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

Факультет інформатики та обчислювальної техніки

Кафедра інформаційних систем та технологій

Індивідуальний дослідницький проєкт

на здобуття ступеня бакалавра
за освітньо-професійною програмою «Інтегровані інформаційні системи»
спеціальності 126 «Інформаційні системи та технології»
на тему: «Веб застосунок для розпізнавання мови жестів за допомогою нейронних мереж»»

Виконав:
студент IV курсу, групи ІА-81
Селезень Артем Васильович _____

Керівник:
Доцент каф. ІСТ, к. т. н.
Шимкович Володимир Миколайович _____

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

Київ – 2022 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 «Інтегровані інформаційні системи»

ЗАВДАННЯ
на індивідуальний дослідницький проєкт студенту
Селезню Артему Васильовичу

1. Тема проєкту «Веб застосунок для розпізнавання мови жестів за допомогою нейронних мереж»», керівник проєкту Шимкович Володимир Миколайович, доцент каф. ІСТ, к. т. н.

2. Термін подання студентом проєкту: 15 червня 2022 року

3. Вихідні дані до проєкту: Веб застосунок для розпізнавання мови жестів за допомогою нейронних мереж.

4. Зміст пояснювальної записки: аналіз предметної області, аналіз існуючих рішень, вибір технологій та реалізації, реалізація програмної частини застосунку з подальшим тестуванням.

5. Перелік графічного матеріалу (із зазначенням обов’язкових креслеників, плакатів, презентацій тощо): діаграма діяльності, діаграма прецедентів, діаграма розгортання, діаграма послідовностей.

6. Дата видачі завдання 1 грудня 2021 року

Календарний план

№ з/п	Назва етапів виконання проєкту	Термін виконання етапів проєкту	Примітка
1	Аналіз предметної області	09.05.2022	
2	Аналіз існуючих рішень	16.05.0222	
3	Попередня підготовка текстових матеріалів	23.05.2022	
4	Реалізація користувацької частини	25.05.2022	
5	Реалізація програмної частини	30.05.2022	
6	Оформлення документації дослідницького проєкту	06.06.2022	

Студент Артем СЕЛЕЗЕНЬ

Керівник Володимир ШИМКОВИЧ

АНОТАЦІЯ

Селезень А.В. Веб-застосунок для розпізнавання мови жестів за допомогою нейронних мереж. КПІ ім. Ігоря Сікорського, Київ, 2022.

Проєкт містить 64 сторінки тексту, 43 рисунка, 1 таблицю, 37 посилання на 29 джерел та 4 конструкторскі документи.

Ключові слова: веб-застосунок, нейронні мережі, мова жестів, глибинне навчання, машинне навчання, комп’ютерний зір.

Об’єкт розробки: веб-застосунок для розпізнавання мови жестів.

Мета розробки – запровадити альтернативний спосіб комунікації між людьми з проблемами голосового або слухового апарату та людьми без подібних порушень.

В результаті виконання індивідуального дослідницького проєкту поставлена мета була досягнута, веб-застосунок може отримати файл з жестом та перекласти його у текстовий формат. Під час користування веб-застосунком не використовується пам’ять та ресурси пристрою користувача.

SUMMARY

Selezen A. Web application for sign language recognition using neural networks. KPI Igor Sikorsky, Kyiv, 2022.

The project contains 64 pages of text, 43 figures, 1 table, 37 references to 29 sources 4 design documents.

Keywords: web application, neural networks, sign language, depth learning, machine learning, computer vision.

The object of development is a web application for sign language recognition.

The purpose of the development is to introduce an alternative way of communicating with people with voice or hearing problems and people without such disorders.

As a result of the diploma project, the goal was achieved, the web application can obtain a file with rigidity and translate it into text format. User’s memory and resources are not used while using the web application.

Номер рядка	Формат	Позначення	Найменування	Кільк. аркушів	Номер екзем.	Примітка
1			Документація загальна			
2						
3			Знову розроблена			
4						
5	А4	ІА81.230БАК.003 ПЗ	Пояснювальна записка	64		
6	А3	ІА81.230БАК.003 Д1	Веб-додаток для	1		
7			розпізнавання мови жестів.			
8			Діаграма діяльності.			
9	А3	ІА81.230БАК.003 Д2	Веб додаток для	1		
10			розпізнавання мови жестів.			
11			Технологічна схема			
12	А3	ІА81.230БАК.003 Д3	Веб-додаток для	1		
13			розпізнавання мови жестів.			
14			Діаграма прецедентів.			
15	А3	ІА81.230БАК.003 Д4	Веб-додаток для	1		
16			розпізнавання мови жестів.			
17			Діаграма розгортання.			
18						
19						
20						
21						
22						
23						
24						
25						
26						
27						
28						

Зм.	Аркуш	№ докум.	Підпис	Дата	ІА81.230БАК.003 ТП			
Розроб.		Селезень А.В.			Веб-додаток для розпізнавання мови жестів. Відомість проєкту	Літ.	Аркуш	Аркушів
Керівн.		Шимкович В.М.				Т	1	64
						КПІ ім. Ігоря Сікорського Група ІА-81		
Затв.								

Пояснювальна записка
до індивідуального дослідницького проєкту на тему: «Веб-застосунок для розпізнавання мови жестів за допомогою нейронних мереж»»

Київ – 2022 року

ЗМІСТ

ПЕРЕЛІК ТЕРМІНІВ ТА СКОРОЧЕНЬ 4
ВСТУП 6
1 ДОСЛІДЖЕННЯ ПРЕДМЕТНОЇ ОБЛАСТІ 8
1.1 Актуальність задачі 8
1.2 Опис наявних даних для аналізу 8
1.3 Попередній аналіз даних 10
1.4 Постановка задачі 11
2 МАТЕМАТИЧНІ ОСНОВИ РОБОТИ 14
2.1 Огляд задачі оптичного розпізнавання жестів 14
2.2 Огляд нейронних мереж 14
2.2.1 Штучний нейрон 15
2.2.2 Функція активації 16
2.2.3 Сигмоїда 17
2.2.4 Гіперболічний тангенс 18
2.2.5 ReLU 20
2.2.6 Arctan 21
2.3 Архітектури нейронних мереж 21
2.4 Огляд сучасних глибоких нейронних мереж 29
2.4.1 VGGnet (Visual Geometry Group network) 29
2.4.2 ResNet (Residual neural network) 30
2.4.3 LeNet(LeNet-5) 32
2.4.4 AlexNet 33
2.4.5 GoogLeNet 34
2.5 Огляд існуючих рішень 36
2.5.1 Wecapable.com 36

2.5.2 Hand Talk Translator 37
2.5.3 Omnibridge.ai 37
3 ВИБІР ТЕХНОЛОГІЙ ТА РЕАЛІЗАЦІЇ 40
3.1 Аналіз мов програмування 40
3.1.1 Мова програмування C++ 40
3.1.2 Мова програмування R 41
3.1.3 Мова програмування Python 42
3.2 Аналіз фреймворків для розробки нейронних мереж 42
3.2.1 Бібліотека Numpy 42
3.2.2 Бібліотека Pandas 43
3.2.3 Бібліотека Matplotlib 44
3.2.4 Бібілотека Seaborn 44
3.2.5 Бібліотека Sci-kit Learn 45
3.2.6 Бібліотека Tensorflow 46
3.2.7 Бібліотека PyTorch 47
3.2.8 Бібліотека Keras 47
3.3 Веб фреймворки 48
3.3.1 Фреймворк Django 48
3.3.2 Фреймворк Flask 49
4 ПРОГРАМНА РЕАЛІЗАЦІЯ СИСТЕМИ 52
4.1 Короткий опис системи 52
4.2 Обробка даних 52
4.3 Модель нейронної мережі 53
4.4 Реалізація веб-застосунку 57
ВИСНОВКИ 61
ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ 63

ПЕРЕЛІК ТЕРМІНІВ ТА СКОРОЧЕНЬ

MNIST database (англ. Modified National Institute of Standards and Technology) – Модифікована база данних Національного інституту стандартів і технологій;

ORM (англ. Object-relational mapping) – Об'єктно-реляційна проекція;

WLASL (англ. World Level American Sign Language) – Американської мови жестів світового рівня;

YOLOv3 (англ. You Only Look Once) – сімейство архітектур і моделей виявлення об'єктів на зображені в режимі реального часу;

Google Play – магазин додатків від компанії Google;

ReLU (англ. Rectifier Linear Unit) – одна з функцій активації в нейронній мережі;

FPS (англ. Frames per second) – кількість кадрів за секунду;
OpenCV (англ. Open Source Computer Vision Library) - бібліотека комп’ютерного бачення з відкритим вихідним кодом;
Json (англ. JavaScript Object Notation) – текстовий формат файлу;

ILSVRC (англ. ImageNet Large Scale Visual Recognition Challenge) – щорічний конкурс комп’ютерного зору, розроблений на основі підмножини загальнодоступного набору даних комп’ютерного зору під назвою ImageNet;

Top5 error – метод бенчмаркінгу моделей машинного навчання;

API(англ. Application Programming Interface) – прикладний програмний інтерфейс;

Pythonic – прикметник, що описує підхід до комп’ютерного програмування, який узгоджується з основоположною філософією мови програмування Python;

Deep learning – тип машинного навчання та штучного інтелекту, який імітує спосіб отримання людьми певних знань;

Data_science –це область дослідження, яка має справу з величезними обсягами даних, використовуючи сучасні інструменти та методи, щоб знайти

невидимі закономірності, отримати значущу інформацію та прийняти бізнес-рішення;

HTTP(англ.Hypertext Transfer Protocol) – це протокол прикладного рівня для передачі гіпермедійних документів;

ONNX(англ. Open Neural Network Exchange) – це відкритий формат, створений для представлення моделей машинного навчання;

REST(англ. representational state transfer) – архітектурний стиль для розподілених гіпермедійних систем.

ВСТУП

Протягом всього існування людства комунікація була одним з найважливіших факторів його розвитку. Саме для обміну інформації між людьми своєї та інших націй були створені мови. Зараз вивчення мов не є чимось незвичайним, навіть навпаки, стало невід'ємною частиною освіти. Але існують люди, які по деяким причинам не можуть використовувати звичні нам звуки або не можуть їх чути. У такому випадку почали використовуватися мови жестів, оскільки для її сприйняття та відтворення необхідні лише зір та жестикуляція.

Свідком спілкування між людьми мовою жестів можна стати скрізь: у громадському транспорті, в магазинах, в кафе та інших багатолюдних містах. Тобто нечуючі люди нерідко беруть активну учать у житті суспільства, а також невід'ємною частиною розвитку суспільства. На жаль, не всі люди здатні зрозуміти та поговорити з людьми, у яких порушення розвитку слухового або голосового апарату. Звісно, мови жестів можна навчитися, або уникати її використання за допомогою письмової мови, як це робиться наприклад на касах магазинів. Продавець просто вказує на суму, що покупець повинен сплатити. Проте можуть скластися ситуації, в яких не можливо буде використати інструменти для написання повідомлення, або просто хочеться додати саме живого спілкування.

Розпізнавання об'єктів за допомогою нейронних мереж – чудовий засіб для використання у створенні посередником для розуміння двох людей, що не знають мови один одного, оскільки мова жестів складається з міміки та жестикуляції. Ще один аргумент на користь нейронних мереж – їх точність. Цей параметр можна покращити майже до абсолютно точного за допомогою різних використання різних архітектур, алгоритмів навчання та самої вибірки даних для навчання. Навіть звичайні онлайн перекладачі, які тлумачать з однієї мови на іншу, наприклад Google translate з англійської на українську, мають неточності у перекладі.

Об’єктом розробки веб-застосунок для розпізнавання мови жестів.

Мета розробки – запровадити альтернативний спосіб комунікації між людьми з проблемами голосового або слухового апарату та людьми без подібних порушень.

Завдання цього проєкту:

– дослідження предметної області, для визначення критеріїв за яким буде оцінюватися наш проєкт та його аналоги;

– на основі проведення досліджень існуючих рішень, визначити вимоги до проєкту;

– огляд технологій;

– реалізація застосунку та аналіз рішень що були прийняті при розробці застосунку, опис реалізації застосунку та його компонентів.

Індивідуально дослідницький проєкт складається з наступних розділів: вступ, дослідження предметної області, математичні основи роботи , вибір технологій та реалізації, програмна реалізація системи, висновки, список використаних джерел із 29 найменувань, 1 додатку. Графічна частина включає 4 креслеників формату А3. Загальний обсяг 65 сторінки.

1 ДОСЛІДЖЕННЯ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Актуальність задачі

Дана задача є дуже актуальною, бо комунікація між людьми завжди буде основою розвитку суспільства. Також на основі результатів, отриманих з цієї роботи можна розпочати більш розробку програмного забезпечення. Наприклад завантажити програму локально у будь-якому сервісному центрі, громадському місті, де спілкування є необхідним для надавання послуг, або створити мобільний додаток, що відправляє та отримує данні з сервера або хмари. Також спілкування між людьми з порушеннями голосового або слухового апарату з людьми без таких порушень є необхідними для безпроблемного розвитку, наприклад дитини. Спілкування з ровесниками додає впевненості у собі, знижує рівень тривожності і допоможе уникнути майбутніх проблем у пристосуванні в суспільстві шляхом підвищення самооцінки.

1.2 Опис наявних даних для аналізу

Не загалу відомий факт, що мова жестів в різних країнах різна. Нечуючий українець не буде здатний порозумітися з іноземцем, у якого також проблеми з слуховим або голосовим апаратом. Кожна мова жестів унікальна, бо звукова мова виникла набагато пізніше. Кількість звуків доволі стисла, порівняно з кількістю жестів, котрі людина може відтворити. Не дивлячись на кількість підтримки з боку Українських влади та суспільства, створення Центру Української Жестової Мови, різних словників, додатків, курсів та відео уроків, збірки для навчання нейронної мережі української жестової мови ще не існує. Звісно можна знайти що найменше по 3 або 4 відео на кожне слово, але для навчання цього буде не достатньо. Саме тому у цьому проекті була взята збірка Американської мови жестів світового рівня «WLASL» з сайту https://dxli94.github.io/WLASL/. Ця колекція містить:

– на відміну від збірки з MNIST, яка представляє лише файли фотографії літер та цифр, що є лише абеткою, та не є повноцінною мовою жестів, містить у собі повні слова, які записані на відео;

– вся колекція супроводжується файлом .json формату, в якому досить детально розписана вся інформація про значення слова, його відео або уривок іншого відео;


```
"root" : [ 7 items
  0 : { 2 items
    "gloss" : string "book"
    "instances" : [ 40 items
      0 : { 11 items
        "bbox" : [ 4 items
          0 : int 385
          1 : int 37
          2 : int 885
          3 : int 720
        ]
        "fps" : int 25
        "frame_end" : int -1
        "frame_start" : int 1
        "instance_id" : int 0
        "signer_id" : int 118
        "source" : string "aslbrick"
        "split" : string "train"
        "url" : string "http://aslbricks.org/New/ASL-Videos/book.mp4"
        "variation_id" : int 0
        "video_id" : string "69241"
      }
```


Рисунок 1.1 – Приклад одного екземпляру json
для подальшої розпаковки[1]

– містить більше 2000 слів, яким нейромережа може вчитися більше ніж 20 000 по відеофрагментів.

1.3 Попередній аналіз даних

У .json файлі є різні додаткові значення до екземпляру відео, а саме:

– gloss: str, файл даних структурований/класифікований на основі блиску знаків, а саме, міток;

– bbox: [int], обмежувальна рамка, виявлена за допомогою конвенції YOLOv3 (xmin, ymin, xmax, ymax). Відповідно до умов OpenCV, (0, 0) є верхнім лівим кутом;

– fps: int, частота кадрів (=25), що використовується для декодування відео, як у папері;

– frame_start: int, початковий кадр глянцю у відео (декодування з FPS=25), індексований з 1;

– frame_end: int, кінцевий кадр глянцю у відео (декодування з FPS=25). -1 означає, що глянець закінчується на останньому кадрі відео;

– instance_id: int, ідентифікатор екземпляра в тому самому класі/глянці;

– signer_id: int, ідентифікатор підписувача. джерело: str, ідентифікатор рядка для вихідного сайту;

– split: str, вказує, до якої підмножини належить вибірка;

– url: str, використовується для завантаження відео;

– variation_id: int, id для діалекту (індексується з 0);

– video_id: str, унікальний ідентифікатор відео.

Приклад відеомартеріалу можна побачити на рисунку 1.2.

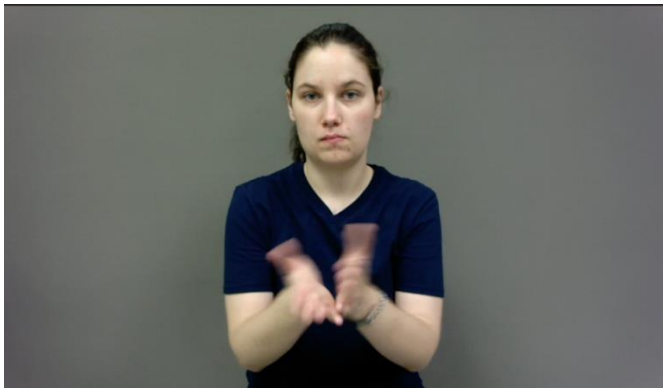

Рисунок 1.2 – Приклад відео, жест, що значить «book»[2]

Оскільки ця колекція була викладена не повністю, за бажанням автора, завантажити її можна тільки через заявлений .json файл. Рисунок 1.3 демонструє кількість відео для найбільш популярних 100 слів у цій колекції. Також видно, що вже велика кількість недоступна за тими посиланнями, які були описані в .json файлі, а загальна кількість відео зменшилася.

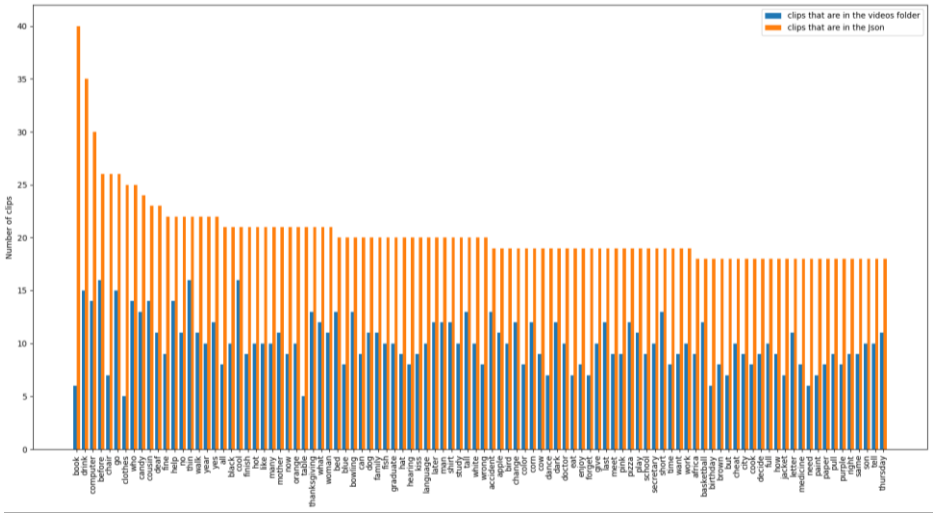

Рисунок 1.3 Кількість відео, що завялені в json файлі(помаранчевій колір) порівняно з кількістю доступного відео(синій колір)

На рисунку 1.4 можна побачити папку з відео з колекції WLASL.

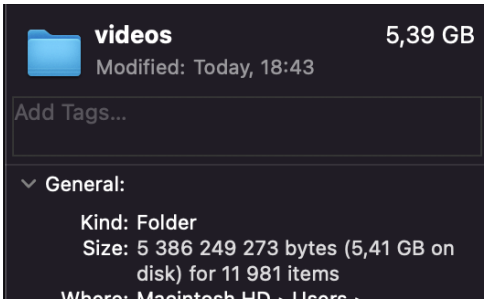

Рисунок 1.4 Кількість завантажених відео

1.4 Постановка задачі

Для початку визначимося з типом задачі, яку ми розв’язуємо. Нам потрібно виділити певні рухи на відео, та класифікувати їх за значенням в Світовій американській мові жестів.

Нехай X:

$$X = \{x_1, x_2, \dots, x_m\} \tag{1.1}$$

Це множина ознак для розпізнавання жестів та обличчя.

$$\begin{gathered} x_i \in \{0,1, \dots, k_i - 1\} \\ k_i \in \{2 \dots N\}, N \in Z \end{gathered} \tag{1.2}$$

Тоді множина розпізнаваних об’єктів:

$$Y = \{y_1, y_2, \dots, y_m\} \tag{1.3}$$

Множина алгоритмів :

$$A = \{a_1, a_2, \dots, a_m\} \tag{1.4}$$

Якість роботи алгоритму на заданому наборі признаків:

$$\begin{gathered} a_j(x_i, y_j) \in \{0,1\} \\ i = 1,2, \dots, l \\ j = 1,2, \dots, n \end{gathered} \tag{1.5}$$

Тобто результат алгоритму може бути оцінений в рамках булевої алгебри:

– 1-алгоритм A_j розпізнав об’єкт y_j по заданим признакам X_i,

– 0-алгоритм A_j не розпізнав об’єкт y_j по заданим признакам X_i

Все тоді зводиться до системи:

$$a_j(X_i, y_j) = \begin{cases} 1, A_i(X_i) = y_j \\ 0, A_i(X_i) \neq y_j \end{cases}, i = 1,2, \dots, l, j = 1,2 \dots, n \quad (1.6)$$

Отже, поставимо задачу. Дані відеоматеріали для навчання та тестування моделі. Необхідно розробити систему, яка на вхід прийматиме файл у форматах .mp4 будь-якого розміру та повертатиме текстове значення відправленого на відео жесту. Для дачі вірної відповіді, застосунок має обробити відео та провести аналіз, спираючись на результати нормалізації. Цей алгоритм зображений на кресленнику ІА81.230БАК.003 Д1. Для зручності користування, системо необхідно зробити кросплатформенною. Система має працювати без встановлення будь-яких файлів на стороні користувача. Система може бути без інтерфейсу, що б не було складності при інтеграції додатку в інші форми програм. Система буде обмінюватися з користувачем повідомленнями .json формату.

Висновки до розділу 1

Отже, у даному розділі була розібрана актуальність задачі. Також був проведений аналіз наявних даних для розробки системи. Вибір коплекту з даними був підкріплений аргументами. Була детальна розібрана та описана структура колекції даних. Окрім відеоматеріалів у ній також знаходиться json файл, в якому і знаходиться вся інформація про те, куди саме належить відео (до тренування, тестування чи варіації). Був представлений вигляд екземпляру даних. Колекція даних була звантажена, та перевірена на цілісність. Майже половина екземплярів відео змінили посилання або обмежили права доступу, тому завантажити їх не вдалося. В кінці розділу була поставлена задача з вимогами та обмеженнями проектованої системи.

2 МАТЕМАТИЧНІ ОСНОВИ РОБОТИ

2.1 Огляд задачі оптичного розпізнавання жестів

Розпізнавання об’єктів (object recognition) — це підгалузь комп’ютерного зору, що вивчає задачу пошуку та ідентифікацій об’єктів у фото або відео. Найпоширенішою практикою оптичного розпізнавання є розпізнавання людського обличчя. Це явище ми можемо спостерігати ледь не щодня, оскільки телефони Apple з функцією Face ID є одними з найпоширеніших пристроїв з цією технологією. Приклад такої технології можна побачити на картинці 2.1.

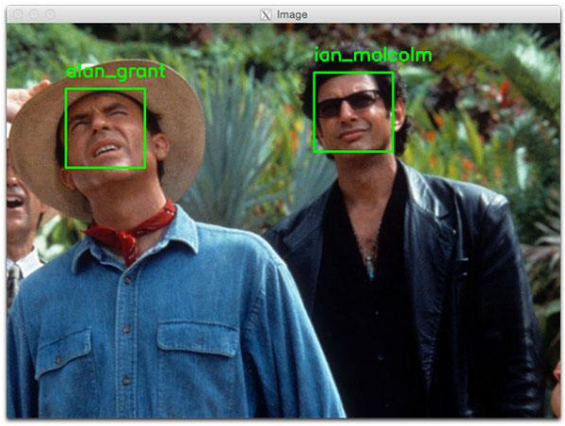

Рисунок. 2.1 – Розпізнавання людських облич[3]

Розпізнавання жестів – це технологія, яку вже досить давно використовують як взаємодію між людиною та машиною. Комп’ютери повинні швидко і точно розпізнавати жести, для того, щоб людина могла легко взаємодіяти з машиною. Це може бути складною задачею, через схожість та порядок жестів та їх застосування.

2.2 Огляд нейронних мереж

Нейронні мережі - це моделі, що взялися за основу з біологічних нейронних мереж мозку, в яких елементарна частина є штучний нейрон.

Нейронну мережу зазвичай представляють спрямованим графом з зваженими зв'язками, де вершинами є самі штучні нейрони, а дуги – зв’язки. Нейронні мережі здатні вирішувати багато задач, через свою здатність “навчатися”.

2.2.1 Штучний нейрон

Штучний нейрон - елемент штучних нейронних мереж. Він отримує якусь інформацію на вході, виконує математичні обчислення та передає іншому нейрону або на вихід.

Структура нейрона складається з трьох частин:

– кілька входів, які сприймають поступаючий сигнал;

– обчислювальний центр, котрий помножує значення сигналу зі значенням ваг w - змінних параметрів. Також обробляє суматором всі вхідні сигнали та застосовує функцію активації до результату;

– кілька виходів, що поширюють отриманий сигнал y від обчислювального центра до інших нейронів.

Самі нейрони пов’язані між собою особливими зв’язками. Ці зв’язки, які називаються синапсами. Вони володіють лише одним параметром – вагою. За його допомогою, сигнал, що передається від одного нейрона до іншого змінюється.

Таким чином вихід нейрона y буде:

$$y = f((w_1x_1 + w_2x_2 + \cdots + w_nx_n) + b) \qquad (2.1)$$

– w_1, w_2, w_n цифрове значення ваг синапсів, які заходять у різні входи нейрона. Чим вище значення, тим важливіший вхід;

– x_1, x_2, x_n цифрове значення сигналу, що передає синапс. Їх може буди декілька, до одного нейрону. Може бути або пряме вхідне значення, або передане з інших нейронів;

– b цифрове значення здвигу, який має назву біас. Цей сигнал може здвинути початок розрахунку функції активації, що надалі призводить до збільшення швидкості навчання. Це значення додається до кожного нейрону, воно також змінюється при навчанні, а також додається до сигналу на вході, а не на виході минулого нейрона;

– f функція активації. Вона є різна для різних типів нейронних мереж.

– z суматор, яка складає всі пари вага-значення та біасом. Також має назву лінійна функція нейрона.

Структура нейрона зображена на рисунку 2.2.

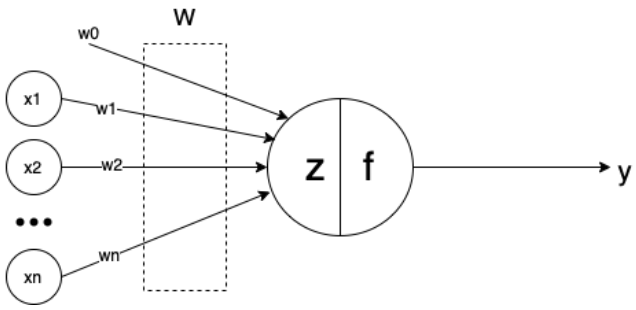

Рисунок. 2.2 – Структура нейрона

2.2.2 Функція активації

Функцій активацій наразі існує безліч, проте найбільшого поширення отримали

– сигмоїдна функція;

– гіперболічний тангенс;

– арктангенс;

– функція ReLU (англ. Rectifier Linear Unit).

На рисунку 2.3 можна побачити графіки цих функцій. Ці функції не мають ніяких змінних параметрів.

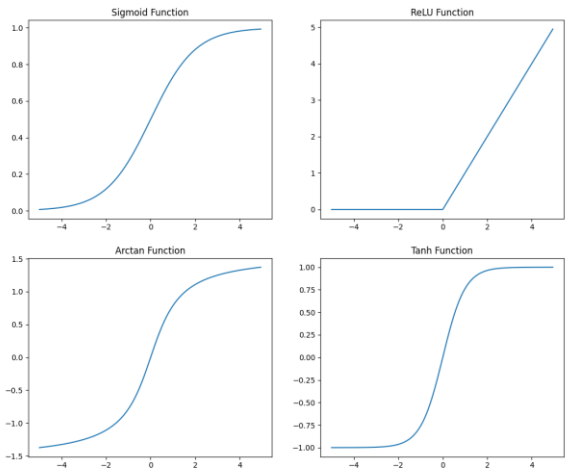

Рисунок 2.3 – Графіки функцій активації

Функція активації – це функція, що використовується в штучних нейронних мережах, яка виводить невелике значення для невеликих входів та більше значення, якщо її входи перевищують поріг. Якщо входи досить великі, функція активації "запалює", інакше вона нічого не робить. Іншими словами, функція активації схожа на вентилятор, який перевіряє, що значення, що входить, більше критичного числа. Функції активації корисні, тому що вони додають нелінійності до нейронних мереж, дозволяючи нейронним мережам вивчати потужні операції. Якби функції активації були видалені з нейронної мережі, вся мережа могла б бути рефакторизована в просту лінійну операцію або матричне перетворення на вході, і вона більше не зможе виконувати складні завдання, такі як розпізнавання зображень. Отже розглянемо ці функції ближче.

2.2.3 Сигмоїда

Основна причина, чому ми використовуємо сигмовидну функцію, полягає в тому, що вона існує між (0 до 1). Тому вона особливо використовується для моделей, де ми повинні передбачити ймовірність як вихідний результат. Оскільки ймовірність чогось існує лише між діапазоном

від 0 до 1, сигмоїдна функція є правильним вибором. Функція диференційована. Це означає, що ми можемо знайти нахил сигмовидної кривої в будь-яких двох точках та тому її легко можна використати для оновлення коефіцієнтів під час зворотнього поширення помилки. Функція монотонна, але похідна функції ні.

Функція softmax є більш узагальненою функцією логістичної активації, яка використовується для багатокласової класифікації.

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

Головними недоліком сигмоїдної функції є те, що вона може спричинити застрягання нейронної мережі під час навчання, бо похідна сигмоїди вже починає губити значення після значень -5 та 5, як це показано на рисунку 2.4.

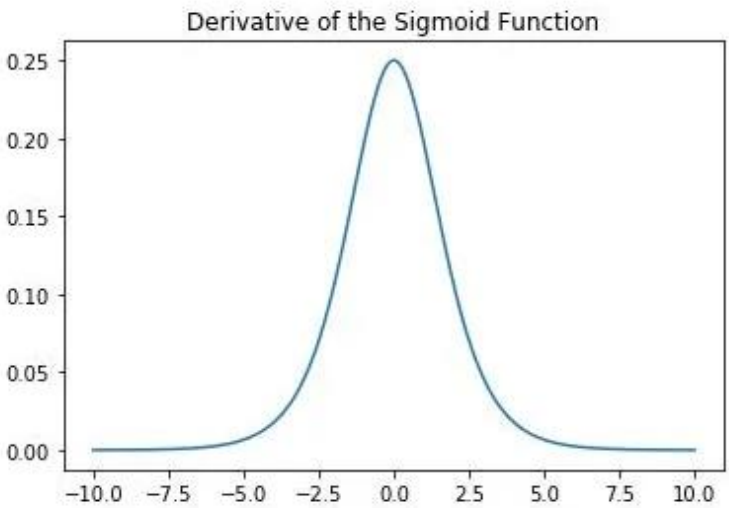

Рисунок 2.4 – Графік похідної сигмоїда [4]

2.2.4 Гіперболічний тангенс

Функція Tanh дуже схожа на функцію сигмоїдної/логістичної активації і навіть має ту саму S-подібну форму з різницею в діапазоні вихідних даних від

-1 до 1. У Tanh, чим більше вхід (більш позитивний), тим ближче вихідне значення буде до 1,0, тоді як чим менше вхід (більш негативний), тим ближче вихід буде до -1,0.

$$\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \tag{2.2}$$

$$\tanh'(x) = (1 - \tanh^2(x)) \tag{2.3}$$

Вихід функції активації tanh має нульовий центр; отже, ми можемо легко відобразити вихідні значення як сильно негативні, нейтральні або сильно позитивні.

Зазвичай використовується в прихованих шарах нейронної мережі, оскільки його значення лежать від -1 до 1; тому середнє значення для прихованого шару виявляється рівним 0 або дуже близьким до нього. Це допомагає центрувати дані та значно полегшує навчання для наступного рівня.

Ця функція також стикається з проблемою зникнення градієнтів, подібний до функції активації сигмоїди. Графік похідної зображений на рисунку 2.5. Крім того, градієнт функції tanh набагато крутіший порівняно з сигмовидною функцією.

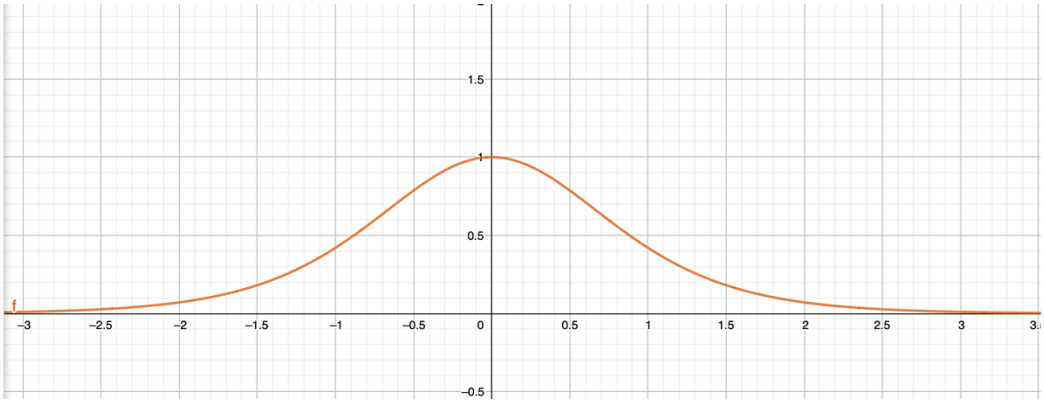

Рисунок 2.5 – Графік похідної гіперболічного тангенсу

2.2.5 ReLU

ReLU (Rectified Linear Unit) вперше була використана саме в згорткових нейронних мережах. Вона не є центровною відносно точки 0.

$$f(x) = \begin{cases} x, & x > 0 \\ 0, & x \leq 0 \end{cases} \tag{2.4}$$

Хоча це створює враження лінійної функції, ReLU має похідну функцію і дозволяє здійснювати зворотне поширення, водночас роблячи її обчислювально ефективною.

Головна фішка полягає в тому, що функція ReLU не активує всі нейрони одночасно. Нейрони будуть дезактивовані, лише якщо вихідний результат лінійного перетворення менше 0.

Переваги використання ReLU як функції активації полягають у наступному:

– оскільки активується лише певна кількість нейронів, функція ReLU є набагато ефективнішою в обчислювальному відношенні порівняно з функціями сигмовидної і гіперболічного тангенсу;

– ReLU прискорює зближення градієнтного спуску до глобального мінімуму функції втрат завдяки своїй лінійній властивості, що не має насичення. Графік похідної ReLU можна побачити на рисунку 2.6. ReLU стикається також з деяким обмеженням.

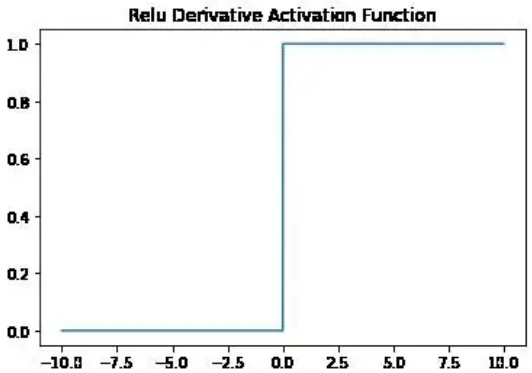

Рисунок 2.6 – Графік похідної ReLU[4]

Від’ємна сторона графіка робить значення градієнта нульовим. З цієї причини під час процесу зворотного поширення ваги та зміщення для деяких нейронів не оновлюються. Це може створити мертві нейрони, які ніколи не активуються. Усі від’ємні вхідні значення відразу стають нульовими, що зменшує здатність моделі правильно підходити або тренуватися з даних.

2.2.6 Arctan

Ця функція активації відображає вхідні значення в діапазоні $(-\pi/2,\pi/2)$. Його похідна квадратично збігається до 0 для великих вхідних значень. З іншого боку, у сигмовидної функції активації похідна збігається експоненціально проти 0, що може викликати проблеми під час зворотного поширення. Його графік похідної трохи плоский, ніж tanh, тому він має кращу тенденцію розрізняти подібні вхідні значення. Цей графік зображено на рисунку 2.7.

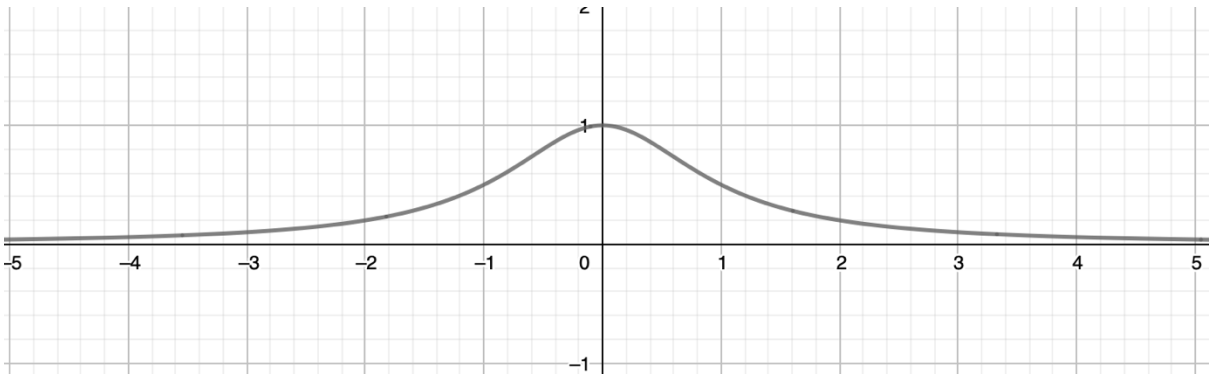

Рисунок 2.7 – Графік похідної арктангенса

2.3 Архітектури нейронних мереж

Дві найбільші базові архітектури нейронних мереж є шарові та повновз’язні.

Принципом шарових мереж є утворення груп нейронів. Такі групи називаються шарами та можуть мати від одного до декількох нейронів. На вхід

кожного нейрону одного шару приходить сигнал від минулого шару з одних і тих самих нейронів. Так сигнал змінюється, корегується та поширюється по нейронній мережі пошарово. Звісно, ключову роль в передачі інформації відіграють ваги синапсів між шарами, бо саме від них залежить, що отримає наступний шар. Інформацію нейрони від шару n сприймають на вхід, обробляють та передають на вхід до кожного нейрону шару n+1 по синапсам, у кожного з яких є своя вага. Число нейронів у кожному шарі обирається довільно та не залежить від кількості нейронів у будь-яких інших шарах.

Одношарова мережа є структурою взаємодії нейронів, у якій сигнали з вхідного шару передаються одразу на вихідній шар, де сигнал не тільки перетворюються, а і одразу видає саму відповідь. Перший шар лише сприймає та передає сигнали по синапсам наступному шару. Бажану відповідь ми можемо отримати вже з другого шару, де і проводилися основні математичний вирахування.

Багатошарові нейронні мережі, на відміну від одношарових, має же внутрішній шар. Цих прихованих шарів може бути безліч, спільним для цього типу нейронних мереж лишається наявність вхідного та вихідних шарів. Таким чином, вони мають вхідний шар, який має зв’язок з вхідними даними, прихований шар, в якому відбуваються всі математичні розрахунки, та вихідний шар, який має зв’язок з вихідними даними, та через який ми отримуємо відповідь. Зв’язок у внутрішньому шарі може реалізуватися по різному, в залежності від типу мережі, тобто прихований шар не обов’язково складається з декількох шарів, що передаються сигнали один від одного до кожного нейрона від кожного нейрона. Нейрони, так само як і шари, можуть бути вхідними, прихованими і вихідними. Вхідні нейрони, отримують сигнал та поширюють його на входи нейронам прихованої мережі. Приховані нейрони – це нейрони, з них складається прихований шар нейронної мережі. Вихідні нейрони, дуже часто це буває лише один нейрон, є частиною вихідного шару, та видає відповідь на своєму виході. Хоча багатошарові нейронні мережі потребують більше часу та інших ресурсів для обчислення,

вони відкривають для користувачів нові додаткові можливості, бо кожний внутрішній шар – це проміжний етап, на якому відбувається обробка та передача інформації.

У шарових мережах є один типовий недолік, пов'язаний з кількістю інформації, що потрапляє на вхід. Великий розмір даних, що передається між нейронами з одного шару в інший може привезти до неточності наближення цільової функції, бо в них будуть існувати дуже складні залежності, що формуються по синапсам і їх вагам. Через це нейронні мережі такого типу не підходили для обробки та аналізу фото та відеофайлів, бо вони мають досить великий розмір. Вирішенням проблеми стало створення загорткових нейронних мереж (CNN). Архітектура та принцип роботи внутрішніх шарів були змінені. По перше вхідні дані подаються у вигляді матриці. Таким чином, кількість математичних операцій та складність зав'язків значно зменшилася. По друге перетворення матриці у нову матрицю, що буде менша за розміром, за допомогою методу «ковзаючого вікна» та використовуючи так звані ядра згортки . Тоді вже зменшена вихідна матриця проходить через функцію активації та подається далі.

Ядро згортки, або Кернел, є невеликою квадратну матрицю ваг, зазвичай непарного розміру (3x3 або 5x5). Таким чином зображення будуть мати лише десяток параметрів, а не сотні або тисячі. Це вирішує проблему вибуху або зниканні градієнта під час тренування. В одному згортковому шарі можуть бути не один Кернел, а декілька. Тоді ковзаюче вікно застосовується стільки разів, скільки ядер згортки присутнє в мережі. Така операція може давати декілька результатів. Це залежить від того, чи має згорнутий зв'язок меншу розмірність порівняно з вхідним, чи більшу або незмінну. Тобто якщо спочатку матриця залежностей (наприклад 5x5), до неї додаються нулі по сторонам(виходить 6x6) і в ролі «ковзаючого вікна» у нас буде матриця 3x3, то згорнута матриця знову вийде 5x5. В такому випадку використовується Same-padding, зображена на рисунку 2.8.

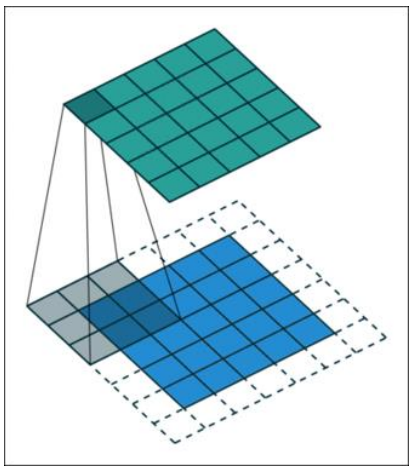

Рисунок 2.8 – Стратегія Same-padding[5]

Якщо ж ми будемо робити те саме, але Кернелом проходити не по кожному рядку, якщо ніби один пропускати, то згорнута матриця вийде 3x3. Така стратегія Valid-padding, зображена на рисунку 2.9.

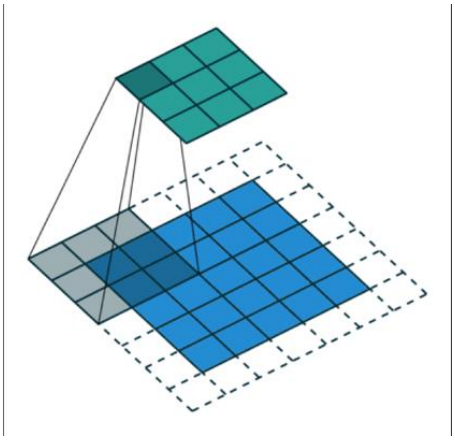

Рисунок 2.9 – Стратегія Valid-padding[5]

Також в згортковий нейронних мережах можуть використовуватися пуллінгові шари. Вони теж виділяють лише домінуючи правила, при цьому зменшуючи кількість обчислень та збільшуючи ефективність навчання моделі. З набору пікселів матриці зображення пуллінговий шар може взяти або найбільше значення, або середнє. Найбільше значення використовується для «шумозаглушення» (Noise Suppressant) та називається максимальний пуллінг (Max pulling). Приклад зображений на рисунку 2.10.

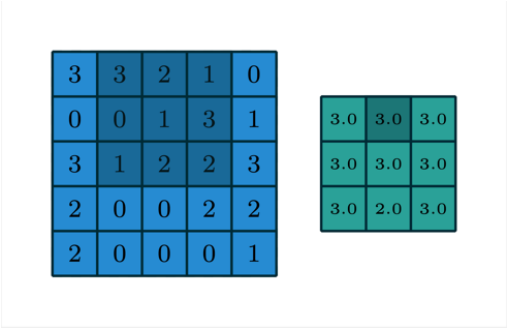

Рисунок 2.10 – Пуллінг максимального значення[5]

У видатку пуллінгу середнього значення відповідно середній пуллінг (Avarage pulling). Такий спосіб зображений на рисункук 2.11.

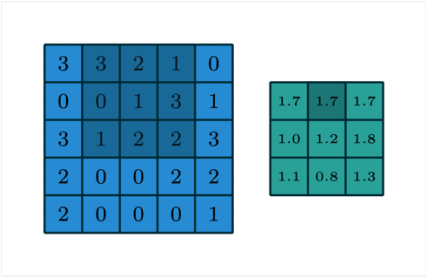

Рисунок 2.11 – Пуллінг середнього значення[5]

Також в проектуванні таких нейронних мереж можуть використовувати, крім згорткових шарів та шарів пуллінга, шари пакетної нормалізації. Це шар, який відповідальний за нормалізацію вхідних данних.

В таких нейронних мережах використовуються також шари пакетної нормалізації - шар, який проводить нормалізацію вхідних даних.

Нормалізація — інструмент попередньої обробки даних, використовуваний для приведення числових даних до загального масштабу без спотворення їх форми. Як правило, коли ми вводимо дані до машини або алгоритму глибокого навчання, ми схильні змінювати значення до збалансованої шкали. Причина, через яку ми нормалізуємо, частково полягає в тому, щоб гарантувати, що наша модель може узагальнювати належним чином.

Тепер повертаючись до пакетної нормалізації, це процес, що дозволяє зробити нейронні мережі швидшими та стабільнішими за рахунок додавання додаткових шарів у глибоку нейронну мережу. Новий шар виконує операції стандартизації та нормалізації на вході шару, що надходить із попереднього шару. Типова нейронна мережа навчається з допомогою зібраного набору вхідних даних, званого пакетом. Так само процес нормалізації при пакетної нормалізації відбувається партіями, а не як один вхід.

Нормалізація — це процес перетворення даних, щоб вони мали середнє нуль і одиницю стандартного відхилення. На цьому кроці існує пакетний вхід із шару h, спочатку потрібно обчислити середнє значення цієї прихованої активації.

$$\mu = \frac{1}{m}\sum h_i$$
$$h_i = f(w_i x_i) \tag{2.5}$$

Тут m – кількість нейронів у шарі h.

Наступним кроком є обчислення стандартного відхилення прихованих активацій.

$$\partial = \left[\frac{1}{m}\sum (h_i - \mu)^2\right]^{1/2} \tag{2.6}$$

Далі, оскільки у нас готові середнє значення і стандартне відхилення. За допомогою цих значень ми нормалізуємо приховані активації. Для цього ми віднімемо середнє з кожного вхідного даних і поділимо ціле значення на суму стандартного відхилення та коефіцієнт згладжування (ε). Згладжування (ε) забезпечує чисельну стабільність операції, зупиняючи поділ на нульове значення.

$$h_{i(norm)} = \frac{(h_i - \mu)}{\partial + \varepsilon} \tag{2.7}$$

Тобто всі вищезгадані матриці, шари є одним згортковим шаром у нейронній мережі. Для кращого результату, після шарів пуллінга, та ядер згортки додається повнозв'язна нейронна мережа, так як на рисунку 2.12.

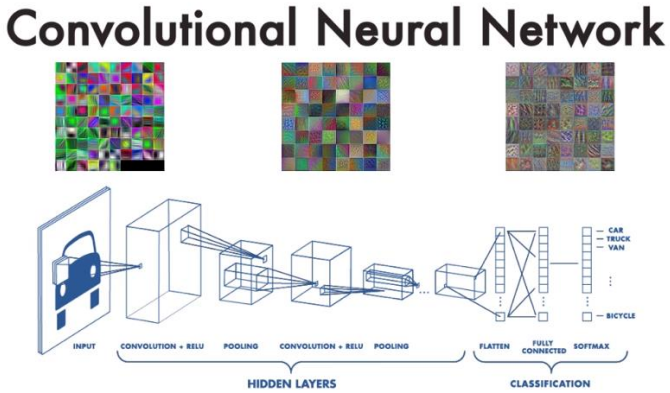

Рисунок 2.12 – Повна згорткова нейронна мережа[6]

У результаті мережа працює у послідовному порядку наступним чином:

–ознаки, що є набагато меншими за вагою, ніж самі данні, фільтруються на згорктовому шарі, та не передаються далі;

–ознаки, що пройшли фільтрацію, класифікуються повнозв'язною частиною шару.

Сучасні класифікаційні нейронні мережі гуртуються на цьому принципі, так як він поєднує відносно невелику кількість параметрів і ефективність роботи.

Крім кількості шарів, нейронні мережі можна розділити по напрямку передачі інформації по синапсам між нейронами.

Нейронні мережі прямого поширення (Feedforward neural network). В цій структурі сигнал переміщується тільки в одному напрямку, строго від вхідного до вихідного шару. Переміщення сигналу в іншому напрямі, від вихідного до вхідного шару, не виконується і фактично неможливе. Мережі такого типу є дуже розповсюдженими, бо вони відмінно виконуються задачі

по розпізнаванню різних образів, прогнозування та кластеризації. Алгоритм роботи такої нейронної мережі зображений на рисунку 2.13.

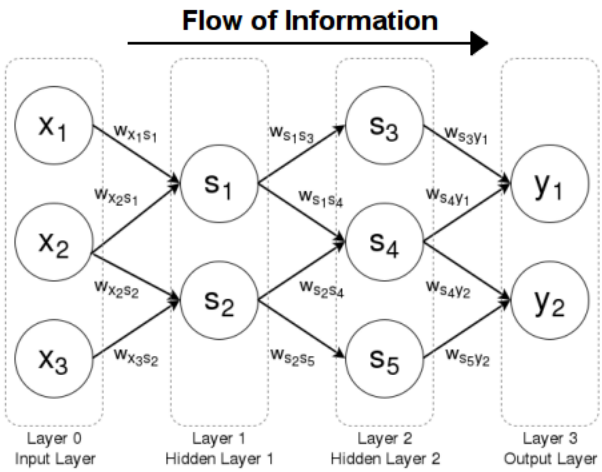

Рисунок 2.13 – Принцип роботи Нейронної мережі прямого поширення[7]

Рекурентні нейронні мережі (зі зворотнім зв’язком). Тут сигнал рухається як в прямому так і в зворотному напрямі. Це дає можливість повертати сигнал з виходу назад на вхід. Вихід нейрона визначається ваговими характеристиками та вхідними сигналами, на додачу доповнюється сигналами, що повернули з виходу мережі назад на вхід. Тут сигнал рухається і в прямому, і в зворотному напрямку. Цим нейронним мережам властива короткочасна пам’ять, на основі якої сигнали відновлюються та доповнюються під час їх обробки. Також існують радіально-базисні функції та самоорганізуючі карти.

Тобто всі нейронні мережі спочатку отримають данні на вхід, нормалізують та виділяють основні ознаки. Процеси тренування та тестування можуть бути різними, та їх відсутність неможлива. Вже після цих процесів, мережа може повертати якийсь результат. Узагальнення цього алгоритму для мережі, що буде розпізнавати жести, зображене на ІА81.230БАК.003 Д2.

2.4 Огляд сучасних глибоких нейронних мереж

2.4.1 VGGnet (Visual Geometry Group network)

Ця модель буда запропонована двома вченими з Оксфорда у 2014 році. На вхід до VGG, яка сама по собі є згортковою нейронною мережею, подається зображення 224x224 у кольоровій моделі RGB. Вхідний шар опрацьовує зображення, присвоюючи кожному пікселю значення від одного до 255 та знаходить їх максимальне значення протягом усієї мережі, застосовуючи пуллінг максимального значення. Подібна схема зображена на рисунку 2.14.

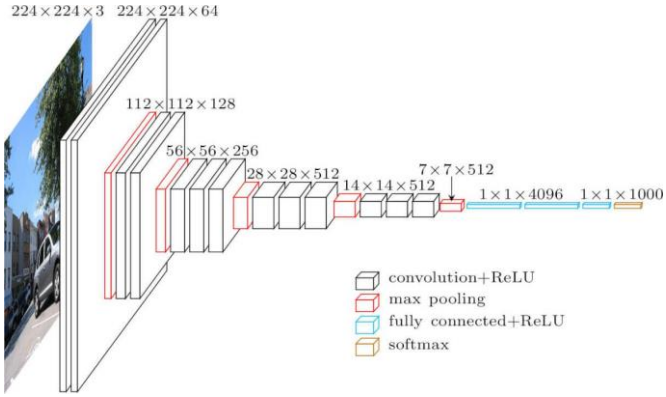

Рисунок 2.14 – Архітектура мережі VGG[8]

Вхідні зображення після попередньої обробки пропускаються через ці вагові шари. Навчальні зображення пропускаються через набір шарів згортки. В архітектурі VGG16 є всього 13 згорткових шарів і 3 повністю пов’язані шари. Структура VGG16 зображена на рисунку 2.15. VGG має менші фільтри (3x3) з більшою глибиною замість великих фільтрів. У підсумку воно має таке ж ефективне сприйнятливе поле, як якщо б у мережі був лише один згортковий шар 7x7.

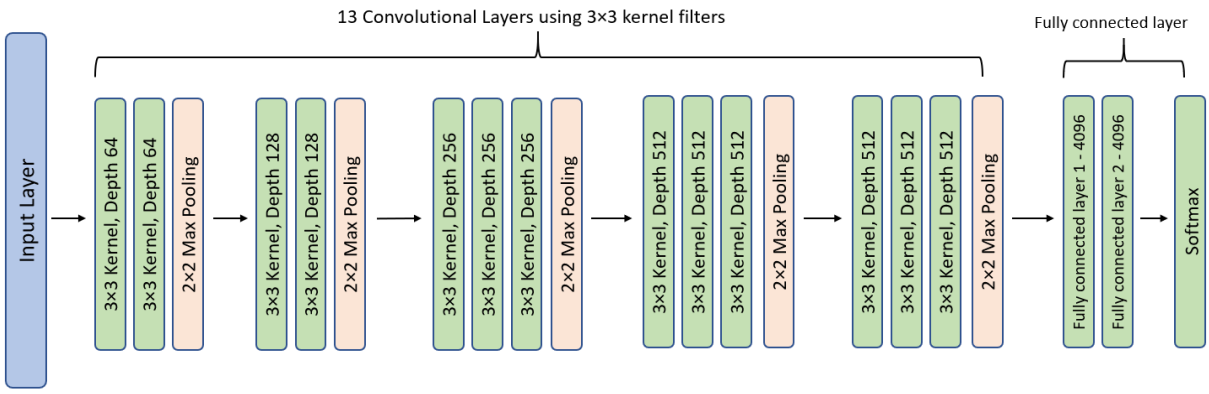

Рисунок 2.15 – Архітектура мережі VGG16[8]

Інша варіація VGGNet має 19 вагових шарів, що складаються з 16 згорткових шарів з 3 повністю пов’язаними шарами і такими ж 5 шарами об’єднання. В обох варіантах VGGNet складається з двох повністю підключених шарів з 4096 каналами кожен, за якими слідує ще один повністю підключений шар з 1000 каналами для прогнозування 1000 міток. Самі нейрони використовують активаційну функцію ReLU. Останній повністю підключений рівень використовує рівень softmax для цілей класифікації.

Перевага VGGNet полягає у великій кількості конфігурацій, що вказує на гнучкість нейрнної мережі та велика кількість внутрішніх шарів, що допомагають виконати навчання біль якісно. Недоліками цієї мережі є велика кількість парментів, більше 138 мільйонів, повнозв’язні шари та велика кількість пам’яті.

2.4.2 ResNet (Residual neural network)

Поштовхом для створення нейронної мережі зі зв’язками швидкого доступу, стало погіршення результатів навчання та тестування нейронних мереж зі збільшенням кількості шарів. Чим глибше нейронна мережа, тим більше помилок вона починала видавати, після 27го шару. Детальний графік ображений на рисунку 2.16.

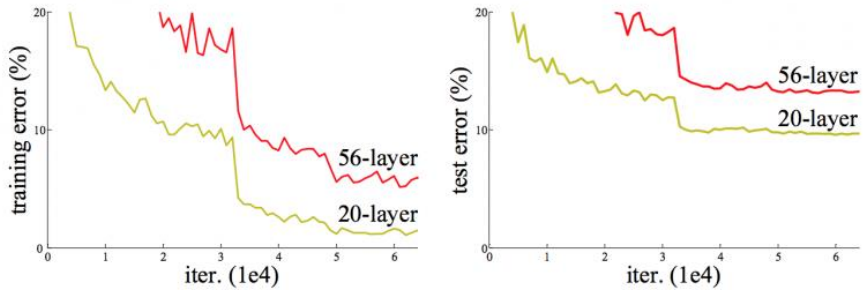

Рисунок 2.16 – Помилка навчання(зліва) та помилка тестування(справа) на данних з бази CIFAR-10 з 20 шарами та 56 шарами нейронних мереж відповідно[9]

Основна ідея ResNet полягає в тому, щоб запровадити так зване «підключення до ідентифікації», яке пропускає один або кілька шарів, як показано на рисунку 2.17:

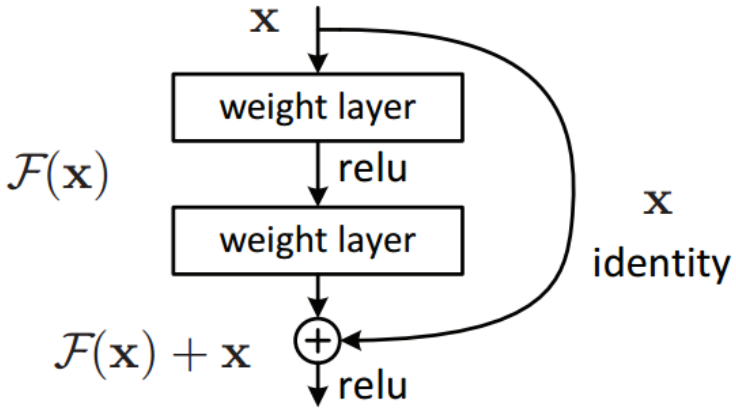

Рисунок 2.17 – Залишковий блок(residual block)[9]

Автори стверджують, що стекування шарів не повинно знижувати продуктивність мережі, тому що ми можемо просто складати відображення ідентифікацій (рівень, який нічого не робить) у поточній мережі, і результуюча архітектура буде працювати так само. Це вказує на те, що більш глибока модель не повинна створювати похибку навчання, більшу, ніж її більш дрібні аналоги. Вони припускають, що дозволити складеним шарам відповідати залишковому відображенню легше, ніж дозволити їм безпосередньо

відповідати бажаному базовому відображенню. І залишковий блок вище явно дозволяє йому робити саме це. Порівняння структур двох нейронних мереж можна побачити на рисунку 2.18. Перевага такої мережі в тому, що завдяки skip connection, повністю вирішена проблема затухання градієнту. Також така архітектура не має обмежень у кількість шарів та є доволі простою у реалізації.

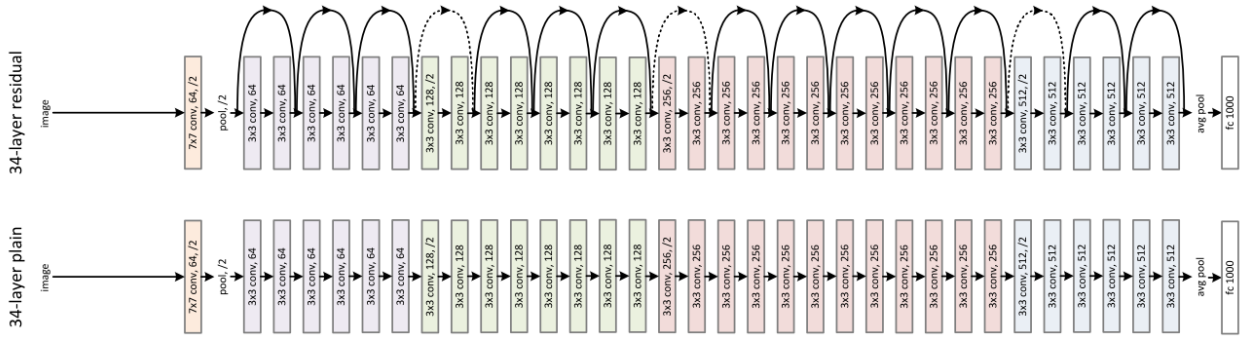

Рисунок 2.18 – Порівняння звичайної згорткової нейронної мержі з 34 шарами з ResNet, також з 34 шарами[9]

2.4.3 LeNet(LeNet-5)

Вперше LeNet була запропонована ще у 1998 році, яка була розроблена Янном Лекуном, Леоном Ботту, Йосухою Бенджіо та Патріком Хаффнером. Також відома як класична нейронна мережа, для розпізнавання рукописних і машинних символів. Вхідні зображення були у сірій шкалі та розміром 32*32*1, за якими слідували дві пари згорковвих шарів з кроком 2 і шаром середньої пуллізації із кроком. повністю з'єднані шари з активацією Softmax у вихідному шарі. Традиційно ця мережа мала 60 000 параметрів. Структура LeNet-5 зображена на рисунку 2.19.

Недоліком такої мережі є відсутність механізму перенавчання нейронної мережі.

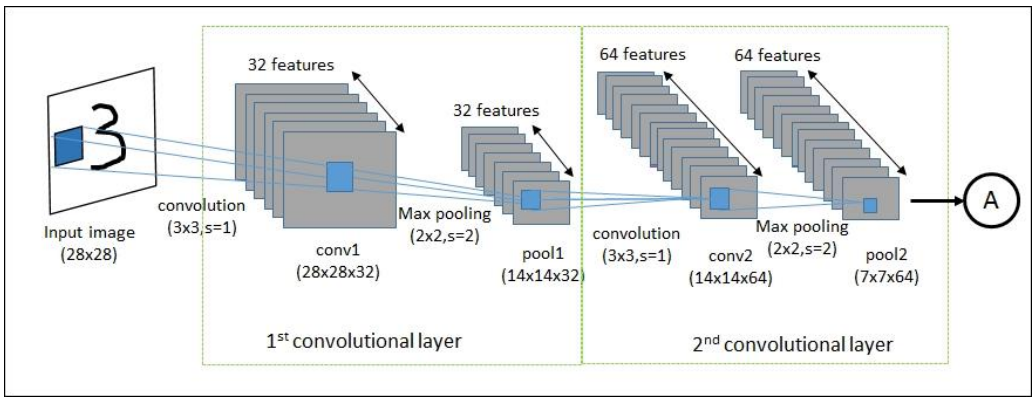

Рисунок 2.19 – Архітектура мережі LeNet-5[10]

2.4.4 AlexNet

AlexNet — згорткова нейронна мережа, яка вплинула на розвиток машинного навчання, особливо — на алгоритми комп'ютерного зору. Мережа з великим відривом виграла конкурс із розпізнавання зображень ImageNet LSVRC-2012 у 2012 році (з кількістю помилок 15,3% проти 26,2% у другого місця).

Архітектура AlexNet схожа із створеною Yann LeCum мережею LeNet. Однак у AlexNet більше фільтрів на шарі та вкладених згорткових шарів. Мережа включає згортки, максимальне об'єднання, дропаут, аугментацію даних, функції активацій ReLU та стохастичний градієнтний спуск. AlexNet була розроблена групою SuperVision, що складається з Алекса Крижевського, Джеффрі Хінтона та Іллі Суцкевера. Архітектура мережі AlexNet показана на рисунку 2.20.

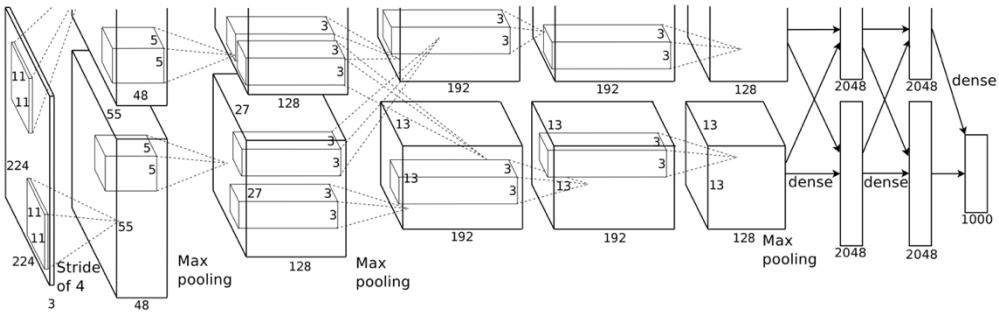

Рисунок 2.20 – Архітектура нейронної мережі AlexNet[11]

Головною перевагою AlexNet стало використання ReLU, що частково виключає затухання градієнту на більш глибоких шарах згортки. Також, у порівнянні з LeNet з'явиввся механізм dropout. Це механізм перенавчання мережі. Dropout — це метод регуляризації, який апроксимує паралельне навчання великої кількості нейронних мереж з різними архітекторами. Під час навчання деяка кількість вихідних даних шарів випадково ігнорується або «випадає». Це призводить до того, що шар виглядає як шар з іншою кількістю вузлів і підключенням до попереднього шару. Фактично, кожне оновлення шару під час навчання виконується з іншим «поглядом» налаштованого шару. Недоліками згортковою нейронної мережі AlexNet стали велика кількість парметрів, більше 60 мільйонів, та повнозв'язні шари. Ці обидва фактори викликають велике навантаження мережі, через це для навчання мережі потрібна більша потужність та більше часу.

2.4.5 GoogLeNet

Google Net (або Inception V1) була запропонована дослідниками Google (у співпраці з різними університетами) у 2014 році в дослідницькій роботі під назвою «Поглиблення згортки». Ця архітектура стала переможцем у конкурсі на класифікацію зображень ILSVRC 2014, досягнувши межі лише 6.67% top5 error. Приклад одного шару – на рисунку 2.21

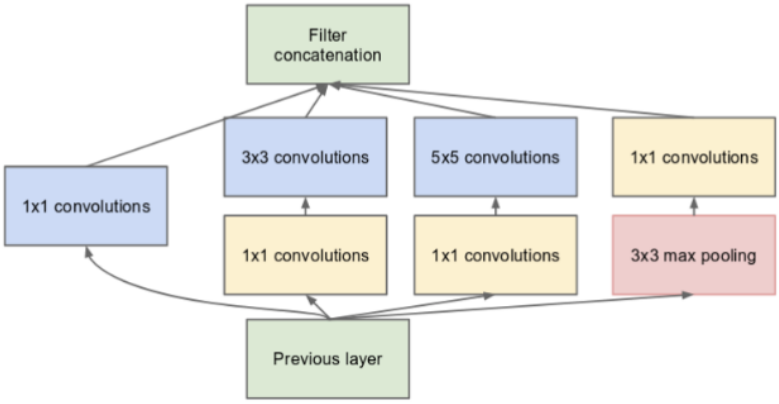

Рисунок 2.21 – Один блок, з якого складається архітектура GoogLeNet[12]

Він забезпечив значне зниження рівня помилок у порівнянні з попередніми переможцями AlexNet (переможець ILSVRC 2012) та ZF-Net (переможець ILSVRC 2013) і значно менший рівень помилок, ніж VGG (переможець 2014 року). Ця архітектура використовує такі методи, як згортки 1×1 в середині архітектури та глобальне середнє об’єднання.

Архітектура GoogLeNet сильно відрізняється від попередніх сучасних архітектур, таких як AlexNet і ZF-Net. Ця нейронна мережа використовує багато різних методів, таких як згортка 1×1 і глобальне середнє об’єднання, що дозволяє створювати більш глибоку архітектуру.

Згортка 1×1 використовувалася для зменшення кількості параметрів (ваги та зміщень) архітектури. Зменшуючи параметри, ми також збільшуємо глибину архітектури. Наприклад потрібно виконати згортку 5x5 матриці 14x14x480 маючи 48 фільтрів, як це показано на рисунку 2.22. Тоді це займе (14x14x48)x(5x5x480)=112.9 мільйонів операцій. За допомогою згортки 1x1 кількість операцій можна зменшити до (14x14x16)x(1x1x480)+(14x14x48)x(5x5x16)=5.3 мільйонів операцій, як на рисунку 2.34.

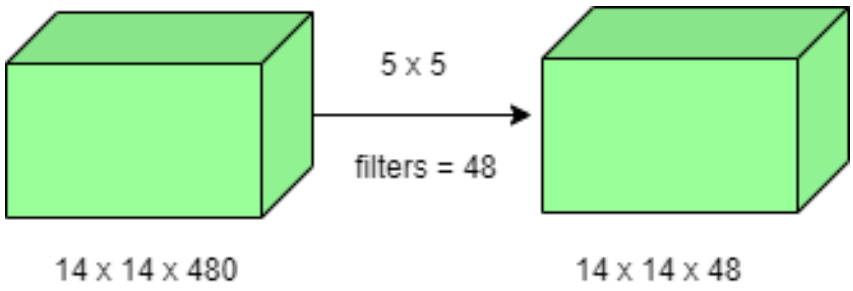

Рисунок 2.22 – Згортка 5x5 без використання згортки 1x1[13]

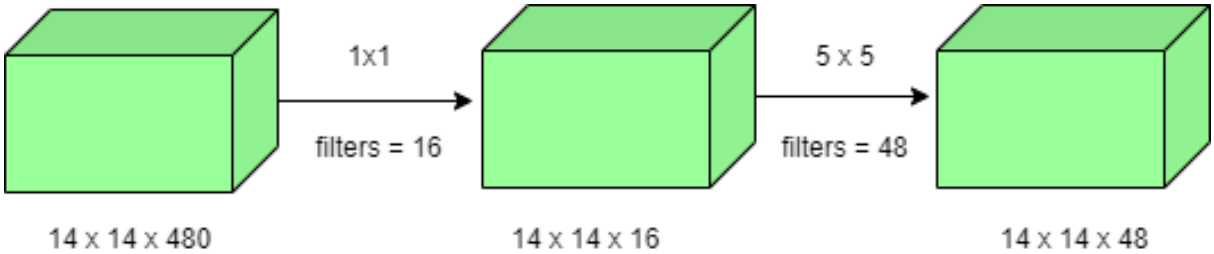

Рисунок 2.23 – Згортка 5x5 з використанням згортки 1x1[13]

Перевага GoogLeNets у тому, що такі шари, як на рисунку 2.21, підсилюються softmax-ом, що не дає градієнту функції втрат затухати. Також ця мережа має меншу кількість параметрів, ніж VGGNet, дякуючи згортці 1x1. З недолік можна назвати досить складну реалізацію та велику кількість повнозв'язних мереж.

2.5 Огляд існуючих рішень

2.5.1 Wecapable.com

Сайт, на якому користувач може ввести фразу, а сервер висилає набір картинок, жести яких утворюють літери, з яких складаються слова. Як було сказано раніше, такий спосіб спілкування не можна назвати повністю мовою жестів, адже така мова налічує жести саме окремих слів, а не літер. Також є очевидним, що переклад видається способом звертання до бази даних, або словника, знаходження потрібної картинки та повертання її. Участі нейронної мережі тут немає. Інтерфейс Wecapable.com зображений на рискну 2.24.

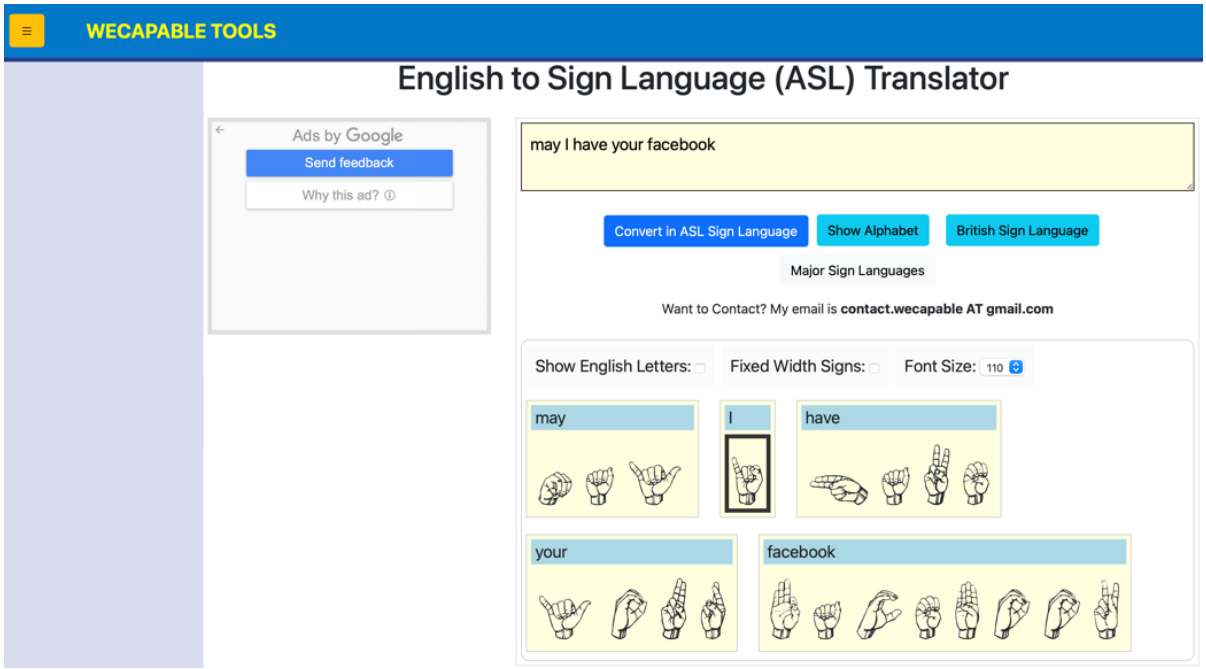

Рисунок 2.24– Інтерфейс сайту wecapable.com[17]

2.5.2 Hand Talk Translator

Мобільний застосунок, що може в результаті відсилання йому голосового повідомлення повертає набір відео з анімаційним персонажем, який показує жести. На відміну від першого варіанту, тут можна побачити вже вживання справжньої мови жестів. Коду у відкритому доступі не було, тому важко сказати, чи бере учать нейронна мережа, хоча б у розпізнаванні слів в записаному голосовому повідомленні, але не думаю, що сама нейронна мережа вирішує яке слово підходить конкретному жесту. Також немає версії для користуванні на комп'ютері. Головна сторінка мобільного застосунку зображена на рисунку 2.25.

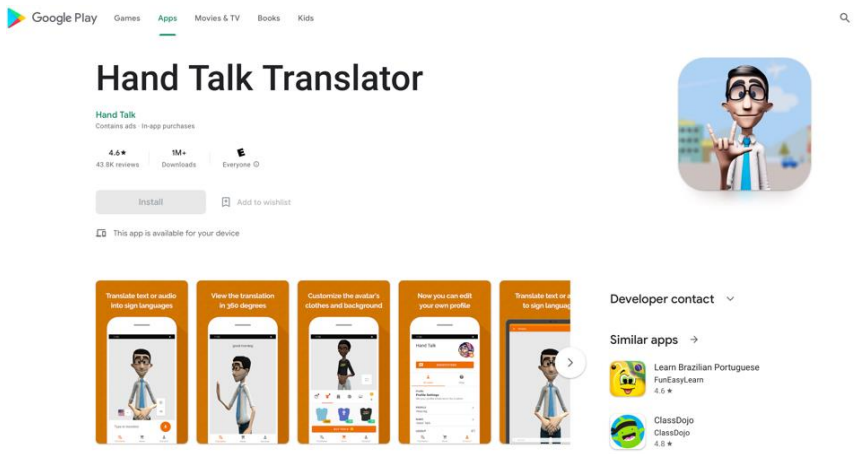

Рисунок 2.25 – Сторінка мобільного застосунку «Hand talk translator» в магазині «Google play»[18]

2.5.3 Omnibridge.ai

Най сайті вказано дуже мало інформації. Не відомо, де та як можна купити, застосувати та протестувати це програмне забезпечення. З іншого сайту, https://www.closingthegap.com/, було зрозуміло одне, що це ПЗ можна встановити на спеціальні засоби для людей з інвалідністю. Ця програма може

перекладати мову жестів у звичайну, так і навпаки у режимі реального часу, при цьому застосовуючи машинне навчання. Проте більшої інформації знайти не вдалося. Вебсайт проекту Omnibridge зображений на рисунку 2.26.

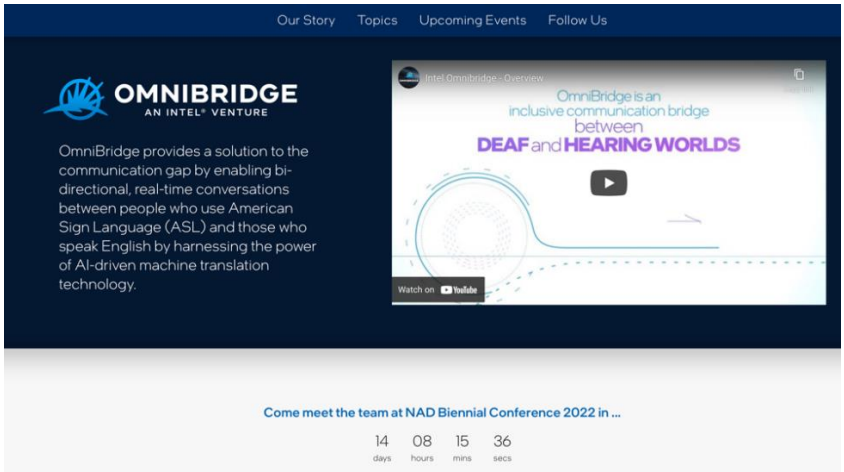

Рисунок 2.26 – Головна сторінка сайту omnibridge.ai[19]

Висновки до розділу 2

В даному розділі були наведені приклади нейронних мереж, які можна було взяти для реалізації. Серед них були: LeNet, AlexNet, VGG, GoogLeNet, ResNet. До кожної із них було наведено переваги та недоліки. Також були названі проблеми, з якими ми можемо зіштовхнутися, та те, на скільки добре, кожна з мереж їх вирішує. На основі цих даних була сформована таблиця 1.

Таблиця 1 – Порівняння різних архітектур згорткових нейронних мереж

Назва	Рік	Кількість шарів	Top-5 Errors
-	До 2012	≤5	>25%
AlexNet	2012	8	15.4%
VGG	2014	19	7.3%
GoogLeNet	2014	22	6.7%
ResNet	2015	152	3.6%

В результаті для реалізації видно, що точність найвища у ResNet, але не дивлячись на свою точність, мережа має забагато шарів, що потребують великих ресурсів. Тому для подальшої розробки була обрана архітектура GoogLeNet або InceptionNet.

Було викладено базове поняття про функцію активації, наведено декілька варіантів готових, вибрано найкращу з них, яка найкраще підходить під нашу задачу. Було викладено базові поняття про згортку, її структуру. Було пояснено різницю між двома типами пулінгу, а саме max-pooling та average-pooling, і чому саме вони були присутні у тій чи іншій згортковій мережі. Після цього була розглянути батч-нормалізація, яка допомагає моделі уникнути перенавчання.

3 ВИБІР ТЕХНОЛОГІЙ ТА РЕАЛІЗАЦІЇ

Для об’єктивного вибору інструментів для реалізації застосунку необхідно спочатку виділити обов’язки, які повинна робити програма По перше, необхідно, щоб вона могла сприймати запити від користувача та надсилати відповіді по закінченню виконання задачі. По друге, веб-застосунок повинен знати, що саме відповісти. Для формування відповіді необхідна модель нейронної мережі. Для тренування моделі, вона сама повинна вміти розділяти кадри, видаляти фон, обмальовувати контури і з цих контурів обирати ключові ознаки, на яких будуть формуватися відповіді. Набор вмінь нейронної мережі та дій користувача з розробником зображені на кресленику ІА81.230БАК.003 ДЗ.

3.1 Аналіз мов програмування

Для написання цього проекту було розглянуто 3 мови програмування: C++, R та Pyhton. Виникла необхідність розглянути кожну з мов окремо та провести порівняльний аналіз.

3.1.1 Мова програмування C++

C++ доволі стара мова програмування, яка була розроблена на мові C, та є повністю з нею сумісна, будь-що написане на C запуститься на C++. Мова є об’єктно-орієнтовною, тобто підтримує поліморфізм, інкапсуляцію, наслідування. C++ є одним із фаворитів в плані швидкодії, що легко пояснює той факт, що на базі цієї мови створена левова частка всіх математичних бібліотек, що в свою чергу може значно спростити розробку нашої задачі. Що стосується deep learning фреймворків, то c++ може запропонувати, наприклад, Torch, OpenNN. Основною складністю, з якою можна зіштовхнутися, вибравши C++ на роль базової для вирішення даної задачі, є те, що поріг

входження є вище середнього, тобто буде витрачатися додатковий час на вивчення певних аспектів цієї мови. Також для цієї мови пропонуються не найкращі середовища розробки, які також збільшують час розробки.

Підсумовуючи, можемо сказати, що серед основних переваг мови C++ є: швидкість роботи, об'єктно-орієнтованість, наявність deep learning фреймворків. Серед основних недоліків такі: високий поріг входження, недосконалість середовищ розробки, які підтримують написання коду на цій мові.

3.1.2 Мова програмування R

R – дуже популярна мова програмування для статистичних обчислень, аналізу та візуалізації у машинному навчанні. Це ідеальна мова на основі графіки для вивчення статистичних даних у вигляді графіків, яка широко використовується фахівцями за даними у Facebook, Google і т. д. Хоча R використовується переважно у біоінженерії та біомедичній статистиці, він також популярний у реалізації машинного навчання, такого як регресія, класифікація, та формування дерева рішень.

Бібліотека xgboost використовується для реалізації фреймворку підвищення градієнта та популярний завдяки своїй продуктивності та швидкості. Він підтримує різні цільові функції, такі як регресія, класифікація та ранжування, і є розширюваним, так що ви можете легко визначити свої власні цілі.

Пакет mlr це структура, що розширюється, для задач класифікації, регресії та кластеризації, що має простий механізм розширення за рахунок успадкування s3.

Пакет Party використовується для рекурсивного розбиття. Цей пакет використовується для побудови рішень на основі алгоритму умовного виведення. Цей пакет також є великим, що скорочує час навчання та систематичну помилку.

Пакет Caret розроблений для об'єднання навчання моделі та прогнозування для кількох різних алгоритмів для даної бізнес-завдання та допомагає вибрати найкращий алгоритм машинного навчання.

3.1.3 Мова програмування Python

Python лідирує усіма іншими мовами: більше 60% розробників машинного навчання використовують і пріоритизують його для розробки, тому що python простий в освоєнні. Масштабований та відкритий вихідний код Python має багато приголомшливих пакетів візуалізації та корисних основних бібліотек, таких як Numpy, scipy, pandas, matplotlib, seaborn, sklearn, що дійсно робить роботу дуже простою та дозволяє машинам вчитися.

3.2 Аналіз фреймворків для розробки нейронних мереж

3.2.1 Бібліотека Numpy

Numeric Python або Numpy – це бібліотека Python, яка використовується для роботи з масивами. Він також має функції для роботи в галузі лінійної алгебри, перетворення Фур'є та матриць. NumPy був створений у 2005 році Тревісом Оліфантом. Це проект з відкритим вихідним кодом і ви можете використовувати його вільно. У Python ми маємо списки, які служать для масивів, але вони повільно обробляються. NumPy прагне надати об'єкт масиву, який працює до 50 разів швидше ніж традиційні списки Python.

Об'єкт масиву в NumPy називається ndarray, він надає безліч допоміжних функцій, що спрощують роботу з ndarray. Масиви дуже часто використовуються в data sciense, де дуже важливі швидкість та ресурси.

Data science: це галузь інформатики, в якій навчаються вивчаємо, як зберігати, використовувати та аналізувати дані для отримання інформації. Масиви NumPy зберігаються в одному безперервному місці в пам'яті, на відміну від списків, тому процеси можуть ефективно звертатися до них і

маніпулювати ними. Така поведінка називається локальністю посилання в інформатиці. Це основна причина, через яку NumPy працює швидше, ніж списки. Також його оптимізовано для роботи з останніми архітектурами ЦП. NumPy частково написана на Python, але більшість частин, які потребують швидких обчислень, написані на C або C++.

3.2.2 Бібліотека Pandas

Це найпопулярніша бібліотека Python, яка забезпечує високу оптимізовану продуктивність для аналізу даних. Pandas – це бібліотека з відкритим вихідним кодом, створена в основному для простої та інтуїтивно зрозумілої роботи з реляційними або розміченими даними. Він надає різні структури даних та операції для обробки числових даних та часових рядів. Ця бібліотека побудована поверх бібліотеки NumPy. Pandas працює швидко і має високу продуктивність та продуктивність для користувачів.

Pandas була спочатку розроблена Уесом МакКінні в 2008 році, коли він працював у AQR Capital Management. Він переконав AQR дозволити йому відкрити вихідний код Pandas. Інший співробітник AQR, Чанг Ше, приєднався до бібліотеки як другий великий учасник у 2012 році. Згодом було випущено безліч версій панд. Pandas стала відомою через те, що вона може швидко та ефективно для обробляти та аналізувати дані.Також з цією бібліотекою можливі завантаження даних із різних файлових об'єктів, проста обробка відсутніх даних (представлених як NaN) з плаваючою комою, а також даних без плаваючої коми. Стали можливі змінюваність розміру стовпця: можна вставляти та видаляти з DataFrame та об'єктів вищого виміру, злиття та об'єднання наборів даних, зміна форми та поворот наборів даних. Pandas забезпечує функціональність часових рядів, потужне угруповання за функціональними можливостями для виконання операцій поділу-застосування-об'єднання наборів даних.

3.2.3 Бібліотека Matplotlib

Це популярна бібліотека графіків Python, яка використовується для створення базових графіків, таких як лінійні діаграми, гістограми, гістограми та багато іншого. Matplotlib – це кросплатформова бібліотека для візуалізації даних та графічної побудови графіків для Python та його числового розширення NumPy. Таким чином він пропонує життєздатну альтернативу MATLAB з відкритим вихідним кодом. Розробники також можуть використовувати інтерфейси API matplotlib (інтерфейси прикладного програмування) для вбудовування графіків у додатки з графічним інтерфейсом.

Сценарій Python matplotlib структурований таким чином, що в більшості випадків для створення графіка візуальних даних достатньо кількох рядків коду. Шар сценаріїв matplotlib перекриває два API:

– API pyplot є ієрархією об'єктів коду Python, очолювану matplotlib.pyplot;

– колекція об'єктно-орієнтованих об'єктів API, які можуть бути зібрані з більшою гнучкістю, ніж pyplot. Цей API забезпечує прямий доступ до внутрішніх шарів Matplotlib.

API-інтерфейс pyplot має зручний інтерфейс із відстеженням стану у стилі MATLAB. Фактично matplotlib спочатку був написаний як альтернатива MATLAB з відкритим вихідним кодом. API та його інтерфейс більш налаштовуються та потужні, ніж pyplot, але вважаються складнішими у використанні. В результаті частіше використовується інтерфейс pyplot.

3.2.4 Бібілотека Seaborn

Matplotlib виявився неймовірно корисним і популярним інструментом візуалізації, але навіть завзяті користувачі визнають, що він деколи може бути

більш вдосконаленим. Є кілька обґрунтованих скарг на Matplotlib, які часто виникають:

– до версії 2.0 значення Matplotlib за замовчуванням не зовсім найкращий вибір. Він був заснований на Matlab приблизно в 1999 році, і це часто видно;

– API Matplotlib щодо низького рівня. Виконання складної статистичної візуалізації можливе, але часто потребує великої кількості шаблонного коду;

– matplotlib передував Pandas більш як на десять років і тому не призначений для використання з Pandas DataFrames. Щоб візуалізувати дані з Pandas DataFrame, ви повинні вилучити кожну серію та часто об'єднувати їх разом у правильному форматі. Було б краще мати бібліотеку побудови графіків, яка може розумно використовувати мітки DataFrame на графіку.

Вирішенням цих проблем є Seaborn. Seaborn надає API поверх Matplotlib, який пропонує розумний вибір стилю графіка та кольорів за замовчуванням, визначає прості високорівневі функції для поширених типів статистичних графіків та інтегрується з функціями Pandas DataFrames.

3.2.5 Бібліотека Sci-kit Learn

Scikit-learn – один з найбільш широко використовуваних пакетів Python для Data Science та Machine Learning. Він дозволяє виконувати безліч операцій та надає безліч алгоритмів. Scikit-learn також пропонує чудову документацію про свої класи, методи та функції, а також опис алгоритмів, що використовуються. Scikit-Learn підтримує:

– попередню обробку даних;

– зменшення розмірності;

– вибір моделі;

– регресії;

– класифікації;

– кластерний аналіз.

Він також надає кілька наборів даних, які можна використовувати для тестування ваших моделей.

Scikit-learn не реалізує все, що пов'язане із машинним навчанням. Наприклад, він не має комплексної підтримки для:

– нейронних мереж;

– самоорганізуються карт (мереж Кохонена);

– навчання асоціативних правил;

– навчання з підкріпленням (reinforcement learning).

Scikit-learn заснований на NumPy та SciPy, тому необхідно зрозуміти хоча б ази цих двох бібліотек, щоб ефективно застосовувати Scikit-learn.

Scikit-learn – це пакет з відкритим вихідним кодом. Як і більшість матеріалів з екосистеми Python, вона безкоштовна навіть для комерційного використання. Його ліцензовано під ліцензією BSD.

3.2.6 Бібліотека Tensorflow

Назва TensorFlow походить від операцій, які нейронні мережі виконують над багатовимірними масивами даних або тензорами. Створена командою Google Brain і спочатку опублікована в 2015 році, TensorFlow — це бібліотека з відкритим вихідним кодом для чисельних обчислень і великомасштабного машинного навчання. TensorFlow об’єднує безліч моделей і алгоритмів машинного навчання та глибокого навчання (він же нейронні мережі) і робить їх корисними за допомогою звичайних програмних метафор. Він використовує Python або JavaScript, щоб забезпечити зручний інтерфейсний API для створення програм, виконуючи ці програми на високопродуктивному C++.

TensorFlow, який конкурує з такими фреймворками, як PyTorch і Apache MXNet, може навчати і запускати глибокі нейронні мережі для рукописної класифікації цифр, розпізнавання зображень, слів, обробки природною мовою та моделювання на основі PDE (диференціальне рівняння в частинних похідних). Найкраще те, що TensorFlow підтримує прогнозування

виробництва в масштабі, з тими ж моделями, що використовуються для навчання. TensorFlow також має широку бібліотеку попередньо навчених моделей, які можна використовувати у ваших власних проектах. Також можна використовувати код із TensorFlow Model Garden як приклад найкращих практик для навчання власних моделей.

3.2.7 Бібліотека PyTorch

Це бібліотека машинного навчання з відкритим вихідним кодом, що використовується для розробки та навчання моделей глибокого навчання на основі нейронних мереж. Він насамперед розроблений дослідницькою групою штучного інтелекту Facebook. PyTorch можна використовувати як з Python, так і C++. Звичайно, інтерфейс Python більш відточений. Pytorch (за підтримки таких гігантів, як Facebook, Microsoft, SalesForce, Uber) є надзвичайно популярним у дослідницьких лабораторіях. Ще не на багатьох виробничих серверах, якими управляють такі компанії, як TensorFlow (за підтримки Google), Pytorch швидко набирає обертів. На відміну від більшості інших популярних середовищ глибокого навчання, таких як TensorFlow, які використовують статичні графи обчислень, PyTorch використовує динамічні обчислення, що забезпечує більшу гнучкість при побудові складних архітектур. Pytorch використовує основні концепції Python, такі як класи, структури та умовні цикли, які нам добре знайомі і, отже, набагато більш інтуїтивно зрозумілі. Це робить це набагато простіше, ніж інші фреймворки, такі як TensorFlow, які використовують власний стиль програмування.

3.2.8 Бібліотека Keras

Keras — бібліотека для мови програмування Python, яка призначена для глибокого машинного навчання. Вона дозволяє швидше створювати та налаштовувати моделі — схеми, якими поширюється і підраховується

інформація під час навчання. Але складних математичних обчислень Keras не виконує та використовується як надбудова над іншими бібліотеками.
Keras з версії 2.3 це надбудова над бібліотекою TensorFlow, яка потрібна для машинного навчання. TensorFlow виконує всі низькорівневі обчислення та перетворення і є своєрідним двигуном, математичним ядром. Keras управляє моделями, за якими проходять обчислення.

До версії 2.3 Keras міг використовувати як двигун обчислювальні бібліотеки Theano або CNTK. Але в нових версіях підтримка припинилася, тепер бібліотека працює лише з TensorFlow. Keras створювалася як гнучка модульна бібліотека, яку легко налаштовувати та модифікувати. Вона безкоштовна, має відкритий вихідний код, який може подивитися будь-хто.
Назва читається як «Керас» і з грецької мови означає «ріг». Це посилання на рядки з «Одіссеї». Keras має вузьку спеціалізацію. Це інструмент для спеціалістів з машинного навчання, які працюють з мовою Python: саме його найчастіше використовують завдяки зручності математичних обчислень. Keras застосовують розробники, які створюють, налаштовують та тестують системи машинного навчання та штучного інтелекту, насамперед нейронні мережі.

3.3 Веб фреймворки

Веб фреймворк – це платформа веб-застосунків або просто веб-платформа є набором бібліотек і модулів, які дозволяють розробникам веб-застосунків писати програми, не турбуючись про низько рівневі деталі, такі як протокол, управління потоками і т.д.

3.3.1 Фреймворк Django

Django це безкоштовне середовище веб-застосінкків з відкритим кодом, написане на Python. Фреймворк - це не що інше, як набір модулів, що

полегшують розробку. Вони згруповані разом і дозволяють створювати програми або веб-сайти з наявного джерела, а не з нуля.

Ось як веб-сайти, навіть прості, розроблені однією людиною, можуть, як і раніше, включати розширені функції, такі як підтримка автентифікації, панелі керування та адміністратора, контактні форми, поля для коментарів, підтримка завантаження файлів та багато іншого. Іншими словами, якщо б ви створювали сайт з нуля, вам потрібно було б розробити ці компоненти самостійно. При використанні замість цього фреймворку ці компоненти вже створені, вам просто потрібно правильно налаштувати їх, щоб вони відповідали вашому сайту.

Офіційний сайт проекту описує Django як «веб-фреймворк Python високого рівня, який сприяє швидкій розробці та чистому прагматичному дизайну. Створений досвідченими розробниками, він бере на себе велику частину клопоту, пов'язаних з веб-розробкою, тому ви можете зосередитися на написанні своєї програми, не винаходячи велосипед. Це безкоштовно та з відкритим вихідним кодом».

Django пропонує велику колекцію модулів, які ви можете використати у своїх проектах. Насамперед, фреймворки існують для того, щоб заощадити розробникам багато витраченого часу та головного болю, та Django не виняток.

Мова шаблонів Django розроблена таким чином, щоб вона була зручною та простою в освоєнні для тих, хто звик працювати з HTML, таких як дизайнери та розробники інтерфейсів. Але він також гнучкий і розширюваний, що дозволяє розробникам розширювати мову шаблонів за необхідності.

3.3.2 Фреймворк Flask

Flask – це веб-фреймворк, модуль Python, який дозволяє легко розробляти веб-програми. У нього невелике ядро, що легко розширюється: це

мікро фреймворк, який не включає ORM (Object Relational Manager) або подібні функції.

У нього є багато цікавих функцій, таких як маршрутизація URL-адрес, механізм шаблонів. Це фреймворк для веб-програм з WSGI.

Flask – це фреймворк для веб-застосунків, написаний на Python. Він був розроблений Армін Ронахером, який очолював команду міжнародних ентузіастів Python під назвою «Pocoo». Flask заснований на наборі інструментів Werkzeg WSGI та механізмі шаблонів Jinja2. Обидва проекти команди «Pocoo». Особливості Flask заключаються у:

– WSGI (інтерфейс шлюзу веб-сервера, використовувався як стандарт для розробки веб-застосунків Python). WSGI – це специфікація загального інтерфейсу між веб-серверами та веб-застосунками;

– Werkzeug (набір інструментів WSGI, що реалізує запити, об'єкти відповідей та службові функції) що дозволяє побудувати на ньому веб-фрейм. Фреймворк Flask використовує Werkzeg як одну зі своїх основ;

– Jinja2 (популярний механізм шаблонів для Python) система веб-шаблонів об'єднує шаблон із певним джерелом даних для відображення динамічної веб-сторінки, це дозволяє передавати змінні Python в HTML-шаблони таким чином, як це зображено на рисунку 3.1.

```
<html>
    <head>
        <title>{{ title }}</title>
    </head>
    <body>
        <h1>Hello {{ username }}</h1>
    </body>
</html>
```

Рисунок 3.1 – Приклад використання механізму Jinja2[29]

Flask часто називають мікро фреймворком. Він призначений для того, щоб ядро програми залишалося простим і масштабованим.

Замість рівня абстракції для підтримки бази даних Flask підтримує розширення для додавання таких можливостей у програму. На відміну від фреймворку Django, Flask дуже «Pythonic». Почати роботу з Flask легко, тому що у нього немає великої кривої навчання. Крім того, це дуже підвищує читабельність. Щоб створити програму Hello World, знадобиться всього кілька рядків коду.

Висновки розділу 3

У цьому розділі були проаналізовані найпопулярніші мови програмування та фреймворки для розробки нейронних мереж. Були перелічені їх недоліки та переваги. У якості основних технологій для розгортки проекту були обрані мова програмування Python з її фреймворками, включаючи і Pythorch, NumPy, Matplotlib та Flask. Останньому було надано перевагу через те, що фреймворк виконує мінімальну роботу в цьому проєкті і веб-застосунок взагалі не потребує бази даних. Замість накопичення та відправки даних, сервер просто зберігає відправлений файл, проводить здогадку щодо значення жесту на кадрі та повертаючи відповідь одразу видалятиме файл зі своєї системи. Django ж більше підходить для адміністрування застосунком та його бази даних.

4 ПРОГРАМНА РЕАЛІЗАЦІЯ СИСТЕМИ

4.1 Короткий опис системи

На даному етапі були вже обрані всі інструменти, які допоможуть в розробці проекту. Оскільки за мету було обрано систему без складного інтерфейсу, для можливостей масштабування в інші типи застосунків, алгоритм роботи з користувачем повинен бути однаково простим, для можливості надбудови над цим проєктом. Найлегшим варіантом буде найпоширеніший протокол HTTP. Користувач за натиском кнопки відправляє запит на сервер. Спочатку цей запит отримує служба маршрутизації і вже потім звертається до файлу з моделлю нейронної мережі, так як зображено на кресленику ІА81.230БАК.003 Д4.

4.2 Обробка даних

Нейронна мережа сприймає ключові ознаки дещо інакше, ніж людина. Комп’ютер робить якісь висновки, залежно від послідовності у масиви або матриці конкретних значень. Тобто перед тим, як сформувати основні ознаки, слід перевести колекцію даних у зрозумілий комп’ютеру вигляд. Для цього слід спочатку написати клас колекції даних. Клас описаний, як на рисунку 4.1.

```
class ASLDataset(Dataset):

    @staticmethod
    def get_mapped_lables():
        list_of_lables = list(range(25))
        list_of_lables.pop(9)
        return list_of_lables

    @staticmethod
    def get_labels_and_samples_from_file(path):
        mapping = ASLDataset.get_mapped_lables()
        labels, samples = [], []
        with open(path) as f:
            _ = next(f)
            for row in csv.reader(f):
                label = int(row[0])
                labels.append(mapping.index(label))
                samples.append(list(map(int, row[1:])))
        return labels, samples
```

Рисунок 4.1 – Клас колекції даних, що успадковується від класу Dataset з бібліотеки Torch

Далі, спираючись на цей каркас, створюються різні файли .csv формату, з переліченими мітками на кожному кадрі з даними, як це зображено на рисунку 4.2. Один з файлі буде використовуватися для тренування моделі, інший для тестування.

```
def __getitem__(self, idx):
    transform = transforms.Compose([
        transforms.ToPILImage(),
        transforms.RandomResizedCrop(28, scale=(0.8, 1.2)),
        transforms.ToTensor(),
        transforms.Normalize(mean=self.features, std=self.standart)])

    return {
        'image': transform(self.samples[idx]).float(),
        'label': torch.from_numpy(self.labels[idx]).float()
    }

def get_sets_loaders(normal):
    train = ASLDataset('/Users/artemselezen/Documents/sign_language/data/sign_mnist_train.csv')
    first_loader = torch.utils.data.DataLoader(train,normal, shuffle=True)

    test = ASLDataset('/Users/artemselezen/Documents/sign_language/data/sign_mnist_test.csv')
    second_loader = torch.utils.data.DataLoader(test, normal, shuffle=False)
    return first_loader, second_loader
```

Рисунок 4.2 – Створення колекції навчальної колекції.

На цьому створення колекцій завершуються. Далі, нам необхідно створити модель нейронної мережі та натренувати її.

4.3 Модель нейронної мережі

Функцію активації було обрано ReLU. Вона була запозичена з бібліотеки Torch. Створення моделі нейронної мережі починається з оголошення аргументів класу. В якості полей класу було оголошені шари згортки з необхідними параметрами та пуллінг, який ми будемо використовувати. Також в якості методу класу було описано тип поширення сигналу в даній нейронній мережі. Під час тренування, були виділені ознаки для кожного екземпляру зі створеного .csv файлу. Розробник здатен сам обирати кількість генерацій нейронів, що будуть навчатися, як це показано на рисунку 4.3. Проте занадто велике число не гарантує кращу якість тренування. Після якогось

конкретного значення параметри нейронної мержі покращуватися перстануть. Весь процес супроводжується детальним описом тренування кожного нейрона та кількістю втрат сигналів.

```
def main():
    neral_network = Net().float()
    basis = nn.CrossEntropyLoss()
    correction = optim.SGD(neral_network.parameters(), lr=0.01, momentum=0.9)
    step = optim.lr_scheduler.StepLR(correction, step_size=10, gamma=0.1)
    loader, _ = get_sets_loaders(32)

    for generation in range(40):  # loop over the dataset multiple times
        train(neral_network, basis, correction, loader, generation)
        step.step()
    torch.save(neral_network.state_dict(), "features.pth")
```

Рисунок 4.3 – Метод, в якому обираються параметри навчання нейронної мережі

Параметри моделі були завантажені у файл features.pth. Пакет torch включає в себе функцію збереження у файл за назвою, а якщо такого файлу не знайдеться, створиться новий. На рисунку 4.4 зображений результат тренування моделі, після створення 40 епох нейронів. На ньому видно, що втрати при такій конфігурації є дуже малими.

```
Generation 38, neurons: 400 loss: 0.00279
Generation 38, neurons: 500 loss: 0.002827
Generation 38, neurons: 600 loss: 0.002835
Generation 38, neurons: 700 loss: 0.002662
Generation 38, neurons: 800 loss: 0.002639
Generation 39, neurons: 0 loss: 0.000545
Generation 39, neurons: 100 loss: 0.001127
Generation 39, neurons: 200 loss: 0.002546
Generation 39, neurons: 300 loss: 0.002902
Generation 39, neurons: 400 loss: 0.002774
Generation 39, neurons: 500 loss: 0.00269
Generation 39, neurons: 600 loss: 0.002581
Generation 39, neurons: 700 loss: 0.002492
Generation 39, neurons: 800 loss: 0.00252
```

Рисунок 4.4 – Вивід програми, після тренування нейронної мережі

Тепер слід оцінити модель розпізнавання мови жестів, обчисливши її точність на контрольній вибірці, наборі даних, які модель не побачила під час навчання. Це дозволить краще зрозуміти продуктивність моделі, ніж за допомогою фінальної втрати.

На рисунку 4.5 зображено алгоритм оцінки параметрів створеної моделі, яка була завантажена у файл features.pth. Для цього використовується оцінка за допомго batch оцінки. Batch – це група зображень, збережена як один тензор. Потім модель PyTorch експортується до бінарного файлу ONNX.

```
def data_validation():
    loader_for_training, loader_for_testing = get_sets_loaders(32)
    network = Net().float().eval()
    saved_network = torch.load("/Users/artemselezen/Documents/sign_language/features.pth")
    network.load_state_dict(saved_network)
    accuracy_of_training = batch_estimation(network, loader_for_training) * 100.
    accuracy_of_testing = batch_estimation(network, loader_for_testing) * 100.
    loader_for_training, loader_for_testing = get_sets_loaders(1)
    print('-' * 10, 'PyTorch model:', '-' * 10)
    print('Training accuracy: ', accuracy_of_training)
    print('Validation accuracy: ', accuracy_of_testing)
    data_file_name = "ASL.onnx"
    random = torch.randn(1, 1, 28, 28)
    torch.onnx.export(network, random, data_file_name, input_names=['input'])
    neural_network_model = onnx.load(data_file_name)
    onnx.checker.check_model(neural_network_model)
    ort_session = ort.InferenceSession(data_file_name)
    network = lambda inp: ort_session.run(None, {'input': inp.data.numpy()})[0]
    accuracy_of_training = batch_estimation(network, loader_for_training) * 100.
    accuracy_of_testing = batch_estimation(network, loader_for_testing) * 100.
    print('-' * 10, 'ONNX model:', '-' * 10)
    print('Training accuracy: ', accuracy_of_training)
    print('Validation accuracy: ', accuracy_of_testing)
```

Рисунок 4.5 – Метод оцінки моделі

Надалі цей бінарний файл можна використовувати у виробничому середовищі для запуску логічного виводу для моделі. Не менш важливо, що для коду, який запускає цей бінарний файл, не потрібна копія визначення оригінальної мережі. Модель у бінарному вигляді також була оцінена. Результат можна побачити на рисунку 4.6.

```
---------- PyTorch model: ----------
Training accuracy:  99.93079584775086
Validation accuracy:  98.24316787506972
---------- ONNX model: ----------
Training accuracy:  99.93808049535605
Validation accuracy:  98.18739542665924
```

Рисунок 4.6 – Результат навчання моделей

Коли результати отримані та є задовільними, можна описати метод прогнозування відповіді. Тут дуже важливе правильне зчитування кадру, тому була використана бібліотека opencv-python, яка була імпортована під названою cv2. У ній також є функції захоплення кадрів спеціально для комп'ютерного зору у режимі реального часу. Ця бібліотека чудово справляється з обрізанням або зміни розміру зображення або відео при правильно заданих параметрах. Варто її виростати в файлі прогнозування, як це зроблено на рисунку 4.7.

```
def predict(file_path):
    basis = 0.485 * 255.
    standart = 0.229 * 255.
    session = onnxruntime.InferenceSession("/Users/artemselezen/Documents/sign_language/data/signlanguage.onnx")
    image = cv2.imread(file_path, cv2.IMREAD_UNCHANGED)
    image = size_correction(image)
    image = cv2.cvtColor(image, cv2.COLOR_RGB2GRAY)
    x = cv2.resize(image, (28, 28))
    x = (x - basis) / standart
    x = x.reshape(1, 1, 28, 28).astype(np.float32)
    y = session.run(None, {'input': x})[0]
    index = np.argmax(y, axis=1)
    result = index_to_answer[int(index)]
    return result
```

Рис 4.7 – Метод прогнозування відповіді

Після закінчення розробки нейронної мережі та всіх методів для її зручного користування наступною стає розробка веб-застосунку, на якому модель мережі буде знаходитися.

4.4 Реалізація веб-застосунку

Документація фреймворку Flask відрізняється від інших бібліотек своєю зручність та читабельністю. На сайті описані всі стандартні процедури по встановленні цього фреймворку, та розробки серверу, який може повертати прості відповіді на запити до нього. Тому тут біде описані лише файли, котрі були дороблені спеціально під цей проект. Стандартом поведінки серверів на Flask є опис дій під час запиту користувача на ту чи іншу адресу. В цьому проєкті, маршрутизація виглядає, як описано на рисунку 4.8.

```
@app.route("/", methods=['GET'])
def home_page():
    return render_template('index.html')


@app.route("/translate", methods=['GET', 'POST'])
def translate():
    if request.method == "POST":
        f = request.files['file']
        filename = secure_filename(f.filename)
        f.save(os.path.join(app.config['UPLOAD_FOLDER'], filename))
        path = os.path.join(app.config['UPLOAD_FOLDER'], f.filename)
        result = predict(path)
        os.remove(path)
        return render_template('result.html', result = result)
```

Рисунок 4.8 – Файл маршрутизації застосунку

Запити до сервера можуть бути у досить різних форматах, проте нам вистачить лише POST та GET. Запит GET нічого не відправляє на сервер, окрім звичайного запиту до інформації. Метод GET використовується для відправки інформації, проте вона формується в адресній стрічці браузера. Це не є загальноприйнятою практикою та абсолютно йде наперекір парадигмам REST. Метод POST навпаки, може передати будь-що, інформацію або файл у своєму тілі. Саме цей метод являться основним методом при відправки будь-чого на сервер. Тому для доступу на веб-застосунок ми використовуємо GET запит. У відповідь сервер відправить файл index.html. Flask візуалізує

шаблонну сторінку за допомогою методу render_template(). Він використовується для створення виводу з файлу шаблону на основі механізму Jinja2, який знаходиться в папці шаблонів програми і має вигляд, як на рисунку 4.9.

```
<!DOCTYPE html>
<html lang="en">
    <body>
        <form action = "http://127.0.0.1:5000/translate" method = "POST" enctype = "multipart/form-data">
            <input type = "file" name = "file" />
            <input type = "submit" value = "Submit" />
        </form>
    </body>
</html>
```

Рисунок 4.9 – Шаблон головної сторінки застосунку

Цей шаблон консервується браузером у сторінку, як показано на рисунку 4.10.

Choose File no file selected

Submit

Рисунок 4.10 – Головна сторінка веб-застосунку

Після вибору потрібного файлу та натисканні кнопки «Submit» відбувається відправка форми з файлом на сервер. Він здатен приймати будь-які запити через утиліту request. Клас Request() модуля flask створює об'єкт запиту при кожному зверненні до URL-програми, який, у свою чергу, запам'ятовує збігається кінцеву точку endpoint і аргументи, що передаються в функцію-подання. Клас flask.Request є підкласом werkzeug.wrappers.Request та надає всі атрибути, визначені Werkzeug, плюс кілька специфічних для Flask.

У відповідь на відправлений користувачем файл веб застосунок переадресує користувача на іншу URL адресу, на якій відбудеться візуалізація шаблону відповіді. Цей шаблон продемонстровано на рисунку 4.11.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Result</title>
</head>
<body>
This is letter {{result}}
<br>
<button onclick="history.back()">Go Back</button>
</body>
</html>
```

Рисунок 4.11 – Шаблон відповіді

Браузер розшифрує цей шаблон та представить його у вигляді, як це зображено на рисунку 4.12.

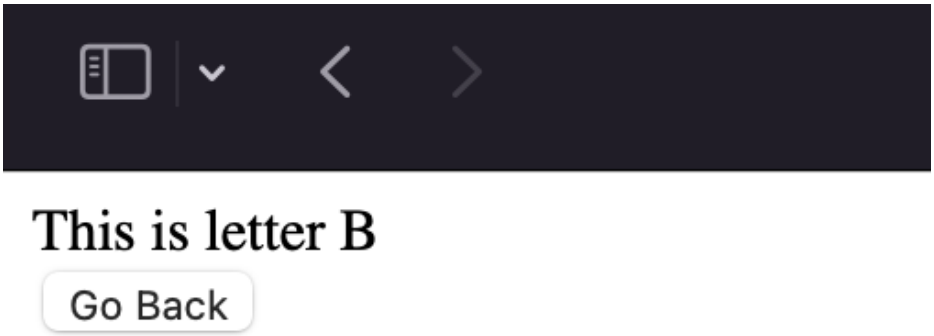

Рисунок 4.12 – Результат прогнозування нейронної мережі

Висновки розділу 4

В даному розділі була розроблена програмна реалізація системи. Для цього спочатку необхідно було обробити колекцію даних, на основні яких проводилося б навчання. Для зручної експлуатації, дані були експортовані в екземпляри класу DataLoader. Для максимальної ефективності оцінки та тренуванні моделі нейронної мережі були створені екзмепляри класу DataLoader з різними конфігураціями, а саме різною batch-нормалізацією. Далі була описана модель нейронної мережі та алгоритми її тренування та оцінювання. Були написані шаблони сторінок веб-застосунок та сценарії відправки запитів на сервер.

ВИСНОВКИ

В ході виконання індивідуально проєкту було спроектовано та розроблено веб-застосунок для розпізнавання мови жестів.

Розробка веб-застосунку почалась з вивчення актуальності проблеми спілкування людей, у яких є проблеми з голосовим або слуховим апаратом, та людей, що не мають таких особливостей. Були розглянуті різні колекції даних, що могли стати матеріалом для навчання. Для навчання була обрана колекція WLASL, бо колекції української мови жестів, ще ніхто не створи. Її структура була ретельно розібрана та завантажена.

У другому розділі було ближче ознайомлено з поняттям нейронної мережі, їх поділи, структури, архітектури. Також було розібрано елементи нейронних мереж.. Були розглянуті та описані найпопулярніші функції активації, а саме їх вигляд, недоліки та переваги. Далі були детально описані згорткові мережі, їх алгоритми згортки та поняття нормалізації. Ще в другому розділі були розглянуті найпопулярніші згорткові мережі, а саме AlexNet, VGG, ResNet та GoogLeNet/InceptionNet. Остання стала прообразом для подальшої розробки програми. Також у другому розділі були розглянуті існуючі рішення. На жаль, ресурсів, що мають схожий функціонал дуже мало.

У третьому розділі були розглянуті програмні можливості для розробки нейронних мереж. Були названі найпоширеніші мови програмування та їх утиліти, їх недоліки та переваги, що можуть бути використані у проєкті. Мовою програмування був обраний Python, бо він ще має досить сильний пакет інструментів для розробки саме веб-застоснку.

В останньому розділі був детально обписаний процес програмної реалізації нейронної мережі. На жаль, під час розробки виявилося, що оперативної пам’яті для навчання саме на відео матеріалах на машині було не достатньо, тому вибірка тренування була змінена на колекцію зображень.

Розроблений веб-застосунок призначений для розпізнавання жесту по зображенню, та відправки результату прогнозування користуву.

Застосунок має переваги над існуючими аналогами:

– кросплатформеність;

– доступність;

– переклад саме з мови жестів на текстову, а не навпаки.

До останнього етапу індивідуального дослідницького проєкту відноситься програмна реалізація та тестування створеного веб-застосунку, та його відповідність заявленим функціональним вимогам. Під час тестування було виявлені невеликі похибки у перекладі, наприклад на запит з літерами «U» та «R», через їх візуальну схожість застоснок висилає однакову відповідь, але у більшості випадках результат був правильним. Було зроблено висновок, що застосунок справний, та відповідає вимогам індивідуального дослідницького проєкту.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. «WLASL: A large-scale dataset for Word-Level American Sign Language (WACV 20' Best Paper Honourable Mention)» URL: https://github.com/dxli94/WLASL/blob/master/start_kit/WLASL_v0.3.json;
2. «Відео жесту «Book» з коллекції WLASL» URL: http://aslbricks.org/New/ASL-Videos/book.mp4;
3. «Facial Detection and Recognition With Dlib» URL: https://www.width.ai/post/facial-detection-and-recognition-with-dlib;
4. «Top 3 Neural Network Activation Functions» URL: https://www.hackdeploy.com/top-3-neural-network-activation-functions/;
5. «A guide to convolution arithmetic for deep learning» URL: https://arxiv.org/pdf/1603.07285.pdf;
6. «Convolutional Neural Network (CNN) In Deep Learning» URL: https://python.plainenglish.io/convolution-neural-network-cnn-in-deep-learning-77f5ab457166;
7. «Feedforward neural network - AshokBhat/ml Wiki» URL: https://github-wiki-see.page/m/AshokBhat/ml/wiki/Feedforward-neural-network
8. «VGG Very Deep Convolutional Networks (VGGNet) – What you need to know» URL: https://viso.ai/deep-learning/vgg-very-deep-convolutional-networks/;
9. «Deep Residual Learning for Image Recognition» URL: https://arxiv.org/pdf/1512.03385.pdf;
10. «Implementing a LeNet-5 step by step» URL: https://subscription.packtpub.com/book/big-data-&-business-intelligence/9781788831109/4/ch04lvl1sec33/implementing+a+lenet-5+step+by+step;
11. «A Walk-through of AlexNet» URL: https://medium.com/@smallfishbigsea/a-walk-through-of-alexnet-6cbd137a5637;
12. «Going deeper with convolutions» URL: https://arxiv.org/pdf/1409.4842.pdf

13. «ML Inception Network V1» URL: https://www.geeksforgeeks.org/ml-inception-network-v1/
14. «Charu C. Aggarwal Neural Networks and Deep Learning: A Textbook Springer 2018 P.5, 350;
15. «Top 5 Programming Languages and their Libraries for Machine Learning in 2020» URL: https://www.geeksforgeeks.org/top-5-programming-languages-and-their-libraries-for-machine-learning-in-2020/;
16. «Very Deep Convolutioal Networks For Large-Scale Image Recognition» URL: https://arxiv.org/pdf/1409.1556.pdf;
17. «English to Sign Language (ASL) Translator» URL: https://wecapable.com/tools/text-to-sign-language-converter/;
18. «Hand Talk Translator» URL: https://play.google.com/store/apps/details?id=br.com.handtalk&hl=en&gl=US;
19. «Omnibridge» URL: https://omnibridge.ai;
20. «Keras» URL: https://keras.io/api/;
21. «NumPy» URL: https://numpy.org/doc/stable/;
22. «Pandas» URL: https://pandas.pydata.org/docs/;
23. «Matplotlib» URL: https://matplotlib.org/stable/users/index;
24. «Seaborn» URL: https://seaborn.pydata.org;
25. «Scikit Learn» URL: https://scikit-learn.org/stable/;
26. «Tesnorflow» URL: https://www.tensorflow.org/learn;
27. «PyTorch» URL : https://pytorch.org/docs/stable/index.html;
28. «Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift» URL: https://arxiv.org/pdf/1502.03167.pdf;
29. «What is Flask Python» URL:https://pythonbasics.org/what-is-flask-python/