_dataset_from_directory(
    data_dir,
    validation_split=0.2,
    subset="validation",
    seed=123,
    image_size=(IMG_SIZE, IMG_SIZE),
    batch_size=BATCH_SIZE)

  global class_names
  class_names = train_ds.class_names

  train_ds = train_ds.prefetch(buffer_size=AUTOTUNE).cache()
  val_ds = val_ds.prefetch(buffer_size=AUTOTUNE).cache()

  return train_ds, val_ds

def validate(model):
  data_dir             =             pathlib.Path("D:/Diploma/Materials/archive
(1)/asl_alphabet_test/asl_alphabet_test/")

  val_ds = tf.keras.preprocessing.image_dataset_from_directory(
    data_dir,
    validation_split=None,
    subset=None,
    seed=123,
    image_size=(IMG_SIZE, IMG_SIZE),
    batch_size=BATCH_SIZE)
  return model.evaluate(val_ds, steps = 20)

def main(test):
  gpus = tf.config.experimental.list_physical_devices('GPU')
  if gpus:
    try:
      for gpu in gpus:
        tf.config.experimental.set_memory_growth(gpu, True)
    except RuntimeError as e:
      print(e)
  print(bcolors.OKGREEN + "PREPARING DATASET" + bcolors.ENDC)
  train_ds,     val_ds     =     prepare_dataset("D:/Diploma/Materials/archive
(1)/asl_alphabet_train/asl_alphabet_train/")
  if(test == False):
    if os.path.isdir(MODEL_FOLDER):
      print(bcolors.OKGREEN + "USING EXISTING MODEL" + bcolors.ENDC)
      model = keras.models.load_model(MODEL_FOLDER)
      model.summary()
    else:
```

```
      print(bcolors.OKGREEN + "FETCHING BASE MODEL" + bcolors.ENDC)
      base_model = setup_base_model()
      print(bcolors.OKGREEN + "CREATING MODEL" + bcolors.ENDC)
      model = setup_model(base_model)
      print(bcolors.OKGREEN + "TRAINING" + bcolors.ENDC)
      train(model, train_ds, val_ds, 3)
      save_model(model)
    camera_stream(model)
  else:
    for epoch_count in TEST_EPOCHS:
      print(bcolors.OKGREEN + "FETCHING BASE MODEL" + bcolors.ENDC)
      base_model = setup_base_model()
      print(bcolors.OKGREEN + "CREATING MODEL" + bcolors.ENDC)
      model = setup_model(base_model)
      print(bcolors.OKGREEN + "TRAINING ON " + str(epoch_count) + " EPOCHS" +
bcolors.ENDC)
      train(model, train_ds, val_ds, epoch_count)
      loss, accuracy = validate(model)
      print(bcolors.OKGREEN + "TRAINING FINISHED ON " + str(epoch_count) + "
EPOCHS. loss = " + str(loss) + " accuracy = " + str(accuracy) + bcolors.ENDC)

main(True)
```