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

Факультет інформатики та обчислювальної техніки

Кафедра інформаційних систем та технологій

До захисту допущено:

Завідувач кафедри

_____ Олександр РОЛІК

«___»_____20__ р.

Дипломний проєкт

на здобуття ступеня бакалавра
за освітньо-професійною програмою «Інформаційні управляючі системи та технології»
спеціальності 126 «Інформаційні системи та технології»
на тему: «Система перетворення мови жестів у текст на основі штучного інтелекту»

Виконала:
студентка IV курсу, групи ІС-92
Мельницька Віталія Богданівна _____

Керівник:
Доцент кафедри ІСТ, к.т.н., доцент
Писаренко Андрій Володимирович _____

Рецензент:
доцент кафедри ІПІ, к.т.н., доцент
Ліщук Катерина Ігорівна _____

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

Київ – 2023 року

Національний технічний університет України

«Київський політехнічний інститут імені Ігоря Сікорського»

Факультет інформатики та обчислювальної техніки

Кафедра інформаційних систем та технологій

Рівень вищої освіти – перший (бакалаврський)

Спеціальність – 126 «Інформаційні системи та технології»

Освітньо-професійна програма «Інформаційні управляючі системи та технології»

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Олександр РОЛІК
«___»_____20__ р.

ЗАВДАННЯ
на дипломний проєкт студентці
Мельницькій Віталії Богданівній

1. Тема проєкту «Система перетворення мови жестів у текст на основі штучного інтелекту», керівник проєкту Писаренко Андрій Володимирович, доцент кафедри ІСТ, к.т.н., затверджені наказом по університету від «31» травня 2023 р. № 2101-с

2. Термін подання студентом проєкту: 12 червня 2023р.

3. Вихідні дані до проєкту: мова програмування Python, штучні нейронні мережі, комп’ютерний зір, розпізнавання мови жестів до 100%.

4. Зміст пояснювальної записки:

Аналіз наявних систем перетворення мови жестів у текст.

Проектування системи перетворення мови жестів у текст на основі штучного інтелекту, побудова структурної схеми та діаграм послідовності.

Вибір нейронної мережі для системи перетворення мови жестів у текст, реалізація програми на мові Python.

Описання результатів навчання та тестування нейронної мережі.

Висновки до пояснювальної записки.

5. Перелік графічного матеріалу (із зазначенням обов'язкових креслеників, плакатів, презентацій тощо):

Структурна схема системи, діаграма послідовностей авторизації користувача, діаграма послідовностей процесу розпізнавання, скріншоти результатів перетворення мови жестів у текст.

6. Дата видачі завдання 13.02.2023

Календарний план

№ з/п	Назва етапів виконання дипломного проєкту	Термін виконання етапів проєкту	Примітка
1	Аналіз та уточнення завдання дипломного проекту	24.04.2023	
2	Аналіз предметної області	26.04.2023	
3	Проектування системи	01.05.2023	
4	Реалізація програмного забезпечення	04.05.2023	
5	Опис програмного забезпечення	08.05.2023	
6	Оформлення звіту дипломного проекту	17.05.2023	
7	Подання ДП на попередній захист	05.06.2023	
8	Подання ДП на основний захист	12.06.2023	
9	Подання ДП рецензенту	13.06.2023	

Студент Віталія МЕЛЬНИЦЬКА

Керівник Андрій ПИСАРЕНКО

АНОТАЦІЯ

Мельницька В.Б. Система перетворення мови жестів у текст на основі штучного інтелекту. КПІ ім. Ігоря Сікорського, Київ, 2023.

Проект містить 60 с. тексту, 29 рисунків, 3 схеми, 2 графіки, посилання на 31 літературні джерела.

Ключові слова: мова жестів, нейронна мережа, штучний інтелект, набір даних, SL-MNIST.

Об’єктом є система перетворення мови жестів у текст на основі штучного інтелекту.

Мета розробки – покращити спілкування між людьми з вадами мови та слуху за допомогою системи перетворення мови жестів у текст на основі штучного інтелекту.

У дипломному проєкті виконане проектування системи перетворення мови жестів у текст на основі штучного інтелекту, та розроблена програма на основі нейронної мережі. Проведено ретельний аналіз та вибір методу розробки. Значну увагу було приділено вибору нейронної мережі для зчитування та перетворення мови жестів у текст.

ABSTRACT

Melnytska V.B. A system for converting sign language into text based on artificial intelligence. National Technical University of Ukraine “Igor Sikorsky Kyiv Polytechnic Institute”, Kyiv, 2023.

The project contains 60 p. text, 29 drawings, 3 diagrams, 2 graphs, references to 31 literary sources.

Keywords: sign language, neural network, artificial intelligence, dataset, SL-MNIST.

The object is a system of converting sign language into text based on artificial intelligence.

The purpose of development is to improve communication between people.

In the diploma project, the design of the system for converting sign language into text based on artificial intelligence was performed, and a program based on a neural network was developed. A thorough analysis and selection of the development method was carried out. Considerable attention was paid to the selection of a neural network for reading and converting sign language into text.

Номер рядка	Формат	Позначення	Найменування	Кільк. аркушів	Номер екзем.	Примітка
1			Документація загальна			
2						
3			Знову розроблена			
4						
5	А4	ІС92.170БАК.004 ПЗ	Пояснювальна записка	65		
6	А3	ІС92.170БАК.004 Д1	Система перетворення мови			
7			жестів у текст на основі			
8			штучного інтелекту.			
9			Структурна схема системи			
10	А3	ІС92.170БАК.004 Д2	Система перетворення мови	1		
11			жестів у текст на основі			
12			штучного інтелекту. Діаграма			
13			послідовностей авторизації			
14			користувача			
15	А3	ІС92.170БАК.004 Д3	Система перетворення мови	1		
16			жестів у текст на основі			
17			штучного інтелекту. Діаграма			
18			послідовностей процесу			
19			розпізнавання жестів			
20	А3	ІС92.170БАК.004 Д4	Система перетворення мови	1		
21			жестів у текст на основі			
22			штучного інтелекту. Графіки			
23						
24						
25						
26						
27						
28						

Зм.	Аркуш	№ докум.	Підпис	Дата	ІС92.170БАК.004 ТП		
Розроб.		Мельницька В.Б.			Система перетворення мови жестів у текст на основі штучного інтелекту. Відомість проєкту	Літ.	Аркуш 1 / Аркушів 1
Керівн.		Писаренко А.В.				Т	
						КПІ ім. Ігоря Сікорського Група ІС-92	
Затв.							

Пояснювальна записка
до дипломного проєкту
на тему: «Система перетворення мови жестів у текст на основі штучного інтелекту»

Київ – 2023 року

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАЧЕНЬ

CNN – Convolutional Neural Network (згорткова нейронна мережа)

КЗ – комп’ютерний зір

ПЗ – програмне забезпечення

SL-MNIST – Sign Language MNIST (набір даних)

ASL – American Sign Language (американська жестова мова)

ШІ – штучний інтелект

ВСТУП

У сучасному швидкоплинному світі комунікація між людьми має вирішальне значення. Мова жестів поширена у світі, але все одно під час розмови можуть створюватись бар’єри. На сьогоднішній день понад 5% населення світу стикаються з проблемами втрати слуху. За прогнозами, до 2050 року ця проблема торкнеться майже 2,5 мільярда людей у різній мірі, а щонайменше 700 мільйонів осіб будуть потребувати реабілітаційних послуг у зв'язку з втратою слуху. Більш ніж 70 мільйонів людей по всьому світу активно використовують різні мови жестів.

Штучний інтелект є потужним інструментом для вирішення цього комунікаційного бар’єру між людьми із вадами слуху та мови. Використовуючи можливості штучного інтелекту можна розробляти інноваційні рішення для перетворення мови жестів у текст.

Тому, головна ідея цього диплому – є покращити спілкування між людьми з вадами мови та слуху за допомогою системи перетворення мови жестів у текст на основі штучного інтелекту. Робота спрямована на вирішення проблеми спілкування мовою жестів за допомогою штучного інтелекту. Завдяки машинному навчанню дана система зможе навчатись і в подальшому точно перекладати мову жестів користувачів у текст у режимі реального часу.

У цій роботі розглядатимуться проектування системи та програмна реалізація, щоб дослідити та показати як працюватиме штучний інтелект у розробленій системі. Розроблено програмне забезпечення, у якому користувачі можуть на камеру показувати жести, які зчитуватимуться програмою та оброблятимуться. Потім за допомогою штучного інтелекту оброблені дані система перекладає на текст і повертатиме користувачу готове слово чи речення.

Також проведено тестування роботи системи на штучно створених наборах даних, які складаються із тисячі зображень жестів, для дослідження та порівняння.

Завданням цієї роботи також є проведення аналізу аналогічних рішень для такої системи та аналіз їхніх переваг та недоліків.

Практична цінність роботи – створення інструменту спілкування між людьми, які мають обмеження в цьому, та покращення їхньої соціалізації. Використання штучного інтелекту у системі – це можливість створити більше соціалізоване суспільство, зводячи розмовні бар'єри до мінімуму, і можливість реалізації потенціалу кожної людини, незалежно від її вад слуху та мови.

1 АНАЛІЗ НАЯВНИХ СИСТЕМ ПЕРЕТВОРЕННЯ МОВИ ЖЕСТІВ У ТЕКСТ

1.1 Огляд наявних методів та технологій

Системи перетворення мови жестів у текст є важливими програмними засобами, які дозволяють людям з обмеженими можливостями спілкуватися з іншими людьми та отримувати доступ до інформації. Жестова мова є важливим способом комунікації для людей, які мають проблеми з мовленням або слухом, а також для тих, хто використовує мову жестів як другу мову.

Наявні системи перетворення мови жестів у текст дозволяють перетворювати жестову мову на письмовий текст, що дозволяє людям з обмеженими можливостями спілкуватися з оточуючими без необхідності використовувати додаткові пристрої чи людей-перекладачів. Окрім того, такі системи дозволяють використовувати жестову мову в інтернеті, в електронній пошті та в інших сферах життя, де необхідний письмовий спосіб комунікації.

Зараз на ринку існує багато різних систем перетворення мови жестів у текст, які відрізняються за своїми функціональними можливостями, швидкістю роботи та точністю розпізнавання жестів.

1.1.1 MotionSavvy UNI

MotionSavvy UNI – це інтерактивний пристрій (рисунок 1.1), який розроблений для допомоги людям зі слуховими проблемами, щоб спілкуватися з іншими людьми за допомогою мови жестів. Цей пристрій використовує технологію розпізнавання рухів, щоб перекладати мову жестів на мову звуків і навпаки [1].

MotionSavvy UNI складається з двох частин: камери та планшета. Камера встановлюється на верхній частині планшета і використовується для розпізнавання рухів жестів користувача. Планшет відображає переклад мови

жестів на мову звуків, що допомагає людям зі слуховими проблемами спілкуватися з оточуючими.

MotionSavvy UNI може перекладати більше 2000 різних жестів і має можливість редагування перекладу, щоб користувачі могли легко коригувати будь-які помилки. Крім того, пристрій може навчатися новим жестам, що робить його корисним для людей, які використовують унікальні мови жестів.

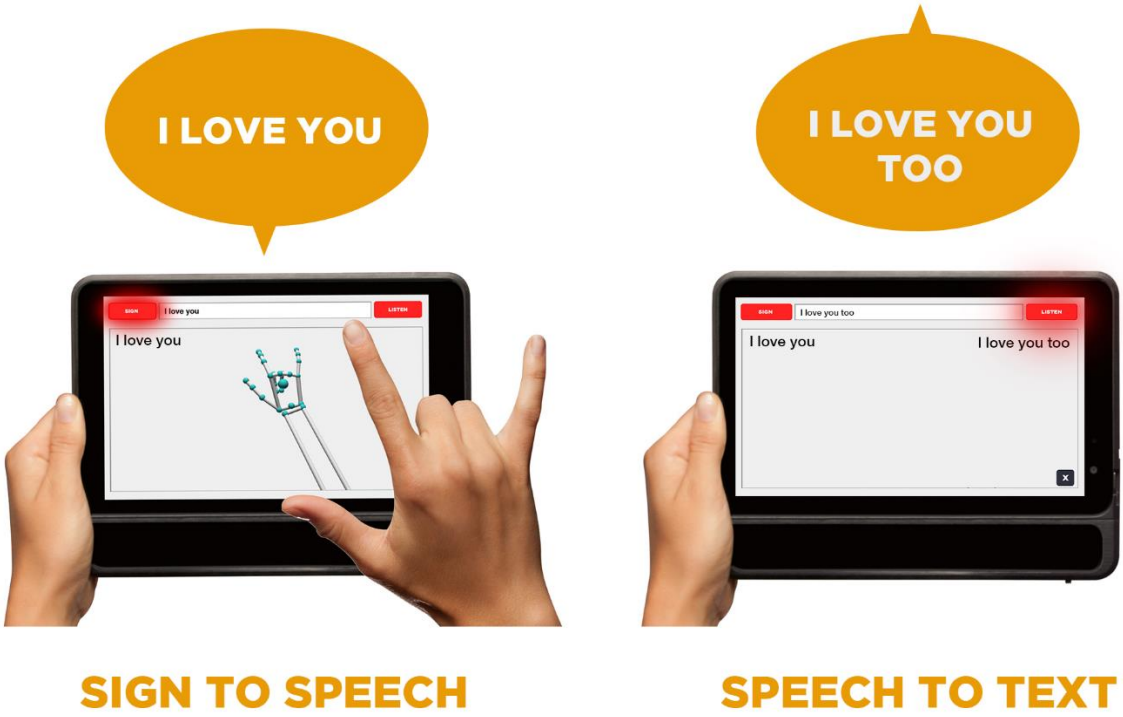


Рисунок 1.1 – Ілюстрація користування застосунком MotionSavvy UNI [2]

Окрім того, MotionSavvy UNI має можливість підключення до Інтернету, що дозволяє користувачам отримувати доступ до онлайн-ресурсів та інформації, а також надсилати повідомлення і електронні листи.

Технології, які використовуються в пристрої:

- камера, яка розпізнає жести рук;
- вбудовані датчики, які допомагають розпізнавати рухи;
- програмне забезпечення, що обробляє дані, отримані з камери та датчиків, та перетворює їх на текст або голос;

- для покращення точності розпізнавання жестів рук пристрій використовує штучний інтелект, який вчиться розпізнавати нові жести та адаптуватися до користувачів;
- може підключатися до Інтернету за допомогою Wi-Fi та з’єднуватися з іншими пристроями за допомогою Bluetooth.

1.1.2 SignAll 1.0

SignAll 1.0 – це програмне забезпечення для автоматичного перекладу мови жестів на англійську мову (рисунок 1.2). Воно створене компанією SignAll Technologies, яка була заснована в 2015 році в Будапешті, Угорщина. Програмне забезпечення знайшло застосування в різних сферах, таких як освіта, медицина та бізнес.

Рисунок 1.2 – Користування застосунком SignAll 1.0 [3]

Застосунок працює на базі технології комп'ютерного зору та машинного навчання, яка дозволяє точно розпізнавати жести та перекладати їх в текстовому форматі на англійську мову. Система використовує камери для

відеозапису жестів та перетворення їх на дані, які потім обробляються за допомогою спеціальних алгоритмів [4].

Програмне забезпечення було розроблене з метою забезпечити доступ до комунікації для людей зі слуховими порушеннями. Завдяки цьому програмному забезпеченню, люди зі слуховими порушеннями можуть легко спілкуватися зі своїми колегами, друзями та родичами.

Особливості цього ПЗ (рисунок 1.3) :

- комунікація людей з вадами слуху та без;
- переклад американської мови жестів (ASL) на англійську. Повне розпізнавання ASL;
- розпізнавання розмовної англійської мови;
- інтерфейс чату, який відображає розмову для обох сторін;
- навчальний компонент освіти, що навчає мови жестів людей із вадами слуху та мовлення.

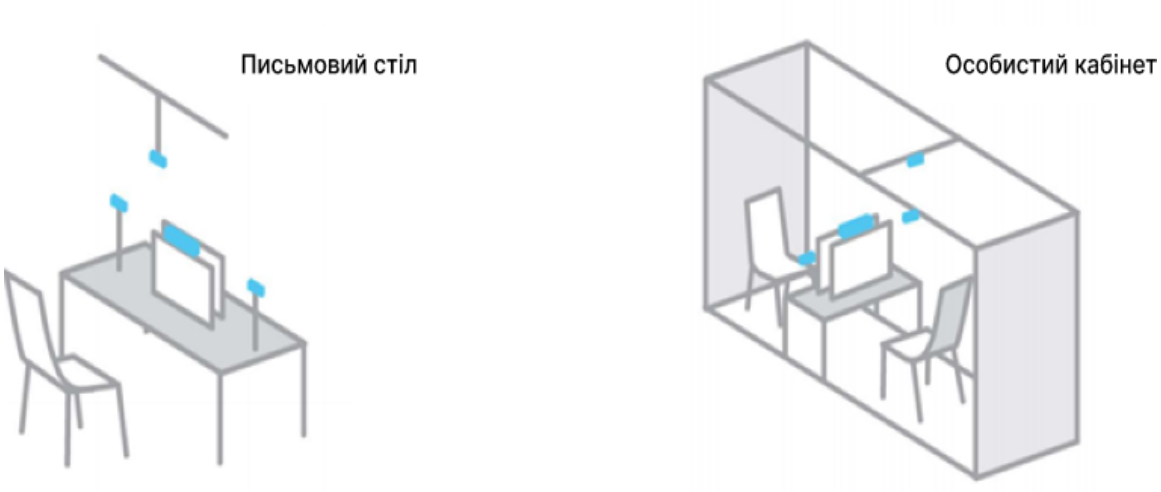


Рисунок 1.3 – Конфігурація продукту SignAll [3]

1.1.3 Myo

Myo – це переносний пристрій для керування електронними пристроями за допомогою жестів руки та м'язів (рисунок 1.4). Він був створений компанією Thalmic Labs (згодом перейменовано на North) та вперше був представлений у 2013 році.

Myo використовує датчики м'язів та жестів, щоб визначити рухи користувача та перетворити їх на команди. За допомогою Myo можна керувати різними електронними пристроями, такими як комп'ютери, телефони, телевізори, гарнітури віртуальної реальності та інші. Крім того, він підтримує програмне забезпечення, що дозволяє розробникам створювати свої власні додатки для використання з Myo.

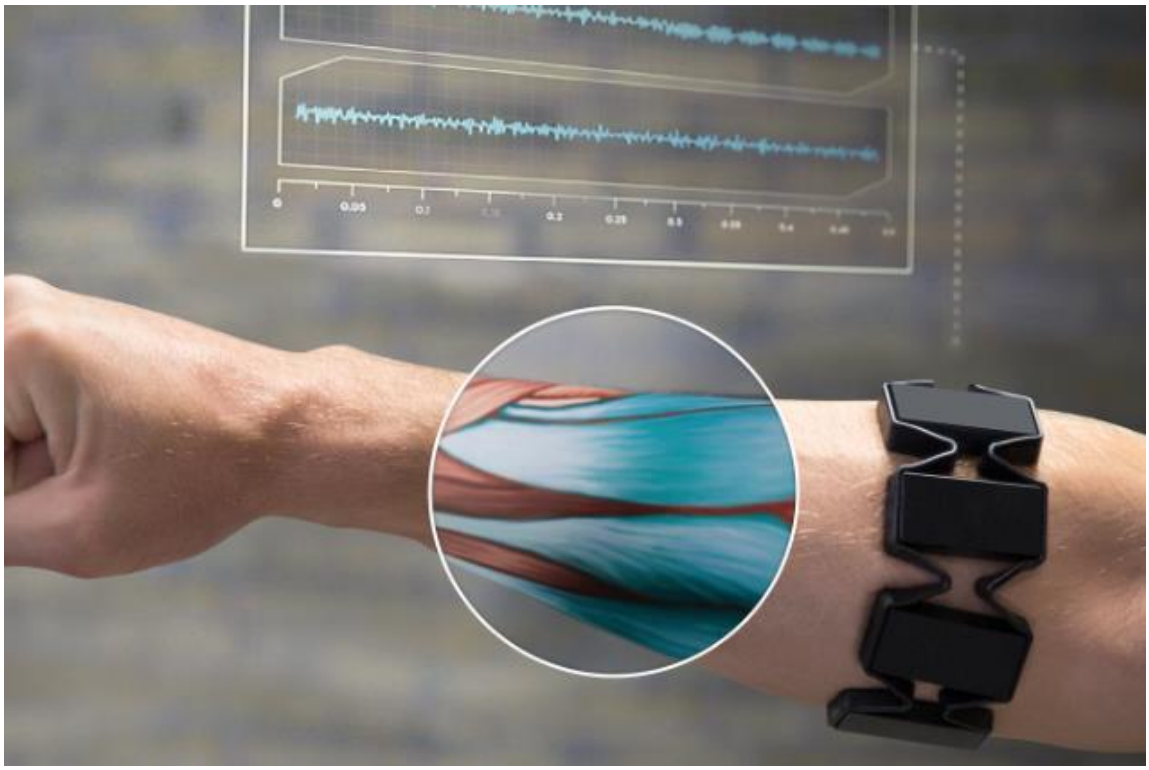


Рисунок 1.4 – Переносний пристрій керування жестами Myo [5]

Myo має бездротове з'єднання Bluetooth, що дозволяє користувачам використовувати його на відстані до дев’яти метрів від пристрою, з яким він

підключений. Він також має вбудований акумулятор, що дозволяє йому працювати протягом декількох годин без підзарядки.

Муо може бути використаний в багатьох різних областях, таких як медицина, спорт, віртуальна реальність, музика та інші. Наприклад, він може бути використаний для керування пристроями під час хірургічних операцій, для тренування м'язів під час фізичних вправ, для керування рухами персонажа віртуальної реальності та багатьох інших застосувань [6].

1.1.4 GestureSign

GestureSign – це програмне забезпечення для комп'ютера, яке дозволяє використовувати жести рук для керування різними функціями та додатками (рисунок 1.5). За допомогою GestureSign можна налаштувати жести для відкриття веб-сторінок, запуску програм, перемикання між вікнами, скролінгу сторінок, виконання різних команд тощо [7].

GestureSign підтримує різні жести, такі як зведення пальців, обертання зап'ястків, рухи вліво-вправо, вверх-вниз та інші. Крім того, в програмі є можливість створювати власні жести та налаштовувати їх для виконання конкретних дій.

GestureSign працює на операційних системах Windows 7, 8 та 10, та не вимагає використання спеціального обладнання для розпізнавання жестів. Програма використовує камеру вбудовану в ноутбук або зовнішню веб-камеру для розпізнавання жестів.

GestureSign може бути корисним для людей з обмеженими можливостями, а також для тих, хто працює з комп'ютером протягом тривалого часу та бажає зменшити навантаження на мишу та клавіатуру.

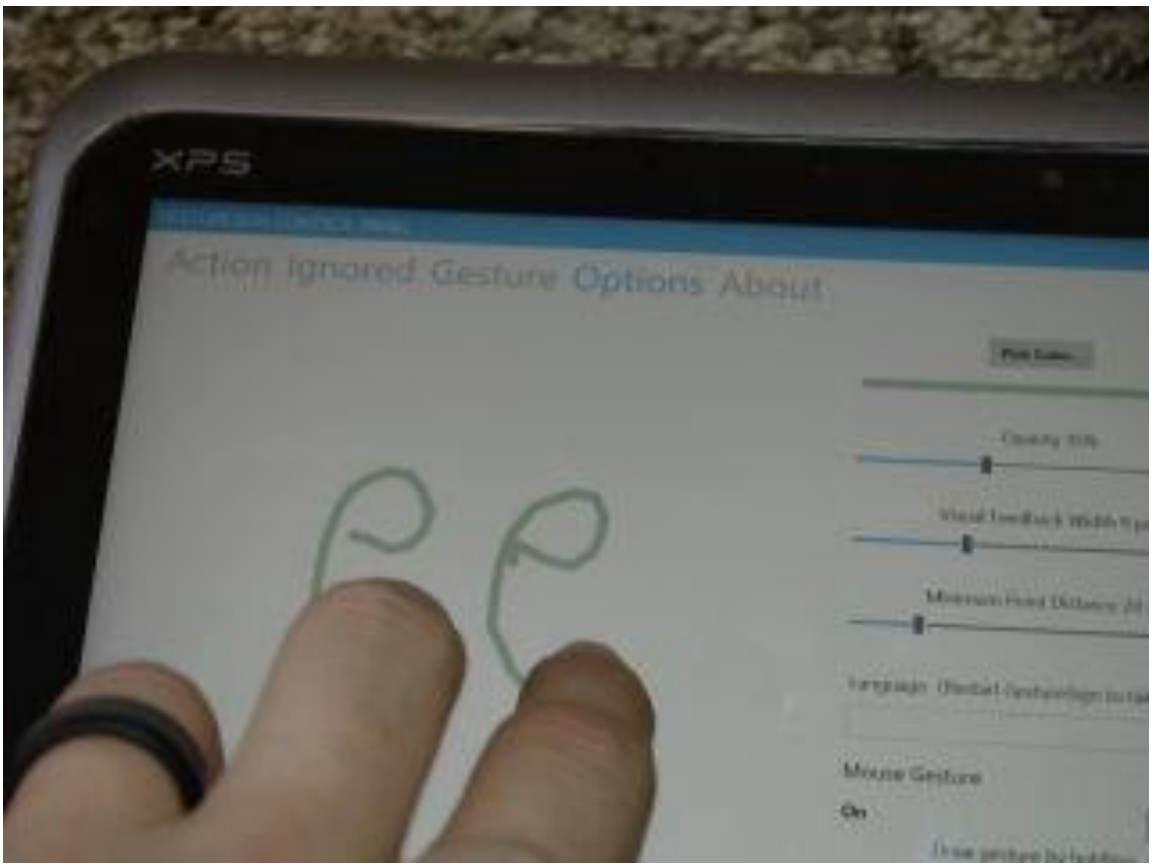


Рисунок 1.5 – Ілюстрація налаштування жестів для керування [8]

1.2 Аналіз переваг та недоліків наявних систем

Хоча вище наведені системи є важливими для людей з обмеженими можливостями, вони мають свої переваги та недоліки.

Система MotionSavvy UNI має наступні переваги. Технологія розпізнавання жестів. MotionSavvy UNI використовує технологію, що дозволяє розпізнавати жести людей, які мають вади слуху, що дає можливість їм комунікувати з іншими людьми, не використовуючи інші методи спілкування.

Висока точність. MotionSavvy UNI відрізняється високою точністю розпізнавання жестів, що дозволяє користувачам взаємодіяти з пристроєм без проблем та без зайвих затримок.

Простота використання. MotionSavvy UNI має простий та інтуїтивно зрозумілий інтерфейс, що дозволяє користувачам швидко та легко навчитися взаємодіяти з пристроєм.

Портативність. MotionSavvy UNI є портативним пристроєм, що дозволяє користувачам взаємодіяти з ним у будь-якому місці та в будь-який час.

Можливість інтеграції з іншими системами. MotionSavvy UNI може бути інтегрований з іншими системами та пристроями, що дозволяє користувачам взаємодіяти з ними, використовуючи ті ж жести, що вони використовують з MotionSavvy UNI.

Ця система непогана, але має свої недоліки. MotionSavvy UNI є досить дорогим пристроєм порівняно з іншими рішеннями для комунікації для людей з вадами слуху.

Потреба в навчанні. Хоча MotionSavvy UNI має простий та інтуїтивно зрозумілий інтерфейс, користувачам потрібно витратити певний час на навчання використанню пристрою.

Обмежена мінімальна відстань. Для того, щоб пристрій розпізнав жести користувача, він повинен бути досить близько до нього, що може бути не зручним у певних ситуаціях.

Обмежена кількість жестів. MotionSavvy UNI має обмежену кількість жестів, які він може розпізнавати, що може обмежувати комунікацію користувачів з пристроєм.

Потреба в освітленні. MotionSavvy UNI потребує достатнього освітлення, щоб правильно розпізнати жести користувача, тому його використання може бути неефективним у певних умовах освітлення.

Переглянемо переваги та недоліки SignAll 1.0. Система використовує технології комп'ютерного зору та нейромереж для точного визначення жестів. Це дозволяє досягнути високої точності перекладу мови жестів на текст.

Швидкість. SignAll 1.0 працює в режимі реального часу, що дозволяє користувачам отримувати миттєвий переклад жестів текст. Це дуже важливо для людей з вадами слуху, які залежать від мови жестів.

Відкритий формат. SignAll 1.0 підтримує відкритий формат мови жестів, що дозволяє додавати нові жести у систему без необхідності відомості про внутрішню архітектуру системи. Це дозволяє розширювати функціональність системи.

SignAll 1.0. як і MotionSavvy UNI також попри свої переваги має декілька недоліків. SignAll 1.0 може розпізнати лише обмежену кількість жестів. Інші системи можуть розпізнавати більшу кількість жестів і мати більшу точність.

Також цей застосунок не може навчитися розпізнавати нові жести, які він не має в своєму довіднику. Інші системи можуть навчатися з нових даних і оновлювати свої алгоритми розпізнавання жестів.

Система обмежується використанням камери. Вона потребує камеру для розпізнавання жестів, що обмежує її використання в деяких ситуаціях, наприклад, у темному приміщенні або на відкритому повітрі. Інші системи можуть використовувати додаткові датчики для розпізнавання жестів.

Висока вартість, що є так само і недоліком MotionSavvy UNI. SignAll 1.0 є досить дорогим рішенням порівняно з іншими системами розпізнавання жестів, що може бути проблемою для багатьох користувачів і організацій.

Для використання SignAll 1.0 необхідно встановлювати спеціальні додатки на комп'ютер або мобільний пристрій. Інші системи можуть працювати без необхідності встановлювати додатки.

Такі переваги має пристрій Myo. Цей пристрій дозволяє користувачам керувати пристроями за допомогою м'язів рук.

Myo має малі вагу і розмір. Досить компактний і легкий, що дозволяє його легко носити на руці. Це робить його зручним для використання в різних ситуаціях, таких як на вулиці або в офісі.

Має широкий спектр застосувань. Він може використовуватись для керування різними пристроями, такими як комп'ютери, телефони, телевізори, роботи і т. д. Це робить його універсальним пристроєм для керування пристроями у різних областях життя.

Висока точність і швидкість відповіді. Муо має досить високу точність і швидкість відповіді, що дозволяє користувачам точно керувати пристроями.

Простота використання. Муо досить простий у використанні. Користувачам не потрібно проводити складне калібрування або вчитися складним жестам, як у деяких інших системах керування.

Немає обмежень у просторі. Муо не потребує обмеження в просторі, що дозволяє користувачам використовувати його в будь-яких умовах і в будь-якому місці.

Але як і інші системи, Муо має свої недоліки. Пристрої Муо призначені виключно для керування різними пристроями за допомогою м'язів рук, що може бути обмежує його застосування в порівнянні з іншими схожими пристроями.

Для користування Муо потрібно досить багато практики та навичок, оскільки для керування пристроями потрібно використовувати відповідні м'язи та жести, які можуть бути доволі складними для деяких користувачів.

Цей пристрій може відрізняти тільки певні м'язи та жести, що може створювати проблеми в разі використання пристрою в складних умовах.

Муо може мати проблеми зі зв'язком, особливо в умовах з високими рівнями перешкод, таких як стіни або металеві поверхні.

Висока вартість. Пристрій може бути досить дорогим порівняно з іншими схожими рішеннями.

І розглянемо деякі недоліки та переваги GestureSign. Програма GestureSign є безкоштовною. Це дозволяє користувачам економити гроші.

Вона досить проста у використанні та налаштуванні, завдяки чому користувачі можуть швидко зрозуміти її функції.

GestureSign дозволяє користувачам створювати власні жести та налаштовувати їх для виконання певних дій.

Також користувачі в програмі можуть налаштувати жести для виконання різних команд, таких як запуск програм, відкриття файлів, перемикання між вікнами, скролінг сторінок тощо.

GestureSign підтримує операційні системи Windows 7, 8 та 10, що дозволяє користувачам використовувати програму на більшості сучасних комп'ютерах.

GestureSign підтримує обмежену кількість жестів, тому може бути складним для використання зі складними комбінаціями жестів або жестами, які не підтримуються програмою.

Програма потребує сенсорного екрану, внаслідок чого, користування програмою неможливе на більшості комп'ютерів без такого обладнання.

GestureSign може бути складною для використання на великих екранах або використання багатьох жестів з більшими масштабами.

Вимоги до користувачів. GestureSign потребує, щоб користувачі навчилися та запам'ятали жести, що може бути складним для новачків або тих, хто користується різними пристроями з різними жестами.

Програма не має додаткових функцій, таких як голосове керування або розпізнавання мови, які можуть бути доступними в інших рішеннях.

Висновки до розділу 1

Отже, переглянувши характеристики вище наведених систем зробимо висновки. MotionSavvy UNI і SignAll 1.0 – це програми, які створені ті для людей з вадами слуху та мовлення. А інші можуть застосовуватись в різних сферах. MotionSavvy UNI більш портативний та простий в використанні за SignAll 1.0, також для роботи SignAll 1.0 потрібно встановлювати додаткові програми на комп’ютер. Що ж до Myo та GestureSign, то Myo може керувати різними пристроями, як GestureSign може тільки комп’ютером. Але GestureSign є безкоштовним що, на мою думку є перевагою.

Для людей з вадами слуху та мовлення більш зручнішим буде пристрій MotionSavvy UNI. Він дозволить безперешкодно спілкуватись з іншими людьми.

Аналізуючи переваги та недоліки наведених систем, прийнято рішення побудувати систему з наступними вимогами:

- система має використовувати камеру та штучний інтелект для розпізнавання мови жестів;
- текст, який перекладатиме програма з мови жестів – має висвітлюватись на екрані ноутбуку чи іншого пристрою;
- також, програма має бути доступною, щоб можна було безкоштовно нею користуватись;
- щоб не витрачати багато часу на навчання користування, програма має бути з простим та зрозумілим інтерфейсом.

2 РОЗРОБЛЕННЯ СИСТЕМИ ПЕРЕТВОРЕННЯ МОВИ ЖЕСТІВ У ТЕКСТ

У даному розділі розглянуто вибір методу для розроблення системи, вибір нейронної мережі для зчитування жестів, пояснення розроблення алгоритму для системи та використання набору даних для навчання моделі нейронної мережі.

2.1 Вибір методу для розроблення системи

2.1.1 Використання згорткової нейронної мережі

Для того, щоб система розпізнавала жести, перед тим необхідно виділити певні ознаки зображення. Після цього можна використовувати будь-який метод класифікації для розпізнавання та інтерпретації отриманих жестів. Щоб класифікувати зображення необхідно, щоб була велика кількість ознак [8].

На сьогоднішній день існує велика кількість різних типів нейронних мереж, тому вибір конкретної реалізації нейронної мережі зазвичай здійснюється на основі експертної оцінки або емпіричних досліджень, що вказують на найбільш ефективну архітектуру. Проте обидва підходи мають свої недоліки: експертна оцінка вимагає наявності експертних знань, які не завжди доступні, а емпіричні дослідження можуть вимагати значних ресурсів і часу через велику різноманітність нейронних мереж.

У зв'язку з цим раціональним підходом є використання простої, реалізованої та добре вивченої архітектури нейронної мережі.

В задачах розпізнавання зазвичай використовуються згорткові нейронні мережі (рисунок 2.1). Вони складаються з трьох шарів: перший шар згортки відповідає за виділення ознак зображення, пулінговий – зменшує зображення для прискорення обчислень, а повнозв’язний призначений для класифікації

результатів. На відміну від простих нейронних мереж, у CNN нейронни розташовані у шарах у трьох вимірах – в ширині, глибині і висоті.

Згорткові нейронні мережі мають такі переваги:

- відсутність необхідності у ручному визначенні особливостей жестів на зображеннях;
- згорткові нейронні мережі можуть бути навчені розпізнавати та розрізняти різних користувачів, навіть якщо вони не брали участі у тренуванні моделі. Це дозволяє побудувати систему, яка ідентифікує конкретних осіб на основі їх жестів;
- згорткові нейронні мережі виявляють високу стійкість до змін масштабу вхідних зображень, різних умов освітлення та часткових закриттів (оклюзій). Вони можуть робити розпізнавання жестів незалежно від таких змін, що покращує загальну надійність системи.

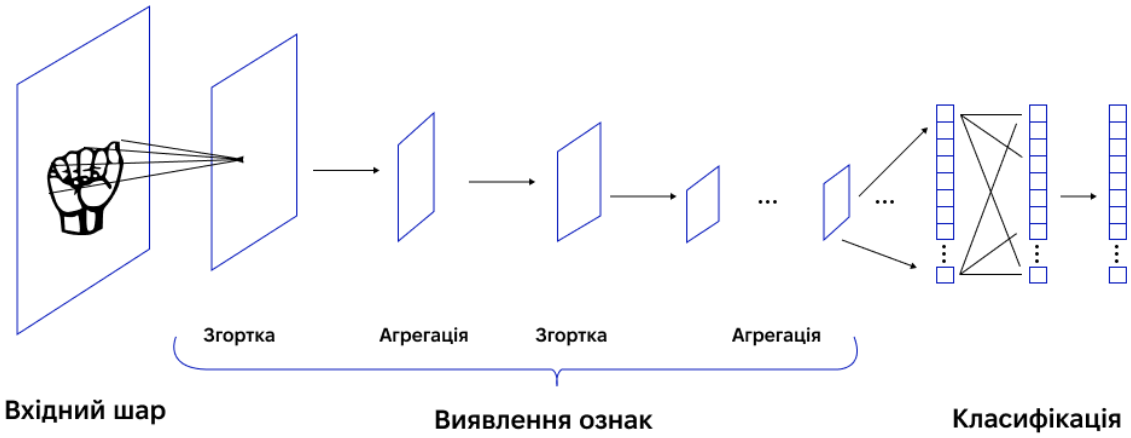


Рисунок 2.1 – Шари згорткової нейронної мережі

Нейронну модель потрібно навчати, і для цього застосовуються задачі глобальної оптимізації. Задачі глобальної оптимізації полягають у пошуку найкращого розв'язку у множині можливих варіантів, яка відповідає заданим обмеженням і максимізує (або мінімізує) певну цільову функцію. Ці задачі виникають у багатьох галузях, включаючи науку, інженерію, економіку та інформатику.

За допомогою оптимізації функції втрат можна знайти оптимальні значення параметрів для побудови найкращої згорткової нейронної мережі. Стохастичний градієнтний спуск є найпоширенішим методом оптимізації для досягнення цього, який використовується для поступового оновлення параметрів моделі шляхом врахування градієнтів функції втрат [10].

2.1.1 Використання комп'ютерного зору

Комп'ютерний зір використовується для розпізнавання об'єктів, виявлення образів, вимірювання розмірів, визначення позиції об'єктів, відстеження руху, розпізнавання жестів та багато іншого. Він може застосовуватись в широкому спектрі галузей, включаючи автоматизовану виробничу лінію, медичну діагностику, розпізнавання обличь, жестів та багато іншого [11].

Комп'ютерний зір (КЗ) є галуззю комп'ютерних технологій, яка спрямована на розробку наукових основ розпізнавання об'єктів та створення практичних систем, здатних виявляти та ідентифікувати об'єкти, аналогічно до сприйняття людиною зором у реальному світі.

Основна мета КЗ полягає в тому, щоб дозволити комп'ютерам "бачити" та розуміти візуальну інформацію, подібно до того, як це робить людський зір.

КЗ використовує різні техніки, включаючи згорткові нейронні мережі, глибоке навчання, методи відстеження об'єктів, детектори ребер Кенні, методи сегментації зображень та багато інших алгоритмів та методів обробки зображень.

Часто вважають, що КЗ – є машинним зором, але це не так, оскільки вони дечим відрізняються. Машинний зір зосереджений на розв'язанні практичних завдань та розробці апаратних і програмних рішень для цих цілей. Ця область включає інженерні знання та прикладне програмування. Однак, для цілей даного дослідження більш відповідним є термін "комп'ютерний зір", оскільки

в рамках комп'ютерних наук оцінюється можливість застосування різних технологій у практичних сценаріях.

OpenCV (Open Source Computer Vision Library) є бібліотекою комп'ютерного зору, яка надає набір інструментів і функцій для обробки зображень, відео та розв'язання завдань комп'ютерного зору.

OpenCV надає реалізацію багатьох алгоритмів комп'ютерного зору, таких як виявлення обличь, вимірювання розмірів об'єктів, відстеження руху, сегментації зображень та інших. Також ця бібліотека включає інструменти для калібрування камер та оптимізації обробки зображень. OpenCV використовується в комп'ютерному зорі як потужний інструментарій для розробки та реалізації рішень з обробки зображень та відео.

Тому, для системи розпізнавання жестів використаємо OpenCV. За допомогою цієї бібліотеки можна відкривати, записувати та відображати зображення, просто відео чи відео з веб-камери, а також захоплювати кадри та проводити обробку зображень.

2.1.2 Набір даних для навчання нейронної мережі

Для навчання CNN потрібно мати велику кількість даних, щоб оптимізовувати роботу моделі. В інтернеті існує безліч готових даних, які можна використовувати. Одним з таких наборів даних є "Sign Language MNIST" (SL-MNIST) (рисунок 2.2). Він є модифікацією набору даних MNIST, який використовується для класифікації рукописних цифр. Оригінальний набір зображень рукописних цифр MNIST є популярним еталоном для методів машинного навчання на основі зображень.

SL-MNIST складається із зображень жестів (рисунок 2.3) американської жестової мови (American Sign Language - ASL). Цей набір даних є загальнодоступним і безкоштовним із інформацією про пікселі для приблизно 1000 зображень кожної 24 літер з ASL.

Рисунок 2.2 – Панель монтажу кадрованих зображень різних користувачів і фонів для літер американської жестової мови» від SL-MNIST [12]

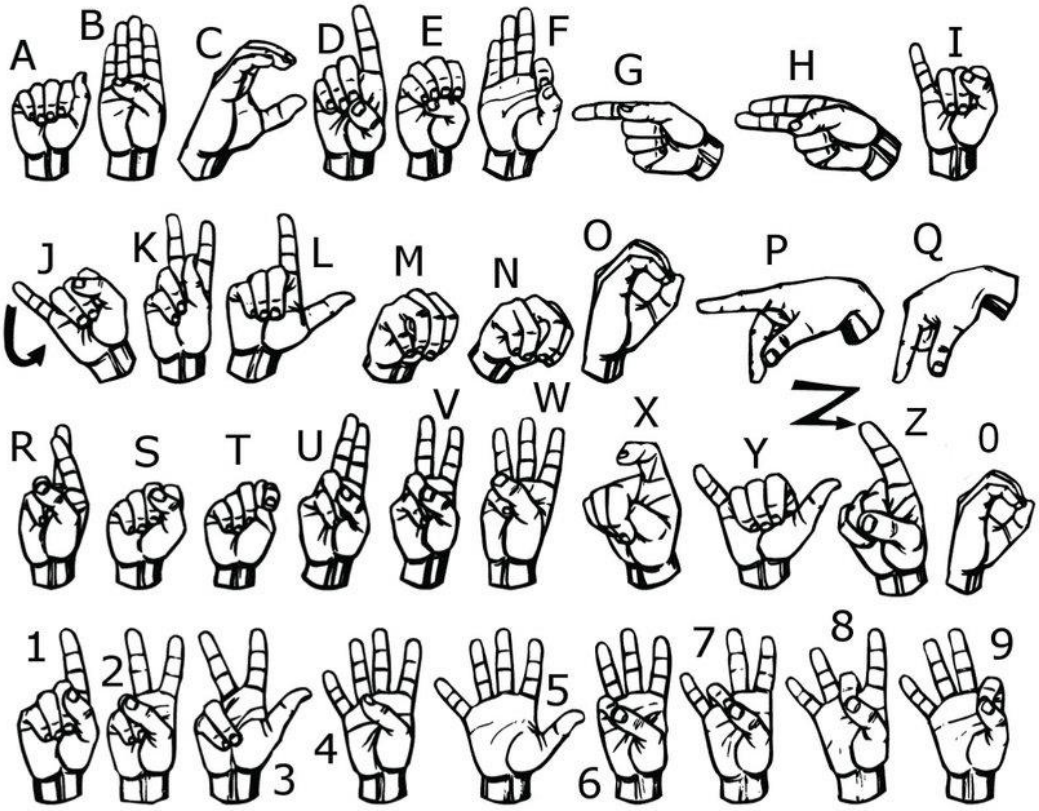


Рисунок 2.3 – Абетка американської жестової мови [13]

Основні характеристики набору даних "Sign Language MNIST":

- набір даних містить 27,455 зображень;
- кожне зображення має розмір 28x28 пікселів і представлене у відтінках сірого, що означає, що кожен піксель має значення інтенсивності від 0 до 255;
- жести відображають різні букви англійського алфавіту, від "A" до "Z". Загалом є 24 класи, бо жести "J" та "Z" відсутні, оскільки вони є динамічними;
- набір даних поділено на дві частини: навчальний набір містить 24,345 зображень, а тестовий набір - 2,110 зображень;
- класи жестів кодуються числами від 0 до 23, де 0 відповідає жесту "A", 1 - жесту "B" і так далі;
- кількість зображень в кожному класі приблизно однакова, що робить набір даних збалансованим;
- набір даних SL-MNIST часто використовується для навчання моделей машинного навчання для класифікації жестів американської жестової мови. За його допомогою можна розпізнавати та класифікувати різні букви у вигляді жестів на основі зображень.

Набір даних SL-MNIST широко використовується у дослідженнях та прикладних проектах з області комп'ютерного зору та машинного навчання. Велика кількість джерел та ресурсів, що стосуються SL-MNIST, забезпечує зручність у розробці та порівнянні моделей.

2.2 Розроблення алгоритму та програмної реалізації системи

2.2.1 Проектування системи

Технологія розпізнавання жестів складається з таких етапів:

- отримання набору даних;
- збільшення набору (для випадків використання власних відео та/або зображень);

- попередня підготовка даних;
- навчання класифікаторів;
- розпізнавання жестів.

Для користування системою користувачу потрібно використовувати веб-додаток або мобільний додаток та веб-камеру.

Збір даних відбувається за такою послідовністю:

- записується відео, потім його система ділить на кадри (відбувається сегментація на окремі зображення);
- обробка даних;
- збереження даних для майбутнього використання для тренування чи тестування нейронної мережі.

Обробка даних потрібна для перетворення кадрів зображень із жестами в один формат для легшого подальшого використання їх. Це дозволить уникнути додаткових навчань та помилок нейронних мереж. Для цього використовується нормалізація та приведення до одного розміру даних.

Розпізнавання відбувається за допомогою вже навченої нейронної мережі.

Для системи спроектовано структурну схему в якій детально візуалізовано всі процеси.

Розглянемо детальніше структурну схему системи розпізнавання жестів та перекладу їх на текст за допомогою ШІ (кресленик ІС92.170БАК.004 Д1).

Загалом, система складається з двох основних частин: клієнтської та серверної. Клієнтська частина відповідає за зчитування жестів з веб-камери та передачу отриманих даних до серверної частини. Серверна частина відповідає за обробку отриманих даних та перетворення жестів на текст. Для цього на серверній стороні буде розташована модель зі штучним інтелектом, яка буде виконувати обробку зображення та розпізнавання жестів. Ця модель буде навчена на великій кількості зображень жестів і буде здатна їх розпізнавати. Отриманий текст буде відображатись на екрані користувача.

У частині «Клієнт» знаходяться такі блоки:

- «Робоча станція»;
- «Мобільний клієнт»;
- «Відеокамера».

У цьому частині «Клієнт» відбувається взаємодія користувача з системою. За допомогою інтерфейсу застосунку користувач може зареєструватись та авторизуватись, щоб почати перекладати жести на текст за допомогою камери.

Для передачі відео на сервіс відкривається потік між клієнтом і сервером в якому дані передаються за кодуванням H.264, найпоширенішим відеокодеком, який широко використовується для онлайн-відео.

У частині «Сервер» використовуються такі мікросервери:

- «Реєстрація/Авторизація»;
- «Розпізнавання ШІ»;
- «Обробка даних»;
- «Account БД» (для запису даних про користувачів);
- «БД значень жестів»;
- «БД» (для функціоналу персонального кабінету).

Отже, відео спершу обробляється у блоці «Обробка даних». Тут виділяються жести за допомогою сегментування відео. Після виділення жестів відбувається процес класифікації, де система визначає, який жест був виконаний на відео. Це здійснюється за допомогою навченої моделі ШІ, яка розпізнає певні жести або рухи. Перед тим як модель розпізнаватиме жести, вона тренується на певному наборі даних.

Блок «Розпізнавання ШІ» відображає процес розпізнавання жестів, результатом якого є текст.

Після отримання результатів дані повертаються у форматі JSON через маршрутизатор до застосунку.

Блок «Реєстрація/Авторизація» створений для авторизації користувачів, для того щоб дані користувачів записувались у БД і в подальшому після

користування розпізнаванням, модель ШІ була більш навчена на отриманих даних користувача.

Для взаємодії користувача детальніше розглянемо діаграми послідовностей авторизації користувача та процесу розпізнавання жесту.

Діаграми послідовностей є одним із типів діаграм, які використовуються для моделювання та візуалізації взаємодії між об'єктами або компонентами в системі. Вони надають зручний спосіб представлення послідовності подій та повідомлень між різними учасниками системи, зокрема об'єктами, компонентами або користувачами.

Діаграма послідовностей показують взаємодію об’єктів, поданих у хронологічному порядку.

Діаграма послідовностей авторизації користувача (кресленик ІС92.170БАК.004 Д2):

- користувач відкриває додаток та пробує ввійти за допомогою логіну та паролю;
- через БД перевіряються дані чи вірно авторизований користувач;
- якщо невдала авторизація, то користувачу пропонується реєстрація, якщо такого користувача не знайдено у БД, якщо дані не дійсні – то користувач не авторизований;
- результати виводяться на екран.

Діаграма послідовностей процесу розпізнавання жестів (ІС92.170БАК.004 Д3) зображує як користувач взаємодіє з програмою з метою отримати розпізнаний текст.

Отже, розглянемо детальніше цю діаграму:

- спершу користувач має увійти в застосунок;
- користувач на камеру показує жести;
- відео з жестами користувача передається у модуль обробки жестів, де відбувається кадрування, тобто у результаті мають вийти сегментовані зображення із жестами;

- далі ці жести записуються у базу даних, для того щоб модель ШІ могла швидше розпізнати жести користувача наступного разу;
- після бази даних модель ШІ розпізнає жести та повертає розпізнаний текст;
- розпізнаний текст теж записується у базу даних;
- потім вже класифіковані дані висвітлюються на екрані користувача.

2.2.2 Інструменти реалізації

Для того, щоб показати, як працюватиме переклад мови жестів у текст за допомогою штучного інтелекту – реалізовано програму на мові Python.

Python – це легка для використання та надійна мова програмування. Python має простий та зрозумілий синтаксис, що робить цю мову доступною. Це сприяє швидкому розробленню прототипів та експериментуванню з алгоритмами розпізнавання жестів.

Python має широкий вибір бібліотек та фреймворків, які полегшують розробку системи зчитування жестів. Наприклад, бібліотека OpenCV, яка широко використовується в комп'ютерному зорі, має підтримку для Python та надає зручні інструменти для обробки зображень та відео.

Python є популярним вибором для розробки моделей машинного навчання. Існує багато бібліотек, таких як TensorFlow, PyTorch, і Keras, які надають потужні інструменти для навчання та застосування моделей розпізнавання жестів.

Оскільки в алгоритмі реалізації системи присутня згорткова нейронна мережа – вирішено використати доступну для користування бібліотеку з інструментами для машинного навчання Keras. Бібліотека Keras є популярним і потужним інструментом для побудови та тренування нейронних мереж, вона широко використовується в галузі глибинного навчання.

Бібліотека Keras надає простий та інтуїтивно зрозумілий інтерфейс для визначення, конфігурування та тренування нейронних мереж. Вона має простий синтаксис, який дозволяє швидко створювати моделі без необхідності писати багато низькорівневого коду.

Keras може використовувати різні бекенди, такі як TensorFlow, Theano або CNTK, що дозволяє легко переключатись між ними в залежності від потреб та вимог проекту.

Для забезпечення ефективного процесу розробки на мові Python використовується інтегроване середовище розробки PyCharm. Це середовище надає зручний інтерфейс для написання, відлагодження та тестування коду на Python, сприяючи продуктивній розробці програмного забезпечення.

Для управління залежностями проекту використовується система управління пакетами pip. Вона дозволяє зручно встановлювати, оновлювати та керувати залежностями, необхідними для виконання проекту. Завдяки pip можна легко встановлювати зовнішні бібліотеки та модулі, які розширюють функціональність проекту на Python.

У коді використовуються такі бібліотеки:

- Keras – це високорівнева бібліотека глибинного навчання, яка дозволяє легко визначати, тренувати та оцінювати нейронні мережі;
- Seaborn – це бібліотека для візуалізації даних у Python. Вона побудована на основі бібліотеки Matplotlib, але надає більш високорівневий та естетичний інтерфейс для створення привабливих та інформативних графіків;
- Pandas – це бібліотека для обробки та аналізу даних у мові програмування Python. Вона надає прості та ефективні структури даних, такі як DataFrame, для роботи з табличними даними, що дозволяє легко виконувати операції з даними, які зазвичай виконуються у статистичному аналізі та обробці даних;
- Matplotlib – це одна з найпопулярніших бібліотек для візуалізації даних у мові програмування Python. Вона надає інструменти для створення різноманітних типів графіків, діаграм та інших візуалізаційних відображень.

2.2.3 Програмна реалізація

Щоб продемонструвати як працює CNN у даній системі реалізовану програмне забезпечення за допомогою якого користувач може перекласти жест за допомогою згорткової нейронної мережі.

Реалізовано модель згорткової нейронної мережі. Модель мережі створюється за допомогою класу Sequential, який дозволяє додавати шари послідовно один за одним.

Основні компоненти CNN включають згорткові шари (Conv2D), пулінгові шари (MaxPool2D), шари нормалізації (BatchNormalization), шари випадкового вимкнення (Dropout), повнозв'язані шари (Dense) та функції активації (relu, softmax).

Розлянемо детальніше побудову моделі:

Створення моделі (рисунок 2.4):

```
model = Sequential()
```

Рисунок 2.4 – Код створення моделі

В наступному кроці додаються згорткові шари, шари пулінгу, нормалізації, повнозв’язні шари, шари випадкового вимкнення (рисунок 2.5). Також тут відбувається перетворення вхідних даних в одновимірний вектор.

Багато нейронних мереж, зокрема повнозв'язані шари, очікують одновимірні вектори як вхідні дані. Перетворення даних в одновимірний вектор дозволяє підключити ці шари до вхідних даних та виконувати подальші операції.

```
model = Sequential()

model.add(Conv2D(75, (3, 3), strides=1, padding='same', activation='relu', input_shape=(28, 28, 1)))
model.add(BatchNormalization())
model.add(MaxPool2D((2, 2), strides=2, padding='same'))
model.add(Conv2D(50, (3, 3), strides=1, padding='same', activation='relu'))
model.add(Dropout(0.2))
model.add(BatchNormalization())
model.add(MaxPool2D((2, 2), strides=2, padding='same'))
model.add(Conv2D(25, (3, 3), strides=1, padding='same', activation='relu'))
model.add(BatchNormalization())
model.add(MaxPool2D((2, 2), strides=2, padding='same'))

model.add(Flatten())

model.add(Dense(units=512, activation='relu'))

model.add(Dropout(0.2))
```

Рисунок 2.5 – Код додавання шарів CNN

Компіляція моделі (рисунок 2.6):

```
model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])
model.summary()
```

Рисунок 2.6 – Код компіляції CNN

Метод «compile» (компіляція моделі) викликається для налаштування параметрів оптимізації та функції втрати.

Тренування та навчання моделі:

Тренування моделі відбувається на вже підготовлених даних (рисунок 2.7).

```
train_df = pd.read_csv("sign_mnist_train.csv")
test_df = pd.read_csv("sign_mnist_test.csv")

y_train = train_df['label']
y_test = test_df['label']
del train_df['label']
del test_df['label']
```

Рисунок 2.7 – Код зчитування та підготовлення набору даних «Sign Language MNIST»

Підготовка даних для генератора даних (рисунок 2.8):

```
datagen.fit(x_train)
```

Рисунок 2.8 – Код для підготовки та адаптації для генератора даних

Тренування моделі у цьому коді відбувається за допомогою методу «fit», який викликається на об'єкт моделі (рисунок 2.9).

```
history = model.fit(datagen.flow(x_train, y_train, batch_size=128),
                    epochs=20, validation_data=(x_test, y_test),
                    callbacks=[learning_rate_reduction])
```

Рисунок 2.9 – Код тренування моделі методом fit()

Навчальні дані «x_train» та «y_train» передаються до функції «flow» об'єкта «datagen» (ImageDataGenerator). Це дозволяє генерувати нові варіанти зображень на основі наявних даних за допомогою заданих параметрів.

Для тренування моделі викликається метод «fit». Сюди передаються навчальні дані «x_train» та «y_train», розмір пакета («batch_size»), кількість проходів моделі через повний набір навчальних даних («epochs»), дані для перевірки «x_test» та «y_test» та додаткові зворотні виклики («callbacks») для налаштування тренування.

Під час тренування модель адаптує свої ваги, оптимізуючи функцію втрат за допомогою алгоритму оптимізації. За кожний прохід через набір даних виводиться інформація про прогрес тренування, включаючи значення функції втрат та точності для навчальних та даних для перевірки.

Збереження моделі (рисунок 2.10):

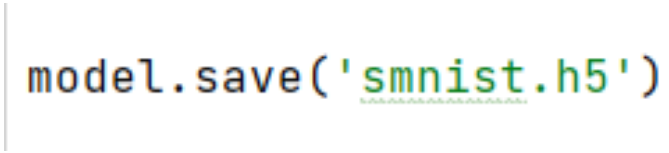


Рисунок 2.10 – Код збереження моделі

Модель зберігається для подальшого використання та розгортання. Коли модель навчена на тренувальних даних, її структура та параметри можуть бути збережені на диск з метою подальшого використання для прогнозування нових даних, оцінки результатів або інших завдань.

Збереження моделі дозволяє використовувати її для прогнозування значень на нових, невідомих даних. Після тренування модель може бути використана для передбачення результатів на нових вхідних даних без необхідності повторного навчання.

Збереження моделі дозволяє відновлювати її після тренування та продовжувати навчання з попередньої точки. Це може бути корисно, якщо навчання моделі займає багато часу або ресурсів, і потрібно продовжити навчання на більшій кількості даних.

2.3 Робота з CNN

2.3.1 Архітектура згорткової нейронної мережі

Для системи використовується згорткова нейронна мережа – CNN.

Архітектура згорткової нейронної мережі (Convolutional Neural Network, CNN) є спеціалізованою структурою нейронної мережі, яка широко використовується для обробки зображень та розпізнавання об'єктів. Основна особливість CNN полягає в застосуванні згорткових шарів, які дозволяють автоматично визначати та виділяти важливі ознаки зображень.

Архітектура цієї мережі складається із трьох згорткових шарів з 75, 50 та 25 фільтрами відповідно, та із пулінгових шарів і шарів нормалізації.

Розглянемо детально структуру CNN. Архітектура нейронної мережі має наступну структуру:

1) Вхідний шар:

- розмір вхідного зображення: 28x28 пікселів;
- вхідний шар типу “Conv2D” з 75 фільтрами розміром 3x3, кроком 1 і підсумовуванням з заповненням “same” (додавання нулів (порожніх значень) на краях вхідного зображення перед застосуванням згортки. Це робиться для збереження розміру вхідного зображення після згортки);
- нелінійна функція активації “relu” застосовується після кожного згорткового шару;
- шар нормалізації пакета “BatchNormalization” додається після першого згорткового шару;

2) Перший згортковий шар:

- застосування пулінгу за допомогою шару “MaxPool2D” з розміром пулінгу 2x2 та кроком 2;
- після пулінгу використовується метод регуляризації «дропаут» з ймовірністю 0.2 для регуляризації моделі (під час навчання, дропаут випадковим чином вимикає (обнуляє) окремі нейрони з певною ймовірністю, що зазвичай встановлюється між 0.2 та 0.5 – це означає, що під час проходження навчальних прикладів, окремі нейрони не приймають участь у процесі обчислення та оновлення ваг) ;

3) Другий згортковий шар:

- шар “Conv2D” з 50 фільтрами розміром 3x3, кроком 1 і підсумовуванням з заповненням “same”;
- шар нормалізації пакета “BatchNormalization” застосовується перед другим згортковим шаром;
- знову використовується дропаут з ймовірністю 0.2;

4) Третій згортковий шар:

- шар “Conv2D” з 25 фільтрами розміром 3x3, кроком 1 і підсумовуванням з заповненням “same”;
- шар нормалізації пакета “BatchNormalization” застосовується перед третім згортковим шаром;

5) Пулінг:

- застосування пулінгу за допомогою шару “MaxPool2D” з розміром пулінгу 2x2 та кроком 2;

6) Повнозв'язаний шар:

- перетворення вихідних даних у форму, придатну для подальшої обробки повнозв'язаним шаром, за допомогою шару Flatten;
- повнозв'язаний шар з 512 нейронами і функцією активації relu;
- дропаут з ймовірністю 0.3 застосовується для регуляризації моделі;

7) Вихідний шар:

- повнозв'язаний шар з 24 нейронами (кількість класів) і функцією активації softmax для отримання ймовірностей належності до кожного класу;

8) Оптимізатор та функція втрат:

- використовується оптимізатор adam;
- в якості функції втрати використовується categorical_crossentropy.

2.1.4 Функція активації нейронів ReLU

Функція ReLU (Rectified Linear Unit) є функцією активації, яка застосовується до виходів нейронів у мережі. Ця функція є однією з найпоширеніших функцій активації, використовуваних у штучних нейронних мережах. Вона використовується для нелінійної активації нейронів,

дозволяючи їм передавати сигнали до наступних шарів мережі. Вона визначається математично наступним чином (2.1):

$$ReLU(x) = max(0, x), \quad (2.1)$$

де x – вхідний сигнал для функції,

а max – функція, яка повертає більше значення з двох.

У практичному розумінні, функція ReLU активує нейрон, якщо вхідний сигнал x є позитивним, в іншому випадку (якщо x є негативним або нульовим), нейрон неактивний.

Застосування функції ReLU до виходів нейронів допомагає моделі нейронної мережі вчитися нелінійним залежностям та вирішувати складні завдання класифікації або регресії. Функція ReLU має просту обчислювальну формулу та виконується швидко, що робить її популярним вибором у багатьох архітектурах нейронних мереж.

Головні переваги функції активації ReLU включають простоту обчислення та відсутність проблеми зі зникненням градієнту. Вона швидко збігається під час навчання, оскільки активовані нейрони не залежать від значень від'ємних входів і можуть ефективно передавати сигнали.

Проте, слабкою стороною ReLU є проблема "мертвих нейронів". Якщо вихідний сигнал нейрона під час навчання стає від'ємним і залишається нульовим, то градієнт не може пройти через нього, що призводить до зупинки оновлення ваг і втрати можливості навчання. Цю проблему можна вирішити застосуванням варіантів функції ReLU, таких як Leaky ReLU або Parametric ReLU.

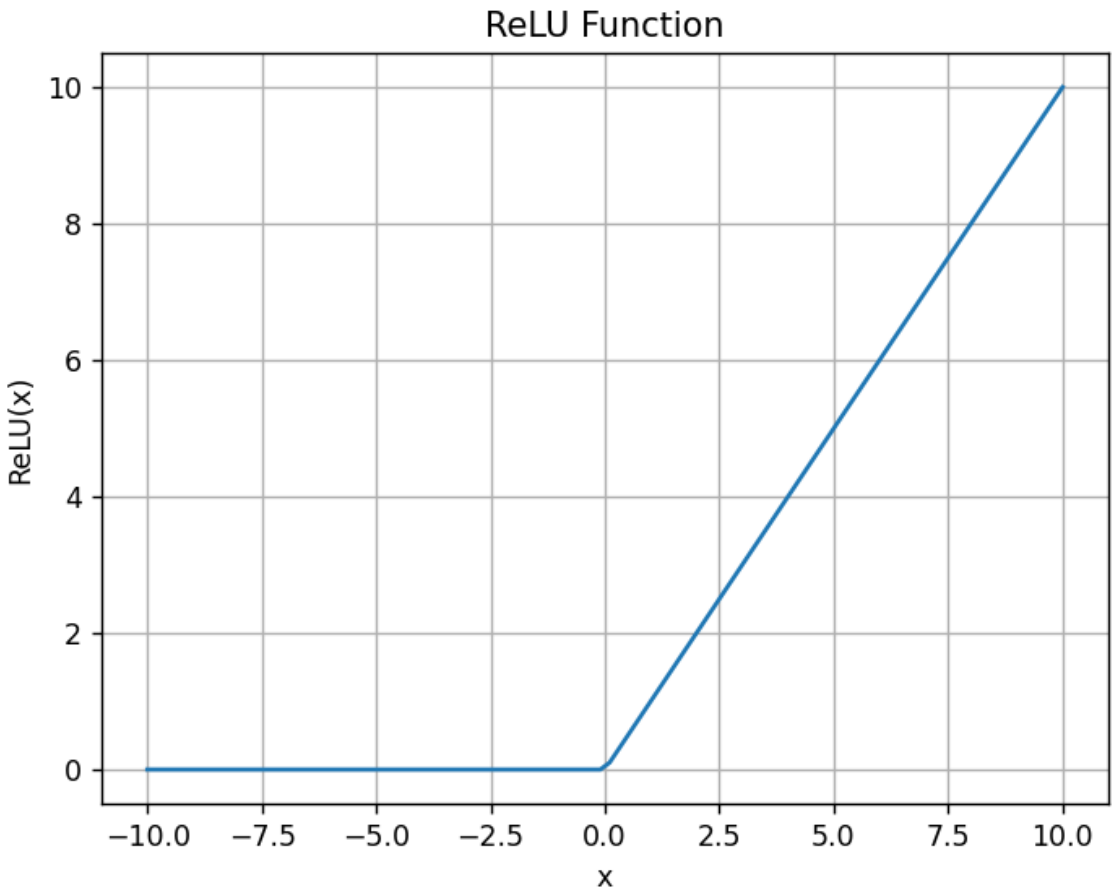


Рисунок 2.11 – Приклад графіку функції ReLU

На графіку (рисунок 2.11) видно, що функція ReLU повертає нуль для всіх негативних значень вхідного сигналу, а для позитивних значень повертає саме ці значення без змін. Таким чином, ReLU активує нейрони, коли вхідний сигнал є позитивним, і вимикає їх, коли вхідний сигнал є негативним.

2.1.5 Алгоритм оптимізації Adam

Оптимізація згорткової нейронної мережі включає в себе кроки, спрямовані на поліпшення її продуктивності, швидкості навчання та загальної точності моделі.

Для даної CNN використовується оптимізатор «Adam». Adam є популярним алгоритмом оптимізації, який комбінує метод моменту та адаптивного швидкісного спуску. Він дозволяє ефективно знаходити

оптимальні ваги моделі, адаптуючи швидкість навчання для кожного параметра окремо.

Алгоритм Adam використовує два основних механізми:

- зберігає покадрово середнє значення градієнтів, що дозволяє адаптивно налаштовувати швидкість навчання для кожного параметра. Це виконується за допомогою експоненційного згладжування, де більший ваговий коефіцієнт надається останнім градієнтам, а менший – старішим;
- використовує метод моменту, який допомагає прискорити навчання та зменшити коливання градієнтів. Обчислюється зважене середнє значення попередніх градієнтів, яке використовується для оновлення ваг моделі.

Оновлення ваг виконується шляхом комбінування цих двох компонентів, де швидкість навчання кожного параметра адаптується в залежності від градієнта та моменту. Це дозволяє здійснювати ефективний та швидкий процес оптимізації моделі під час навчання.

У даному коді моделі, оптимізатор Adam ініціалізується без явного задання гіперпараметрів. У більшості випадків значення за замовчуванням алгоритму Adam працюють добре, але в деяких випадках може бути потрібна налаштування гіперпараметрів для досягнення кращої продуктивності моделі.

2.1.6 Функція втрат

У вибраній згортковій нейронній мережі застосовується функція втрат «Cross-Entropy Loss» (категоріальна перехресна ентропія або логарифмічна функція втрат) для оптимізації параметрів. Ця функція втрат використовується у задачах класифікації, де модель намагається визначити клас чи категорію вхідних даних. Основна мета функції втрат полягає в оцінці різниці між прогнозованими значеннями моделі та правильними мітками у тренувальних даних.

Функція втрат «Cross-Entropy Loss» має такий вигляд (2.2):

$$CCE = -\log\left(\frac{e^{sp}}{\sum_j^C e^{s_j}}\right), \quad (2.2)$$

де s – вектор вихідних значень;

C – кількість класів.

Найчастіше функція Cross-Entropy зустрічається у інтерпретації функції активації нейрона Softmax (рисунок 2.12). Для даної моделі застосовується така інтерпретація.

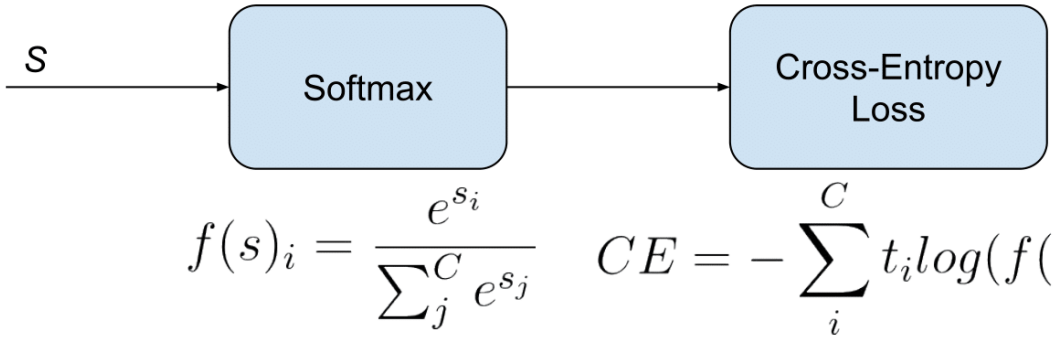


Рисунок 2.12 – Візуалізація функції Softmax [14]

Функція активації softmax працює шляхом перетворення вхідних значень на ймовірності. Вона приймає вектор значень і повертає новий вектор, в якому кожен елемент відображає ймовірність належності до певного класу. Таким чином, елементи знаходяться в діапазоні від 0 до 1, а їх сума рівна 1.

Отже, для розпізнавання жестів вирішено використовувати просту та добре вивчену архітектуру згорткової нейронної мережі, яка здатна автоматично виділяти ознаки зображення. Для реалізації системи розпізнавання жестів можна використовувати бібліотеку комп'ютерного зору OpenCV, яка надає інструменти для обробки зображень та відео, а також дає можливість реалізовувати різні алгоритми комп'ютерного зору. За допомогою OpenCV можна працювати із зображеннями та відео, включаючи їх запис, відображення та обробку.

2.4 Навчання CNN

Для навчання моделі згорткової нейронної мережі застосовується метод навчання з учителем. Метод навчання з учителем має довгу історію, що розпочалася з появою перших моделей штучних нейронних мереж у 1950-х роках. Основні ідеї і концепції методу навчання з учителем сформувалися впродовж наступних десятиліть та було зроблено багато важливих відкриттів і вдосконалень.

Одним з перших успішних застосувань методу навчання з учителем було навчання персептрона, що стало популярним у 1960-х роках. Це була одношарова штучна нейронна мережа, яка мала здатність класифікувати лінійно-роздільні дані. Однак, персептрон мав обмеження в розв'язуванні складніших задач, які не можна було лінійно розділити.

У 1990-х роках метод навчання з учителем отримав популярність. Застосування нейронних мереж для задач класифікації, регресії та інших завдань стало більш поширеним і було розроблені нові архітектури для мереж та алгоритми для навчання.

З появою глибоких нейронних мереж (Deep Neural Networks - DNN), конволюційних нейронних мереж (Convolutional Neural Networks - CNN) та рекурентних нейронних мереж (Recurrent Neural Networks - RNN), метод навчання з учителем здобув нових успіхів в багатьох областях, включаючи комп'ютерний зорі, обробку жестів та інші.

Сучасні методи навчання з учителем використовуються для вирішення складних завдань машинного та глибокого навчання. Вони вимагають великих обчислювальних ресурсів та великих наборів даних, але можуть досягти вражаючих результатів у багатьох областях дослідження.

Основна ідея методу навчання з учителем полягає в тому, що моделі надається набір даних, для яких відомі правильні відповідні їм дані, і модель має знайти правильний зв'язок між ними. Модель використовує цей набір

даних для побудови функціональної залежності між вхідними та вихідними змінними.

У цій моделі використовується метод fit() (рисунок 2.13) з бібліотеки Keras для тренування моделі. Метод fit() приймає набори даних тренування та перевірки, кількість разів проведення тренування, розмір партії (batch size) та інші параметри.

Метод fit() є основним методом для навчання моделей машинного навчання в бібліотеці Keras. Він використовується для підгонки моделі до тренувальних даних шляхом знаходження оптимальних значень ваг і зсувів (параметрів моделі) за допомогою оптимізаційних алгоритмів.

```
history = model.fit(datagen.flow(x_train, y_train, batch_size=128),
                    epochs=20, validation_data=(x_test, y_test),
                    callbacks=[learning_rate_reduction])
```

Рисунок 2.13 – Код навчання моделі

У загальному контексті машинного навчання, метод fit() можна класифікувати як один з методів навчання з учителем, оскільки він використовує цільові дані (мітки класів) для навчання моделі. В процесі навчання модель адаптується до вхідних даних і навчається робити прогнози на основі набутого досвіду.

Основні параметри методу fit():

- x: вхідні дані тренування;
- y: цільові дані тренування;
- batch_size: розмір партії (кількість прикладів), яка використовується під час однієї ітерації навчання;
- epochs: кількість разів, скільки модель буде проходити через весь набір даних тренування;

- verbose: режим виведення інформації під час навчання. Значення 0 – без виведення, 1 – виведення прогресу навчання, 2 – виведення тільки кінцевих результатів для кожної прогонки;

- callbacks: список об'єктів зворотного виклику, які викликаються на певних етапах навчання (наприклад, зменшення швидкості навчання, збереження моделі тощо);

- validation_data: дані валідації, за допомогою яких оцінюється ефективність моделі під час навчання.

Навчання моделі проводиться 20 епох (повний прохід моделі через увесь набір тренувальних даних). Метою цих епох є оновлення параметрів моделі. Кожен приклад даних в тренувальному наборі передається моделі, і на основі прогнозів моделі порівнюються з очікуваними значеннями. Потім застосовуються алгоритми оптимізації для підрахунку помилки і коригування ваг моделі.

Кількість епох визначається перед початком тренування і є параметром, який варто налаштовувати.

Результати навчання моделі (рисунок 2.14):

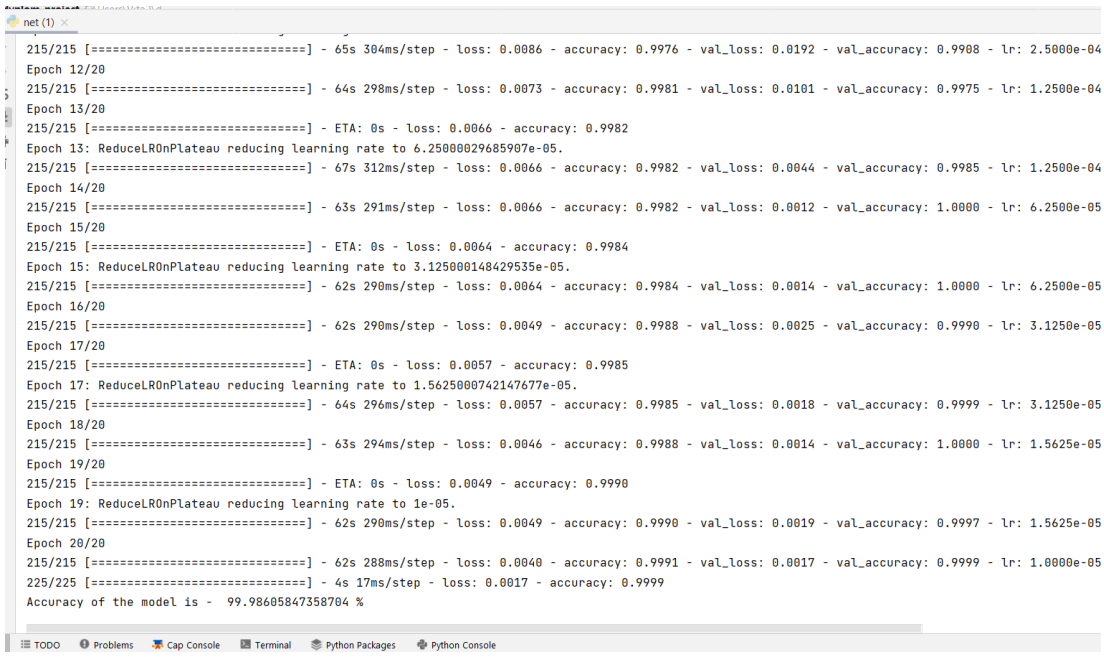


Рисунок 2.14 – Результат навчання моделі на кожній епосі

Занадто мала кількість епох може бути недостатньою для моделі, щоб вивчити закономірності в даних, тоді як занадто велика кількість епох може призвести до перенавчання моделі на тренувальних даних.

У наведеному коді (рисунок 2.15), після тренування моделі на навчальних даних та її оцінки на тестових даних, виводиться повідомлення про точність моделі.

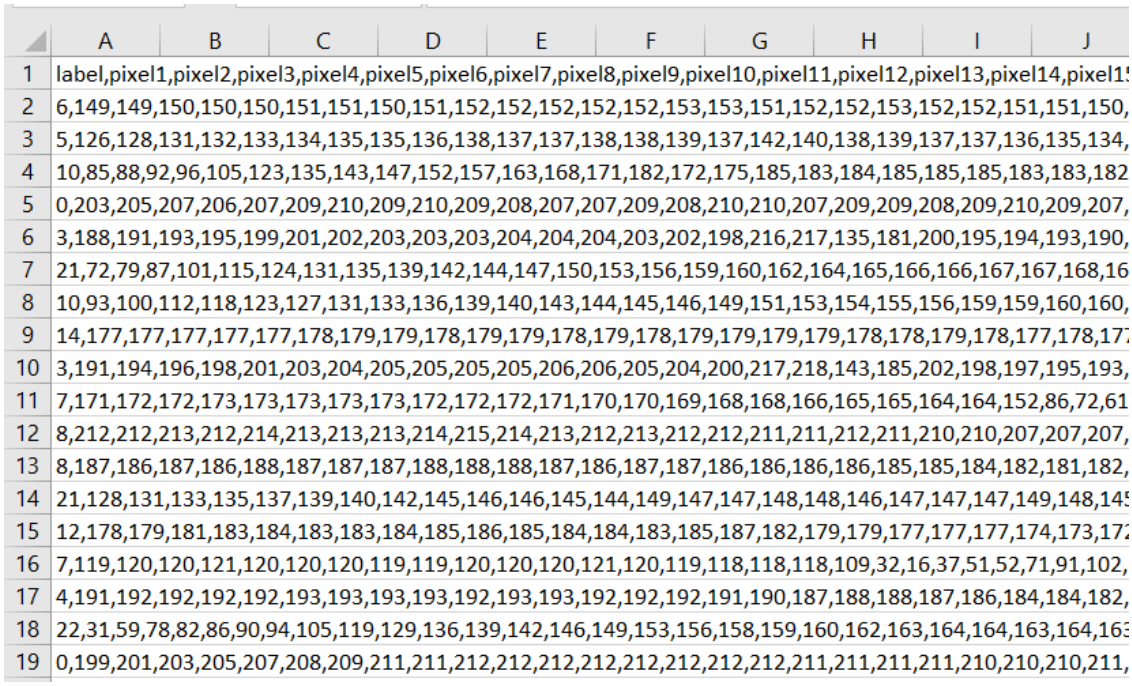
```
datagen = ImageDataGenerator(
    featurewise_center=False,  # середня значення = 0
    samplewise_center=False,  # для кожного екземпляра середнє значення = 0
    featurewise_std_normalization=False,  # розділення вхідних даних
    samplewise_std_normalization=False,
    zca_whitening=False,  # застосовується "відбілювання"
    rotation_range=10,  # вибадкове обернення зображення (від 0 до 180 градусів)
    zoom_range=0.1,  # випадкове збільшення (зум) зображення
    width_shift_range=0.1,  # випадкове зміщення зображень по горизонталі (по ширині)
    height_shift_range=0.1,  # випадкове зміщення зображень по вертикалі (по висоті)
    horizontal_flip=False,  # випадкове обертання зображень
    vertical_flip=False)  # випадкове обертання зображень
```

Рисунок 2.15 – Код генератора випадкового внесення змін до даних

Точність моделі вказує на відсоток правильно класифікованих зразків на тестових даних. Вона обчислюється як відношення кількості правильних передбачень до загальної кількості зразків у тестовому наборі даних. Чим вище точність, тим краще модель здатна передбачати класи зразків.

Першим кроком підготовки даних для навчання є перетворення та формування всіх піксельних даних із набору даних у зображення, щоб їх міг читати алгоритм.

Вигляд даних (рисунок 2.16) у файлах sign_mnist_train.csv та sign_mnist_test.csv.



	A	B	C	D	E	F	G	H	I	J
1	label,pixel1,pixel2,pixel3,pixel4,pixel5,pixel6,pixel7,pixel8,pixel9,pixel10,pixel11,pixel12,pixel13,pixel14,pixel1									
2	6,149,149,150,150,150,151,151,150,151,152,152,152,152,152,153,153,151,152,152,153,152,152,151,151,150,									
3	5,126,128,131,132,133,134,135,135,136,138,137,137,138,138,139,137,142,140,138,139,137,137,136,135,134,									
4	10,85,88,92,96,105,123,135,143,147,152,157,163,168,171,182,172,175,185,183,184,185,185,185,183,183,182									
5	0,203,205,207,206,207,209,210,209,210,209,208,207,207,209,208,210,210,207,209,209,208,209,210,209,207,									
6	3,188,191,193,195,199,201,202,203,203,203,204,204,204,203,202,198,216,217,135,181,200,195,194,193,190,									
7	21,72,79,87,101,115,124,131,135,139,142,144,147,150,153,156,159,160,162,164,165,166,166,167,167,168,16									
8	10,93,100,112,118,123,127,131,133,136,139,140,143,144,145,146,149,151,153,154,155,156,159,159,160,160,									
9	14,177,177,177,177,177,178,179,179,178,179,179,178,179,178,179,179,179,179,178,178,179,178,177,178,17									
10	3,191,194,196,198,201,203,204,205,205,205,205,206,206,205,204,200,217,218,143,185,202,198,197,195,193,									
11	7,171,172,172,173,173,173,173,173,172,172,172,171,170,170,169,168,168,166,165,165,164,164,152,86,72,61									
12	8,212,212,213,212,214,213,213,213,214,215,214,213,212,213,212,212,211,211,212,211,210,210,207,207,207,									
13	8,187,186,187,186,188,187,187,187,188,188,188,187,186,187,187,186,186,186,186,185,185,184,182,181,182,									
14	21,128,131,133,135,137,139,140,142,145,146,146,145,144,149,147,147,148,148,146,147,147,147,149,148,14									
15	12,178,179,181,183,184,183,183,184,185,186,185,184,184,183,185,187,182,179,179,177,177,177,174,173,17									
16	7,119,120,120,121,120,120,120,119,119,120,120,120,121,120,119,118,118,118,109,32,16,37,51,52,71,91,102,									
17	4,191,192,192,192,192,193,193,193,193,192,193,193,192,192,192,191,190,187,188,188,187,186,184,184,182,									
18	22,31,59,78,82,86,90,94,105,119,129,136,139,142,146,149,153,156,158,159,160,162,163,164,164,163,164,16									
19	0,199,201,203,205,207,208,209,211,211,212,212,212,212,212,212,212,212,211,211,211,211,210,210,210,211,									

Рисунок 2.16 – Вигляд набору даних для навчання та тестування

Вхідні дані для тренування передаються через datagen.flow(), що дозволяє генерувати пакети даних з використанням ImageDataGenerator. Параметр batch_size встановлює розмір партії для тренування моделі. Параметр epochs визначає кількість епох тренування.

Параметр validation_data приймає дані валідації (x_test, y_test), щоб оцінити ефективність моделі на них під час тренування.

Наступним кроком є створення генератора даних для випадкового внесення змін до даних, збільшення кількості навчальних прикладів і надання більш реалістичних зображень шляхом додавання шуму та трансформацій до різних екземплярів. Це може покращити здатність моделі до розпізнавання та класифікації об'єктів у реальному середовищі. Генератор даних може бути реалізований шляхом застосування різних технік, таких як аугментація даних, зміщення, обрізка, розмиття,

До даних також необхідно додати нормалізацію даних, рівномірно збалансувавши класи з меншою кількістю зображень. Зазвичай, нормалізація зводить значення даних до діапазону від 0 до 1 або до середнього значення 0 та стандартного відхилення 1. Це можна зробити шляхом віднімання

середнього значення та поділу на стандартне відхилення або шляхом масштабування даних до відповідного діапазону. Виконання нормалізації даних та рівномірного збалансування класів перед тренуванням моделі допомагає забезпечити більш стійкі та точні результати навчання.

2.5 Математичне забезпечення

Маємо задачу класифікації жестів.

Нехай маємо N різних класів жестів, які ми хочемо розпізнати.

Нехай кожне зображення жесту представлене у векторному форматі, наприклад вектор розміром M, де M – це кількість пікселів у зображенні. Позначимо цей вектор як x.

Тоді сформулюємо задачу класифікації таким чином. Дано набір тренувальних прикладів $\{(x\_1, y\_1), (x\_2, y\_2), \ldots, (x\_n, y\_n)\}$, де x_i – зображення жесту, y_i – мітка класу жесту, яка відповідає цьому зображенню. Мітка класу є цілочисельним значенням від 1 до N, де N – загальна кількість класів жестів.

Мета полягає у побудові моделі $f(x)$, яка приймає зображення жесту x та передбачає його клас – y. Модель $f(x)$ може бути, наприклад, нейронною мережею з різними шарами згортки, пулінгу та повнозв'язаними шарами.

Після побудови моделі ми можемо тренувати її на наборі тренувальних даних, використовуючи різні методи оптимізації. Мета тренування полягає у підборі параметрів моделі так, щоб мінімізувати функцію втрат та точно класифікувати жести.

Після тренування моделі її можна використовувати для класифікації нових зображень жестів. Для кожного нового розпізнавання жесту подається його зображення на вхід моделі $f(x)$, і модель передбачає його клас y.

Висновки до розділу 2

У цьому розділі розглянуте проектування системи та наведено інструменти реалізації.

Також описана архітектура згорткової нейронної мережі, реалізоване програмне забезпечення, яке дозволяє користувачу перекладати жести за допомогою згорткової нейронної мережі (CNN). Описано функцію активації нейронів для згорткової нейронної мережі, алгоритм оптимізації Adam, функція втрат.

Для тренування моделі використовуються підготовлені дані – датасет SL-MNIST.

Для навчання моделі використаний метод навчання з учителем.

Також у розділі описано математичне забезпечення. Для задачі класифікації створено модель $f(x)$, яка приймає зображення тексту і передбачає його клас.

3 ТЕСТУВАННЯ СИСТЕМИ

В даному розділі розглядається тестування створеного програмного забезпечення для демонстрації роботи штучного інтелекту в системі зчитування і перекладу жестів у текст. Тестування є важливою частиною розроблення системи. Цей процес дозволяє оцінити ефективність і точність розробленої моделі, перевірити її здатність правильно класифікувати різні жести та визначити можливі обмеження або проблеми.

3.1 Побудова зображення жестів із вибірки даних

Вибірка даних складається з масивів пікселів, за допомогою яких навчається CNN. Якщо побудувати за допомогою коду (рисунок 3.1) зображення жестів із набору пікселів – отримаємо чорно-білі зображення (рисунок 3.2).

Розглянемо код та результат його виконання:

```
def show_images(images,labels):
    fig,ax = plt.subplots(2,5)
    fig.set_size_inches(10, 6)
    k =0
    for i in range(2):
        for j in range(5):
            ax[i,j].imshow(images[k] , cmap='gray')
            ax[i,j].set_title(str(unique_labels[np.argmax(y_train[k])]))
            k = k+1;
    plt.tight_layout()
```

Рисунок 3.1 – Код побудови перших десяти зображень жестів із навчальної вибірки

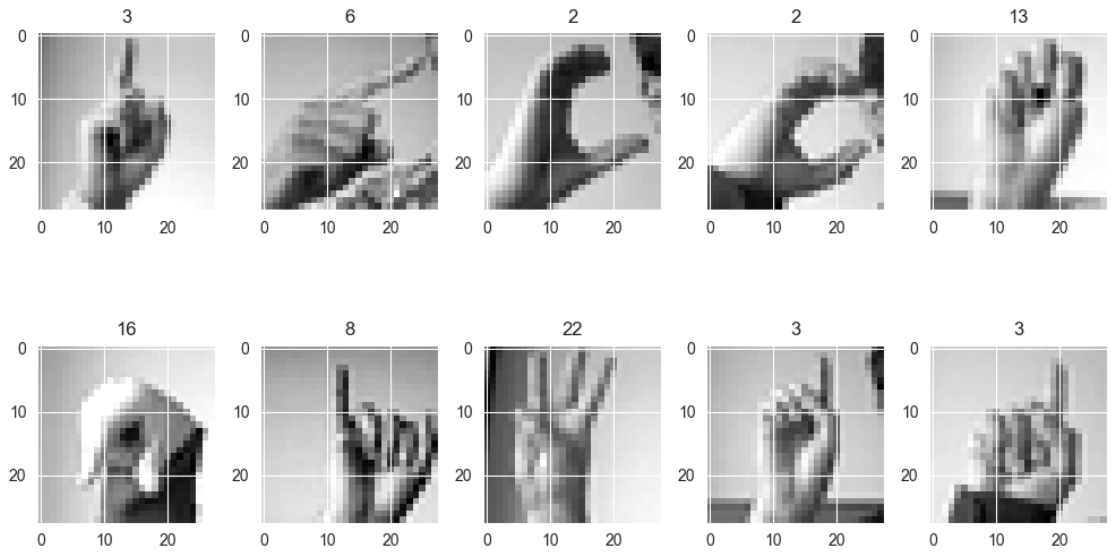


Рисунок 3.2 – Результат побудови зображень із набору даних для тренування

3.2 Побудова графіків функції втрат та функції точності

Після навчання згорткової нейронної мережі побудовано графіки функції втрат та функції точності розпізнавання (кресленик ІС92.170БАК.004 Д4) за допомогою коду (рисунок 3.3).

Додання графіка функції точності та втрат є важливим етапом у валідації та аналізі навчання моделі розпізнавання жестів. Графік відображає залежність точності та втрат від кількості навчальних епох.

Розглянемо ці графіки (кресленик ІС92.170БАК.004 Д4).

Точність визначається як відсоток правильних передбачень, зроблених протягом однієї прогонки. Втрата вимірюється відповідною функцією втрат, яка описує розбіжність між реальними і передбаченими значеннями. Цей показник використовується для оптимізації параметрів нейронних мереж.

```
def predictions_to_labels(pred):
    labels =[]
    for p in pred:
        labels.append(unique_labels[np.argmax(p)])
    return labels
```

Рисунок 3.3 – Функція для побудови графіків точності та втрат

Для побудови графіків цих метрик використовувалися інструменти бібліотеки Matplotlib.

На графіках (кресленик ІС92.170БАК.004 Д4) можна помітити, що точність моделі з часом збільшується, а значення втрат зменшуються. Це свідчить про те, що модель ефективно навчається розпізнавати різні жести і здатна знаходити правильні зв'язки між вхідними зображеннями та їх класами.

Початково точність може бути низькою, а втрати високими, що є нормальним на ранніх етапах навчання моделі. Проте, з кожною епохою точність зростає, а втрати зменшуються, що свідчить про навчання моделі.

Якщо було багато епох навчання, але точність залишається низькою, можливо, є необхідність у внесенні змін у роботу моделі або навчального процесу. Якщо точність збільшується, а втрати зменшуються з кожною епохою, це підтверджує ефективність моделі і її здатність до розпізнавання жестів.

Графіки функцій точності та втрат (кресленик ІС92.170БАК.004 Д4) є важливим інструментом для оцінки якості моделі та визначення оптимальної кількості епох навчання. Вони надають візуальне відображення прогресу навчання і допомагають зрозуміти, чи модель навчається розпізнавати жести.

3.3 Запуск програми

Програмне забезпечення використовує веб-камеру для зчитування та оброблення зображення із жестом, щоб в подальшому розпізнати цей жест та вивести результат текстом.

Для тестування програми на живих прикладах було вибрано такі жести літер ASL (американської жестової мови) «P», «F», «O», «H», «Y», «V». Точність перекладу може залежати від освітлення приміщення, від точності відтворення жесту та від якості веб-камери яка використовується.

Результати перекладу жестів у текст:

Жест «P» (рисунок 3.4).

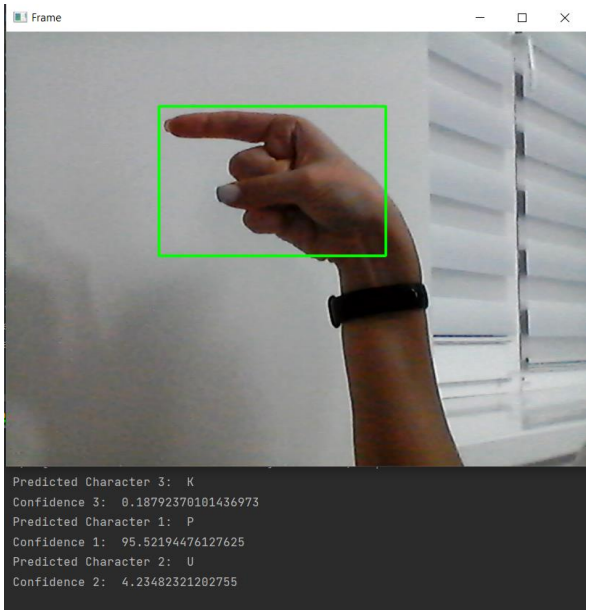


Рисунок 3.4 – Результат перекладу жесту «P»

Жест «F» (рисунок 3.5).

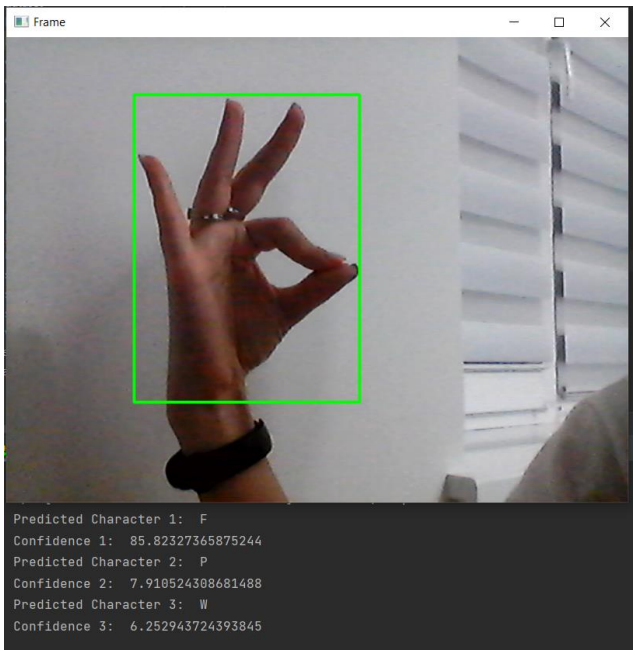


Рисунок 3.5 – Результат перекладу жесту «F»

Жест «W» (рисунок 3.6).

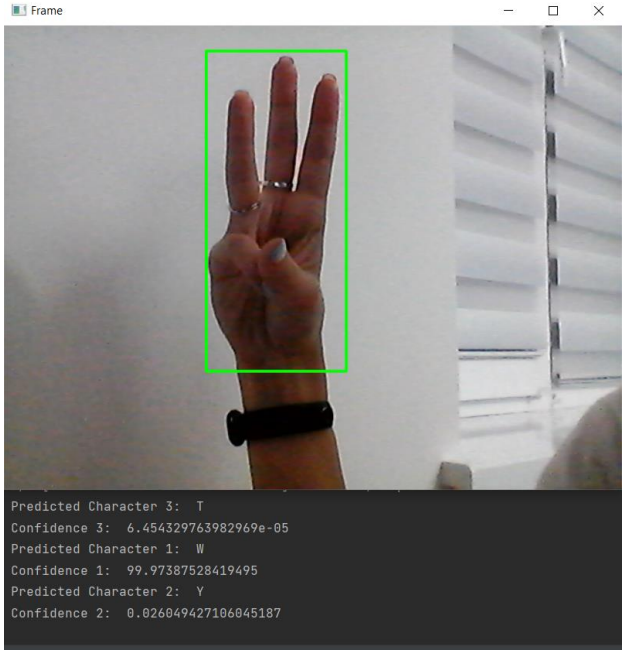


Рисунок 3.6 – Результат перекладу жесту «W»

Реалізована згорткова нейронна мережа навчалась на вибірці даних SL-MNIST (доступні для навчання таких нейронних мереж). Під час тестування

використовувались реальні жести людини. Тому точність використання для деяких жестів була дещо менша.

Результат прогнозування перекладу жестів виводиться передбаченням класифікування жесту. Як бачимо результати передбачення нейронної мережі досить високі.

3.4 Оцінювання результатів

З оцінки результатів тестування видно, що побудовані зображення жестів із набору пікселів відображаються як чорно-білі зображення. Це підтверджує правильність коду побудови зображень та датасету.

Графіки функції втрат та функції точності (кресленик ІС92.170БАК.004 Д4) є важливими інструментами для оцінки якості моделі розпізнавання жестів. З результатів видно, що точність моделі з часом збільшується, а значення втрат зменшуються. Це свідчить про ефективне навчання моделі та її здатність розпізнавати різні жести.

Результати тестування на реальних жестах показали досить високу точність передбачень нейронної мережі. Переклад жестів "P", "F" і "W" був виконаний правильно, що підтверджує успішність програми у розпізнаванні жестів.

Загалом, результати тестування можна вважати вдалими, оскільки модель успішно розпізнає жести з високою точністю.

3.5 Керівництво користувача

Для запуску програми потрібно запустити файл *run.py* в папці з проектом (рисунок 3.7).

Після запуску має з’явитись вікно в якому буде відображатися пряма трансляція через веб-камеру.

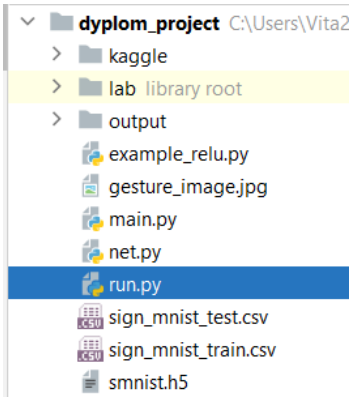


Рисунок 3.7 – Розташування файлу запуску програми у папці проекту

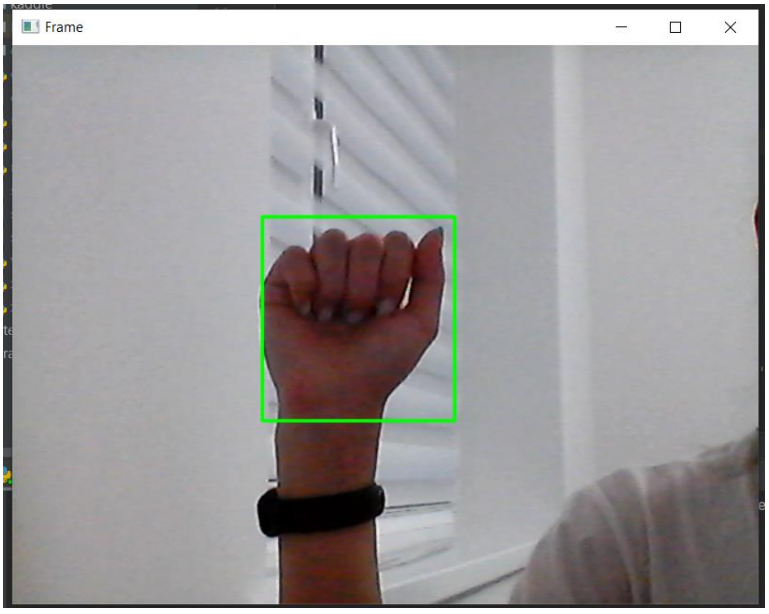


Рисунок 3.8 – Вікно програми із прямою трансляцією

На рисунку 3.8 видно, як виділяється жест користувача за допомогою зеленої рамки. Тоді користувач може зобразити жест і нажати «Space», щоб зафіксувати зображення.

Після фіксування зображання відображається результат передбачення як на рисунках 3.4 – 3.6.

Щоб вийти з програми потрібно натиснути «Esc».

Висновки до розділу 3

Отже, у цьому розділі було розглянуто тестування програмного забезпечення для системи перетворення мови жестів у текст на основі штучного інтелекту.

Було перетворено 3 жести рук людини у текст. Точність розпізнавання була досить високою. Для першого жесту точність перекладу склала – 95,5%, для другого – 85,8%, для третього 99,9%

Результати проведення тестування підтверджують задовільну роботу ПЗ.

Створено керівництво користувача, в якому описується інструкція до користування програмним забезпеченням.

Було також побудовано графіки функцій втрат та точності (кресленик ІС92.170БАК.004 Д4)за допомогою яких можна було зрозуміти, що модель навчилась розпізнавати жести та перетворювати їх у текст.

ВИСНОВКИ

Протягом виконання дипломного проекту покращені практичні та теоретичні навички із проектування систем та використання штучного інтелекту в задачах розпізнавання жестів.

У даному дипломному проекті описаний проведений аналіз предметної області, спроектовано систему перекладу мови жестів у текст за допомогою штучного інтелекту та розроблене програмне забезпечення для роботи нейронної мережі за допомогою мови програмування Python та бібліотеками Keras, Tensorflow, OpenCV. За допомогою цього програмного забезпечення користувач може перекласти літери мови жестів у текст.

Для системи перекладу мови жестів у текст вибрана згорткова нейронна мережа (CNN), що є найоптимальнішим варіантом для розв’язання задачі розпізнавання для цієї системи, так як були важливі такі критерії як точність результатів та швидкість.

У першому розділі було проведено аналіз наявних існуючих рішень, після чого зроблено висновок, як саме має виглядати система перекладу мови жестів у текст за допомогою штучного інтелекту.

У другому розділі детально розглядається вибір методу розробки системи, її проектування, інструменти реалізації, згорткова нейронна мережі, набір даних для навчання мережі, архітектура нейронної мережі та функції для навчання мережі. Також у цьому розділі описано математичну постановку задачі. Математична постановка допомагає чітко визначити об'єкт дослідження та його властивості. Вона дозволяє точно сформулювати поставлену проблему та встановити критерії її вирішення.

Також описано набір даних для навчання мережі SL-MNIST. Кожне зображення у цьому наборі даних представлено у форматі 28x28 пікселів і відтворює жест, який виконує людина. Цей набір даних дозволяє тренувати та тестувати моделі нейронних мереж на розпізнавання жестів, що використовуються в американській жестовій мові.

У розділі тестування детально розглянуто тестування нейронної мережі на жестах, які показувала людина на веб-камеру, за допомогою якої нейронна мережа перекладала жести у текст. Набір даних для навчання складався із уже перетворених зображень у пікселі, тому точність розпізнавання жестів живої людини була дещо меншою.

Для програмного забезпечення, яке розроблене, створено керівництво користувача, в якому описуються кроки користуванням програмою.

Можливими напрямками розвитку системи перекладу мови жестів у текст на основі штучного інтелекту є реалізації даного проектування, навчання моделі з високою точністю розпізнавати жести, та розширення розпізнавання мов дактилю.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. M. Ayaz Ahmad, Volodymyr Gorokhovatskyi, Iryna Tvoroshenko, Nataliia Vlasenko, Syed Khalid Mustafa (2021) The Research of Image Classification Methods Based on the Introducing Cluster Representation Parameters for the Structural Description, International Journal of Engineering Trends and Technology, 69(10), pp. 186-192.
2. MotionSavvy UNI: 1st sign language to voice system. URL: https://www.indiegogo.com/projects/motionsavvy-uni-1st-sign-language-to-voice-system#/ (дата звернення: 27.04.2023).
3. Product Specs: Signall 1.0. URL: https://oarklibrary.com/viewers/pdf/viewer.html?file=https%3A%2F%2Fapp.oarklibrary.com%2Fproxy%2F%3Furl%3Dhttps%253A%252F%252Fcurriculumoark.blob.core.windows.net%252Foark-library-container%252Fexternal-search-provider%252F4a514ab0-dbff-49f1-87c5-0514a285945e%252Fa46873bc-3324-4954-8f0d-a25a1ba2585f%252F400a824d-07f2-4a05-a139-07372a1aceb5.pdf (дата звернення: 27.04.2023).
4. Dorner, B. (1994). Chasing the colour glove: Visual hand tracking (Doctoral dissertation, Theses (School of Computing Science)/Simon Fraser University).
5. Контроллер жестами Myo Armband: обзор. URL: https://futurenow.com.ua/ru/kontroler-zhestamy-myo-armband-harakterystyky-obzor-y-prymenenye/ (дата звернення: 29.04.2023).
6. This Futuristic Armband Lets You Control Your Computer Like Magic. URL: https://time.com/4173507/myo-armband-review/ (дата звернення: 01.05.2023).
7. GestureSign. URL: https://apps.microsoft.com/store/detail/gesturesign/9N45WQVK2QQW?hl=en-us&gl=us (дата звернення: 01.05.2023).

8. GestureSign makes Windows 10 even more touch-friendly. URL: https://www.windowscentral.com/gesturesign-makes-windows-10-even-more-touch-friendly (дата звернення: 01.05.2023).

9. Y. LeCun et al. Deep learning URL: https://www.nature.com/articles/nature14539 (дата звернення: 08.05.2023) .

10. Masood S., Srivastava A., Thuwal H.C., Ahmad M. (2018) Real-Time Sign Language Gesture (Word) Recognition from Video Sequences Using CNN and RNN. In: Bhateja V., Coello Coello C., Satapathy S., Pattnaik P. (eds) Intelligent Engineering Informatics. Advances in Intelligent Systems and Computing, vol 695. Springer, Singapore

11. J. Gua et al. «Recent Advances in Convolutional Neural Networks». URL: https://arxiv.org/abs/1512. 07108/ (дата звернення: 10.05.2023).

12. Sign Language MNIST. URL: https://www.kaggle.com/datasets/datamunge/sign-language-mnist (дата звернення: 12.05.2023).

13. The 26 letters and 10 digits of American Sign Language (ASL). URL: https://www.researchgate.net/figure/The-26-letters-and-10-digits-of-American-Sign-Language-ASL_fig1_328396430 (дата звернення: 12.05.2023).

14. Understanding Categorical Cross-Entropy Loss, Binary Cross-Entropy Loss, Softmax Loss, Logistic Loss, Focal Loss and all those confusing names. URL: https://gombru.github.io/2018/05/23/cross_entropy_loss/ (дата звернення: 13.05.2023).

15. Бахтеев Д.В. Компьютерное зрение и распознавание образов в криминалистике / Д.В. Бахтеев // Российское право: образование, практика, наука. – 2019. – №3. – С. 66-73.

16. Булатников Е.В. Сравнение библиотек компьютерного зрения для применения в приложении, использующем технологию распознавания плоских изображений / Е.В. Булатников, А.А. Гоева // Вестник МГУП имени Ивана Федорова. – 2015. – №6. – С. 85-91.

17. Давидов М.В., Нікольський Ю.В. Пасічник О.В. Дослідження ефективності методів розпізнавання у моделях жестової мови / М.В. Давидов, Ю.В. Нікольський, О.В. Пасічник // Вісн. Нац. ун-ту «Львів. Політехніка». – 2008. – № 621. – С. 117-123.

18. Сіряк Р.В. Модель обробки потокових даних для розпізнавання окремих одиниць жестової мови / Р.В. Сіряк, І.С. Скарга-Бандурова // Вісник Національного технічного університету «ХПІ». Серія: Інформатика та моделювання. – 2018. – № 42. – С. 73-81.

19. Pigou, L., Dieleman, S., Kindermans, P.-J., Schrauwen, B.: Sign language recognition using convolutional neural networks [Electronic resource]. – 2015. – Access mode: https://ieeexplore.ieee.org/document/5204291/ (lastaccess: 31.05.2021). – Title from the screen.

20. Дятлов Е.И. Машинное зрение (аналитический обзор) / Е.И. Дятлов // Математичні машини і системи. – 2013. – № 2. – С. 32-40.

21. Яшина М.В., Толмачев А.А. Методы распознавания образов для оценки характеристик пешеходных потоков // T-Comm: Телекоммуникации и транспорт. – 2017. – Том II. – №8. – С. 45-51.

22. Нагапетян В.Э. Распознавание жестов ручной азбуки asl / В.Э. Нагапетян // Вестник РУДН. Серия Математика. Информатика. Физика. – 2013. – №2. – С.105-113.

23. What is Python. URL: https://www.w3schools.com/python/python_intro.asp (дата звернення: 13.05.2023).

24. TensorFlow module. URL: https://www.tensorflow.org/learn (дата звернення: 13.05.2023).

25. Import module. URL: https://docs.python.org/3/reference/import.html. (дата звернення: 13.05.2023).

26. Flah P. (2015) Machine learning. The science and art of building algorithms that extract knowledge from data, Moscow, Russia: DMK Press, 400 p.

27. Nong Ye. (2013) Data Mining: Theories, Algorithms, and Examples, Florida, USA: CRC Press, 349 p.

28. Davis, J.W., & Bobick, A. F. (1998, November). A robust human-silhouette extraction technique for interactive virtual environments. In International Workshop on Capture Techniques for Virtual Environments (pp. 12-25). Springer, Berlin, Heidelberg.

29. Wang, R.Y., & Popović, J. (2009). Real-time hand-tracking with a color glove. ACM transactions on graphics (TOG), 28(3), 1-8.

30. Daradkeh Y.I., Gorokhovatskyi V., Tvoroshenko I., Gadetska S., and Al-Dhaifallah M. (2021) Methods of Classification of Images on the Basis of the Values of Statistical Distributions for the Composition of Structural Description Components, IEEE Access, 9, pp. 92964-92973.

31. Творошенко І.С. (2021) Технології прийняття рішень в інформаційних системах: навч. посібник. Харків: ХНУРЕ, 120 с.