НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

Факультет прикладної математики

Кафедра програмного забезпечення комп'ютерних систем

«До захисту допущено»
Науковий керівник кафедри
_____ І.А. Дичка
«___»_____2019 р.

Дипломний проект

на здобуття ступеня бакалавра

з напряму підготовки 6.050103 «Програмна інженерія»

на тему: «Автоматизована система розпізнавання жестової мови»

Виконав (-ла):
студент (-ка) IV курсу, групи КП-51
Пеня Олександр Романович _____

Керівник:
Доцент кафедри ПЗКС, к.т.н., доцент,
Сулема Є. С. _____

Консультант з нормоконтролю:
Доцент кафедри ПЗКС, к.т.н.,
Онай М.В. _____

Рецензент:
В.О. зав. кафедри ММСА ІПСА, к.т.н., доцент
Тимощук О.Л. _____

Засвідчую, що у цьому дипломному проекті немає запозичень з праць інших авторів без відповідних посилань.
Студент _____

Київ – 2019 року

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Факультет прикладної математики
Кафедра програмного забезпечення комп'ютерних систем

Рівень вищої освіти – перший (бакалаврський)

Напрям підготовки (програма професійного спрямування) – 6.050103 «Програмна інженерія»

ЗАТВЕРДЖУЮ
Науковий керівник кафедри
_____ І.А. Дичка
«___»_____2018 р.

ЗАВДАННЯ

на дипломний проект студенту

Пені Олександру Романовичу

1. Тема проекту «Автоматизована система розпізнавання жестової мови», керівник проекту Сулема Євгенія Станіславівна, к.т.н., доцент, затверджені наказом по університету від «22» травня 2019 р. № 1331-С

2. Термін подання студентом проекту «14» червня 2019 р.

3. Вихідні дані до проекту: див. Технічне завдання.

4. Зміст пояснювальної записки:
   - аналіз технологій розроблення систем комп'ютерного зору;
   - проектування системи розпізнавання жестів;
   - розроблення автоматизованої системи розпізнавання жестів;
   - аналіз отриманих результатів.

5. Перелік обов'язкового графічного матеріалу:
   - діаграма класів (креслення);
   - діаграма діяльності (креслення);
   - структура нейронної мережі (плакат);
   - елементи жестової системи (плакат).

6. Консультанти розділів проекту

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Нормоконтроль	Онай М.В., доцент		

7. Дата видачі завдання «31» жовтня 2018 р.

Календарний план

№ з/п	Назва етапів виконання дипломного проекту	Термін виконання етапів проекту	Примітка
1.	Розроблення та узгодження технічного завдання	21.01.2019	
2.	Дослідження методів розпізнавання образів	11.02.2019	
3.	Визначення засобів розроблення	18.02.2019	
4.	Підготовка матеріалів першого розділу дипломного проекту	21.01.2019	
5.	Підготовка матеріалів другого розділу дипломного проекту	04.02.2019	
6.	Збір та підготовка даних для навчання нейронної мережі	18.02.2019	
7.	Програмна реалізація та тестування системи	27.04.2019	
8.	Підготовка матеріалів третього розділу дипломного проекту	13.05.2019	
9.	Підготовка матеріалів четвертого розділу дипломного проекту	20.05.2019	
10.	Підготовка графічних матеріалів проекту	03.05.2019	
11.	Оформлення документації дипломного проекту	10.06.2019	

Студент О.Р. Пеня

Керівник проекту Є. С. Сулема

АНОТАЦІЯ

Даний дипломний проект присвячено розробленню програмної системи автоматичного розпізнавання жестів.

У ході роботи було проаналізовано методи розпізнавання зображень, розглянуто принципи функціонування, навчання нейронних мереж, сучасні архітектури глибоких високоточних мереж та їх характеристики, побудовано та навчено згорткову нейронну мережу для розпізнавання української дактильної абетки та розроблено програмний застосунок для її застосування.

Робота над проектом здійснювалась відповідно до життєвого циклу програмного забезпечення. Таким чином, було сплановано процес розроблення, виявлено та проаналізовано вимоги до розробленого програмного застосунку, спроектовано архітектуру, визначено найбільш підходящі засоби розроблення, розроблено виділені компоненти, протестовано їх поведінку, проведено інтеграційне тестування, проаналізовано точність, швидкодію та інші атрибути якості системи, створено технічну документацію.

Розроблена система є програмним застосунком, який дозволяє в режимі реального часу обробляти відео-потік, який надходить із локального відео-файлу або з веб-камери, проводить обробку кадрів відео за допомогою нейронної мережі, виявляє елементи жестової мови та виводить їх значення. Програма має віконний графічний інтерфейс, який забезпечує взаємодію з користувачем, засоби захоплення відео-потоку, глибоку згорткову нейронну мережу за допомогою якої відбувається класифікація зображень, локальну вбудовану базу даних значень жестів.

Результати роботи можуть бути використані для розробки ефективних систем розпізнавання та класифікації, наприклад, для розроблення застосунку для перекладу жестової мови або комунікації в нестандартних умовах.

ABSTRACT

This project is devoted to the development of automatic gesture recognition software.

In this work image recognition methods, principles of neural network functioning and training, modern precise networks and their properties were analyzed, a convolutional neural network was built and trained to recognize the Ukrainian sign alphabet and a software application was developed for its usage.

The development was carried out according to the software lifecycle. Thus, the development process was planned, the requirements for the developed software application were identified and analyzed, the architecture was designed, the most appropriate development tools were chosen, defined components were created, their behavior was tested, integration testing was carried out, the system's accuracy, speed and other attributes were analyzed and a technical documentation was created.

The developed system is a software application that allows real-time video stream processing from a local video file or a webcam, processes frames using a neural network, detects elements of the sign language and displays their meanings. The program has a windowed graphical interface that provides user interaction, means of capturing a video stream, a deep convolutional neural network for image classification and a local built-in database of gesture meanings.

The results of the project can be used to develop effective recognition and classification systems, for example, to create an application for translating sign language or communication in non-standard conditions.

АННОТАЦИЯ

Данный дипломный проект посвящен разработке программной системы автоматического распознавания жестов.

В ходе работы были проанализированы методы распознавания изображений, рассмотрены принципы функционирования, обучения нейронных сетей, современные архитектуры глубоких высокоточных сетей и их характеристики, построена и обучена сверточная нейронная сеть для распознавания украинской дактильной азбуки и разработано программное приложение для ее применения.

Работа над проектом осуществлялась в соответствии с жизненным циклом программного обеспечения. Таким образом, был спланирован процесс разработки, выявлены и проанализированы требования к разработанному программному приложению, спроектирована архитектура, определены наиболее подходящие средства разработки, созданы выделенные компоненты, протестировано их поведение, проведено интеграционное тестирование, проанализированы точность, быстродействие и другие атрибуты качества системы, создана техническая документация.

Разработанная система является программным приложением, которое позволяет в режиме реального времени обрабатывать видео-поток, который поступает из локального видео-файла или веб-камеры, проводит обработку кадров видео с помощью нейронной сети, обнаруживает элементы жестового языка и выводит их значения. Программа имеет оконный графический интерфейс, который обеспечивает взаимодействие с пользователем, средства захвата видео-потока, глубокую сверточную нейронную сеть, с помощью которой происходит классификация изображений, локальную встроенную базу данных значений жестов.

Результаты работы могут быть использованы для разработки эффективных систем распознавания и классификации, например, для создания приложения для перевода жестового языка или коммуникации в нестандартных условиях.

ДП.045480-01-90 Автоматизована система розпізнавання жестової мови.
Відомість проекту

Позначення	Найменування	Кіл-ть	Примітка
	Документація проекту		
ДП.045480-02-91	Автоматизована система		
	розпізнавання жестової мови.		
	Технічне завдання	5	
ДП.045480-03-81	Автоматизована система		
	розпізнавання жестової мови.		
	Пояснювальна записка	79	
ДП.045480-04-51	Автоматизована система		
	розпізнавання жестової мови.		
	Програма та методика		
	тестування	13	
ДП.045480-05-34	Автоматизована система		
	розпізнавання жестової мови.		
	Керівництво користувача	7	
ДП.045480-06-99	Автоматизована система		
	розпізнавання жестової мови.		
	Діаграма класів системи.		
	Діаграма класів	1	
ДП.045480-07-99	Автоматизована система		
	розпізнавання жестової мови.		
	Схема роботи системи.		
	Діаграма діяльності	1	

Позначення	Найменування	Кіл-ть	Примітка
ДП.045480-08-98	Автоматизована система		
	розпізнавання жестової мови.		
	Компакт-диск	1	

Факультет прикладної математики

Кафедра програмного забезпечення комп'ютерних систем

"ЗАТВЕРДЖЕНО"

Науковий керівник кафедри

_____ І.А. Дичка

"___" _____ 2018 р.

АВТОМАТИЗОВАНА СИСТЕМА РОЗПІЗНАВАННЯ ЖЕСТОВОЇ МОВИ

Технічне завдання

ДП.045440-02-91

"ПОГОДЖЕНО"

Керівник проекту:

_____ Є.С. Сулема

Нормоконтроль:

_____ М.В. Онай

Виконавець:

_____ О.Р. Пеня

2018

ЗМІСТ

1. НАЙМЕНУВАННЯ ТА ГАЛУЗЬ ЗАСТОСУВАННЯ

Назва розробки: Автоматизована система розпізнавання жестів

Галузь застосування: інформаційні технології.

2. ПІДСТАВА ДЛЯ РОЗРОБЛЕННЯ

Підставою для розроблення є завдання на дипломне проектування, затверджене кафедрою програмного забезпечення комп'ютерних систем Національного технічного університету України «Київський політехнічний інститут імені Ігоря Сікорського» (КПІ ім. Ігоря Сікорського).

3. ПРИЗНАЧЕННЯ РОЗРОБКИ

Розробка призначена для автоматичного розпізнавання жестів у режимі реального часу із відео-потоку з веб-камери пристрою або локального відео файлу.

4. ВИМОГИ ДО ПРОГРАМНОГО ПРОДУКТУ

Програмний за стосунок має виконувати такі основні функції:

1) визначення доступних пристроїв відео введення;
2) отримання відео із веб-камери пристрою або локального відео файлу;
3) вибір жестової системи;
4) визначення жестів обраної жестової системи у режимі реального часу.

Забезпечити виведення значень жестів на екран та можливість зручного редагування.

Додаткові вимоги:

1) можливість розширення функціональності системи без зміни програмного коду;

2) можливість регулювання параметрів системи для покращення роботи;
3) забезпечити достатню точність розпізнавання (імовірність помилки першого роду при розпізнаванні не більше ніж 10%, помилки другого роду при розпізнаванні не більше ніж 15%);
4) наявність довідки у системі.

Розроблення системи виконати на платформі Microsoft Windows, передбачити можливість перенесення на інші платформи, в тому числі і мобільні. Під час розробки використовувати відкриті засоби.

5. ВИМОГИ ДО ПРОЕКТНОЇ ДОКУМЕНТАЦІЇ

У процесі виконання проекту повинна бути розроблена наступна документація:

1) пояснювальна записка;
2) програма та методика тестування;
3) керівництво користувача;
4) креслення:
   - «Автоматизована система розпізнавання жестової мови. Діаграма класів»;
   - «Автоматизована система розпізнавання жестової мови. Діаграма діяльності».

6. ЕТАПИ ПРОЕКТУВАННЯ

Розроблення технічного завдання..... 21.01.2019
Проектування архітектури застосунку..... 04.02.2019
Вибір засобів розроблення..... 18.02.2019
Розроблення та тестування модулів системи..... 27.04.2019
Інтеграція, тестування застосунку..... 20.05.2019
Оформлення технічної документації проекту..... 10.06.2019

7. ПОРЯДОК ТЕСТУВАННЯ РОЗРОБКИ

Тестування розробленого програмного продукту виконується відповідно до “Програми та методики тестування”.

Факультет прикладної математики

Кафедра програмного забезпечення комп'ютерних систем

"ЗАТВЕРДЖЕНО"

Завідувач кафедри

_____ І. А. Дичка

"___" _____ 2019 р.

АВТОМАТИЗОВАНА СИСТЕМА РОЗПІЗНАВАННЯ ЖЕСТОВОЇ МОВИ

Пояснювальна записка

ДП.045480-03-81

"ПОГОДЖЕНО"

Керівник проекту:

_____ Є. С. Сулема

Нормоконтроль:

_____ М. В. Онай

Виконавець:

_____ О. Р. Пеня

2019

ЗМІСТ

СПИСОК ТЕРМІНІВ, СКОРОЧЕНЬ І ПОЗНАЧЕНЬ

Рівень абстракції – узагальнена характеристика рівня приховання деталей реалізації певного набору функціональних можливостей.

Переносимість (англ. portability) – можливість розгортання програмних систем на різних програмно-апаратних платформах.

Парадигма програмування – сукупність ідей та принципів, які визначають стиль написання програм.

Збирання сміття (англ. garbage collection) – механізм автоматичного звільнення ресурсів, коли вони не використовуються.

Фреймворк (англ. framework) – програмна платформа, яка надає частину коду та передбачає розширення стороннім кодом для досягнення бажаних результатів.

Комп'ютерний зір – технологія створення машин, які здатні виявляти, позиціонувати та класифікувати об'єкти за їх зображеннями.

Штучна нейронна мережа – програмна або програмно-апаратна система поєднаних між собою простих обробників сигналів – нейронів.

Навчання нейронної мережі – процес визначення синаптичних ваг у мережі.

API (англ. applied programming interface) – специфікація способів, якими одна програмна система може взаємодіяти з іншою.

GPU (англ graphics processing unit) окремий компонент комп'ютера, який виконує графічний рендеринг.

Гіперпараметр – параметр, оптимальне значення якого потрібно визначити експериментальним шляхом.

Архітектура системи – набір високорівневих елементів системи та зв'язків між ними, які визначають принципи функціонування системи на найвищому рівні абстракції.

СУБД (система управління базою даних) – сукупність програмних засобів, які забезпечують управляння створенням, маніпуляцією та використанням баз даних.

SQL (structured query language) – декларативна мова, призначення для управління створенням та модифікацією структури та даних в реляційних СУБД.

Модуль – функціонально ізольований і певною мірою незалежний програмний компонент.

XML (extensible markup language) – розширювана мова розмітки.

ВСТУП

Розвиток технологій та науковий прогрес сьогодення сягають значних висот і відкривають величезні можливості. Обчислювальна потужність сучасних комп'ютерів, навіть дешевших і менш продуктивних, дозволяє використання методів і технологій, не доступних раніше.

Паралельно з можливостями збільшуються і потреби користувачів сучасних інформаційних систем, які постійно підштовхують прогрес в різних суміжних областях, а особливо в дизайні програмного та апаратного забезпечення. З урахуванням цих потреб формуються поточні передові напрямки розвитку сфери інформаційних технологій, такі як штучний інтелект та нейронні мережі, альтернативні інтерфейси, системи прогнозування та прийняття рішень, які обробляють велику кількість даних у реальному часі, тощо.

Так, проблеми, які вирішують досягнення в цих напрямках, зараз є дуже актуальними в сучасному суспільстві, але їх застосування нових і потужних засобів є складним у багатьох випадках з різних причин: технологічні досягнення ще занадто нові та не набули популярності, їх застосування потребує обчислювально потужного обладнання, , застосування технологій потребує тривалого і складного процесу підготовки для досягнення прийнятного результату, тощо. Тому дослідження та напрацювання в цих областях важливі і актуальні для їх розвитку та поширення.

Дана робота стосується багатьох із цих актуальних напрямків і надає цікавий досвід їх дослідження та застосування в умовах із багатьма обмеженнями, в тому числі з обмеженими можливостями апаратних ресурсів, що заохочує до їх ефективного використання, а її результати можуть бути застосовані в практичних задачах, таких як автоматичний перекладач жестової мови, засіб комунікації в нестандартних умовах або система жестового керування.

1. АНАЛІЗ ТЕХНОЛОГІЙ РОЗРОБЛЕННЯ СИСТЕМ КОМП'ЮТЕРНОГО ЗОРУ

1.1. Обґрунтування актуальності

Значний розвиток обчислювальної техніки та технологічних засобів дозволяє створення принципово нових комп'ютерних систем, які все більшою мірою інтегруються в усі сфери людської діяльності, починаючи з передових розробок для вивчення космосу і закінчуючи звичайними повсякденними справами, і можливості, які зовсім нещодавно вважались суто науково-теоретичними тепер активно застосовуються у передових компаніях по всьому світу.

Технологічний прогрес рухається у напрямку автоматизації, роботизації та розвитку штучного інтелекту, тобто технологій, які радикально змінюють можливості людини і взаємодію з такими системами, яка стає все більш природною, тому виникає потреба у спеціалістах, які володіють сучасними засобами організації взаємодії комп'ютерних систем з людиною (нові, більш природні інтерфейси, такі як розпізнавання та синтез усної мови, рухів людини, засоби віртуальної реальності та BCI і CBI – безпосередній контакт обчислювальних систем із нервовою системою людини), оточуючим світом (засоби комп'ютерного зору та слуху, машинного навчання, аналізу та прийняття рішень, позиціонування та стеження, побудови моделей процесів і прогнозування) та з іншими електронними системами (гетерогенні мережі Інтернету речей та інформаційні системи різного рівня складності).

Ще один впливовий світовий тренд – це надання рівного доступу і можливостей використання технологій усім, незалежно від будь-яких особливостей або вад, особливо фізичних – сучасні технології адаптуються до спеціальних потреб користувачів як у повсякденному житті, так і у роботі та промисловості.

Дана розробка є актуальною, оскільки є дослідженням та прикладним засобом використання штучного інтелекту з широкими можливостями застосування, одним із яких є надання простих можливостей комунікації людям із вадами слуху та/або мовлення, які спілкуються переважно жестовою мовою, при чому така можливість є достатньо гнучкою: можлива побудова як застосунку-перекладача жестової мови на різних, в тому числі і мобільних платформах (метою даної роботи є побудова ядра системи, яке можна розгорнути на іншій платформі та прикладу на конкретній платформі), так і API для людинно-машинної взаємодії, який можна використати в інших системах або розширити його функціональність.

1.2. Порівняльний аналіз мов програмування

1.2.1. Визначення критеріїв та методів порівняння

Вибір засобів і технологій розроблення може сильно впливати як на процес розроблення системи, так і на кінцевий результат проекту, тому необхідний попередній аналіз засобів розроблення та зважений вибір найбільш підходящих технологій для розроблення. Для мови програмування, з використанням якої розроблятиметься система важливі такі характеристики:

- Ефективність доступу до апаратних ресурсів. Система працюватиме з відео-потоком у реальному часі, що створює значне навантаження на обчислювальні системи пристрою, особливо мобільного, тому максимально ефективне використання апаратного забезпечення є важливою характеристикою.
- Рівень абстракції. Мови програмування, які оперують більш високорівневими сутностями дозволяють створення програмного коду, який по-перше, пишеться швидше, по-друге, легше розуміється, а отже легше підтримується і модифікується, по-третє, приховує реалізацію операцій над програмними сутностями за

допомогою абстрактного інтерфейсу, який визначає їх поведінку так, щоб її (реалізацію) можна було змінити чи оптимізувати без впливу на логіку роботи.

- Наявність вільних засобів розроблення. Ресурси проекту обмежені, тому необхідне їх найбільш раціональне використання. Таким чином, наявність бібліотек та засобів, які частково або повністю реалізують необхідну функціональність дозволяє зменшити час розроблення, а відкриті засоби, такі як інтегровані середовища розробки та бібліотеки з відкритим кодом зменшують і витрати на розроблення проекту, і час розроблення за рахунок додаткових зручностей, тому такі засоби є значною перевагою.
- Переносимість. Можливість створення версій системи на різних апаратних платформах без внесення значних змін до коду та втрати ефективності часто є важливою характеристикою сучасних систем.
- Перевага розробника. Хоча вибір мови здійснюється виходячи з необхідних для проекту характеристик, особиста перевага та досвід розробника теж позначається на якості кінцевого результату, тому повністю нехтувати цією характеристикою теж не можна.
- Інтеграція з іншими технологіями. Можливість зручного та ефективного зв’язку програми зі сторонніми компонентами, такими як програмні інтерфейси що використовують інші мові програмування, низькорівневі системні виклики, системи управління базами даних, мережеві сервіси, периферійні пристрої підвищує ефективність розробки та якість кінцевого продукту.

Мови, що розглядатимуться для розроблення оцінюватимуться за умовною шкалою від одного до десяти, де один відповідає найгіршій оцінці, а десять, відповідно, найкращій. Остаточна оцінка є середнє зважене оцінок за всіма критеріями. Вагові коефіцієнти критеріїв відображають вплив критерію на кінцеву оцінку і обрані наступним чином (табл. 1.1).

Таблиця 1.1

Вагові коефіцієнти критеріїв оцінювання

Критерій оцінювання	Ваговий коефіцієнт
Ефективність доступу до апаратних ресурсів	0.45
Рівень абстракції	0.15
Наявність вільних засобів розроблення	0.1
Переносимість	0.15
Перевага розробника	0.05
Інтеграція з іншими технологіями	0.1

1.2.2. Мова Java

Java – це С-подібна об’єктно-орієнтована мова програмування та однойменна платформа розроблення. Вона набула широкого розповсюдження з ряду причин, серед яких простота освоєння та розгортання додатків, середовище виконання (віртуальна машина Java (JVM), яка дозволяє виконувати код на будь-якій апаратній платформі без перекомпіляції, звісно, за умови, що код не використовує засоби певної системи), можливості створення клієнт-серверних застосунків, тощо.

До характеристик Java відносяться:

- Строго об’єктно-орієнтована парадигма програмування. Програми мовою Java обов’язково об’єктно-орієнтовані, що полегшує процес проектування та тестування та створення документації.
- Велика вбудована бібліотека класів. Платформа Java надає широкі можливості розроблення без встановлення додаткових сторонніх засобів.
- Автоматичне керування пам’яттю. Віртуальна машина надає механізм «збирання сміття» – керованого системою звільнення областей динамічної пам’яті, які більше не задіяні програмою.

- Переносимість. Можливість розроблення і тестування на одній платформі та розгортання на іншій – на персональному комп’ютері, мобільному пристрої, або універсальний веб-застосунок, процес розроблення незмінний.
- Вбудована система XML-документування JavaDoc. Коментування коду автоматично перетворюється в стандартизовану технічну характеристику пакету за допомогою утиліти, яка входить до складу платформи Java [1].

Переваги та недоліки: Java є міжнародним стандартом у галузі розроблення веб-застосунків і середовище виконання доступне і навіть попередньо встановлене на багатьох пристроях і дозволяє побудувати зручний для підтримки, проведення та моніторингу процес розроблення промислового рівня. Втім, галузь застосування не співпадає з обраною для даного проекту, механізми, що використовуються у середовищі віртуальної машини сильно сповільнюють виконання коду, а консервативні стандарти позбавляють мову багатьох зручних методів та можливостей, наявних в інших мовах. Зважаючи на особливості мови Java у табл. 1.2 наведено оцінку даної мови.

Таблиця 1.2

Оцінка мови Java

Критерій оцінювання	Оцінка
Ефективність доступу до апаратних ресурсів	4
Рівень абстракції	7
Наявність вільних засобів розроблення	9
Портабельність	8
Перевага розробника	3
Інтеграція з іншими технологіями	7

1.2.3. Мова C#

C# – це проста, сучасна, строго об’єктно-орієнтована мова програмування, яка стандартизується та підтримується корпорацією Microsoft. Також підтримується компонентно-орієнтований підхід – розроблення програмних компонентів в формі автономних пакетів, що реалізують певні функціональні можливості на основі методів, властивостей та подій. Мова надає засоби створення таких компонентів будь-якої функціональності, які містять вбудовані елементи технічної документації.

До особливостей мови відносяться:

- Інтеграція у платформу Microsoft .NET Framework. Дана платформа надає функціональні можливості, схожі на віртуальну машину Java, описану вище, а саме JIT-компіляцію коду у інструкції тої апаратної платформи, на якій виконується розгортання, портативність між усіма пристроями, на яких встановлено .NET відповідної або вищої версії, вбудовану бібліотеку класів, які покривають більшість можливостей взаємодії з операційною системою та її можливостями.
- Сильно оптимізовані методи керування пам’яттю. Середовище виконання CLR, яке здійснює виконання коду, надає garbage collector – механізм автоматичного звільнення динамічної пам’яті, зайнятої знищеними та недоступними об’єктами, який перекладає цю відповідальність з розробника на середовище виконання.
- Обробка виключень – гнучкий метод обробки та реагування на помилки та виняткові ситуації, що стаються як у вбудованих та і самостійно розроблених програмних сутностях, що також підтримує можливості об’єктно-орієнтованого програмування.
- Строга типізація. Стандарти мови не дозволять таких конструкцій, як використання неініціалізованих об’єктів, некоректне приведення типів, вихід за межі масивів, тощо, які часто викликають проблеми та дефекти у програмах на інших мовах.

- Широкі можливості розроблення. Мовою C# можна створювати майже будь-який вид застосунків: віконні програми, веб-застосунки, мобільні та кросплатформні програми з використанням технології .NET Core та багато інших [2].

Переваги та недоліки: Мова C# є універсальним засобом розроблення програмних застосунків з широкими та зручними можливостями, створення як інтерфейсу, так і бізнес-логіки програм. Також наявні безкоштовні засоби розроблення, тестування, версіонування та документування коду. Зважаючи на особливості мови C# у табл. 1.3 наведено оцінку даної мови.

Таблиця 1.3

Оцінка мови C#

Критерій оцінювання	Оцінка
Ефективність доступу до апаратних ресурсів	6
Рівень абстракції	8
Наявність вільних засобів розробленя	8
Портабельність	4
Перевага розробника	9
Інтеграція з іншими технологіями	8

1.2.4. Мова Python

Python – високорівнева мова програмування, побудована на ідеях як об’єктно-орієнтованого, так і функціонального програмування. Завдяки високій абстрактності та високому рівню програмної логіки, що реалізується простими командами ця мова є однією з найпопулярніших у світі.

Особливості Python:

- Динамічна та неявна типізація. Мова Python не потребує попереднього визначення типу об’єкту та допустимих операцій на етапі написання коду – це визначається динамічно під час виконання.

- Широкий спектр доступних сторонніх бібліотек. Підтримувані бібліотеки дозволяють значно розширити можливості мови, надаючи доступ до майже будь-яких за напрямком засобів, від веб-фреймфорків до розрахунку ядерних реакторів.
- Інтерпретована мова програмування. Код на Python не компілюється у виконуваний модуль, а обробляється інтерпретатором безпосередньо під час виконання, тому програму можна запустити будь-де, де наявний інтерпретатор [3].

Переваги та недоліки: Python дозволяє швидко створювати логіку програми з використанням високорівневих абстрактних конструкцій та сторонніх бібліотек, але будучи інтерпретованою мовою, не дозволяє ефективного управління апаратним забезпеченням, високорівневі команди приховують їх реалізацію, тому змінити або налаштувати її дуже складно, скриптові мови передбачені для використання на серверах, а не створення портативних застосунків, підтримка та тестування коду в силу особливостей синтаксису мови є складнішим, ніж з використанням інших розглянутих мов. Оцінку мови Python з урахуванням її особливостей наведено у табл. 1.4.

Таблиця 1.4

Оцінка мови Python

Критерій оцінювання	Оцінка
Ефективність доступу до апаратних ресурсів	7
Рівень абстракції	10
Наявність вільних засобів розроблення	10
Портабельність	7
Перевага розробника	7
Інтеграція з іншими технологіями	9

1.2.5. Мова C++

C++ – логічний наступник C, мова програмування високого рівня з підтримкою різних парадигм програмування: об'єктно-орієнтованої, функціональної та узагальненої, яка набула широкої популярності ще у дев'яностих роках минулого століття.

Особливості мови C++:

- Максимальна оптимізація роботи апаратного забезпечення. C++ базується на C та підтримує бібліотеки C, які мають можливості керування апаратурою, близькі до мов низького рівня. Використання пам'яті та її звільнення повністю покладено на розробника, що дозволяє самостійно і своєчасно керувати задіяними у програмі ресурсами.
- Оптимізація коду. За значний час розвитку стандарту C++ компілятори набули можливості оптимізації створеного коду, що може прискорювати виконання коду до трьохсот разів.
- Значний обсяг сторонніх засобів. Завдяки своїй ефективності мова C++ стала мовою загального призначення, яку величезна спільнота програмістів використовувала для розв'язання типових задач, результатом чого є велика кількість бібліотек з відкритим кодом, що дозволяє їх модифікацію за необхідності, для найрізноманітніших завдань, від з розширень-утиліт до ігрових рушіїв.

Переваги та недоліки: можливості мови дозволяють створювати ефективні програми у будь-який зручний для розробника спосіб (ООП, функціональне програмування), багато типових задач вже розв'язані спільнотою і можна використати або модифікувати ці рішення. Розроблення та забезпечення прийнятного рівня якості є складнішими ніж на інших розглянутих мовах, але і більш гнучкими та контрольованими розробником, на відміну від контрольованості системою. Зважаючи на особливості мови C++ у табл. 1.5 наведено оцінку даної мови.

Таблиця 1.5

Оцінка мови C++

Критерій оцінювання	Оцінка
Ефективність доступу до апаратних ресурсів	10
Рівень абстракції	5
Наявність вільних засобів розроблення	10
Портабельність	5
Перевага розробника	6
Інтеграція з іншими технологіями	6

1.2.6. Вибір мови програмування

Зважаючи на отримані результати аналізу мов програмування та обраний метод підсумкового оцінювання отримаємо остаточні характеристики розглянутих мов програмування (табл. 1.6).

Таблиця 1.6

Підсумкова оцінка мов програмування

Мова	Критерії оцінювання						∑
	Ефективність	Абстрактність	Засоби розроблення	Переносимість	Розробник	Інтеграція	
Java	4,0	7,0	9,0	8,0	3,0	7,0	5,8
C#	6,0	8,0	8,0	4,0	9,0	8,0	6,55
Python	7,0	10,0	10,0	7,0	7,0	9,0	7,95
C++	10,0	5,0	10,0	5,0	6,0	6,0	7,9

Таким чином, маємо, що найефективнішими з близьким результатом є мови C++ та Python. Враховуючи їх майже однаковий результат, обрати можна довільно, тому виходячи з особистої переваги розробника використовуватиметься мова Python.

1.3. Огляд засобів обробки зображення та комп’ютерного зору

1.3.1. Бібліотека OpenCV

Бібліотека алгоритмів комп’ютерного зору, обробки зображень та численних алгоритмів загального призначення з відкритим кодом OpenCV надає неймовірно широкі можливості: отримання зображення з відеопотоку файлу або камери, численні методи обробки зображення та математичних перетворень.

Бібліотека реалізована на C/C++, що дозволяє виконувати складні операції з високою точністю та швидкістю і є дуже популярним засобом у галузі комп’ютерного зору, тому для неї існує велика кількість доповнень, не передбачених стандартом, що надають додаткові можливості, різноманітні обгортки, які дозволяють використання бібліотеки у інших мовах програмування, таких як C# та Python. Також існують вичерпні керівництва та документація, приклади розв’язання численних задач у багатьох сферах застосування комп’ютерного зору: робототехніка, розпізнавання образів, перетворення зображення, тощо [4].

Розробники надають реалізацію бібліотеки на різні платформи, а саме Windows, Android, IOS, MacOS, Linux, що дозволяє використовувати її у різних застосунках.

1.3.2. Бібліотека LTI

Бібліотека LTI – об’єктно-орієнтована бібліотека, що включає часто використовувані у задачах комп’ютерного зору, розпізнавання образів та обробки зображень алгоритми та структури даних. Була розроблена у Аахенському технологічному інституті для ряду дослідницьких проектів і галузях робототехніки, розпізнавання та класифікації зображень, тощо і стала публічним надбанням.

Основна ціль LTI – надання об’єктно-орієнтованої бібліотеки на C++, яка спрощує підтримання та розповсюдження коду, в той же час на-

даючи швидкі алгоритми, які можна використовувати у прикладних програмах.

Бібліотека розроблена для C++ під операційні системи Windows та Linux та надає можливості роботи з лінійною алгеброю для обробки зображень, класифікацією та кластеризацією зображень за допомогою багатьох популярних та ефективних алгоритмів, високоріневою обробкою та корекцією зображення, засобами візуалізації.

Бібліотека розповсюджується за ліцензією GNU LPGL, що дозволяє її безкоштовне використання, модифікацію та розповсюдження [5].

1.3.3. Бібліотека VXL

VXL – набір засобів та бібліотек мовою C++, передбачених для наукових досліджень та реалізації технологій комп'ютерного зору. VXL була написана в ANSI / ISO C ++ і призначена для портативних платформ. Бібліотека складається з декількох основних складових: VNL (числа) – чисельні алгоритми і контейнери, наприклад, матриці, вектори, оптимізатори і т.д., VIL (зображення) – завантаження, збереження і редагування зображень у багатьох найбільш поширених форматах (Також існує можливість роботи з дуже великими зображеннями), VGL (геометрія) – геометрія точок, кривих та інших елементарних об'єктів в одно-, дво- і тривимірному просторах, VSL (вхідний і вихідний потоки), VBL (основні шаблони), VUL (утиліти) – різний функціонал для незалежних платформ. Крім основних бібліотек що входять до складу VXL, є також і додаткові. Додаткові бібліотеки відповідають за такі поняття, як чисельні алгоритми, обробка зображень, системи координат, геометрія камери, стерео, маніпуляції з відео потоком, відновлення структури при русі камери, графічний дизайн, функції відстеження, топологія, класифікатори, 3D-візуалізація і багато іншого. Особливість бібліотеки полягає в тому, що кожен її компонент може використовуватися окремо, не посилаючись на інші компоненти бібліотеки. Таким чином, в додатку можна використовувати тільки ті бібліотеки, які дій-

сно необхідні. VXL використовується по всьому світу в сфері навчання і промисловості, деякі провідні світові експерти в сфері комп'ютерного зору користуються даною бібліотекою. Існує документація по VXL з описом кожного класу і функцій [6].

1.4. Загальні підходи до розпізнавання образів та класифікації

Для розв’язання задач розпізнавання образів використовується ряд методів та алгоритмів. Так, для визначення належності зображення або його частин для до певних класів і прийняття рішень у зв’язку отриманими результатами як правило використовується один із наступних методів класифікації:

- Перебір можливих виглядів об’єкта під різними кутами, масштабами, зміщеннями, тощо. Використання такого методу не є доцільним через складність (практичну неможливість) побудови вичерпної системи зображень для перебору та загальна неефективність підходу у задачах розпізнавання природних образів.
- Математичні методи передбачають виділення певних характеристик зображення, наприклад, контурів, окремих частин, тощо, побудову вектора ознак як елемента певного простору та математичне дослідження отриманого вектора, яке дає можливість співвіднести образ з одним із класів або зробити висновок про відсутність класу, до якого можна віднести визначений образ.
- Побудова штучних нейронних мереж – метод заснований на принципах машинного навчання та побудови моделі збуджень, що викликаються певними характеристиками зображення. Серія збуджень, згрупованих по рівням дозволяє використовуючи результати збуджень останнього рівня отримати імовірнісну класифікацію об’єкта. Такий метод потребує великої кількості прикладів для навчання системи (корекції збуджень) або штучної її

побудови з урахуванням специфіки задачі. Метод навчання потребує часу, проте є загалом простішим та більш застосовним.

1.5. Обґрунтування вибору технологій розроблення

Для розроблення даного проекту будуть використовуватись такі технології та засоби:

1. Мова програмування Python. Python – високорівнева мова, яка надає широкі можливості для створення програмних засобів, великий набір компонентів та розширень як у стандартній бібліотеці так і від спільноти, і використовується в задачах інтелектуального аналізу даних, штучного інтелекту та машинного навчання з високою ефективністю.
2. Бібліотека OpenCV. Вичерпний набір засобів та можливостей, інтеграція з Python та висока ефективність алгоритмів, відкритий код роблять цю бібліотеку корисним засобом отримання та обробки зображення. Також реалізація на різних платформах дозволяє розглянути можливість побудови програми для користувачів різних ОС, а велика кількість прикладів, доповнень та документації дозволить спростити процес освоєння та розроблення.
3. Розпізнавання за допомогою нейронної мережі. Використання нейронних мереж у задачах розпізнавання та класифікації є ефективним і прогресивним засобом, який демонструє високу точність та ефективність, а також загальну гнучкість системи, хоча і потребує більших, порівняно з іншими методами витрат обчислювальних ресурсів, а відкриті засоби та бібліотеки машинного навчання дозволяють побудувати такі системи достатньо швидко і ефективно. Також можна використати попередню обробку зображення, для отримання даних, які простіше і швидше оброблюються нейронною мережею.

1.6. Висновки до розділу 1

У даному розділі було проведено ґрунтовний аналіз наявних засобів та технологій для розроблення системи, розглянуто їх переваги, недоліки та перспективи для використання при розробленні системи і зроблено обґрунтований вибір на користь зазначених засобів і технологій.

В результаті проведення аналізу було обрано наступні технології:

- мова програмування Python;
- бібліотека доступу до відео-введення та попередньої обробки зображення OpenCV;
- класифікація зображень засобами нейронної мережі.

2. ПРОЕКТУВАННЯ СИСТЕМИ РОЗПВЗНАВАННЯ ЖЕСТІВ

2.1. Розроблення та аналіз вимог до розроблюваної системи

Функціональні вимоги до системи впливають із її очікуваної поведінки та варіантів використання, які описують сценарії взаємодії користувача з системою.

2.1.1. Варіанти використання

Система використовується для розпізнавання попередньо визначеного набору жестів (жестової мови) із відео-потоку у режимі реального часу. Відео-потік може бути отримано як із локального файлу, так і з веб-камери пристрою.

При роботі з відео-файлом програма надає інтерфейс вибору з файлової системи, фільтруючи файли на допустиме розширення. При захопленні відео-потоку камери пристрою система має надавати можливість вибору серед наявних камер, якщо їх декілька, наприклад вбудована та підключена USB веб-камера, або з’єднатись із пристроєм відео-введення за замовчанням. При відсутності такого пристрою або доступу до нього – вивести відповідне повідомлення.

Підсистеми розпізнавання жестів (нейронні мережі) мають бути ізольовані від виконуваного коду, і можуть бути замінені на інші аналогічні або можливий вибір серед декількох наявних мереж (для користувача це означає можливість використання декількох режимів роботи програми, наприклад розпізнавання української або англійської жестової мови за вибором). Це дозволяє розширювати можливості застосунку без перекомпіляції, шляхом додання нових конфігураційних файлів з параметрами нейронної мережі.

Жести, що розпізнаються, умовно поділяються на два класи: елементарні та неелементарні лексеми, наприклад, елементарними є дактильні жести (такі, що означають одну літеру), а неелементарними – жести що

означають цілі слова або словосполучення. Значення елементарних жестів зберігається в базі даних і зазвичай не потребує зміни або доповнення в рамках даної жестової системи і при визначенні такого жесту його значення просто виводиться в поле результату. Неелементарні жести є більш складними, тому окрім їх початкового значення користувач може додати свої, які будуть збережені і при повторному входженні буде запропоноване найбільш уживане значення і запропоновано список можливих значень із попередньо збережених.

У процесі розпізнавання елементарні лексеми групуються у складніші конструкції, але без можливості швидкого редагування або зміни значення, бажано забезпечити узгодження між окремими елементами (словами та словосполученнями у реченні), але ця функція не є обов'язковою у початковій версії системи.

Під час розпізнавання отриманий текст залишається на екрані деякий час після отримання останнього елементу, час затримки регулюється користувачем, після вказаної затримки накопичений текст очищується.

Описані способи використання відображені на діаграмі варіантів використання (рис. 2.1).

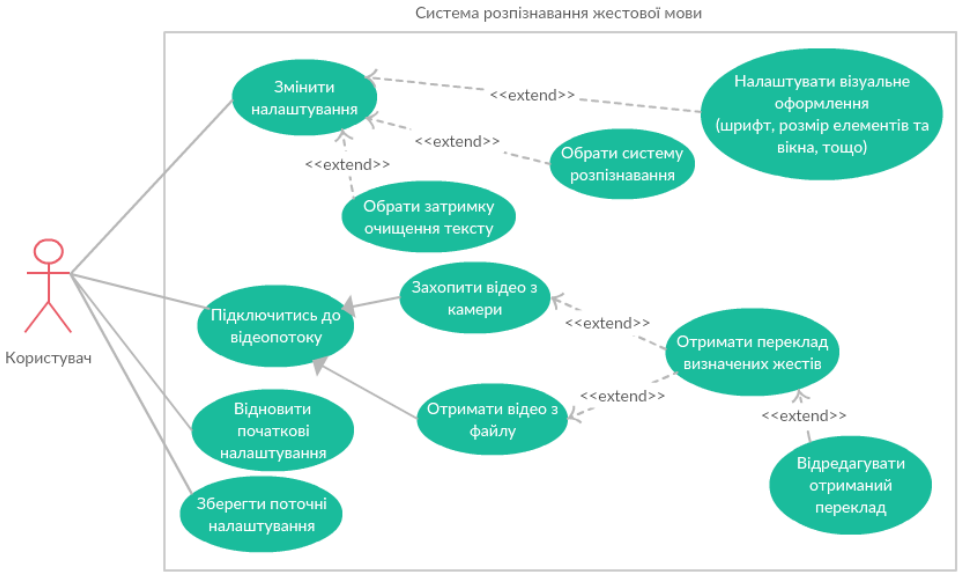


Рис. 2.1. Діаграма варіантів використання

2.1.2. Вимоги до застосунку

Результат збору вимог до функціональності застосунку та проектування і планування процесу розроблення узагальнено у табл. 2.1-2.4.

Функціональні вимоги описують особливості поведінки та функції системи.

Таблиця 2.1

Функціональні вимоги до системи

№ з/п	Вимога	Пріоритет	Трудо-ємність	Опис
1	Отримання відео-потоку з камери пристрою	Критичний	Низька	Можливість використання однієї з камер пристрою як джерела введення для програми.
2	Обробка доступних пристроїв введення відео	Критичний	Низька	За відсутності камер видати відповідне повідомлення, за наявності декількох – можливість вибору між ними, за наявності однієї – підключитись до неї за замовчанням
3	Отримання відео-потоку з файлу	Середній	Низька	Можливість обробки існуючих локальних відео файлів.
4	Автоматичне розпізнавання елементів обраної жестової системи	Критичний	Висока	Основна функція системи полягає в автоматичному визначенні жестів у режимі реального часу
5	Швидке редагування неелементарних лексем	Високий	Середня	Можливість зміни значення жесту, попередньо визначеного в системі та швидкого вибору із списку внесених значень

Продовження табл. 2.1

6	Можливість вибору жестової системи	Середній	Висока	Можливість без перекомпіляції змінювати набір жестів, що розпізнаються шляхом завантаження параметрів нейронної мережі
7	Обрати затримку перед очищенням тексту	Низький	Низька	Дозволяє очистити накопичений текст, якщо довгий час не визначається ніякого введення
8	Налаштування пропуску кадрів	Середній	Низька	Підвищення швидкодії завдяки обробці не усіх кадрів відеопотоку, а з пропуском деяких із них. Можлива незначна втрата точності
9	Налаштування зовнішнього вигляду	Середній	Середня	Можливість налаштувати елементи керування, наприклад розмір вікна, розмір шрифту, повно екранний режим
10	Збереження списку користувацького введення або відновлення значень до початкових	Низький	Середня	Надає користувачу можливість зберегти свої корективи до значень у системі, щоб перенести їх на іншу машину або зберегти копію про всяк випадок або відновити початкові значення.

Нефункціональні вимоги описують вимоги до характеристик системи, які не відображаються безпосередньо у функціональності, наприклад, швидкодія або точність.

Таблиця 2.2

Нефункціональні вимоги до системи

№ з/п	Вимога	Пріоритет	Трудо-ємність	Опис
1	Забезпечити імовірність помилки першого роду при розпізнаванні не більше ніж 10%	Критичний	Висока	Вимоги щодо точності забезпечують можливість використання застосунку як такого, тому є критично важливими у його реалізації
2	Забезпечити імовірність помилки другого роду при розпізнаванні не більше ніж 15%	Критичний	Висока	–
3	Можливість використання у режимі реального часу	Високий	Середня	Для зручного користування застосунок має бути достатньо продуктивним, щоб працювати з потоковим відео і давати достатньо точні результати. На машинах низької потужності для підвищення швидкодії дозволяється пропускати частину кадрів
4	Робота системи обмежена локальною машиною, для роботи та збереження даних не потрібен доступ до мережі	Високий	Низька	Забезпечити можливість автономної роботи застосунку без необхідності використання додаткових серверів, як на машині користувача, так і у мережі

Вимоги до розроблення описують специфічні умови, які необхідно витримати розробнику.

Таблиця 2.3

Вимоги до розроблення та проектування

№ з/п	Вимога	Пріоритет	Трудо-ємність	Опис
1	Використання XML-документування для створення актуальної і вичерпної документації коду	Середній	Низька	Дозволяє автоматично генерувати та оновлювати документацію API
2	Розмежування модулів застосунку і їх взаємодія через абстрактні інтерфейси	Високий	Середня	Вимагається розділення абстракції і реалізації модулів таким чином, щоб заміна реалізації викликала якомога менше змін в загальній структурі програми
3	Використання машинно-незалежних компонентів	Середній	Висока	Потребується можливість компіляції застосунку для різних платформ, у тому числі і мобільних, тому необхідне використання бібліотек та засобів, доступних на усіх платформах, а машинно-залежні частини, наприклад система введення, інкапсулювати абстрактним інтерфейсом

Програмно-апаратні вимоги описують вимоги до середовища, у якому виконується система

Таблиця 2.4

Вимоги до середовища виконання та апаратного забезпечення

№ з/п	Вимога	Опис
1	Операційна система Windows NT	Вимоги до апаратного забезпечення формулюються виходячи з початкової версії системи на платформі Windows NT. При переході застосунку на інші середовища виконання відповідні вимоги до апаратної частини будуть сформульовані
2	Оперативна пам’ять 2 Гб	
3	Частота процесора 2 ГГц	
4	Вільний дисковий простір 200 Мб	
5	Бажана наявність графічного процесора для підвищення швидкодії	

2.2. Побудова та навчання нейронних мереж

2.2.1. Принципи функціонування штучних нейронних мереж

Штучні нейронні мережі були розроблені як математична модель біологічних нейронних мереж, які зустрічаються у нервовій системі живих організмів для використання у задачах, для яких не вдається побудувати детермінований алгоритм розв’язання, або об’єм даних для нього занадто великий, наприклад, в задачах прогнозування та прийняття рішень, класифікації, адаптивного керування, розпізнавання.

Штучна нейронна мережа представляє собою множину пов’язаних синапсами штучних нейронів – простих програмних або апаратних сигнальних процесорів. Таким чином, нейрон приймає множину вхідних даних – сигналів – виконує відносно просте обчислення і видає вихідний сигнал.

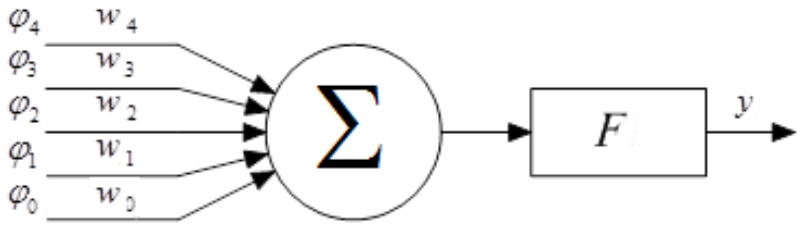


Рис. 2.2. Структура штучного нейрона

На рис. 2.2 зображено загальну структуру нейрона, де φ_i – вхідні сигнали, φ_0 – зміщення (необов'язковий елемент, який завжди видає однаковий сигнал, як правило одиницю), w_i – синаптична вага і-того синапсу, коефіцієнт, який визначає, наскільки сильно даний сигнал впливає на вихідне значення нейрона, $\sum$ – суматор нейрона, який обчислює скалярний добуток векторів вхідних значень і ваг синапсів, отримуючи проміжне значення, яке передається в функцію активації F, яка обчислює вихідне значення нейрона y, яке іноді називається збудженням. Зазвичай нейрони працюють з нормованими значеннями в певному діапазоні, наприклад [0; 1], [-1; 1], [0, ∞) або в іншому визначеному діапазоні. Для отримання таких значень використовується функція активації, яка обирається так, щоб вихідні значення відповідали заданому діапазону значень. Найчастіше використовуються такі: лінійна $F(x) = x$, яка використовується тільки коли потрібно передати дані без змін або протестувати мережу; сигмоїда (2.1), яка видає збудження в діапазоні [0; 1]; гіперболічний тангенс (2.2), якщо необхідно отримувати також від'ємні значення; випрямляч (2.3) або його неперервне наближення (2.4).

$$F(x) = \frac{1}{1+e^{-x}} \tag{2.1}$$

$$F(x) = \frac{e^{2x}-1}{e^{2x}+1} \tag{2.2}$$

$$F(x) = \max(0; x) \tag{2.3}$$

$$F(x) = \log\left(1 + e^{x}\right) \tag{2.4}$$

Нейронні мережі складаються із великої кількості нейронів, які умовно поєднуються у шари. Так, у кожної мережі є вхідний шар, нейрони якого не виконують обробку даних, а лише приймають і розподіляють вхідні сигнали, вихідний шар, який видає результат роботи нейронної мережі, та внутрішні шари (також називаються прихованими), які виконують більшу частину обробки. На рис. 2.3 нейрони вхідного шару позначено синім, внутрішнього – червоним, вихідного – зеленим кольором. Залежно від зв'язків між шарами нейронні мережі поділяються на послідовні, в яких сигнали передаються в одному напрямку та рекурентні, в яких сигнали також повторно передаються на попередні шари.

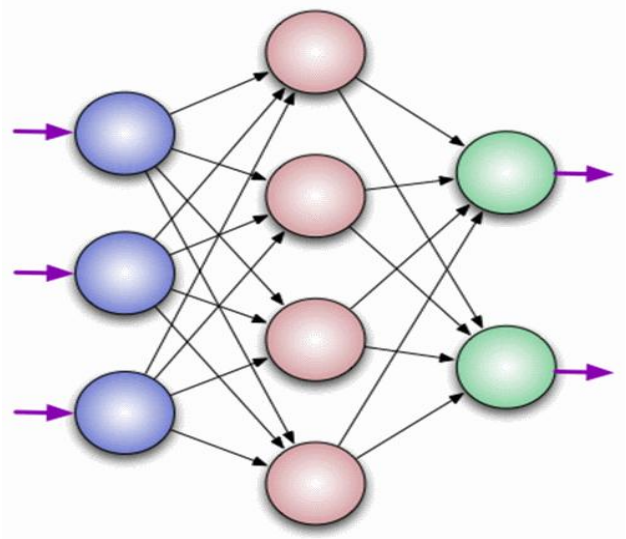


Рис. 2.3. Схема простої послідовної нейронної мережі

Як зазначалось раніше, нейрони пов'язані між собою синапсами – зв'язками, які характеризуються синаптичною вагою – мірою значимості сигналу, що надходить за даним синапсом. Сам же нейрон обчислює середнє зважене сигналів, що надходять до нього [7].

Поведінка мережі визначається її топологією та синаптичними вагами, оскільки нейрони завжди виконують одне і те саме обчислення. Тому

для вирішення практичних задач необхідно обрати топологію мережі та отримати необхідні синаптичні ваги – навчити мережу.

2.2.2. Аналіз загальних підходів побудови та навчання нейронних мереж

Для отримання бажаного результату обробки даних необхідно обрати модель мережі та визначити шляхом навчання ваги синапсів мережі.

Модель мережі – це її структура, яка описує кількість шарів, кількість нейронів у кожному шарі, синапси, функцію активації нейронів, наявність або відсутність нейронів зміщення. Модель мережі обирається окремо для кожної задачі з урахуванням самої задачі, вхідних даних та очікуваних результатів і всі її характеристики є гіперпараметрами, тобто величинами, які підбираються експериментальним шляхом. Прийняття рішення про модель мережі здійснюється різними методами. Для деяких задач вже знайдено оптимальні конфігурації, отримання бажаної моделі експериментальним шляхом зазвичай проводиться або з використанням спрощення мережі, або з використанням нарощування. Спрощення передбачає створення мережі, яка очевидно перевищує достатні для розв’язання показники і після навчання визначають надлишкові елементи, без яких можна досягти бажаного результату з отриманою спрощеною мережею. Нарощування навпаки передбачає створення простої структури, недостатньої для розв’язання задачі і на кожному кроці додають або по одному нейрону на кожен шар або ще один шар і повторюють навчання доки мережа не досягне необхідної точності або не стане неприпустимо великою з точки зору обчислювальної складності. Втім процес навчання різних моделей займає багато часу, обчислювальної потужності і потребує певних надлишкових зусиль на побудову моделей, що гірше підходять для даної задачі, щоб впевнитись, що оптимальний результат було досягнуто і навіть при цьому проведені експерименти не гарантують оптимальності вибору у

загальному сенсі. Тому за наявності існуючих конфігурацій мережі і проведених досліджень краще використати їх результати.

Так, для задач розпізнавання образів та класифікації найбільш ефективним типом мереж визнано згорткові нейронні мережі, які є прогресивним напрямком розвитку. Окрім вже перевірених структур мереж, постійно з'являються нові види загорткових мереж, які демонструють кращі результати за точністю або швидкістю роботи. На відміну від повнозв'язних мереж, у яких кожен нейрон пов'язаний з усіма нейронами наступного шару, згорткові мережі мають ряд особливостей структури, які дозволяють обробляти великі масиви даних, такі як зображення або звук. Згорткові мережі складаються із згорткових і агрегувальних шарів, що чередуються.

Згорткові шари виконують операцію згортки (операція згортки полягає в по членному множенні невеликої частини вхідного зображення та фільтр, тобто матрицю такого ж розміру, і обчисленні суми отриманого добутку), пересуваючи фільтр по зображенню з невеликим (меншим за розмірність фільтра) кроком. Саме така структура дозволяє зменшити кількість параметрів у порівнянні з повнозв'язною мережею, оскільки нейрони поєднуються не з усіма нейронами наступного шару, а тільки з тими, які були задіяні під час виконання згортки. Операцію згортки пов'язують з виділенням певних елементарних ознак на вхідному зображенні, таких, наприклад, як прямі лінії, дуги, чи межа контрасту на зображенні під певним кутом, при чому що далі за шарами просувається сигнал, то більш абстрактні ознаки виділяють згорткові шари, тому результати роботи згорткових шарів називають картами ознак, які виявляють наявність та положення певної ознаки.

Агрегувальні шари використовуються для того, щоб зменшити розмірність карт ознак за рахунок ущільнення. У задачах класифікації факт наявності ознаки є більш важливим, ніж визначення його точних координат. Найбільш часто застосовується метод максимуму: карти ознак діляться на фрагменти, як правило квадрати 2×2 пікселя, які не перетинаються у

вихідний результат потрапляє тільки найбільше значення пікселя, як показник наявності або відсутності ознаки в даному фрагменті (рис. 2.4). Також відкидання менш важливих частин зображення дозволяє зменшити вірогідність перенавчання (overfitting) мережі.

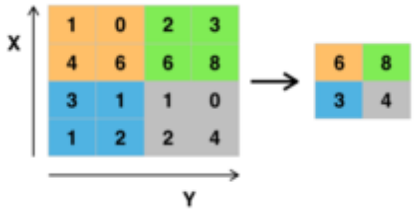


Рис. 2.4. Приклад ущільнення агрегувальним шаром з використання максимуму

Після проходження декількох етапів згортки за агрегації виділяються абстрактні високорівневі ознаки, які обробляються класифікатором, як правило декілька шарів повнозв'язної мережі, яка на виході визначає приналежність груп ознак до певного класу [8] (рис. 2.5).

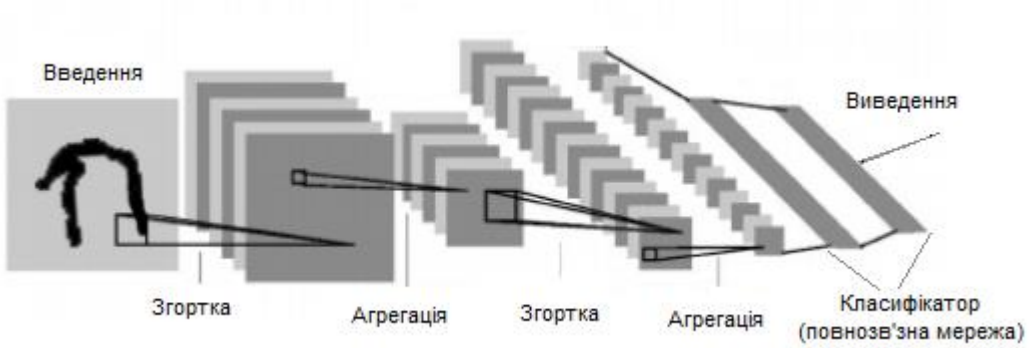


Рис. 2.5. Типова структура згорткової мережі

Навчання мережі – це процес визначення підходящих синаптичних ваг у мережі (у згорткових мережах під час навчання також визначаються коефіцієнти фільтрів, які застосовуються в операціях згортки). Існує декілька розповсюджених методів навчання, серед яких широко використову-

ються метод зворотного розповсюдження, метод пружного розповсюдження, генетичний алгоритм.

Метод зворотного розповсюдження застосовується досить часто і є основою багатьох інших методів навчання мережі. Він ґрунтується на послідовній зворотній (в напрямку від вихідних нейронів до вхідних) зміні синаптичних ваг залежно від отриманих вхідних даних. У даному методі розглядається функція залежності похибки мережі від ваги кожного синапса і вага, за якої досягається мінімум цієї функції є шуканою вагою. Пошук мінімуму виконується методом градієнтного спуску, звідки випливають його основні недоліки: можливість помилки визначення оптимальної ваги через знайдений локальний мінімум замість глобального (щоб подолати цей недолік використовують додаткові параметри: момент, який дозволяє оминути локальний мінімум, але так само можна оминути і глобальний мінімум, що зробить процес пошуку значно довшим, і швидкість навчання, яка задає елементарну зміну ваги на кожному кроці), довгий процес навчання (саме навчання займає багато часу, особливо на пристроях з невеликим обчислювальним ресурсом і відсутності високопаралельних можливостей, таких як GPU або спеціальні апаратні прискорювачі, момент і швидкість навчання є гіперпараметрмами, тобто, щоб знайти їх найбільш підходящі їх значення потрібно виконати навчання декілька разів з різними значеннями параметрів).

Метод пружного розповсюдження є модифікацією попереднього методу і використовує інші правила для коригування ваг таким чином, щоб уникнути використання гіперпараметрів (вони все одно існують у вигляді величин, на які збільшується чи зменшується вага при коригуванні, але для них встановлені рекомендовані значення за замовчанням). Так, цей алгоритм має ряд переваг, серед яких швидша збіжність мережі (величина, на яку змінюється вага збільшується кожного разу, коли шуканий мінімум помилки не знайдено, забезпечуючи швидший пошук необхідного значення), можливе досягнення більшої точності (значення синаптичних ваг ко-

ригується тільки після проходження епохи, тобто усіх тестових прикладів даних, на яких навчається мережа, а не одного прикладу, як у попередньому методі, завдяки чому швидше і, можливо, точніше досягається узгодженість між прикладами), як правило не потребується виконання навчання декілька разів для визначення гіперпараметрів навчання мережі [9].

Генетичний алгоритм використовує принципово інший метод визначення синаптичних ваг, також націлений на швидше отримання результату. Генетичний алгоритм використовує послідовність дій, схожу на природний відбір найбільш пристосованих особин, звідки і назва. На початку алгоритму створюється початкова популяція – множина мереж із випадково ініціалізованими синаптичними вагами. Для кожної з них обчислюється помилка, при чому чим менша помилка у варіанта мережі, тим імовірніше він буде використаний у подальшому. Після цього за встановленими правилами відбувається створення нового покоління із наявної популяції за допомогою схрещування, внесення випадкових змін в популяції (“мутація”). Після цього відбувається скорочення популяції до її початкового розміру шляхом відкидання варіантів, у яких помилка більша за інші. Таким чином, популяція “еволюціонує”, зберігаючи частини найбільш підходящих варіантів і відкидаючи помилкові. За такою схемою алгоритм працює доки не буде отримано варіант із бажаною точністю або не вичерпається відведена кількість ітерацій. Такий алгоритм може функціонувати значно швидше ніж метод зворотного розповсюдження помилки та його модифікації і охоплювати широкий простір пошуку, завдяки випадковим змінам варіантів під час схрещування та мутацій, завдяки чому він уникає проблеми локального мінімуму інших методів, але ефективність алгоритму залежить від великої кількості параметрів: розмір початкової популяції, коефіцієнти схрещування та мутації, методи схрещування, мутації, обчислення помилки, допустима кількість ітерацій. Також алгоритм опирається на випадкові величини. Таким чином, генетичний алгоритм навчання нейронних мереж є імовірнісним – він не гарантує збіжність мережі і може безус-

пішно продовжуватись нескінченно, але якщо збіжність досягається, то цей процес буде швидшим, ніж у методах розповсюдження помилки. Тому вводиться обмеження на кількість ітерацій і бажану точність результату і якщо алгоритм не збігається за задану кількість кроків, то слід розпочати спочатку з новою популяцією, можливо змінивши деякі параметри методу [10].

Також на навчання мережі сильно впливають вхідні дані, на яких навчається мережа. Якщо даних недостатньо, або вони занадто подібні між собою, то можна зіткнутись з проблемою надлишкового навчання (перенавчання, overfitting), яка появляється в тому, що мережа адаптується під конкретний набір даних і після навчання не реагуватиме на інші, навіть дуже подібні дані коректно. Навпаки, якщо вхідних даних забагато, то топологія мережі, яка необхідна, щоб забезпечити збіжність алгоритму навчання (незалежно від обраного методу навчання), може виявитись занадто складною в плані обчислювальних ресурсів і часу на обробку. Але набір вхідних даних також є гіперпараметром навчання мережі і визначення підходящого набору вхідних даних для навчання також є складною задачею, проте певні рекомендації є мало не очевидними передумовами навчання: наявність достатньої кількості прикладів для кожного класу (від ста до тисячі прикладів), приклади мають містити елементи, що розпізнаються, під різними кутами, у різних масштабах, кольорах (якщо це властиво об’єкту), з різним рівнем освітленості, зашумлення, різним фоном, частково перекриті іншими об’єктами. Так мережа може більш правильно визначити ознаки, що характеризують об’єкт.

2.2.3. Оцінка ефективності та вибір структури мережі

Для розв’язання поставлених задач використаємо згорткову нейронну мережу, як найкращий загальний варіант розв’язання поставленої задачі. Процес роботи мережі має бути достатньо швидким для обробки відеопотоку у реальному часі (при обробці відео частотою тридцять кадрів на

секунду, швидкість обробки зображення має бути близько 30 мс, проте враховуючи предметну область час може бути і значно більшим, аж до 2 секунд). Також необхідне забезпечення точності розпізнавання на рівні не меншому, ніж 75-80%. Враховуючи ресурси проекту і можливості розвитку потрібно забезпечити виконання обчислень на центральному процесорі.

Із state-of-the-art глибоких нейронних мереж, що використовуються для класифікації зображень, можна виділити такі архітектури та їх модифікації, які мають різні варіанти реалізації:

- VGGNet.
- ResNet.
- Faster-RCNN.
- MobileNet.
- Google Inception.

Всі варіанти ефективні у задачах розпізнавання, що підтверджується результатами ImageNet visual recognition challenge, міжнародного відкритого конкурсу розпізнавання зображень, для якого представляється набір вхідних даних із десяти тисяч зображень, що належать до тисячі класів. Втім, за результатами досліджень дані структури є певним компромісом між точністю (і, відповідно, обчислювальною складністю) та швидкістю. Так, структури, що забезпечують високу точність потребують сильно розподілених систем для навчання та обчислення, як правило для таких мереж розробляються спеціальні апаратні прискорювачі, а навчання і/або використання проводиться на кластерах комп’ютерів. В той же час, мережа MobileNet була розроблена спеціально для використання на пристроях з меншими обчислювальними можливостями і порівняно невеликим розміром вхідного зображення за рахунок точності [11]. Тому, враховуючи конструктивні особливості та специфіку проекту, для розв’язання задачі класифікації використовується мережа MobileNet, яка має таку структуру [12], описану в табл. 2.6.

Таблиця 2.6

Структура мережі MobileNet v1

Тип шару	Розмір фільтра	Вхідне зображення
Згортка з кроком 2	3 × 3 × 3 × 32	320 × 240 × 3
Згортка в глибину з кроком 1	3 × 3 × 32	160 × 120 × 32
Згортка з кроком 1	1 × 1 × 32 × 64	160 × 120 × 32
Згортка в глибину з кроком 2	3 × 3 × 64	160 × 120 × 64
Згортка з кроком 1	1 × 1 × 64 × 128	80 × 60 × 64
Згортка в глибину з кроком 1	3 × 3 × 128	80 × 60 × 128
Згортка з кроком 1	1 × 1 × 128× 128	80 × 60 × 128
Згортка в глибину з кроком 2	3 × 3 × 128	80 × 60 × 128
Згортка з кроком 1	1 × 1 × 128× 256	40 × 30 × 128
Згортка в глибину з кроком 1	3 × 3 × 256	40 × 30 × 256
Згортка з кроком 1	1 × 1 × 256× 256	40 × 30 × 256
Згортка в глибину з кроком 2	3 × 3 × 256	40 × 30 × 256
Згортка з кроком 1	1 × 1 × 256× 512	20 × 15 × 256
Згортка в глибину з кроком 1	3 × 3 × 512	20 × 15 × 512
Згортка з кроком 1	1 × 1 × 512× 512	20 × 15 × 512
Згортка в глибину з кроком 1	3 × 3 × 512	20 × 15 × 512
Згортка з кроком 1	1 × 1 × 512× 512	20 × 15 × 512
Згортка в глибину з кроком 1	3 × 3 × 512	20 × 15 × 512
Згортка з кроком 1	1 × 1 × 512× 512	20 × 15 × 512
Згортка в глибину з кроком 1	3 × 3 × 512	20 × 15 × 512
Згортка з кроком 1	1 × 1 × 512× 512	20 × 15 × 512
Згортка в глибину з кроком 1	3 × 3 × 512	20 × 15 × 512

Продовження табл. 2.6

Згортка з кроком 1	1 × 1 × 512× 512	20 × 15 × 512
Згортка в глибину з кроком 2	3 × 3 × 512	10 × 7 × 512
Згортка з кроком 1	1 × 1 × 512× 1024	10 × 7 × 512
Згортка в глибину з кроком 2	3 × 3 × 1024	10 × 7 × 1024
Згортка з кроком 1	1 × 1 × 1024× 1024	10 × 7 × 1024

В якості класифікатора використаємо SSD (single shot detector), блоки визначення опрацьовують результати виділених шарів, визначаючи в блоках різної просторової величини імовірність наявності та положення об’єктів.

2.3. Висновки до розділу 2

Даний розділ є описом проекту системи, який визначає визначає поведінку системи через специфікацію вимог: як функціональних, що визначають принципи роботи системи, так і нефункціональних, які описують бажані характеристики та середовище роботи системи, виділених із варіантів використання системи, що забезпечує виконання поставлених у рамках даної роботи задач і враховує можливі перспективи розвитку системи у комерціалізований програмний засіб або проект по його створенню. Також здійснено аналіз головного модуля системи, що розробляється – нейронної мережі, яка здійснює розпізнавання жестів із кадрів вхідного потоку: визначено принципи дії, які зумовлюють використання саме нейронної мережі для задачі розпізнавання і дозволяють забезпечити гнучкість як роботи системи, так і самої системи, розглянуто сучасні архітектури глибоких згорткових нейронних мереж, які із значним успіхом використовуються для розпізнавання об’єктів, проведено порівняння їх обчислювальної складності під час навчання та роботи і швидкості класифікації, обрано конфі-

гурацію MobileNet-SSD як таку, що найбільшою мірою задовольняє вимоги проектів до швидкодії та обчислювальних ресурсів.

3. РОЗРОБЛЕННЯ АВТОМАТИЗОВАНОЇ СИСТЕМИ РОЗПІЗНАВАННЯ ЖЕСТІВ

3.1. Архітектура застосунку

Архітектура програмної системи – це базова високорівнева організація системи, що визначає її основні компоненти та способи взаємодії між ними. Архітектура визначає загальні абстрактні елементи, що забезпечують функціонування структурних блоків системи, і має безпосередній вплив на засоби і принципи розроблення, оскільки обумовлює конкретні проектні рішення для дотримання спроектованих архітектурних конструкцій.

Правильно обрана архітектура системи дозволяє організувати процес розроблення та супроводу системи значно легше, особливо коли доводиться працювати з регулярною модифікацією програмних модулів завдяки ізоляції компонентів та виділення абстрактних зв’язків між ними, що дозволяє змінювати та розширювати можливості програми локально, не змінюючи інших її частин. Такий підхід дуже важливий для сучасного програмного забезпечення, тому було виділено певні шаблони – перевірені ефективні рішення питання архітектури системи, які мають певні передумови застосування, переваги та недоліки:

- багаторівнева архітектура;
- мікросервісна архітектура;
- мікроядерна архітектура;
- керована подіями архітектура;
- інші підходи [13, 14].

Система розробляється з використанням шаблону багаторівневої архітектури, який передбачає розділення функціональності системи на окремі незалежні блоки – рівні, які взаємодіють ніж собою через визначені програмні інтерфейси і можуть бути замінені на іншу реалізацію, при цьому ніяк не впливаючи на інші рівні, що дозволяє створити більш гнучку

та простішу у супроводі, модифікації та тестуванні систему. Використовується поєднання об’єктно-орієнтованого та процедурного принципів програмування.

Таким чином, система складається з 4 рівнів, зображених на рис. 3.1.

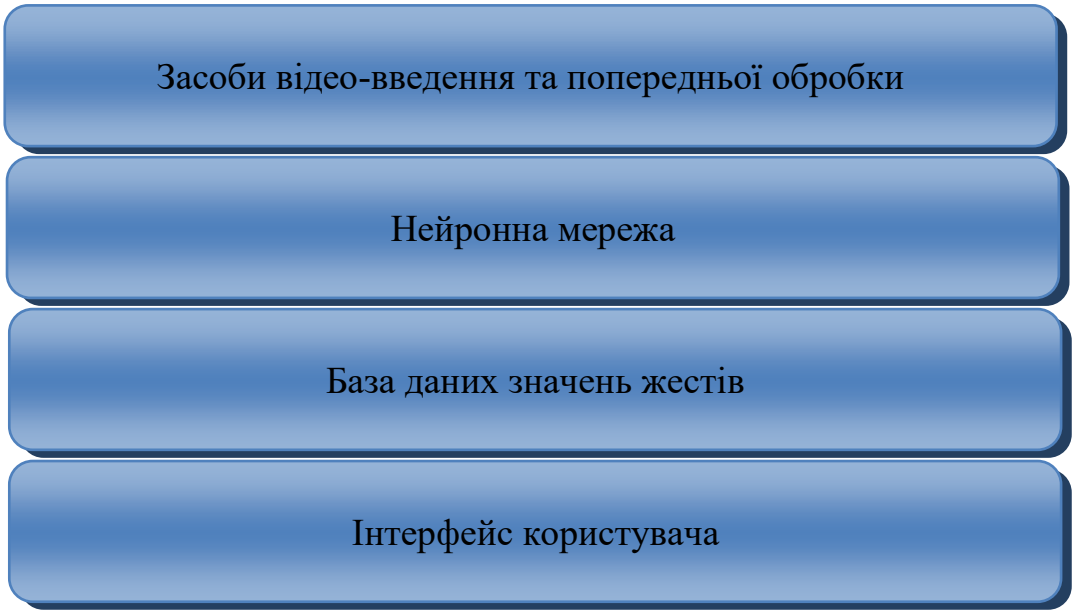


Рис. 3.1. Рівні архітектура системи

Засоби отримання доступу до відео-потоку складають машино-залежний компонент, який реалізовується по-різному в залежності від конкретної платформи. У реалізації початкової версії системи для Windows використовується відкрита бібліотека комп’ютерного зору OpenCV, яка приховує особливості роботи з пристроями введення відео і так само легко дозволяє отримати відео-потік із файлу. З її допомогою також виконується попередня обробка зображення, підвищення чіткості та контрастності, фільтрація шумів. Задача цього рівня – інкапсуляція взаємодії з операційною системою для отримання відео-потоку і передача послідовності кадрів на наступний рівень для подальшої обробки.

Основне робоче навантаження на систему припадає на рівень нейронної мережі, яка виконує розпізнавання жестів із кадрів відео-потоку. При фіксованому програмному інтерфейсі можна досягти гнучкості систе-

ми, завантажуючи різні моделі нейронних мереж, попередньо навчених на різних вхідних даних, щоб система була легко розширюваною іншими жестовими системами. Приймаючи на вхід кадр відео із попереднього рівня, мережа проводить розпізнавання відповідно до сформованих під час навчання для даної жестової системи параметрів та видає клас, до якого найбільш імовірно відноситься даний жест або виносить висновок про відсутність на кадрі відомих жестів. Для побудови нейронної мережі використовується бібліотека Tensorflow – відкрита бібліотека машинного навчання від Google.

Локальна база даних зберігає початкові значення елементів жестової системи, їх тип, інші значення, додані користувачем та повертає усі наявні значення жесту відповідно до його класу відповідно до частоти їх використання (значення, яке використовується найчастіше буде значенням за замовчанням для цього класу). Враховуючи особливості роботи системи та невеликий розмір, необхідний для такої бази даних варто використати вбудовану систему управління базами даних SQLite, яка не потребує окремого процесу-сервера і надає можливості створення та використання реляційних баз даних та доступом і керуванням за допомогою SQL.

Інтерфейс користувача представляє собою вікно, у якому відображається відео, що обробляється та значення розпізнаних жестів у текстовому вигляді. Призначення інтерфейсу – здебільшого виведення даних, отриманих системою, введення даних від користувача відбувається досить нечасто, при редагуванні результатів перекладу та настроюванні параметрів системи. Інтерфейс системи достатньо простий та зрозумілий і будується за допомогою засобів бібліотеки wxPython – програмної обгортки над кросплатформним API графічного віконного інтерфейсу.

3.2. Модуль нейронної мережі

Нейронна мережа виконує основну роботу системи – розпізнавання жестів із кадрів відео-потоку.

Машинне навчання та нейронні мережі є актуальним напрямком різноманітних досліджень та розробок, що постійно розвивається, досягаючи високої ефективності у точності та швидкодії. Звідси велика кількість проектів і засобів, присвячених нейронним мережам. Порівняння потужних та універсальних засобів наводиться у [15]. Важливими для проекту є такі характеристики засобів побудови та навчання нейронних мереж, як відкритий доступ, кросплатформність та можливість використання на мобільних платформах, підтримка згорткових мереж, вичерпна документація та успішне застосування, перевагою є наявність інтерфейсу для мови Python. Таким чином, підходящими засобами для даного проекту є бібліотеки Tensorflow та Caffe. Враховуючи підтримку спільноти, постійний розвиток, підтримку різноманітних апаратних засобів та наявність проектів з використанням Tensorflow, у проекті застосовано саме цей засіб.

Використання нейронних мереж є обчислювально затратним, тому необхідно обрати структуру мережі, яка буде ефективною в обмежених апаратних умовах проекту, а саме достатньо швидкою, щоб працювати з відео в реальному часі та забезпечувати достатню точність, визначену вимогами. Вибір структури мережі проводився раніше і було обрано архітектуру Mobilenet-SSD. Процес навчання мережі має певні особливості. Для навчання і перевірки мережі було зібрано вхідні дані із зображеннями жестів української дактильної абетки у кількості в середньому близько двоста зображень на клас із різним фоном, масштабом, поворотом, нахилом та освітленням у форматі RGB і розміром 320×240 пікселів. Для навчання мережі необхідно сформувати спеціальну структуру, яку використовує Tensorflow, tfrecords. Для цього потрібно промаркувати зібрані зображення, тобто вказати, де на зображенні знаходиться об’єкт, які його розміри та до якого класу він відноситься, що зручно зробити за допомогою утиліти LabelImg (рис. 3.2), яка зберігає маркування в форматі XML.

Після цього невелику частину зображень (5-10 %) з відповідним маркуванням потрібно виділити як тестові дані, які використовуються для

оцінки точності мережі, інші зображення будуть використані для навчання мережі.

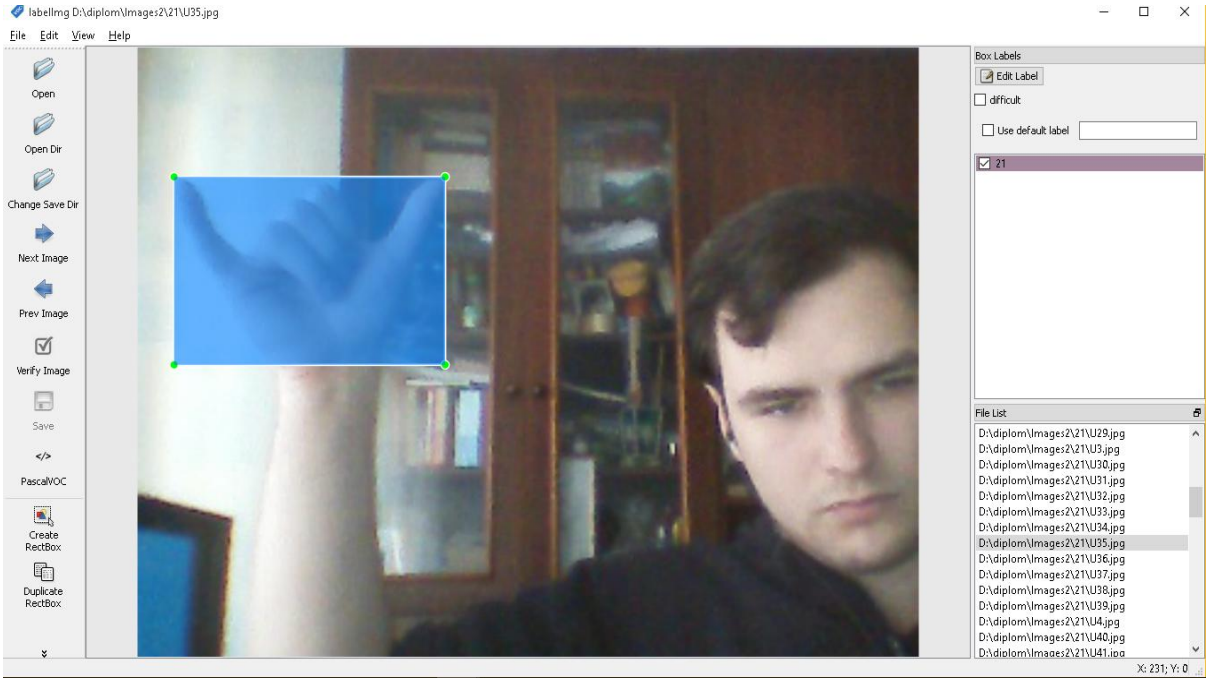


Рис. 3.2. Маркування зображення в LabelImg

Після цього потрібно конвертувати тестові та навчальні зображення і їх маркування в формат tfrecord. Також для навчання мережі потрібно встановити конфігурацію у спеціальному форматі (JSON з визначеними полями), яка використовуватиметься під час навчання. Приклади правильно сформованих конфігурацій можна знайти у офіційних джерелах та відредагувати їх відповідно до параметрів навчання мережі.

Важливі параметри класифікатора: num_classes – кількість класів, feature_extractor – тип мережі виділення ознак, loss – метод оцінки нев'язки, score_converter – функція оцінки впевненості класифікації; параметри сесії тренування: batch_size – кількість зображень, за якими відбувається коригування синаптичних ваг за один крок, optimizer – метод навчання і його параметри (швидкість навчання, момент та їх інкремени), fine_tune_checkpoint – файл, який зберігає проміжний стан мережі, вказується, коли навчання проводиться на за один підхід, а за декілька, можли-

во, з різними конфігураціями, train_input_reader – джерело даних для навчання мережі, eval_input_reader – джерело даних для оцінки точності мережі, num_steps – максимальна кількість кроків навчання. Після підготовки даних та налаштувань навчання можна розпочати навчання мережі.

Стандартне діагностичне виведення фіксує помилку мережі на кожному кроці, але для детального моніторингу процесу навчання Tensorflow надає спеціальний діагностичний інструмент – Tensorboard [16]. Ця утиліта дозволяє у реальному часі за допомогою інтерактивного веб-інтерфейсу відстежувати процес навчання (рис. 3.3).

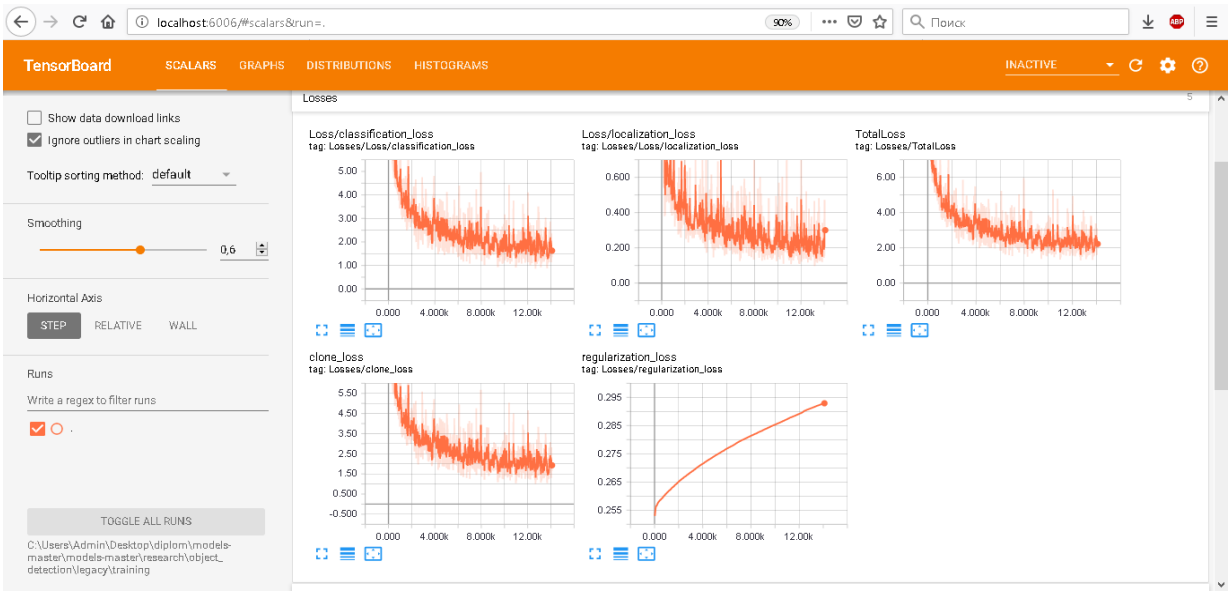


Рис. 3.3. Оцінка помилки у Tensorboard

Таким чином, можна отримати багато важливої актуальної інформації про процес навчання, втім практичний інтерес в даній задачі становлять помилка класифікації (classification_loss) та штраф регуляризації (regulariztion_loss). Помилка класифікації відображає точність розпізнавання мережі і зменшується, якщо мережа збігається. Бажане отримати її середнє значення не більше одиниці. Штраф регуляризації відображає стан мережі в контексті перенавчання. Великий штраф регуляризації (близький або більший за одиницю) означає високу імовірність перенавчання мережі

і в такому випадку варто припинити навчання та змінити гіперпараметри мережі.

Спостереження за навчанням свідчить, що після дванадцяти тисяч кроків алгоритму навчання помилка класифікації коливається близько до одиниці, але не зменшується стабільно, при цьому штраф регуляризації поступово збільшується. Тому на даному етапі навчання мережі варто зупинити і використати набір ваг який дає найменшу помилку класифікації як остаточний варіант.

Для перевірки точності мережі перевіримо її на зображеннях, які не використовувались у навчанні (тестові екземпляри). Результат роботи мережі на них зображено на рис. 3.4, 3.5. Ідентифікатори класів співпадають з присвоєними під час маркування і впевненість мережі висока (найменша впевненість на тестових зображеннях – 66%), це означає, що мережа достатньо натренована і видає очікуваний результат.

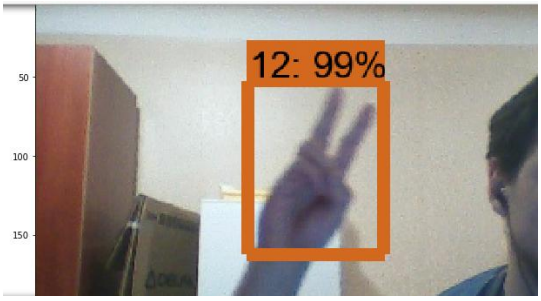


Рис. 3.4.Результати класифікації(1)

Рис. 3.5.Результати класифікації(2)

Програний інтерфейс модуля нейронної мережі передбачає операції завантаження параметрів та виконання класифікації за зображенням.

3.3. База даних елементів жестової мови

Елементи жестової мови є зовнішнім відносно системи контекстом, тому для підтримання незалежності системи від зовнішніх елементів потрібно забезпечити постійне зберігання значень жестів у зовнішньому модулі визначеної структури, який буде завантажуватись у систему і може бути замінений на інший або відкоригований без змін для розроблюваної системи. Варіанти збереження, які дозволяють таку модель функціонування, це точка розширення, конфігураційний файл або база даних.

Точка розширення припускає завантаження програмного коду динамічно під час роботи програми і використовується в основному в мікроядерній архітектурі для забезпечення модульності. Така структура дозволяє під час виконання розширити можливості системи за умови, що модуль, який завантажується має визначений програмний інтерфейс та попередньо підготований належним чином (наприклад, скомпільований та містить метадані, що відповідають даному модулю). Таким чином, цей метод застосовується переважно для розширення функціональності за рахунок додаткового виконуваного коду і менш застосовний для даних.

Конфігураційний файл дозволяє постійне збереження зовнішніх даних у файловій системі у попередньо встановленому форматі, наприклад XML-документ у стандартизованому кодуванні, такому як Unicode, і часто застосовується для постійного зберігання параметрів системи, звернення до яких відбувається не дуже часто, наприклад зчитування під час запуску та запис перед завершенням роботи. Збереження у файлі також застосовне і в інших випадках, але тоді зазвичай копію частини або всієї інформації, наявної у файлі зберігають в оперативній пам’яті доступ до якої здійснюється значно швидше і простіше (такий підхід часто використовується для збереження стану цілісних сутностей системи при серіалізації та подаль-

шому відновленні, а не окремих елементів даних системи). Використання файлу, як засобу постійного зберігання даних також потребує наявності модуля-парсера, який дозволяє отримати необхідні дані і, залежно від формату файлу або створюється спеціально для такого формату, або є загальним засобом (наприклад, парсери XML або JSON, реалізовані у багатьох мовах програмування). Таким чином, даний метод збереження має декілька переваг, серед яких можливість збереження різних варіантів конфігурації, можливості легко відновити параметри або змінити їх, але за наявності складних, пов'язаних між собою структур даних, особливо у великих кількостях використання файлів стає складнішим, більш імовірними стають можливості порушення визначеного формату або узгодженості даних, що можуть призвести до непередбачуваної поведінки системи.

База даних дозволяє забезпечити ефективне збереження зовнішніх даних, та доступ до них, процедури обробки (запити), підтримання структури та контролю за даними, їх однозначний формат, визначений структурою бази, збереження стану бази та його відновлення, методами системи управління базою даних, таким чином, спрощуючи роботу системи з зовнішніми даними, перекладаючи контроль та організацію даних на СУБД, в той час як система, яка отримує доступ до даних, використовує для цього запити.

Бази сучасні СУБД поділяються на реляційні, які використовують таблиці строго визначеного формату та зв'язки між ними для побудови моделі даних, та нереляційні, які використовують інші принципи організації, наприклад, колекції, хеш-таблиці, колонки, тощо. Нереляційні бази даних використовуються у випадках, коли об'єм даних у системі надзвичайно великий або якщо необхідно забезпечити певні специфічні властивості системи, оскільки нереляційні СУБД зазвичай оптимізовані під певні специфічні задачі. У системах загального призначення з невеликим обсягом даних зазвичай застосовуються реляційні бази даних. Маніпулювання даними у реляційних СУБД виконується за допомогою запитів мовою SQL,

яка є декларативною і визначає високорівневі операції, такі як вибірка, додання, видалення, тощо, тому СУБД використовують більш ефективні, оптимізовані алгоритми безпосереднього доступу до даних (деякі діалекти SQL, наприклад, T-SQL, який підтримується у СУБД Microsoft SQL Server, надають можливості імперативного програмування, використовуючи які можна власноруч визначити алгоритм виконання запиту, але ці можливості слід використовувати у складних запитах, які неможливо побудувати засобами стандартної мови SQL).

Ефективність роботи БД сильно залежить від вибору СУБД та побудови структури даних.

Вибір СУБД обумовлюється такими факторами, як вимоги до продуктивності та допустимого навантаження системи, апаратні вимоги, кількість даних. Враховуючи вимоги до розроблюваної системи та апаратні можливості проекту, застосування повноцінної СУБД є неможливим, оскільки такі системи, як SQL Server, Oracle, MySQL, вимагають значних апаратних ресурсів, використовують клієнт-серверну парадигму, створюючи додаткові процеси для ядра СУБД та застосовуються для систем з великим об’ємом даним та багатокористувацьким режимом роботи. Тому для реалізації бази даних у системі потрібно просте рішення, яке забезпечить переваги використання бази даних без створення навантаження на обчислювальні ресурси. Таким рішенням є технологія SQLite, яка представляє собою вбудовану СУБД, тобто її ядро не функціонує, як окремий процес, а є частиною програми. Бази даних SQLite зберігаються як окремі файли спеціального формату, тому такий засіб також має переваги збереження даних у файлі.

Організація даних у базі також достатньо проста і складається з двох таблиць: жестів та значень, пов’язаних відношенням один до багатьох. Жести характеризуються їх ідентифікатором та типом. Значення характеризуються ідентифікатором жесту, до якого вони відносяться, самим значенням та кількістю застосувань (рис. 3.6).

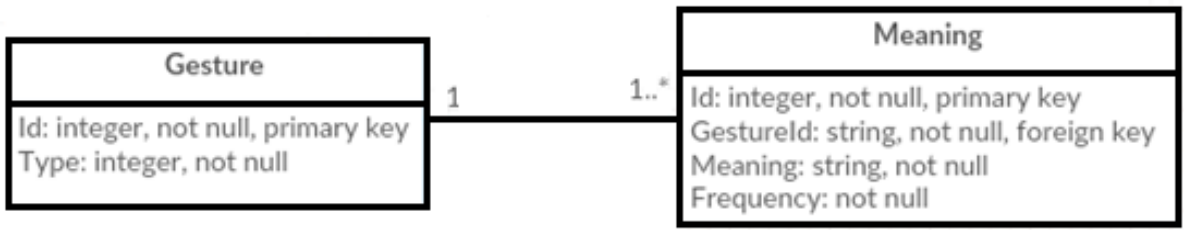


Рис. 3.6. Схема даних системи

Програмний інтерфейс модуля бази даних передбачає операції отримання значень жесту за ідентифікатором та оновлення статистики використання значень.

3.4. Інтерфейс користувача

Інтерфейс надає можливості взаємодії користувача з системою і забезпечує введення та передачу даних до інших модулів та у зворотному порядку виведення оброблених даних у прийнятному для користувача форматі. Розроблювана система є локальним застосунком, що повністю виконується в рамках одного локального процесу, тому доцільно використати графічний віконний інтерфейс.

Розроблення графічного інтерфейсу як правило передбачає використання певного набору віджетів – елементів керування, які є зручними як для користувача, оскільки використовуються в більшості застосунків і мають приблизно однаковий вигляд і поведінку в рамках однієї платформи, так і для розробника, оскільки вони вже реалізовані в численних бібліотеках, які реалізують самі елементи керування, а їх властивості та поведінка швидко та просто налаштовується відповідно потреб проекту. Також у розробленні такого роду інтерфейсів використовуються CASE-засоби дизайну та автоматичної генерації коду інтерфейсу, який залишається тільки доповнити обробниками подій.

Для мови Python, яка використовується для розроблення даного проекту, існує велика кількість фреймворків, які дозволяють побудову вікон-

ного графічного інтерфейсу: Tkinter, Dabo, Kivy, Pyforms, PyQt, IronPython, wxPython та багато інших. Для реалізації інтерфейсу в проекті використовується пакет wxPython, через ряд особливостей, серед яких кросплатформність обгортки над віджетами, які використовує операційна система, наявність багатьох елементів різного призначення, наявність автоматичних інструментів розроблення, багато інших фреймворків використовують пакет wxWidgets, на якому побудовано дану бібліотеку.

Дизайн інтерфейсу має відповідати функціональним вимогам проекту. Враховуючи вимоги, інтерфейс має містити область відображення відео, текстова область, яка містить інтерпретацію розпізнаних елементів жестової системи та можливості налаштування системи у вигляді меню.

Меню способу введення дозволяє користувачу обрати доступні пристрої відео-введення або файли, що містять відео для обробки, меню налаштувань містить загальні параметри, такі як доступні жестові системи, властивості введення (частота кадрів, попередня обробка зображення, автоматичне очищення текстової області).

Програмний інтерфейс модуля графічного інтерфейсу передбачає отримання введення: отримання відео-потоку та передачу кадрів на обробку нейронній мережі та отримання користувацького коригування результатів і відображення його в базі даних; та виведення результатів розпізнавання.

3.5. Модуль отримання відео-потоку

Модуль захоплення відео надає системі можливість отримання введення із відповідного периферійного пристрою або локального файлу. Отримуючи окремі кадри відео, система проводить розпізнавання за допомогою нейронної мережі і обробляє отримані результати.

Захоплення відео реалізується за допомогою пакету opencv-python, реалізації відкритої бібліотеки OpenCV мовою Python, яка надає широкі можливості комп’ютерного зору і також використовує сумісний з нейрон-

ною мережею формат даних – зображення в OpenCV зберігаються у багатоканальних масивах numpy, використовуючи кольорову схему BGR. Таким чином, для використання зображення OpenCV модулем нейронної мережі достатньо здійснити перетворення кольорової схеми та розширити масив додатковими вимірами:

```
def prepare_image(self, image):
    image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
    return np.expand_dims(image.astype(np.uint8), axis=0)
```

OpenCV надає зручні можливості роботи з відео-потоком, але визначення джерел введення покладається на зовнішній код. Таким чином, для забезпечення гнучкості системи необхідно передбачити методи виявлення доступних систем відео-введення в операційній системі. Ця частина застосунку є платформо залежною, на основній платформі розроблення і розгортання MS Windows можна застосувати DirectShow – API Windows, який забезпечує роботу з мультимедійними засобами, в тому числі і доступ до периферійних пристроїв.

Приклад застосування, який забезпечує необхідну функціональність наведено у [17]. Для безпосереднього використання C++ коду у системі необхідно створити обгортку-розширення, яка міститиме необхідні метод у спеціальному форматі і скрипт, за допомогою якого розширення створюється і встановлюється як пакет інтерпретатора.

```
from distutils.core import setup, Extension

module_device = Extension('device', sources = ['device.cpp'],
library_dirs=["""C:\Program Files (x86)\Microsoft
SDKs\Windows\v7.1A\Lib"""])

setup (name = 'WindowsDevices',
        version = '1.0',
        description = 'Get device list with DirectShow',
        ext_modules = [module_device])
```

Незалежно від джерела введення, програмний інтерфейс захоплення відео-потоку має забезпечувати можливість отримання даних. Так, інтерфейс для джерела відео-потоку має вигляд:

```
class ImageSource(object):
    def get_next_image(self):
        pass

    def get_properties(self):
        pass
```

Засоби, які забезпечують доступ до відео-потоку реалізують цей інтерфейс і поліморфно надають іншим засобам доступ до фактичних ресурсів. Подальша реалізація даного інтерфейсу в інших класах дозволяє використання інших ресурсів, наприклад, таких як IP-камери.

Програмний інтерфейс модуля захоплення відео-потоку передбачає описані вище операції отримання кадру та властивостей об’єкта. Враховуючи динамічну типізації мови Python, можливості використання такого інтерфейсу достатньо широкі, оскільки формат значень, які повертають ці методи може бути довільним.

3.6. Особливості реалізації системи

3.6.1. Застосування паралельних обчислень

Використання нейронних мереж без апаратного прискорення (без використання графічного процесора або інших, спеціалізованих апаратних модулів, наприклад TPU або без використання спеціальних інструкцій центрального процесора, які пришвидшують операції нейронних мереж, наприклад AVX, SEE) може займати значний час, особливо на менш обчислювально потужних або частково завантажених машинах. Таким чином, використання синхронних викликів нейронної мережі значно сповільнювати роботу інших елементів системи, в тому числі віконного

інтерфейсу, який не відповідатиме на дії користувача під час обробки зображення.

Для попередження такої особливості системи можна використати паралельне виконання обробки зображення, забезпечуючи повноцінне функціонування інших елементів системи. Хоча підтримання паралельного виконання створює ряд проблем як для роботи, так і для підтримки і розвитку системи (використання паралельності створює додаткові накладні витрати, пов’язані з організацією багатозадачності в операційних системах, а побудова коду стає дуже специфічною), використання такого підходу є виправданим.

Для забезпечення коректного паралельного функціонування нейронної мережі слід застосувати шаблон паралельного програмування producer-consumer, суть якого полягає у спільному використанні буфера даних багатьма процесами/потоками, які додають або вичитують дані із забезпеченням синхронізації та мінімальною необхідністю блокувань [18]. У випадку даної задачі формулювання задачі має певні модифікації: усі задіяні процеси одночасно і створюють дані, і зчитують їх. Схема міжпроцесної взаємодії показана на рис. 3.7.

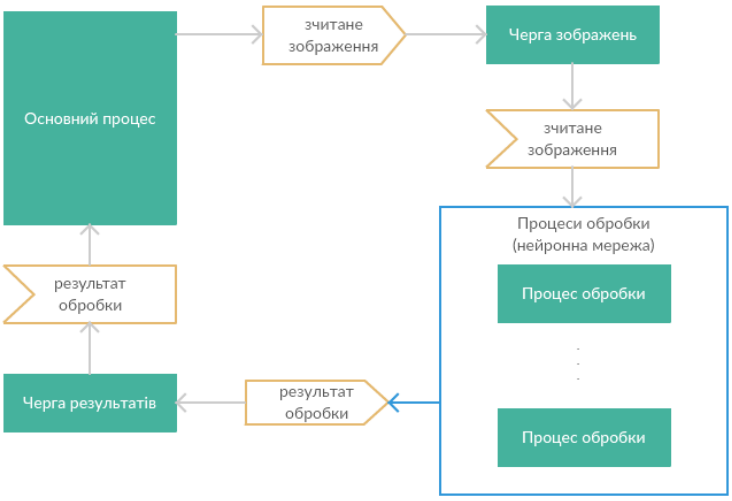


Рис . 3.7. Обмін даними між процесами системи

Таким чином, можна застосувати декілька процесів-обробників даних, які використовують однакову мережу, хоча у цьому не завжди є сенс, оскільки додаткові витрати на підтримання і синхронізацію процесів як правило більші, ніж виграш у швидкодії за рахунок паралельності.

Використання паралельності мовою Python можливе з використанням потоків та процесів (інші засоби, такі як асинхронні виклики та набори процесів неявно використовують ці два основних засоби). Використання потоків є зручнішими і більш зрозумілим, оскільки потоки використовують спільний адресний простір, а отже не потребують окремих засобів взаємодії та оперують одними і тими програмними об'єктами, але при цьому менш ефективним, через особливості реалізації паралельних обчислень на потоках у Python, а саме механізму GIL (global interpreter lock), який забезпечує блокування усіх ресурсів, задіяних одним з потоків для уникнення неконтрольованих операцій. Тому у системі використовується паралельність на процесах. Окремі процеси-обробники виконують процедуру розпізнавання зображення і повертають результат розпізнавання до головного процесу:

```
def mp_processing(network_graph_path, img_q, res_q):
    n_network = ImportNetwork.Network()
    n_network.load(network_graph_path)
    while True:
        result = n_network.predict(img_q.get())
        res_q.put(result)
```

Після цього основний процес послідовно отримує та виконує обробку результатів розпізнавання.

3.6.2. Розширення системи за рахунок змістового наповнення

Основна частина функціональності системи завантажується динамічно під час роботи системи: нейронна мережа, яка здійснює розпізнавання, база даних значень, джерело введення, тому можливо досягти розширення функціональних можливостей системи без внесення змін до програмної

частини, якщо передбачити пошук доступних ресурсів. Таким чином, завантажуючи різні конфігурації мережі і/або бази даних можна змінити поведінку системи.

Така можливість забезпечена за рахунок динамічного завантаження ресурсів під час запуску системи. Таким чином, система шукає доступні в операційній системі пристрої відео введення і перебирає наявні жестові системи у директорії Resources/Systems. Кожна піддиректорія, яка містить файл Description.xml розглядається як доступна жестова система, характеристики елементів якої зберігаються в цьому файлі. Для коректної роботи системи він має відповідати формату

```
<system>
    <modelName>name</modelName>
    <modelGraph>relative_path_to_graph</modelGraph>
    <database>relative_path_to_database</database>
    <description>system_description</description>
</system>
```

де name – назва моделі, relative_path_to_graph – відносний або абсолютний шлях до графу операцій мережі, наприклад Resources/Systems/DactileUA/frozen_inference_graph.pb,
relative_path_to_database – відносний або абсолютний шлях до бази даних значень, наприклад Resources/Systems/DactileUA /DactileGesture.db, system_description – довільний рядок, що містить опис даної жестової системи. Також необхідна наявність вказаних у описі файлів бази даних та графу операцій відповідного формату.

Таким чином, можна розширювати функціональні можливості системи, додаючи підготовані жестові системи у Resources/Systems і використовувати їх без модифікації програмного коду.

3.6.3. Переносимість системи

Розроблення виконується у спеціальному засобі, передбаченому для мови Python – віртуальному середовищі. Мова Python є інтерпретованою,

тобто програмний код не компілюється в окремий виконуваний файл, тому для запуску системи на іншій машині необхідна наявність інтерпретатора з усіма необхідними встановленими бібліотеками. Віртуальне середовище створює ізольовану копію інтерпретатора та усіх встановлених розширень, що дозволяє запустити програму на іншій, апаратно сумісній машині без встановлення та конфігурування, просто скопіювавши систему у віртуальному середовищі і виконавши команду запуску.

3.7. Висновки до розділу 3

У даному розділі визначено важливі елементи рішення саме як програмної системи, проведено дослідження доступних можливостей, обрані засоби та визначені принципи реалізації компонентів:

- Обрано архітектурний шаблон для системи, а саме багаторівневу архітектуру, що складається з ізольованих модулів, які взаємодіють між собою через визначені інтерфейси: інтерфейс користувача, засоби отримання відео, нейронна мережа, база даних значень жестів.
- Для кожного модуля виділено функціональне призначення, проаналізовані наявні засоби реалізації, прийняте рішення щодо використання певних компонентів та обрано загальний підхід реалізації, визначено абстрактні інтерфейси, через які рівні взаємодіють між собою:
  - засоби отримання та попередньої обробки відео – бібліотека OpenCV;
  - інтерфейс користувача – графічний віконний інтерфейс з використанням фреймфорку wxPython;
  - нейронна мережа Mobilenet-SSD, реалізована засобами Tensorflow;
  - вбудована реляційна СУБД SQLite.
- Модулі системи спроектовані таким чином, щоб полегшити їх модифікацію та супровід, а також перенесення на інші платформи (усі

засоби розроблення кросплатформні, як і інтерпретована мова Python, тому хоч система розробляється на базі MS Windows, вона має коректно функціонувати на інших платформах персональних комп’ютерів, таких як Linux, MacOS із незначними змінами коду або без них взагалі). Перехід на мобільні платформи потребує певних змін системи, пов’язаних переважно з інтерфейсом та базовим введенням-виведенням.

4. АНАЛІЗ ОТРИМАНИХ РЕЗУЛЬТАТІВ

4.1. Критерії оцінювання та експертна оцінка розробленої системи

Результати розроблення потребують детального аналізу для оцінки якості створеного програмного забезпечення. Якісне програмне забезпечення має відповідати технічній специфікації та документації, технічному завданню та сформульованим вимогам, і щоб впевнитись у цьому проводяться тестування та валідація, які визначають придатність системи до використання.

Тестування системи на відповідність технічній специфікації, яка витікає із сформульованих функціональних вимог та особливостей проектування, проводиться згідно з методикою тестування. Аналіз результатів тестування дозволяє впевнитись у правильності роботи компонентів системи та їх поєднання, хоча і не гарантує її у всіх випадках.

Валідація системи націлена на перевірку відповідності системи вимогам користувачів або замовників та її застосовність для розв’язання задач, для яких вона розроблялась. Критерії успішності валідації випливають із особливостей предметної області та нефункціональних вимог.

Таким чином, можна виділити наступні критерії якості для розробленої системи (табл. 4.1). Для проведення аналізу результатів розроблення визначимо відносний вплив кожного критерію на результуючу оцінку системи, щоб відобразити більш пріоритетні критерії, яким має відповідати система. Пріоритетність критеріїв відобразимо відносною шкалою від одного до десяти, за якою більший пріоритет відповідає більшому значенню. Відповідність системи встановленим критеріям після перевірки оцінюється за такою самою шкалою. Інтегральним показником якості системи є середнє зважене показників відповідності системи у порівнянні з максимальним можливим значенням. Окрім цього показника окремої уваги потребують найбільш пріоритетні критерії, оскільки вони є критичними для ефектив-

ного використання системи, і критерії, яким система слабо відповідає. Добрим вважається загальний результат 80-100%, задовільним –70-80%.

Таблиця 4.1

Критерії якості системи

Критерій	Пріоритет	Зауваження
Відповідність технічній специфікації	8	Відповідність специфікації забезпечує функціонування компонентів системи відповідно до проектування і перевіряється тестуванням
Швидкодія	7	Швидкодія є важливим параметром системи, оскільки вона має використовуватись в реальному часі, проте не найважливішим в умовах предметної області
Портативність	6	Можливість перенесення системи на інші машини є необхідною характеристикою прикладного програмного забезпечення. Необхідно забезпечити портативність в межах цільового програмно-апаратного середовища, визначеного вимогами, можливість перенесення або версії для інших платформ є перспективним напрямком розвитку системи.
Можливість розширення вмісту системи	6	Можливість розширення та розвитку системи новою функціональністю, що реалізовується як зміною програмного коду так і без неї є важливою характеристикою сучасних проектів, якої можна досягти за допомогою рефакторінгу пізніше у життєвому циклі.

Продовження табл. 4.1

Відсутність помилок першого роду	9	Дані критерії якості націлені на забезпечення точності системи. Помилки першого роду означають визначення жесту системою, коли його фактично немає, в той час як помилки другого роду – не визначення наявного жесту. Більш критичними є помилки першого роду, оскільки вони викликають небажану реакцію системи, а при наявності помилки другого роду можна повторити введення або отримати бажаний результат із іншого кадру відео-потоку
Відсутність помилок другого роду	7	
Документація на систему	5	Вичерпна документація системи включає програмну специфікацію, опис системи для розробника та кінцевого користувача і значно полегшує супровід та використання системи, проте не є критичним пріоритетом на даному етапі проекту
Стабільність	9	Стабільність означає коректність роботи системи і її відповідність іншим характеристикам в різних умовах

Відповідність системи визначеним критеріям відбувається за допомогою тестування роботи системи. Узагальнені результати тестування наводяться у табл. 4.2.

Таблиця 4.2

Результати тестування та валідації системи

Критерій	Пріоритет	Відповідність критерію
Відповідність технічній специфікації	8	10
Швидкодія	7	7

Продовження табл. 4.2

Портативність	6	9
Відсутність помилок першого роду	9	7
Відсутність помилок другого роду	7	8
Документація на систему	5	6
Можливість розширення вмісту системи	6	8
Стабільність	9	7
Підсумок:		443 із 570 (78%)

Отже, система має задовільну загальну якість реалізації. Найбільшими недоліками, які потребують доопрацювання є документація (некритичний недолік) та важливі характеристики роботи, такі як точність та продуктивність. Покращення цих характеристик можна домогтись за рахунок додаткових досліджень нейронної мережі: підбір іншої архітектури та набору даних і гіперпараметрів навчання можуть збільшити ефективність роботи системи.

4.2. Проведення та аналіз результатів тестування

Проведемо тестування системи згідно методики тестування та проаналізуємо отримані результати.

4.2.1. Модульне тестування

Модульне тестування проводиться для перевірки правильності роботи окремих компонентів системи, спрощуючи процес тестування, за рахунок звуження кількості операцій, що виконуються, та дозволяє швидко локалізувати та відлагодити виявлені помилки синтаксису та логіки, та доз-

воляє повторне використання компонентів у інших системах. Програмою тестування передбачено проведення таких тестів та їх сценарії:

1. Перевірка розширення для пошуку пристроїв відео введення.
2. Зчитування даних про жестову систему з XML-файлу.
3. Зчитування даних про програмну систему з XML-файлу.
4. Тестування бази даних.
5. Перевірка нейронної мережі.
6. Тестування інтерфейсу.

Проведення модульного тестування наведено в табл. 4.3-4.9.

Таблиця 4.3

Тестування пошуку пристроїв

Передумови	Вхідні дані	Очікуваний результат	Отриманий результат
Додаткові пристрої введення не підключені	–	1 [*назва вбудованого пристрою*]	1 ['HP Truevision HD']
Підключено USB-веб-камеру	–	2 [*список назв пристроїв*]	2 ['HP Truevision HD', 'USB Camera']

У випадку з підключеною USB-веб-камерою та без неї розширення коректно визначає перелік доступних пристроїв введення відео, тому його функціонування є коректним.

Таблиця 4.4

Тестування обробки даних про жестову систему з XML-файлу

Передумови	Вхідні дані	Очікуваний результат	Отриманий результат
Наявний файл example.xml відповідного формату	path = ".\example.xml"	model Resources/Systems/DactileUA/frozen_inference_graph.pb Resources/Systems/DactileUA/DactileGesture.db Українська дактильна абекта	model Resources/Systems/DactileUA/frozen_inference_graph.pb Resources/Systems/DactileUA/DactileGesture.db Українська дактильна абекта

Дані сформованих інформаційних об’єктів співпадають, тому функціонування xml-парсера даних про жестову систему є коректним.

Таблиця 4.5

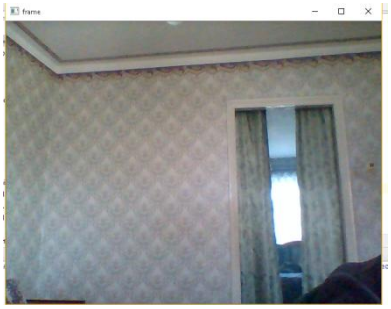
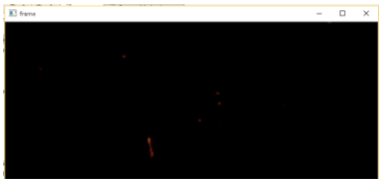
Тестування обробки даних про програму з XML-файлу

Передумови	Вхідні дані	Очікуваний результат	Отриманий результат
Наявний файл Info.xml відповідного формату	path = "./Info.xml"	Gesture recognition system v 1.0.0 build date: 02.06.2019	Gesture recognition system v 1.0.0 build date: 02.06.2019

Формат і зміст зчитаних даних співпадають з очікуваним, тому робота xml-парсера даних про програму є коректною.

Таблиця 4.6

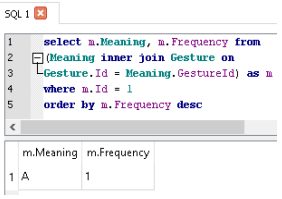
Тестування захоплення відео

Передумови	Вхідні дані	Очікуваний результат	Отриманий результат
Доступний хоча б один пристрій відео введення	cam_id=0	Вікно, у яккому відображається відео з камери	
Наявний файл Video.avi	filename= ".\Video.avi"	Вікно, у якому відображається відео з файлу	

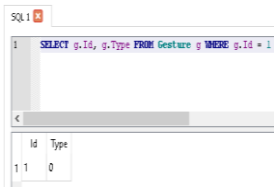
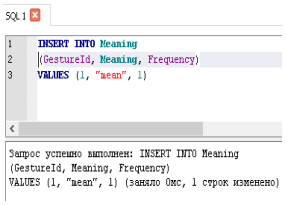
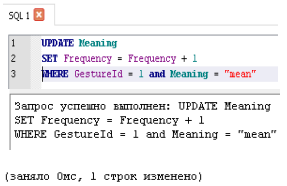
Модуль захоплення відео-потоку коректно отримує дані з пристрою введення та локального файлу, тому його роботу можна вважати коректною.

Таблиця 4.7

Відлаголдження запитів

Передумови	Вхідні дані	Очікуваний результат	Отриманий результат
Створена база даних відповідної структури, заповнена даними	%id = 1 SELECT m.Meaning, m.Frequency FROM (Meaning INNER JOIN Gesture On Gesture.Id = Meaning.GestureId) AS m WHERE m.Id = %id ORDER BY m.Frequency DESC	Таблиця формату (один або більше рядків) m.Meaning | m.Frequency … | …	

Продовження табл. 4.7

<table>
<tr>
<td rowspan="3"></td>
<td>%id = 1
SELECT g.Id, g.Type FROM Gesture g WHERE g.Id = %id</td>
<td>Таблиця формату (один рядок)<table><tr><td>g.Id</td><td>g.Type</td></tr><tr><td>…</td><td>…</td></tr></table></td>
<td>  </td>
</tr>
<tr>
<td>%id=1, %mean=mean, %frq=1
INSERT INTO Meaning (GestureId, Meaning, Frequency) VALUES (%id, `%mean`, %frq)</td>
<td>Вставка одного рядка в таблицю Meaning</td>
<td>  </td>
</tr>
<tr>
<td>%id=1, %mean=mean, %dfrq=1
UPDATE Meaning SET Frequency = Frequency + %dfrq WHERE GestureId = %id and Meaning = '%mean'</td>
<td>Оновлення одного рядка таблиці Meaning</td>
<td>  </td>
</tr>
</table>

Результати запитів в СУБД відповідають очікуваним результатам, запити є коректними і можуть застосовуватись у програмній реалізації.

Таблиця 4.8

Тестування програмного компоненту доступу до бази даних

Передумови	Вхідні дані	Очікуваний результат	Отриманий результат
Створена база даних відповідної структури, заповнена даними. Файл даної БД знаходиться за заданим шляхом	path= ".\DactileGesture.db" gesture_id = 1 new_meaning = "sdg" init_frequency = 3 added_frequency = 39	[('A', 1)] 1 0 [('A', 1)] [('sdg', 3), ('A', 1)] [('sdg', 42), ('A', 1)]	[('A', 1)] 1 0 [('A', 1)] [('sdg', 3), ('A', 1)] [('sdg', 42), ('A', 1)]

Компонент доступу до бази даних коректно виконує запити, ініціалізує та звільняє підключення до бази даних, тому його функціонування є коректним.

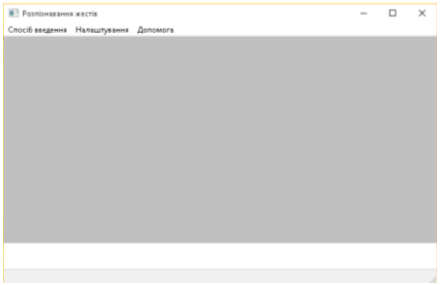
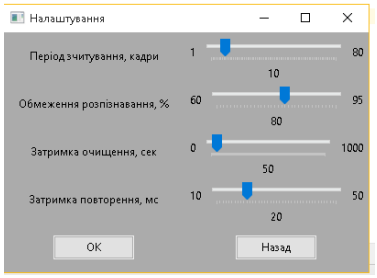
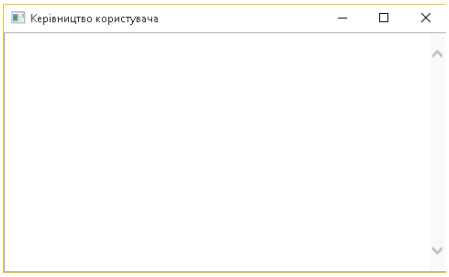
Перевірка характеристик нейронної мережі виконується для оцінки основних характеристик, таких як точність та швидкість розпізнавання. З результатів тестування (див. Додаток 3) можна отримати наступні характеристики мережі:

- Швидкодія. Обробка зображення в середньому виконується за 0.267 с. Такої швидкості достатньо для обробки відео в реальному часі. В залежності від умов і враховуючи специфіку реалізації системи (в основному, паралельність) цей показник може дещо відрізнятись.
- Імовірність помилки першого роду (кількість помилкових розпізнавань до загальної кількості зображень без жестів): 6%.
- Імовірність помилки другого роду (кількість неправильних розпізнавань до загальної кількості зображень з жестами): 4%.

Такі характеристики мережі є задовільними і відповідають сформульованим вимогам.

Таблиця 4.9

Тестування віконного інтерфейсу

Передумови	Вхідні дані	Очікуваний результат	Отриманий результат
–	frame = MainWindow (None)	*<головне вікно системи>*	
	frame = SettingsFrame (None)	*<вікно налаштувань системи>*	
	frame = HelpFrame (None)	*<вікно керівництва користувача>*	

Тести інтерфейсу призначені для перевірки дизайну і його відповідності очікуваному вигляду. Функціональність інтерфейсу перевірятиметься під час інтеграційного тестування.

4.2.2. Інтеграційне тестування

Інтеграційне тестування перевіряє правильність зв’язків між компонентами, сумісність їх інтерфейсів та загальну роботу системи, її відповідність заявленим вимогам. Проведемо перевірку роботи системи за сценарієм, наведеним у методиці тестування (табл. 4.10).

Таблиця 4.10

Проведення інтеграційного тестування

№ з/п	Дія користувача	Відповідність очікуваній реакції
1	Запуск програми на виконання	+
2	Перехід у меню «Допомога», пункт «Про програму»	+
3	Натиснути «ОК»	+
4	Перехід у меню «Допомога», пункт «Довідка»	+
5	Закрити вікно довідки	+
6	Перехід у меню «Налаштування», пункт «Налаштування системи…»	+
7	Змінити налаштування довільним чином	+
8	Натиснути «Назад»	+
9	Перехід у меню «Налаштування», пункт «Налаштування системи…»	+
10	Змінити налаштування довільним чином	+
11	Натиснути «ОК»	+
12	Перехід у меню «Спосіб введення», пункт «Відео з камери», підпункт «HP TrueVision HD»	+
13	Перехід у меню «Налаштування», пункт «Жестові системи», підпункт «Українська дактильна абетка»	+
14	Показати послідовність введення, наприклад “А”, “Б”, “В”, “Г”, “Д”	+
15	Натиснути на одне із отриманих значень	+
16	Ввести інше значення та натиснути Enter	+

Продовження табл. 4.10

17	Ввести жест із новим значенням ще раз	+
18	Натиснути на одне із отриманих значень	+
19	Обрати випадний список значень	+
20	Обрати нове значення та натиснути Enter	+
21	Почекати певний час, не вводячи жестів	+
22	Закрити головне вікно програми	+

Результати проведених дій користувача відповідають очікуваним, що означає правильність роботи системи за даних умов. Хоча результати цього тестування не гарантують коректність роботи системи за будь-яких умов, вони дозволяють пересвідчитись в тому, що компоненти системи поєднані правильно і можуть забезпечити коректне функціонування системи.

4.2.3. Висновки з тестування

Проведення модульного тестування показало відповідність компонентів системи очікуваній поведінці. Особливо важливим є тест нейронної мережі, який на тестових даних дозволяє отримати такі характеристики:

Інтеграційне тестування показало відповідність очікуваній поведінці системи, достатню точність і продуктивність.

Початкову версію системи можна вважати задовільною і розробленою у відповідності до сформульованих вимог. Можливий також і подальший розвиток та модифікація системи у різних напрямках.

4.3. Перспективи подальшого розвитку

Життєвий цикл програмного забезпечення передбачає супровід та розвиток розробленого програмного забезпечення.

Детальний аналіз характеристик системи дозволяє прийти висновку, що покращення та розвиток системи можливі у декількох різних напрямках, серед яких :

1. *Оптимізація та покращення ефективності.*

Система розрахована на використання у режимі реального часу як варіант людинно-машинного інтерфейсу, тому її застосування має бути зручним та ефективним, тобто достатньо швидким, щоб обробляти відеопотік і достатньо точним у розпізнаванні. Досягти покращення у цій області можна за рахунок оптимізації нейронної мережі: інша, більш підходяща архітектура мережі, більший та різноманітніший набір даних для навчання мережі, інші гіперпараметри навчання, зміни процесу навчання (наприклад, наявність валідаційного набору даних в додаток до тестового для точнішого відстеження процесу навчання). Додаткові дослідження, направлені на вивчення цих характеристик можуть збільшити її загальну продуктивність, але потребують значних витрат часу та кращого обладнання для більш швидкого процесу навчання та дослідження результатів.

2. *Розширення змісту системи.*

Розвиток проекту можливий за рахунок додання нових змістових елементів замість зміни поведінки системи. Наприклад, підтримка різних жестових систем та збільшення словника значень існуючих, що допоможе розширити застосування системи без змін до її програмної будови. Такий метод розвитку потребує вивчення контексту та збору даних для навчання мережі – в цілому достатньо простий спосіб, який потребує певних часових витрат на дослідження, навчання та тестування.

3. *Повторне використання модулів.*

Архітектура системи передбачає певний рівень ізоляції компонентів, тому їх можна використати повторно в іншому середовищі. Особливо це стосується ядра системи – нейронної мережі. Можливе її використання з іншим інтерфейсом, наприклад як система управління або метод безпосереднього введення в операційній системі (HID-система). Для

застосування даного методу розвитку системи необхідно підвищити ефективність (в основному точність, оскільки швидкодія є менш критичною у більшості варіантів такого використання підсистем) та універсальність системи, дозволяючи її використання в складніших умовах, наприклад, коли інші методи недоступні або неефективні.

4. *Підтримка інших платформ розгортання.*

Поширення системи або її окремих версій для інших середовищ виконання, особливо для мобільних пристроїв, може значно розширити застосовність системи, але потребує застосування інших технологій, покращення продуктивності, та зміни внутрішньої будови системи для розсортування на інших платформах. Також, значно збільшується необхідність підтримки системи на різних платформах, при чому модифікації як правило потребують специфічні для платформи компоненти, оскільки кросплатформні компоненти системи змінюються один раз для усіх платформ.

5. *Рефакторинг та розширення функціональності.*

Даний підхід включає зміну програмної частини системи, при чому можливі як зміни, які не впливають на поведінку системи, а лише підвищують якість коду (впровадження шаблонів проектування, підтримка коду для кращого розуміння та супроводження, коментування та приведення до стандартів) для спрощення підтримки та введення нових можливостей, так і зміни, що змінюють поведінку системи, додаючи або змінюючи функціональність. Рефакторинг системи можна проводити майже за будь-яких умов, оскільки він не відображається на поведінці системи, а покращує стан кодової бази і для впровадження потребує тільки проведення локального тестування компонентів, які були змінені, щоб упевнитись, що поведінка системи не змінилась. Зміни, що відображаються на поведінці потребують попередніх висновків про їх необхідність, досліджень стосовно суті змін та можуть проводитись у вигляді окремого проекту із повним життєвим циклом, якщо дані вони достатньо значні.

4.4. Висновки до розділу 4

У даному розділі проведено тестування розробленої системи, аналіз результатів розроблення та тестування, виділено критерії якості системи та проведено оцінку відповідності системи даним критеріям. Визначено характеристики, яким має відповідати система для досягнення допустимого рівня якості.

Проведена оцінка системи дозволяє визначити характеристики системи, які потребують покращення та відповідні дії, які можуть вплинути на дані характеристики і додаткові наслідки за затрати, необхідні для їх проведення.

Визначено напрямки подальшого розвитку, покращення та супроводу розробленого програмного забезпечення, методи їх впровадження, наслідки та необхідні засоби, потрібні для проведення змін.

В цілому, розроблена система відповідає сформульованим вимогам із задовільним рівнем якості продукту, але має недоліки, виправлення або покращення яких підвищить ефективність системи або розширить сферу її застосування. Розроблений програмний продукт має широкі можливості до розширення як функціональності, так і варіантів використання системи, завдяки чому вона може супроводжуватись та зберігати відповідність користувацьким вимогам значно довше, ніж статичний монолітний застосунок.

ВИСНОВКИ

Отже, у ході даного проекту було розроблено програмний застосунок із повним життєвим циклом: створено та узгоджено технічне завдання, проаналізовано та обрано засоби розроблення, які найбільше підходять до умов проекту, досліджено принципи роботи та навчання штучних нейронних мереж, зібрано та підготовано набір даних для навчання мережі, проведено декілька експериментів з навчання з різними гіперпараметрами та даними, спроектовано архітектуру прикладного застосунку, відповідно до якої створено та протестовано компоненти системи, проведено інтеграційне тестування та перевірку системи на відповідність вимогами та критеріям якості.

Окрім цього, дана робота зачіпає важливі та актуальні галузі в сучасній сфері інформаційних технологій, серед яких:

- нейронні мережі та штучний інтелект;
- паралельні обчислення;
- нестандарті інтерфейси людинно-машинної взаємодії;
- проектування ПЗ, яке можна розширювати або повторно використовувати;
- управління проектами в умовах невизначеності та недостатності ресурсів.

Після доопрацювання результати даного проекту можна застосувати для створення програних систем промислового рівня для розв’язання прикладних проблем, наприклад, проблеми комунікації людей з вадами слуху або мовлення.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Характеристика технології Java [Електронний ресурс]. – Режим доступу: https://www.java.com/ru/about/. – Дата доступу: лютий 2019.
2. Огляд мови C# [Електронний ресурс]. – Режим доступу: https://docs.microsoft.com/ru-ru/dotnet/csharp/tour-of-csharp/. – Дата доступу: лютий 2019.
3. Документація Python [Електронний ресурс]. – Режим доступу: https://docs.python.org/3/. – Дата доступу: лютий 2019.
4. Документація OpenCV [Електронний ресурс]. – Режим доступу: https://opencv.org/ . – Дата доступу: лютий 2019
5. Документація LTI [Електронний ресурс]. – Режим доступу: http://ltilib.sourceforge.net/doc/homepage/index.shtml. . – Дата доступу: лютий 2019
6. Порівняння бібліотек комп’ютерного зору для використання в застосунку, який використовує технологію розпізнавання пласких зображень [Електронний ресурс]. – Режим доступу: https://cyberleninka.ru/article/v/sravnenie-bibliotek-kompyuternogo-zreniya-dlya-primeneniya-v-prilozhenii-ispolzuyuschem-tehnologiyu-raspoznavaniya-ploskih. – Дата доступу: лютий 2019
7. Структура і принципи роботи нейронної мережі. [Електронний ресурс]. – Режим доступу: https://studbooks.net/2030599/informatika/struktura_printsipy_raboty_neyronnoy_seti . – Дата доступу: квітень 2019
8. Принцип роботи згорткової нейронної мережі [Електронний ресурс]. – Режим доступу: https://habr.com/ru/post/416777/ – Дата доступу: квітень 2019

9. Аналіз і порівняння методів навчання нейронних мереж [Електронний ресурс]. – Режим доступу: https://cyberleninka.ru/article/v/analiz-i-sravnenie-metodov-obucheniya-neyronnyh-setey – Дата доступу: квітень 2019
10. Використання генетичного алгоритму для навчання нейронних мереж [Електронний ресурс]. – Режим доступу: https://www.science-education.ru/ru/article/view?id=5138. – Дата доступу: квітень 2019
11. Легковагова мережа MobileNet v1/v2 [Електронний ресурс]. – Режим доступу: https://bzdww.com/article/144579/. – Дата доступу: березень 2019.
12. Порівняння характеристик архітектур згорткових нейронних мереж [Електронний ресурс]. – Режим доступу: https://github.com/tensorflow/models/blob/master/research/object_detection/g3doc/detection_model_zoo.md – Дата доступу: квітень 2019.
13. Огляд популярних шаблонів архітектури програмного забезпечення [Електронний ресурс]. – Режим доступу: https://towardsdatascience.com/10-common-software-architectural-patterns-in-a-nutshell-a0b47a1e9013 . – Дата доступу: квітень 2019.
14. Архітектура програмного забезпечення [Електронний ресурс]. – Режим доступу: . https://ru.wikipedia.org/wiki/%D0%90%D1%80%D1%85%D0%B8%D1%82%D0%B5%D0%BA%D1%82%D1%83%D1%80%D0%B0_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%BD%D0%BE%D0%B3%D0%BE_%D0%BE%D0%B1%D0%B5%D1%81%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%B8%D1%8F. – Дата доступу: квітень 2019.
15. Порівняння засобів машинного навчання [Електронний ресурс]. – Режим доступу: https://ru.wikipedia.org/wiki/%D0%A1%D1%80%D0%B0%D0%B2%

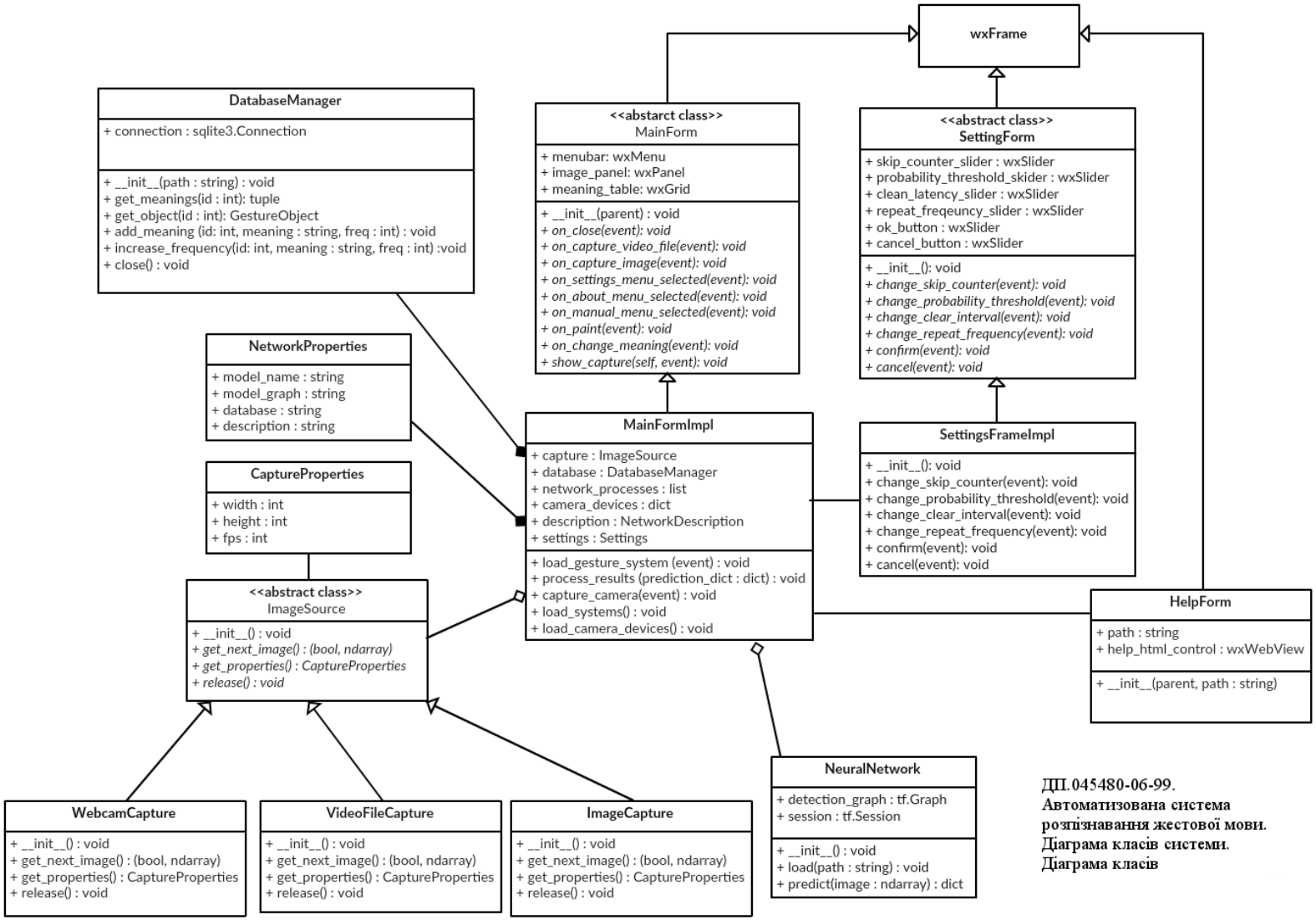
D0%BD%D0%B5%D0%BD%D0%B8%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC_%D0%B3%D0%BB%D1%83%D0%B1%D0%B8%D0%BD%D0%BD%D0%BE%D0%B3%D0%BE_%D0%BE%D0%B1%D1%83%D1%87%D0%B5%D0%BD%D0%B8%D1%8F

16. Документація TensorBoard [Електронний ресурс]. – Режим доступу: . https://www.tensorflow.org/tensorboard/r1/overview . – Дата доступу: квітень 2019
17. Документація DirectShow [Електронний ресурс]. – Режим доступу: https://docs.microsoft.com/ru_ru/windows/desktop/DirectShow/selecting-a-capture-device. – Дата доступу: квітень 2019
18. Задача producer-consumer [Електронний ресурс]. – Режим доступу: https://en.wikipedia.org/wiki/Producer%E2%80%93 consumer_problem. – Дата доступу: травень 2019

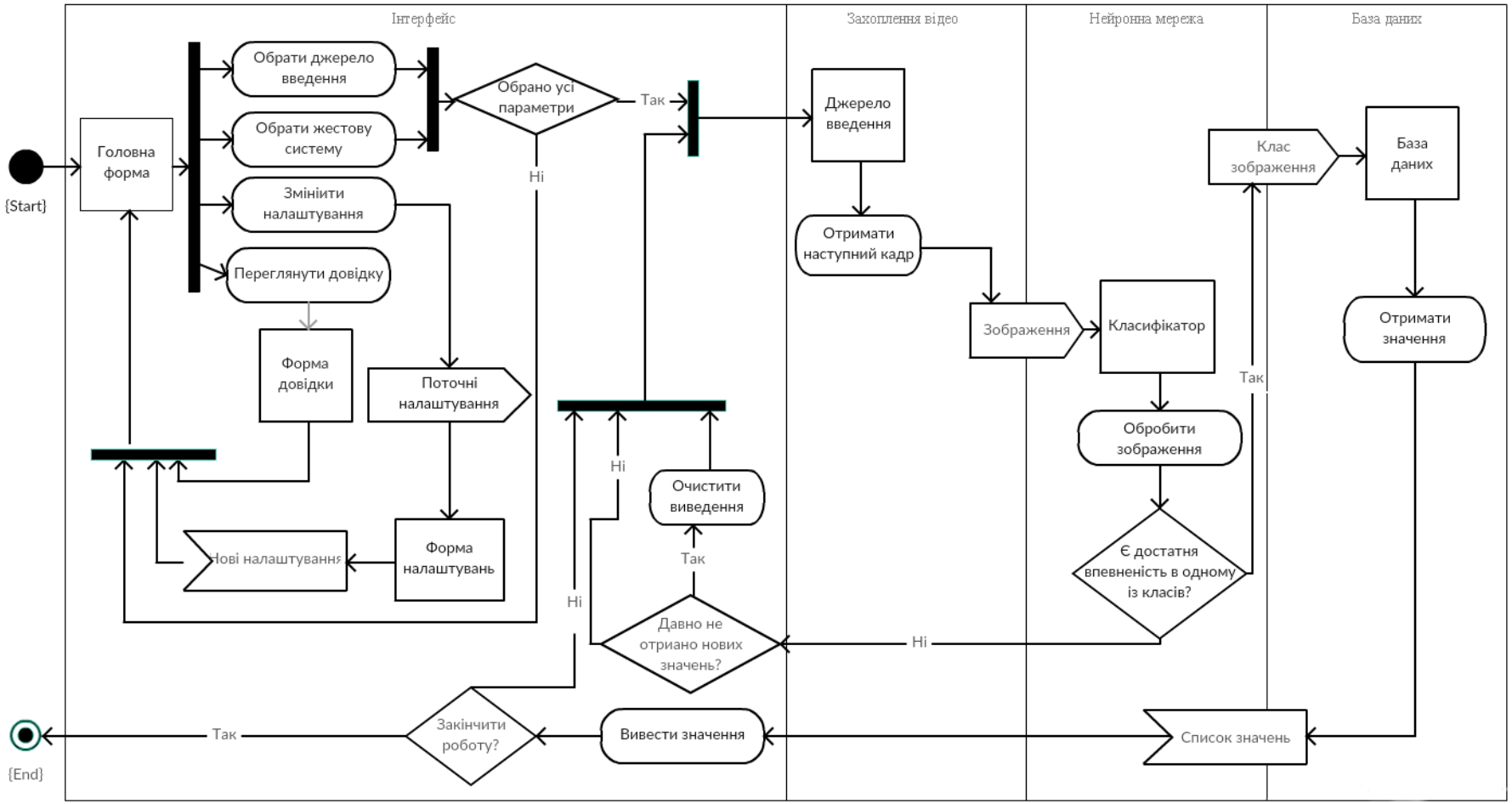
ДОДАТКИ

Додаток 1

Копії графічних матеріалів



ДП.045480-06-99.
Автоматизована система розпізнавання жестової мови.
Діаграма класів системи.
Діаграма класів



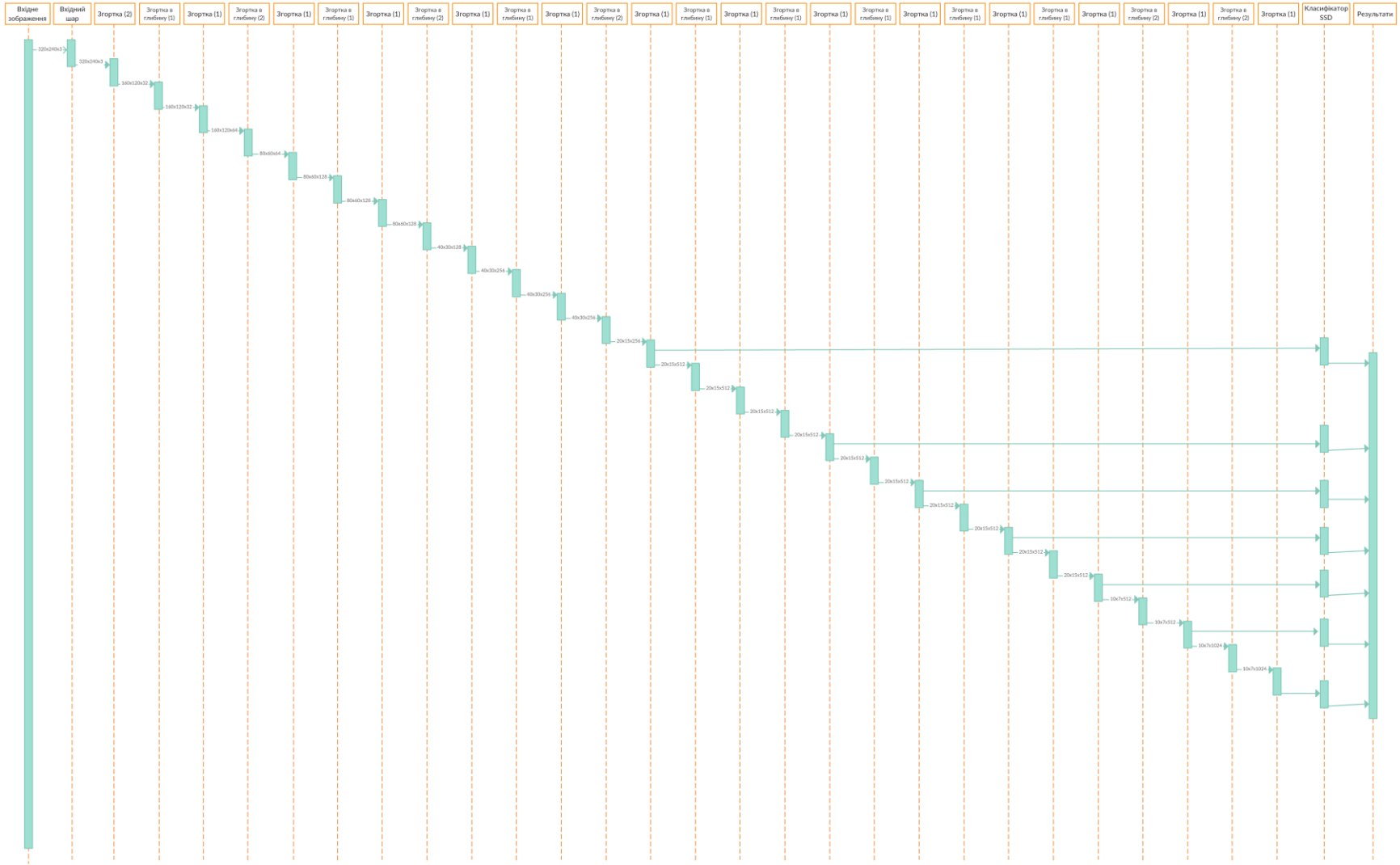
ДП.045480-07-99.
Автоматизована система розпізнавання жестової мови.
Послідовність роботи системи.
Діаграма діяльності

Елементи жестової мови

 **А**	 Б	 В	 Г	 Ґ	 Д
 Е	 Є	 Ж	 З	 И	 І
 Ї	 К	 Л	 М	 Н	 О
 П	 Р	 С	 Т	 У	 Ф
 Х	 Ц	 Ч	 Ш	 Щ	 Ь
		 Ю	 Я		

Пеня О.Р., група КП-51

Структура мережі MobileNet-SSD



Пеня О.Р., група КП-51

Додаток 2

Лістинг програми

2.1. ImageSource.py. Інтерфейс джерела відео-потоку.

```
class ImageSource(object):
  def __init__(self):
    pass


  def get_next_image(self):
    pass


  def get_properties(self):
    pass


  def release(self):
    pass
```

2.2. CaptureMethods.py. Методи захоплення відео-потоку

```
import cv2


from Capture import ImageSource

from Utils import CaptureProperties




class WebcamCapture(ImageSource.ImageSource):
  def __init__(self, cam_id=0):
    ImageSource.ImageSource.__init__(self)
    self.cam_id = cam_id
    self.capture = cv2.VideoCapture(self.cam_id)


  def get_properties(self):
    if self.capture is None:
      props =CaptureProperties.CaptureProperties (0, 0, 0)
    else:
      props =CaptureProperties.CaptureProperties(
self.capture.get(cv2.CAP_PROP_FRAME_WIDTH),
self.capture.get(cv2.CAP_PROP_FRAME_HEIGHT),
self.capture.get(cv2.CAP_PROP_FPS))
    return props



  def get_next_image(self):
    if self.capture is not None:
      return self.capture.read()
    else:
      return (False, None)



  def release(self):
    if self.capture is not None:
```

```
      self.capture.release()



class VideoFileCapture(ImageSource.ImageSource):


  def __init__(self, filename):
    ImageSource.ImageSource.__init__(self)
    self.filename = filename
    self.capture = cv2.VideoCapture(self.filename)


  def get_properties(self):
    if self.capture is None:
      props = CaptureProperties.CaptureProperties(0, 0, 0)
    else:
      props = CaptureProperties.CaptureProperties
(self.capture.get(cv2.CAP_PROP_FRAME_WIDTH),
self.capture.get(cv2.CAP_PROP_FRAME_HEIGHT),
self.capture.get(cv2.CAP_PROP_FPS))
    return props


  def get_next_image(self):
    if self.capture is not None:
      return self.capture.read()
    else:
      return (False, None)


  def release(self):
    if self.capture is not None:
      self.capture.release()


class ImageFileCapture(ImageSource.ImageSource):


  def __init__(self, filename):
    ImageSource.ImageSource.__init__(self)
    self.filename = filename
    self.img = cv2.imread(self.filename)


  def get_properties(self):
    if self.capture is None:
      props = CaptureProperties.CaptureProperties(0, 0, 0)
    else:
      (rows, cols) = self.img.shape
      props = CaptureProperties.CaptureProperties (cols, rows, 1)
    return props


  def get_next_image(self):
    if self.img is not None:
      return (True, self.img)
```

```
    else:
      return (False, None)


  def release(self):
    pass
```

2.2. GestureObject.py. Структура, що зберігає дані жесту

```
class GestureObject(object):
  def __init__(self, id, type, meanings):
    self.id = id
    self.type = type
    self.meanings = meanings


  def __str__(self):
    return " ".join((self.id.__str__(), self.type.__str__(),
self.meanings.__str__()))
```

2.3. DatabaseConnector.py. Модуль доступу до БД.

```
import sqlite3

from Database import GestureObject




class DatabaseManager(object):
  def __init__(self, db_path):
    self.connection = sqlite3.connect(db_path)


  def get_meanings(self, gesture_id):
    table = self.connection.cursor()
    query_params = (str(gesture_id),)
    table.execute("""select m.Meaning, m.Frequency from (Meaning inner
join Gesture on Gesture.Id = Meaning.GestureId) as m where m.Id = ?
order by m.Frequency desc""", query_params)
    meanings = table.fetchall()
    table.close()
    return meanings



  def get_object(self, gesture_id):
    result = None
    gesture_data = self.connection.cursor()
    gesture_data.execute("""select g.Id, g.Type from Gesture g where
g.Id = ?""", (gesture_id,))
    gesture = gesture_data.fetchone()
    if gesture is not None:
      meanings = self.get_meanings(gesture_id)
      result = GestureObject.GestureObject (gesture[0], gesture[1],
meanings)
    gesture_data.close()
    return result
```

```
  def add_meaning(self, gesture_id, meaning, freq=1):
    table = self.connection.cursor()
    # params = (gesture_id, meaning, freq)
    query = "insert into Meaning (GestureId, Meaning, Frequency)
values(?, '?', ?)"
    table.execute(query, (gesture_id, meaning, freq))
    self.connection.commit()
    table.close()



  def increase_frequency(self, gesture_id, meaning, inc_frequency=1):
    table = self.connection.cursor()
    params = (inc_frequency, gesture_id, meaning)
    query = "update Meaning set Frequency = Frequency + ? where
GestureId = ? and Meaning = '?'"
    table.execute(query, params)
    self.connection.commit()
    table.close()


  def close(self):
    try:
      self.connection.commit()
    finally:
      try:
        self.connection.close()
      except:
        pass
```

2.4. NeuralNetwork.py. Модуль нейронної мережі.

```
import numpy as np

import tensorflow as tf

import cv2




class NeuralNetwork(object):
  def __init__(self):

    self.detection_graph = tf.Graph()
    self.session = None



  # TODO : put this outside of the class
  def prepare_image(self, image):
    image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
    return np.expand_dims(image.astype(np.uint8), axis=0)



  def load(self, path_to_graph):
    with self.detection_graph.as_default():
      od_graph_def = tf.GraphDef()
      with tf.gfile.GFile(path_to_graph, 'rb') as fid:
        serialized_graph = fid.read()
        od_graph_def.ParseFromString (serialized_graph)
```

```
        tf.import_graph_def(od_graph_def, name='')
    self.session = tf.Session (graph=self.detection_graph)



  def predict(self, input_image):
    image = self.prepare_image(input_image)
    ops = self.detection_graph.get_operations()
    all_tensor_names = {output.name for op in ops for output in
op.outputs}
    tensor_dict = {}

    for key in ['num_detections', 'detection_scores',
'detection_classes']:
      tensor_name = key + ':0'
      if tensor_name in all_tensor_names:
        tensor_dict[key] = self.detection_graph.
get_tensor_by_name(tensor_name)

    image_tensor = self.detection_graph.
get_tensor_by_name('image_tensor:0')



    output_dict = self.session.run(tensor_dict, feed_dict =
{image_tensor: image})
    output_dict['num_detections'] =
int(output_dict['num_detections'][0])
    output_dict['detection_classes'] =
output_dict['detection_classes'][0].astype(np.uint8)
    output_dict['detection_scores'] =
output_dict['detection_scores'][0]


    return output_dict



  def close(self):
    self.session.on_close()
```

2.5. ReadBuildInfo.py. Утиліта доступу до інформації про систему

```
from lxml import etree




class BuildInfo(object):
  def __init__(self, title, version, date, desc=""):
    self.title = title
    self.version = version
    self.date = date
    self.description = desc



  def __str__(self):
    return self.title + "\n" + self.version + "\n" + self.date + "\n"
+ self.description
```

```
def get_build_info_from_xml(path):
  tree = etree.parse(path)


  tags = (tree.find("name"), tree.find("version"),
tree.find("builddate"), tree.find("description"))

  params = []
  for tag in tags:
    if tag is None:
      params.append("")
    else:
      params.append(tag.text)


  return BuildInfo(params[0], params[1], params[2], params[3])
```

2.6. NetworkDescription.py. Утиліта доступу до інформацію про жестову систему

```
from lxml import etree



class NetworkDescription(object):
  def __init__(self, m_name, m_graph, db, desc):
    self.model_name = m_name
    self.model_graph = m_graph
    self.database = db
    self.description = desc


def get_network_description_from_xml(path):
  tree = etree.parse(path)

  tags = (tree.find("modelName"), tree.find("modelGraph"),
tree.find("database"), tree.find("description"))
  params = []
  for tag in tags:
    if tag is None:
      return None
    else:
      params.append(tag.text)

  return NetworkDescription(params[0], params[1], params[2],
params[3])
```

2.7. IternalSettings.py. Структура налаштувань системи

```
class Settings(object):
  def __init__(self):
    self.skip_counter = 20
    self.probability_threshold = 80
    self.clear_interval = 1000
    self.repeat_frequency = 5
```

2.8. CaptureProperties.py. Структура властивостей відео-потоку

```
class CaptureProperties(object):
  def __init__(self, width, height, fps):
    self.width = width
    self.height = height
    self.fps = fps
```

2.9. MainFormImpl.py. Головне вікно програми

```
import wx
import cv2
import numpy as np


import os
import multiprocessing as mp
import time


import device


from UI import MainForm, OutputControl, HelpFrame, SettingsFrameImpl
from Capture import CaptureMethods
from NeuralNetwork import NeuralNetwork
from Utils import NetworkDescription, ReadBuildInfo, InternalSettings
from Database import DatabaseConnector


process_count = 2
network_processes = []
image_queue = mp.Queue(maxsize=200)
results_queue = mp.Queue(maxsize=200)
device_list = device.getDeviceList()


def mp_processing(network_graph_path, img_q, res_q):
  print("started net_process")
  n_network = NeuralNetwork.NeuralNetwork()
  n_network.load(network_graph_path)
  started = time.time()
  while True:
    print("net_ready ", img_q.qsize(), time.time())
    result = n_network.predict(img_q.get())
    res_q.put(result)
    print("processing done in ", time.time() - started)
    started = time.time()


# Implementing MainWindow
class MainFormImpl(MainForm.MainWindow):
  settings = InternalSettings.Settings()


  def load_systems(self):
    for folder in os.walk("Resources//Systems"):
      if len(folder[1]) == 1:
        new_item = wx.MenuItem(self.system_submenu, wx.ID_ANY,
                   folder[1][0].__str__(), wx.EmptyString,
wx.ITEM_NORMAL)
```

```
        self.Bind(wx.EVT_MENU, self.load_gesture_system,
id=new_item.GetId())
        self.system_submenu.AppendItem(new_item)


  def load_camera_devices(self):
    global device_list
    index = 0
    for name in device_list:
      self.camera_devices[name] = index
      index += 1
      new_item = wx.MenuItem(self.camera_submenu, wx.ID_ANY, name,
wx.EmptyString, wx.ITEM_NORMAL)
      self.Bind(wx.EVT_MENU, self.capture_camera, id=new_item.GetId())
      self.camera_submenu.Append(new_item)



  def __init__(self, parent):
    MainForm.MainWindow.__init__(self, parent)
    self.camera_devices = {}
    self.load_systems()
    self.load_camera_devices()
    self.capture = None
    self.description = None
    self.database = None


    self.network_ready = False
    self.data_ready = False


    self.frame_counter = 0
    self.timestamp = time.time()
    self.last_symbol = -1
    self.last_timestamp = 0


  # Handlers for MainWindow events.
  def on_close(self, event):
    self.timer.Stop()


    if self.capture is not None:
      self.capture.release()
    if self.database is not None:
      self.database.close()
    for pr in network_processes:
      pr.terminate()

    quit()



  def capture_camera(self, event):
    obj = event.GetId()
    for item in self.camera_submenu.MenuItems:
      if item.GetId() == obj:
        name = item.GetLabel()
        break

    else:
      return
```

```
    device_index = self.camera_devices[name]
    self.capture = CaptureMethods.WebcamCapture(device_index)
    self.timer.Start(1000.0 / 30)
    self.data_ready = True


  def on_capture_video_file(self, event):
    with wx.FileDialog(self, "Оберіть відео файл", wildcard="Відео
файли|*.avi,*.mkv",
                style=wx.FD_OPEN | wx.FD_FILE_MUST_EXIST) as
fileDialog:
      if fileDialog.ShowModal() == wx.ID_CANCEL:
        return
      else:
        pathname = fileDialog.GetPath()
        self.capture = CaptureMethods.VideoFileCapture(pathname)
        self.timer.Start(1000.0 / self.capture.get_properties().fps)
        self.data_ready = True

  def on_capture_image(self, event):
    with wx.FileDialog(self, "Оберіть зображення", wildcard="Файли
зображень|*.jpg,*.jpeg,*.bmp,*.png", style=wx.FD_OPEN |
wx.FD_FILE_MUST_EXIST) as fileDialog:



      if fileDialog.ShowModal() == wx.ID_CANCEL:
        return
      else:
        pathname = fileDialog.GetPath()
        self.capture = CaptureMethods.ImageFileCapture(pathname)
        self.timer.Start(1000.0 / self.capture.get_properties().fps)
        self.data_ready = True


  def process_results(self, prediction_dict):
    array = prediction_dict['detection_scores']
    high_confidence = np.argmax(array)


    # TODO: remove this debug input

    print(str(array[high_confidence]), " for ",
prediction_dict['detection_classes'][high_confidence])
    print("Unprocessed images: ", image_queue.qsize())


    if array[high_confidence] >= self.settings.probability_threshold /
100.0:
      image_class =
prediction_dict['detection_classes'][high_confidence]


      # TODO: remove this debug input
      print("Detected class ", image_class, " with ",
array[high_confidence], " confidence")
      print("Unprocessed images: ", image_queue.qsize())


      if self.last_symbol != image_class:
        self.last_symbol = image_class
        self.last_timestamp = time.time()
      else:
        if time.time() - self.last_timestamp <
self.settings.repeat_frequency:
```

```
          return


      meanings = self.database.get_meanings(image_class)
      meanings_raw_text = [word[0] for word in meanings]


      self.meaning_table.AppendCols(1, False)
      self.meaning_table.SetCellEditor(0,
self.meaning_table.GetNumberCols()-1,
OutputControl.OutputControl(image_class, meanings_raw_text, True))

      self.meaning_table.SetCellValue(0,
self.meaning_table.GetNumberCols()-1, meanings_raw_text[0])



      if self.meaning_table.GetNumberCols() > 8:
        self.meaning_table.DeleteCols(0, 1, False)


      self.timestamp = time.time()
    else:
      if self.timestamp - time.time() > self.settings.clear_interval:
        self.meaning_table.DeleteCols(0,
self.meaning_table.GetNumberCols(), False)


  def show_capture(self, event):
    if self.capture is not None:
      self.image_panel.Refresh()
      if not results_queue.empty():
        prediction = results_queue.get(block=True, timeout=5)
        self.process_results(prediction)


      if self.data_ready and self.network_ready:
        if self.frame_counter == self.settings.skip_counter:
          ret, frame = self.capture.get_next_image()
          if ret:
            if not image_queue.full():
              print("putting image into queue, unprocessed: ",
image_queue.qsize())
              image_queue.put(frame)
          self.frame_counter = 0
        else:
          self.frame_counter += 1
    event.Skip()


  def on_paint(self, event):
    if self.capture is not None:
      dc = wx.BufferedPaintDC(self.image_panel)
      ret, frame = self.capture.get_next_image()
      if ret:
        frame = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
        properties = self.capture.get_properties()
        bmp = wx.BitmapFromBuffer(properties.width, properties.height,
frame)
        dc.DrawBitmap(bmp, 0, 0, True)
    event.Skip()
```

```
  def on_change_meaning(self, event):
    column = event.GetCol()
    row = event.GetRow()
    new_meaning = self.meaning_table.GetCellValue(row, column)
    if new_meaning == "":
      self.meaning_table.DeleteCols(column, 1, False)
    else:
      gesture_to_update = self.meaning_table.GetCellEditor(row,
column).index
      if new_meaning in self.meaning_table.GetCellEditor(row,
column).my_choices:
        self.database.increase_frequency(gesture_to_update,
new_meaning)
      else:
        self.database.add_meaning(gesture_to_update, new_meaning)



  def on_about_menu_selected(self, event):
    build_info =
ReadBuildInfo.get_build_info_from_xml("Resources//BuildInformation//In
fo.xml")
    msg = wx.MessageBox(str(build_info), u"Про програму")



  def on_manual_menu_selected(self, event):
    help_window = HelpFrame.HelpFrame(None)
    help_window.refresh()
    help_window.Show()  -

  def on_settings_menu_selected(self, event):
    settings_window = SettingsFrameImpl.SettingsForm(None)
    settings_window.Show()


  def load_gesture_system(self, event):
    global network_processes, image_queue, results_queue,
process_count


    obj = event.GetId()
    for item in self.system_submenu.MenuItems:
      if item.GetId() == obj:
        name = item.GetLabel()
        break
    else:
      return


    self.description =
NetworkDescription.get_network_description_from_xml("Resources//System
s//"+ name + "//Description.xml")
    self.database =
DatabaseConnector.DatabaseManager(self.description.database)
    image_queue = mp.Queue(maxsize=200)
    results_queue = mp.Queue(maxsize=200)



    self.status_bar.SetStatusText("Initializing network...")
    for i in range(1, process_count):
      process = mp.Process(target=mp_processing,
args=(self.description.model_graph, image_queue, results_queue),
```

```
            name="network_prc" + str(i))
  process.start()
  network_processes.append(process)
self.status_bar.SetStatusText("Done...\nWaiting for input")

self.network_ready = True
```

Додаток З

Тестування нейронної мережі

Таблиця 5.1

Перевірка мережі на зображеннях, що містять жести, що розпізнаються

Передумови	Вхідні дані	Очікуваний результат	Отриманий результат
Навчена нейронна мережа, граф операцій якої знаходиться за вказаним шляхом. Тестові зображення містять жести, що розпізнаються, але не використо-вувались в навчанні. Інші зображення не містять жестів, що розпізнаються	path = "model/frozen_inference_graph.pb" true_image_count = 50 true_image_path = "model/test_images/" 50 тестових зображень, що містять жести	*<50 результатів із високою (більше 0.5) імовірністю і класами 1, 2, … , 25 ,по два результати>*	0.99689794: 1 0.99543214: 1 0.9992799: 2 0.9996563: 2 0.999395: 3 0.9994842: 24 0.9839191: 4 0.9972065: 4 0.99819034: 5 0.99616915: 5 0.99956614: 6 0.99033934: 6 0.9997774: 7 0.99962366: 7 0.94497645: 8 0.98628634: 8 0.9935597: 9 0.95149064: 9 0.99204236: 10 0.9949126: 10 0.99630797: 11 0.9986053: 11 0.99666256: 12 0.98706985: 5 0.99631643: 13 0.99588376: 13 0.9805504: 14 0.97964877: 14 0.7047125: 15 0.99823004: 15 0.9738321: 16 0.9311324: 16 0.9861536: 17 0.9881952: 17 0.9994862: 18 0.9990602: 18 0.9963075: 19 0.9870372: 19 0.8904584: 20 0.58957378: 20 0.99995196: 21 0.99996626: 21 0.98308325: 22 0.99829763: 22 0.5685087: 23 0.5894744: 23 0.75597364: 24 0.9990841: 24 0.9911686: 25 0.9805712: 25

Таблиця 5.2

Перевірка мережі на зображеннях, що не містять жести, що розпізнаються

Передумови	Вхідні дані	Очікуваний результат	Отриманий результат
Навчена нейронна мережа, граф операцій якої знаходиться за вказаним шляхом. Тестові зображення містять жести, що розпізнаються, але не використовувались в навчанні. Інші зображення не містять жестів, що розпізнаються	path = "model/frozen_inference_graph.pb" false_image_count = 50 false_image_path = "model/false_images/" 50 тестових зображень, що не містять жестів	*<50 результатів із низькою (менше 0.5)>*	0.4380282: 8 0.24247324: 18 0.5083244: 23 0.5121578: 18 0.009836732: 19 0.09421866: 22 0.38944948: 24 0.15763412: 20 0.023486959: 1 0.052397486: 20 0.1906895: 24 0.007962229: 1 0.319444: 7 0.017883128: 20 0.11145924: 15 0.012775108: 21 0.06933077: 7 0.22955072: 15 0.019649675: 21 0.40603757: 3 0.15642892: 20 0.0289792: 20 0.29591167: 22 0.009243557: 3 0.13479201: 3 0.2880282: 8 0.41247324: 18 0.5083244: 23 0.5121578: 18 0.009836732: 19 0.09421866: 22 0.38944948: 24 0.15763412: 20 0.023486959: 1 0.052397486: 20 0.1906895: 24 0.007962229: 1 0.319444: 7 0.017883128: 20 0.11145924: 15 0.012775108: 21 0.06933077: 7 0.22955072: 15 0.019649675: 21 0.40603757: 3 0.15642892: 20 0.0289792: 20 0.29591167: 22 0.009243557: 3 0.13479201: 3

Таблиця 5.3

Перевірка швидкодії мережі

Передумови	Вхідні дані	Очікуваний результат	Отриманий результат
Навчена нейронна мережа, граф операцій якої знаходиться за вказаним шляхом. Тестові зображення містять жести, що розпізнаються, але не використо-вувались в навчанні. Інші зображення не містять жестів, що розпізнаються	path = "model/frozen_inference_graph.pb" true_image_count = 50 false_image_count = 50 true_image_path = "model/test_images/" false_image_path = "model/false_images/" 50 тестових зображень, що містять жести 50 тестових зображень, що не містять жестів	*<час роботи мережі на 100 зображеннях>*	26.72

Додаток 4

Копія презентації

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО”

ФАКУЛЬТЕТ ПРИКЛАДНОЇ МАТЕМАТИКИ

КАФЕДРА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ КОМП’ЮТЕРНИХ СИСТЕМ

АВТОМАТИЗОВАНА СИСТЕМА РОЗПІЗНАВАННЯ ЖЕСТОВОЇ МОВИ

Виконав: Пеня Олександр Романович

Науковий керівник: к.т.н., доц. Сулема Євгенія Станіславівна

Київ – 2019

ПОСТАНОВКА ЗАДАЧІ

Мета проекту: розробити програмну систему розпізнавання жестів у реальному часі.

Завдання:

- дослідити методи розпізнавання зображень;
- спроектувати архітектуру застосунку;
- розробити та протестувати програмні компоненти системи;
- розробити та провести інтеграційне тестування системи;
- оформити технічну документацію.

АКТУАЛЬНІСТЬ

Дана розробка є актуальною, тому що стосується прогресивних напрямків сучасної галузі інформаційних технологій:

- штучний інтелект;
- системи роботи в реальному часі;
- розпізнавання образів;
- альтернативні інтерфейси;
- управління проектами в умовах обмежених ресурсів.

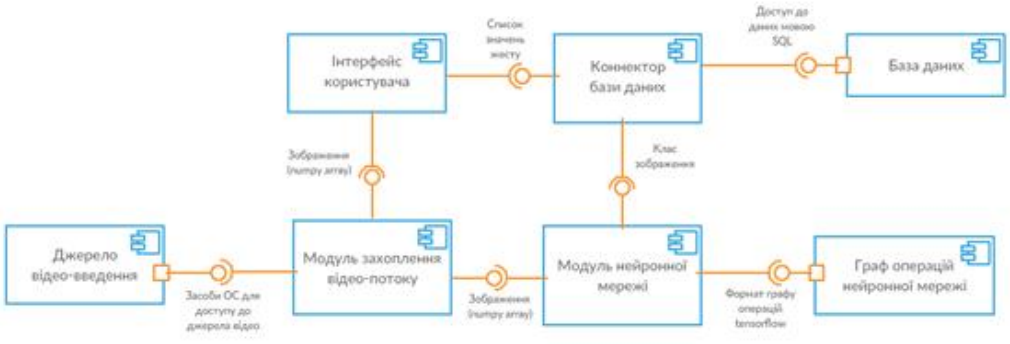
АНАЛІЗ ВИМОГ ДО РОЗРОБЛЮВАНОГО ПЗ

На основі поставленої задачі дипломного проектування було виявлено та проаналізовано вимоги до системи: функціональні, нефункціональні, апаратні, вимоги до розробки, визначено їх пріоритетність та трудоємність.

ЗАСОБИ РОЗРОБЛЕННЯ

- Платформа Microsoft Windows NT.
- Мова програмування Python 3.
- Бібліотека комп'ютерного зору OpenCV-python.
- СУБД SQLite.
- Бібліотека машинного навчання Tensorflow.
- Фреймворк побудови віконного інтерфейсу wxPython.

АРХІТЕКТУРА СИСТЕМИ



КОМПОНЕНТИ СИСТЕМИ: ЗАХОПЛЕННЯ ВІДЕО-ПОТОКУ

Модуль захоплення відео надає системі можливість отримання введення із відповідного периферійного пристрою або локального файлу.

Для джерела потоку передбачено абстрактний інтерфейс, реалізації якого надають системі можливості введення відео.

Отримання відео використовується за допомогою OpenCV, яка надає зображення у форматі, сумісному з іншими компонентами системи.

КОМПОНЕНТИ СИСТЕМИ: ІНТЕРФЕЙС КОРИСТУВАЧА

Для системи розроблено віконний інтерфейс, який надає користувачу можливість взаємодії з системою у звичний спосіб.

Інтерфейс розділено на дизайн (абстрактні класи, які будують вікна та визначають події, на які реагує система) та безпосередню обробку (класи, що реалізують обробники подій).

Інтерфейс побудовано за допомогою wxPython – кросплатформної обгортки над віджетами ОС, які мають вигляд відповідний до платформи виконання.

КОМПОНЕНТИ СИСТЕМИ: БАЗА ДАНИХ

БД зберігає значення жестів, які розпізнаються системою.

Для бази даних створено просту, але гнучку схему даних, що дозволяє поступово ускладнювати можливості системи.

Для управління базою даних обрано вбудовану СУБД SQLite для того, щоб зменшити навантаження на систему.

КОМПОНЕНТИ СИСТЕМИ: НЕЙРОННА МЕРЕЖА

У ході роботи досліджено принципи функціонування та навчання нейронних мереж, проаналізовано сучасні архітектури глибоких мереж, які використовуються для розпізнавання образів, обрано та реалізовано мережу SSD-Mobilenet v1 засобами бібліотеки Tensorflow, зібрано та підготовано дані для навчання мережі (фото жестів української дактильної абетки), навчено та протестовано створену мережу.

ОСОБЛИВОСТІ РЕАЛІЗАЦІЇ СИСТЕМИ

Систему розроблено з такими особливостями:

- застосування паралельних обчислень;
- розширення системи за рахунок змістового наповнення;
- переносимість системи.

КРИТЕРІЇ ЯКОСТІ СИСТЕМИ

Для оцінки якості системи обрано наступні критерії:

- Відповідність технічній специфікації
- Швидкодія
- Переносимість
- Точність
- Наявність документації на систему

АНАЛІЗ РЕЗУЛЬТАТІВ ТЕСТУВАННЯ

Було розроблено програму тестування системи, за якою здійснено модульне та інтеграційне тестування системи.

За результатами тестування було встановлено, що система відповідає очікуваній поведінці та проведено її оцінку за виділеними критеріями якості.

ВИСНОВКИ

У ході даної роботи:

1. Визначено вимоги до розроблюваної системи.
2. Проаналізовано та обрано засоби розробки.
3. Спроектовано архітектуру і сплановано розробку програмної системи.
4. Створено та протестовано визначені компоненти системи.
5. Розроблено та протестовано ПЗ, яке відповідає заявленим вимогам та критеріям якості.

УНІКАЛЬНІСТЬ

Розділ	Відсоток унікальності
Вступ	100 %
Перший розділ	96 %
Другий розділ	95 %
Третій розділ	97 %
Четвертий розділ	99 %
Висновки	95 %
Весь текст	94 %

Дякую за увагу!

Факультет прикладної математики

Кафедра програмного забезпечення комп'ютерних систем

"ЗАТВЕРДЖЕНО"

Завідувач кафедри

_____ І. А. Дичка

"___" _____ 2018 р.

АВТОМАТИЗОВАНА СИСТЕМА РОЗПІЗНАВАННЯ ЖЕСТОВОЇ МОВИ

Програма та методика тестування

ДП.045480-04-51

"ПОГОДЖЕНО"

Керівник проекту:

_____ Є. С. Сулема

Нормоконтроль:

_____ М. В. Онай

Виконавець:

_____ О. Р. Пеня

2018

ЗМІСТ

1. ОБ’ЄКТ ВИПРОБУВАНЬ

Програмне забезпечення автоматичного розпізнавання жестів з відео-потоку.

2. МЕТА ТЕСТУВАННЯ

У процесі тестування має бути перевірено наступне:

1. Синтаксична правильність коду.
2. Правильне функціонування компонентів системи.
3. Відповідність системи сформульованим вимогам.
4. Стабільність і реакція на помилки.

3. МЕТОДИ ТЕСТУВАННЯ

Тестування проводиться в два етапи: модульне тестування, яке націлене на перевірку правильності роботи окремих програмних компонентів системи та інтеграційне тестування, націлене на перевірку правильності роботи системи в цілому та її відповідність заявленим вимогам.

Модульне тестування проводиться напівавтоматичним способом за методом «білого ящика», інтеграційне тестування проводиться методом «чорного ящика» за сценарієм поведінки користувача в ручному режимі.

4. ЗАСОБИ ТА ПОРЯДОК ТЕСТУВАННЯ

Тестування системи проводиться в два етапи: модульне тестування для перевірки коректності функціонування компонентів та інтеграційне тестування для перевірка правильності роботи системи та відповідності до заявлених вимог. Модульне тестування проводиться в напівавтоматичному

режимі за допомогою тестових сценаріїв (скриптів), інтеграційне тестування – у ручному режимі за наведеним сценарієм дій користувача.

4.1. Модульне тестування

4.1.1. *Тестування допоміжних компонентів*

1. Перевірка розширення для пошуку пристроїв відео введення.

Тестування проводиться за допомогою скрипта:

```
import device

device_list = device.getDeviceList()
print(len(device_list))
print(device_list)
```

Таблиця 1

Сценарій тестування розширення пошуку пристроїв

Передумови	Вхідні дані	Очікуваний результат
Додаткові пристрої введення не підключені	–	1 [*назва вбудованого пристрою*]
Підключено USB-вебкамеру	–	2 [*список назв пристроїв*]

2. Зчитування даних про жестову систему з XML-файлу.

Тестування проводиться за допомогою скрипта:

```
import NetworkDescription

path = """D:\diplom\Test_int\example.xml"""
desc = NetworkDescription.get_network_description_from_xml(path)

output = "\n".join((desc.model_name, desc.model_graph, desc.database,
desc.description))
print(output)
```

Таблиця 2

Сценарій тестування зчитування даних про жестову систему

Передумови	Вхідні дані	Очікуваний результат
Наявний файл example.xml відповідного формату	path = "D:\diplom\Test_int\example.xml"	model Resources/Systems/DactileUA/frozen_inference_graph.pb Resources/Systems/DactileUA/DactileGesture.db Українська дактильна абетка

3. Зчитування даних про програмну систему з XML-файлу.

Тестування проводиться за допомогою скрипта:

```
import ReadBuildInfo

path = "Info.xml"
desc = ReadBuildInfo.get_build_info_from_xml(path)

output = desc.__str__()
print(output)
```

Таблиця 3

Сценарій тестування зчитування даних про програму

Передумови	Вхідні дані	Очікуваний результат
Наявний файл Info.xml відповідного формату	path = ".\Info.xml"	Gesture recognition system v 1.0.0 build date: 02.06.2019

4.1.2. *Тестування захоплення відео-потоку*

1. Отримання відео з камери.

Тестування проводиться за допомогою скрипта:

```
import cv2
from Capture import CaptureMethods

cap = CaptureMethods.WebcamCapture(cam_id=0)
```

```
while True:
    (ret, frame) = cap.get_next_image()

    if ret:
        cv2.imshow('frame', frame)
        if cv2.waitKey(1) & 0xFF == ord('q'):
            break
    else:
        break

cap.release()
cv2.destroyAllWindows()
```

Таблиця 4

Сценарій тестування отримання відео з камери

Передумови	Вхідні дані	Очікуваний результат
Доступний хоча б один пристрій відео введення	cam_id=0	Вікно, у якому відображається відео з камери

2. Отримання відео з файлу.

Тестовий скрипт подібний до попереднього пункту, змінюється тільки джерело введення:

```
cap = CaptureMethods.VideoFileCapture(filename="Video.avi")
```

Таблиця 5

Сценарій тестування отримання відео з файлу

Передумови	Вхідні дані	Очікуваний результат
Наявний файл Video.avi	filename="Video.avi"	Вікно, у якому відображається відео з файлу

4.1.3. *Тестування бази даних*

1. Тестування запитів

У табл. 6 наведено запити, що відлагоджуються та відповідні очікувані результати.

Таблиця 6

Сценарій тестування запитів до БД

<table>
<tr><th>Передумови</th><th>Вхідні дані</th><th>Очікуваний результат</th></tr>
<tr><td>Створена база даних відповідної структури, заповнена даними</td><td>%id = 1
SELECT m.Meaning, m.Frequency FROM
(Meaning INNER JOIN Gesture On Gesture.Id = Meaning.GestureId) AS m
WHERE m.Id = %id
ORDER BY m.Frequency DESC</td><td>Таблиця формату
(один або більше рядків)
<table><tr><td>m.Meaning</td><td>m.Frequency</td></tr><tr><td>…</td><td>…</td></tr></table></td></tr>
<tr><td rowspan="3"></td><td>%id = 1
SELECT g.Id, g.Type FROM Gesture g WHERE g.Id = %id</td><td>Таблиця формату
(один рядок)
<table><tr><td>g.Id</td><td>g.Type</td></tr><tr><td>…</td><td>…</td></tr></table></td></tr>
<tr><td>%id=1, %mean=mean, %frq=1
INSERT INTO Meaning (GestureId, Meaning, Frequency) VALUES (%id, `%mean`, %frq)</td><td>Вставка одного рядка в таблицю Meaning</td></tr>
<tr><td>%id=1, %mean=mean, %dfrq=1
UPDATE Meaning SET Frequency = Frequency + %dfrq WHERE GestureId = %id and Meaning = '%mean'</td><td>Оновлення одного рядка таблиці Meaning</td></tr>
</table>

2. Тестування програмного компонентну бази даних.

Тестування проводиться за допомогою скрипта:

```
from Database import DatabaseConnector, GestureObject

path = "DactileGesture.db"
gesture_id = 1
new_meaning = "sdg"
init_frequency = 3
added_frequency = 39

db = DatabaseConnector.DatabaseManager(path)

print(db.get_meanings(gesture_id))
print(db.get_object(gesture_id))
db.add_meaning(gesture_id, new_meaning, init_frequency)
print(db.get_meanings(gesture_id))
db.increase_frequency(gesture_id, new_meaning, added_frequency)
print(db.get_meanings(gesture_id))
db.close()
```

Таблиця 7

Сценарій тестування компоненту доступу до БД

Передумови	Вхідні дані	Очікуваний результат
Створена база даних відповідної структури, заповнена даними. Файл даної БД знаходиться за заданим шляхом	path = "DactileGesture.db" gesture_id = 1 new_meaning = "sdg" init_frequency = 3 added_frequency = 39	[('A', 1)] 1 0 [('A', 1)] [('sdg', 3), ('A', 1)] [('sdg', 42), ('A', 1)]

4.1.4. *Тестування нейронної мережі*

Тестування проводиться за допомогою скрипта:

```
from NeuralNetwork import ImportNetwork
import cv2
import time
import numpy as np

path = """model/frozen_inference_graph.pb"""
true_image_count = 50
false_image_count = 50
true_image_path = """model/test_images/"""
false_image_path = """model/false_images/"""

neural_network = ImportNetwork.Network()
neural_network.load(path)
```

```
t_images = []
f_images = []

for i in range(true_image_count + 1):
    t_images.append(cv2.imread(true_image_path+str(i) + ".jpg"))

for i in range(false_image_count + 1):
    f_images.append(cv2.imread(false_image_path + str(i) + ".jpg"))

start = time.time()
for img in t_images:
    result = neural_network.predict(img)
    high_confidence = np.argmax(result['detection_scores'])
    img_class = result['detection_classes'][high_confidence]
    print(str(high_confidence) + ": " + str(img_class))

for img in f_images:
    result = neural_network.predict(img)
    high_confidence = np.argmax(result['detection_scores'])
    img_class = result['detection_classes'][high_confidence]
    print(str(high_confidence) + ": " + str(img_class))
end = time.time()
print(end-start)
```

Таблиця 8

Сценарій тестування нейронної мережі

Передумови	Вхідні дані	Очікуваний результат
Навчена нейронна мережа, граф операцій якої знаходиться за вказаним шляхом. Тестові зображення містять жести, що розпізнаються, але не використовувались в навчанні. Інші зображення не містять жестів, що розпізнаються	path = "model/frozen_inference_graph.pb" true_image_count = 50 false_image_count = 50 true_image_path = "model/test_images/" false_image_path = "model/false_images/" 50 тестових зображень, що містять жести 50 тестових зображень, що не містять жестів	*<50 результатів із високою (більше 0.5) імовірністю і класами 1, 2, … , 25 ,по два результати на клас>* *<50 результатів із низькою (менше 0.5) імовірністю >* *<час роботи мережі на 100 зображеннях>*

4.1.5. *Тестування інтерфейсу*

Тестування інтерфейсу, як окремого елементу дає лише можливість переконатись в тому, що наявні всі необхідні вікна інтерфейсу та елементи керування на цих вікнах. Безпосередню функціональність інтерфейсу забезпечують реалізації абстракцій інтерфейсу, які реалізують поведінку в відповідності до дій користувача.

Тестування проводиться за допомогою скриптів наступного вигляду, у яких об’єкт frame замінюється на клас відповідного вікна інтерфейсу:

```
import wx
from UI import ProjectMainFrame

app = wx.App()
frame = ProjectMainFrame.ProjectMainFrame(None)
frame.Show()
app.MainLoop()
```

Таблиця 9

Сценарій тестування інтерфейсу

Передумови	Вхідні дані	Очікуваний результат
–	frame = ProjectMainFrame(None)	*<головне вікно системи>*
	frame = SettingsFrameImpl(None)	*<вікно налаштувань системи>*
	frame = HelpFrame(None)	*<вікно керівництва користувача>*

4.2. Інтеграційне тестування

Інтеграційне тестування проводиться за відповідним сценарієм, наведеним у табл. 10.

Таблиця 10

Сценарій інтеграційного тестування

№ з/п	Дія користувача	Очікувана реакція системи
1	Запуск програми на виконання	Відкривається головне вікно системи. У інтерфейс динамічно завантажуються доступні пристрої відео-введення та наявні жестові системи. Встановлюються налаштування за замовчанням.
2	Перехід у меню «Допомога», пункт «Про програму»	Відображається повідомлення з даними про систему, версією, датою створення. Головне вікно стає неактивним
3	Натиснути «ОК»	Повідомлення закривається, головне вікно стає активним
4	Перехід у меню «Допомога», пункт «Довідка»	Відкривається вікно з керівництвом користувача
5	Закрити вікно довідки	Вікно з керівництвом користувача закривається, головне вікно стає активним
6	Перехід у меню «Налаштування», пункт «Налаштування системи…»	Відкривається вікно налаштувань
7	Змінити налаштування довільним чином	Значення параметрів у вікні змінюються відповідним чином
8	Натиснути «Назад»	Вікно налаштувань закривається, параметри системи не зміню-ються

Продовження табл. 10

9	Перехід у меню «Налаштування», пункт «Налаштування системи…»	Відкривається вікно налаштувань
10	Змінити налаштування довільним чином	Значення параметрів у вікні змінюються відповідним чином
11	Натиснути «ОК»	Вікно налаштувань закривається, параметри системи змінюються відповідно до встановлених значень
12	Перехід у меню «Спосіб введення», пункт «Відео з камери», підпункт «HP TrueVision HD»	Система захоплює відео-потік з вбудованої камери пристрою, відео відображається у головному вікні
13	Перехід у меню «Налаштування», пункт «Жестові системи», підпункт «Українська дактильна абетка»	Система завантажує граф операцій нейронної мережі, запускає окремі процеси обробки зображень, починає обробку відео-потоку
14	Показати послідовність введення, наприклад “А”, “Б”, “В”, “Г”, “Д”	Система проводить розпізнавання жестів, можливо з певною затримкою або прийнятною кількістю помилок. Отримані значення виводяться на екран
15	Натиснути на одне із отриманих значень	Відображається поле редагування значення
16	Ввести інше значення та натиснути Enter	Нове значення відображається на екрані і вноситься до бази даних
17	Ввести жест із новим значенням ще раз	Отримане значення виводяться на екран
18	Натиснути на одне із отриманих значень	Відображається поле редагування значення
19	Обрати випадний список значень	Відображається список збережених значень, серед яких нове введене значення

Продовження табл. 10

20	Обрати нове значення та натиснути Enter	Нове значення відображається на екрані, частота його використання збільшується у базі даних
21	Почекати певний час, не вводячи жестів	Робота системи продовжується, через встановлений час розпізнані значення очищуються
22	Закрити головне вікно програми	Головне вікно програми закривається. Система коректно звільняє задіяні ресурси: закриває підключення до бази даних, знищує контекст нейронної мережі в пам’яті, звільняє пристрій відео-введення, закриває процеси обробки зображень. Робота системи припиняється.

Описаний сценарій дій покриває більшу частину функціональності системи, достатню для отримання висновків щодо якості роботи системи та відсутності помилок внутрішньої логіки, сумісності інтерфейсів компонентів, тощо.

Факультет прикладної математики

Кафедра програмного забезпечення комп'ютерних систем

"ЗАТВЕРДЖЕНО"

Завідувач кафедри

_____ І. А. Дичка

"___" _____ 2019 р.

АВТОМАТИЗОВАНА СИСТЕМА РОЗПІЗНАВАННЯ ЖЕСТІВ

Керівництво користувача

ДП.045480-05-34

"ПОГОДЖЕНО"

Керівник проекту:

_____ Є. С. Сулема

Нормоконтроль:

_____ М. В. Онай

Виконавець:

_____ О. Р. Пеня

2019

ЗМІСТ

Призначення і функціональність програми

Система призначена для розпізнавання жестів у режимі реального часу із потокового відео з веб-камери пристрою або відео файлу. Вона обробляє кадри відео та виводить у текстовому форматі значення розпізнаних жестів, які можна відредагувати після цього вручну і призначена для інтерпретації жестової мови.

Початок роботи

Для роботи з системою запустіть її за допомогою командного файлу Start.bat. Після завантаження має відкритись головне вікно програми (рис. 1).

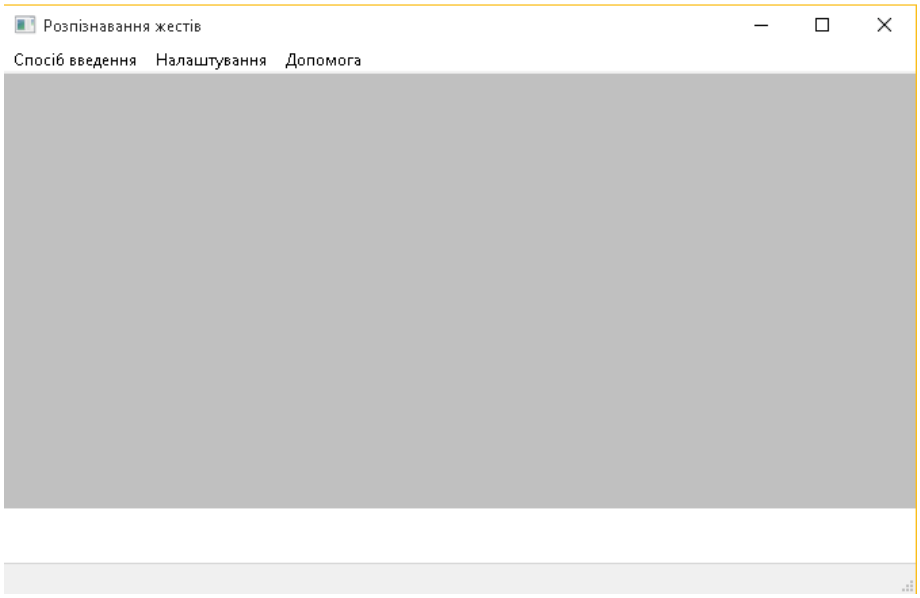


Рис. 1. Головне вікно програми

Для початку роботи необхідно обрати джерело введення відео, жестову систему та параметри роботи.

Вибір джерела введення

Для того, щоб обрати джерело відео введення оберіть пункт меню «Спосіб введення», натиснувши на нього лівою кнопкою миші. Оберіть бажане джерело введення. Підменю «Відео з камери» містить периферійні пристрої введення, доступні для роботи (рис. 2).

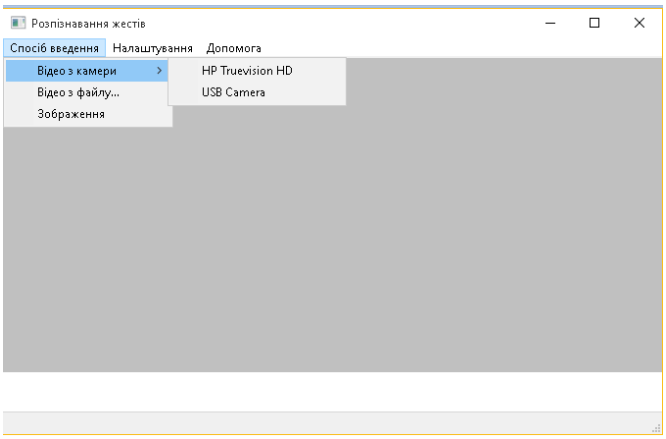


Рис. 2. Доступні системі веб-камери

При виборі однієї з камер система підключиться до неї та почне зчитування відео, яке відображатиметься на екрані (рис. 3).

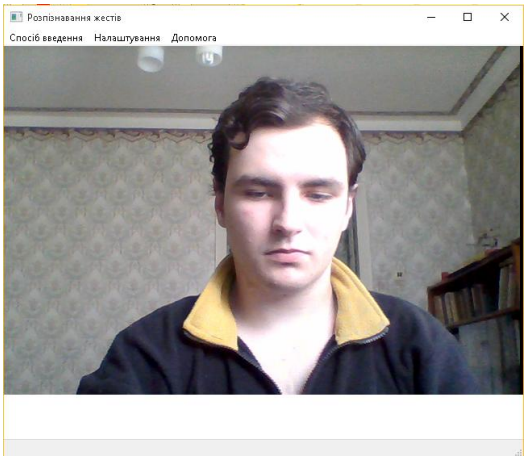


Рис. 3. Відео з камери

Обравши підпункт «Відео з файлу…», оберіть необхідний файл у файловій системі (рис. 4) і програма почне зчитування з нього.

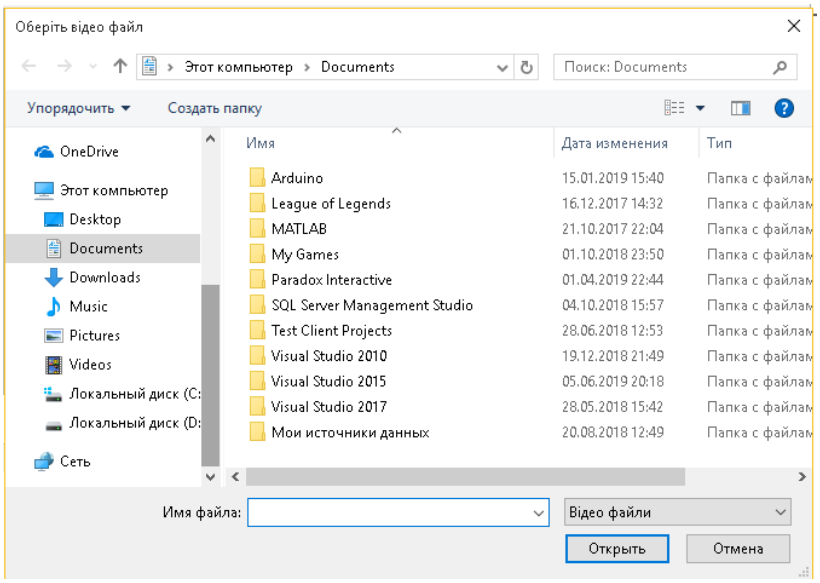


Рис. 4. Вибір відео-файлу

Вибір жестової системи

Жестова система – це набір жестів, які розпізнаватиме система.

Для вибору жестової системи оберіть пункт меню «Налаштування», підпункт «Жестова система» (рис. 5) та оберіть зі списку доступних систем бажану. Після цього почнеться завантаження системи, яке може займати певний час, особливо на менш потужних комп’ютерах, тому бажано, але не обов’язково обрати жестову систему перед джерелом відео.

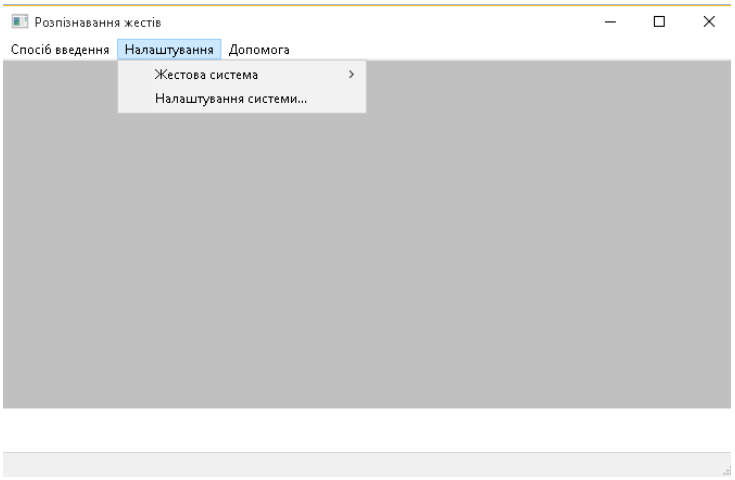


Рис. 5. Вибір жестової системи

Тепер система готова до розпізнавання і виводитиме виявлені жести у поле під відео.

Параметри системи

Параметри містять набір налаштувань, які дозволяють покращити роботу системи. На відміну від джерела відео та жестової системи, параметри містять певні значення за замовчанням, тому їх не обов'язково вказувати або змінювати для початку роботи.

Змінити параметри можна обравши меню «Налаштування», підпункт «Налаштування системи…». У вікні параметрів (рис. 6) можна вказати бажані значення параметрів.

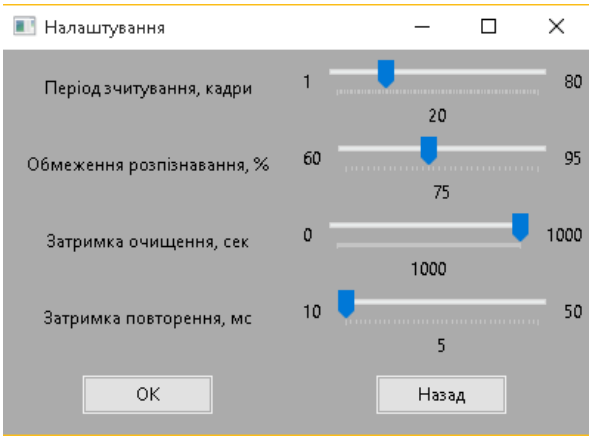


Рис. 6. Налаштування системи

Зміст цих значень полягає у наступному:

- Період зчитування. Розпізнавання займає певний час, тому для пришвидшення роботи система пропускає частину кадрів. Що більше це значення, то більше кадрів буде пропущено, при цьому зменшується час затримки системи для розпізнавання, але зменшується точність і, можливо, введенні жести буде потрібно затримувати на певний час або показувати повторно. Збільшуйте це значення, якщо затримка на розпізнавання жесту на Вашому комп'ютері велика.
- Обмеження розпізнавання. Цей параметр визначає точність роботи системи. Чим його значення більше, тим менше система реагує на неточні результати розпізнавання. Збільшуйте цей параметр, якщо система хибно реагує та жести і виводить ті, які Ви не показували і зменшуйте, якщо вона погано реагує на жести, що Ви показуєте.
- Затримка очищення. Якщо система певний час не визначає нових жестів, то для зручності попередні розпізнані жести будуть очищені. Час, через який значення очищуються регулюється цим параметром в секундах.
- Затримка повторення. При визначенні одного і того ж жесту декілька разів система не виводить однакові значення одразу, подібного до того, як при утриманні однієї кнопки клавіатури символи повторюються не одразу. Цей параметр регулює затримку між повторним виведенням одного і того ж значення.

Для збереження змін у параметрах натисніть «ОК», для того, щоб повернутись до головного вікна без збереження змін натисніть «Назад» або просто закрийте вікно налаштувань.

Довідка

Пункт меню «Допомога» призначений для отримання довідки за програмою. Підпункт «Про програму» відображає дані про дану програму, а підпункт «Керівництво користувача» відобразить дану інструкцію у окремому вікні.