Міністерство освіти і науки України

Харківський національний університет радіоелектроніки

Факультет Комп'ютерної інженерії та управління

Кафедра Комп'ютерних інтелектуальних технологій та систем

КВАЛІФІКАЦІЙНА РОБОТА

Пояснювальна записка

рівень вищої освіти другий (магістерський)

(рівень вищої освіти)

Оцінки 3D пози і форми людини за допомогою штучних нейронних мереж

(тема)

Виконав:
студент 2 курсу, групи КІТм-21-1

Романов Р.Г.

(прізвище, ініціали)

Спеціальність 123 Комп'ютерна інженерія

(код і повна назва спеціальності)

Тип програми освітньо-професійна

Освітня програма Комп'ютерні інтелектуальні технологі

(повна назва освітньої програми)

Керівник проф. Безсонов О.О.

(посада, прізвище, ініціали)

Допускається до захисту

Зав. кафедри _____ проф. Руденко О.Г.
(підпис) (прізвище, ініціали)

2022 р.

Харківський національний університет радіоелектроніки

Факультет Комп’ютерної інженерії та управління

Кафедра Комп’ютерних інтелектуальних технологій та систем

Рівень вищої освіти другий(магістерський)

Спеціальність 123 – Комп’ютерна інженерія

Тип програми Освітньо-професійна

Освітня програма Комп’ютерні інтелектуальні технології
(повна назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
“ ____ “ _____ 2022 р.

ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ

студентові Романову Руслану Георгійовичу
(прізвище, ім’я, по батькові)

1. Тема роботи Оцінки 3D пози і форми людини за допомогою штучних нейронних мереж

затверджена наказом по університету від “ 07 листопада 2022р № 1455Ст

2. Термін подання студентом роботи до екзаменаційної комісії

3. Вхідні дані до роботи Рекомендаційні системи

Побудова штучної нейронної мережі

Алгоритм визначення пози людини

4. Перелік питань, що потрібно опрацювати в роботі

Аналіз предметної області.

Вибір технологій для реалізації проекта

Розробка плану використання технологій для реалізації проекта

Програмна реалізація проекта

Висновки

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, компьютерних ілюстрацій (слайдів) Презентація 10 слайдів

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1	Аналіз предметної області та постановка	08.11 – 13.11	виконано
	Задачі дипломної роботи		
2	Огляд сучасної літератури за напрямком	13.11 – 20.11	Виконано
	магістерської роботи		
3	Вибір методів рішення	21.11 – 23.11	Виконано
4	Застосування гібридних рекомендаційних	24.11 – 26.11	Виконано
	систем		
5	Експериментальні дослідження	26.11 – 30.12	Виконано
6	Оформлення пояснювальної записки	01.12 – 07.12	Виконано
7	Оформлення графічного матеріалу	08.12 – 10.12	Виконано
8	Захист проекту	19.12 – 22.12	виконано

Дата видачі завдання 07 листопада 2022 р.

Студент _____
(підпис)

Керівник роботи _____ проф. Безсонов О.О.
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ

Пояснювальна записка маг. кваліфікаційної роботи: 78 с., 43 рис., 1табл., 33 джерела.

НЕЙРОННІ МЕРЕЖІ, 3D ФОРМИ І ПОЗИ ЛЮДИНИ, PYTHON, РОЗПІЗНАВАННЯ ЗОБРАЖЕНЬ, РОЗПІЗНАВАННЯ ОБ’ЄКТІВ, МАШИННЕ НАВЧАННЯ

Мета даної роботи є створення веб-додатку для оцінки 3D пози за допомогою нейронних мереж. Штучні нейронні мережі є похідними від глобального спрямування штучного інтелекту. Нейронні мережі можна використовувати для вирішення широкого кола завдань, у тому числі для обробки зображень. Технологія обробки зображень, що ґрунтується на розпізнаванні образів, дозволяє з високою точністю розпізнавати становище людського тіла, тому з'являється можливість відстежувати рухи людини та підраховувати обсяг людської активності.

Об’єктом дослідження є веб-застосунок для оцінки 3D поз людини, а також кількості та якості виконання фізичних вправ на основі технологій використання штучних нейронних мереж.

Предметом дослідження є нейронна мережа для розпізнавання 3D пози і форми людини.

ABSTRACT

Master’s thesis 78 pages, 43 figures, 1 table, 33 sources.

NEURAL NETWORKS, 3D HUMAN SHAPES AND POSES, PYNTHON, IMAGE RECOGNITION, OBJECT RECOGNITION, MACHINE LEARNING

The purpose of this work is to create a web application using neural networks to determine the quantitative and qualitative parameters of human activity. Artificial neural networks are derived from the global direction of artificial intelligence. Neural networks can be used to solve a wide range of tasks, including image processing. Image processing technology based on pattern recognition can recognize the position of the human body with high accuracy, allowing to track human movements and calculate the amount of human activity.

The object of the research is a web application for evaluating the quantity and quality of physical exercises based on artificial neural network technologies.

The subject of the study is a neural network for recognizing the posture and shape of a person.

Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет Комп’ютерної інженерії та управління

Кафедра Комп’ютерних інтелектуальних технологій та систем

АНОТАЦІЯ
КВАЛІФІКАЦІЙНОЇ РОБОТИ

рівень вищої освіти другий (магістерський)

Оцінки 3D пози і форми людини за допомогою штучних нейронних мереж

Виконав:
студент 2 курсу, групи: КІТм-21-1
Романов Р.Г.
Спеціальність 123 Комп’ютерна інженерія
Тип програми освітньо-професійна
Освітня програма Комп’ютерні інтелектуальні технології
Керівник проф. Безсонов О.О.

2022р.

АНОТАЦІЯ

Романов Р.І. Оцінки 3D пози і форми людини за допомогою штучних нейронних мереж – Магістерська кваліфікаційна робота.

Актуальність теми дослідження. Сьогодні штучний інтелект є одним з найважливіших напрямків розвитку в сфері інформаційних технологій. Штучні нейронні мережі виросли з глобального напрямку штучного інтелекту. Нейронні мережі можна використовувати для різних завдань, включаючи обробку зображень. Технологія обробки зображень розпізнавання образів може з високою точністю розпізнавати положення та позу людського тіла, обчислювати обсяг людської діяльності та відстежувати рухи.

Об’єкт дослідження – веб-додаток, який оцінює кількість та якість вправ на основі технології штучної нейронної мережі.

Предмет дослідження - нейронна мережа, яка розпізнає форму та позу людини.

Метою даної роботи є створення веб-додатка з використанням нейронних мереж для визначення кількісних і якісних параметрів діяльності людини. Штучні нейронні мережі є похідними від глобального спрямування штучного інтелекту. Існують деякі системні вимоги:

- необхідність правильного вибору моделі розпізнавання для правильного розпізнавання об'єктів
- використовувати модель, яка не навантажує систему, щоб охопити більше пристроїв, ніж підтримується завантаження.

У першому розділі проаналізовано історичні причини створення систем штучного інтелекту. Розібрали основи побудови систем нейронної мережі. Загальною рисою багатьох типів неконтрольованих і керованих моделей глибокого навчання є велика кількість прихованих шарів нейронів, навчених у поєднанні з градієнтами помилок зворотного поширення та стохастичним градієнтним спадом.

Основна перевага нейронних мереж перед стандартними підходами до

комп'ютерних алгоритмів у тому, що, регулюючи силу зв'язків між нейронами, тобто. змінюючи значення ваги, мережа може сильно зменшувати дані, можна класифікувати точно. У ній описано принципи технології розпізнавання зображень, набір методів виявлення та аналізу зображень з метою автоматизації певних завдань.

Розпізнавання фото- та відеоматеріалів може виконуватися з різним ступенем точності. Моделі можуть виявляти певні функції або призначати зображення широким категоріям з урахуванням їх характеристик. У завдання розпізнавання зображень використовуються методи глибокого навчання, підкатегорія машинного навчання. Належить до серії методів автоматичного навчання, заснованих на штучних нейронних мережах.

Надає огляд ключових архітектур нейронних мереж. Серед них пряме поширення, згорткова та рекурсивна архітектура. Найпростішою формою прямої нейронної мережі є одношаровий персептрон. У нейронній мережі з прямим зв'язком інформація переміщається лише в одному напрямку (від вхідного шару через приховані шари до вихідного шару). Інформація проходить по мережі безпосередньо, ніколи не проходячи через один і той самий вузол двічі.

Рекурентна нейронна мережа - це тип штучної нейронної мережі, яка використовує тимчасові ряди або послідовні дані. У RNN інформація проходить через цикл. Коли система приймає рішення, вона враховує як поточну інформацію, і те, що вона дізналася з раніше отриманої інформації.

У глибокому навчанні згорткові нейронні мережі є найчастіше використовуваним класом глибоких нейронних мереж для аналізу візуальних зображень. Архітектура згорткової нейронної мережі складається з ряду будівельних блоків, які набувають ознак, що відрізняють зображення відповідних класів від інших.

У другому розділі розглянули засоби навчання та розробки нейронних мереж, рекомендується використовувати мову програмування Python для розробки рішень для розпізнавання людини.

Python - одна з мов комп'ютерного програмування, що використовується для

створення програмного забезпечення, особливо нейронних мереж. Вважається бібліотекою TensorFlow для Python. TensorFlow - ця бібліотека Python використовуються для розробки та навчання моделей з використанням нейронних мереж. Однією з основних переваг TensorFlow є те, що програми, написані з використанням цієї бібліотеки, можуть працювати практично на будь-якому пристрої: хмарних кластерах, локальних машинах, пристроях Android та iOS, GPU або CPU. Однак, функції TensorFlow є частиною реалізації TensorFlow, і отримати повністю детерміновані результати навчання моделі складно. Модель, навчена в одній системі, може трохи відрізнятися від моделі, навченої в іншій системі з тими самими вхідними даними. Оскільки створення моделі стеження за руками потребує багато часу та ресурсів, а створення програми, яка ґрунтується на відстеженні людського тіла, утруднено, я використовував MediaPipe для створення нейронної мережі. MediaPipe використовується для моделів ML для досить простих завдань, таких як відстеження окремих частин тіла або поз людини для оптимізації програм машинного навчання.

Основною перевагою MediaPipe є його підтримка прискорення GPU на мобільних платформах. Ми рекомендуємо використовувати фреймворк Flask для створення веб-частин для вашого проекту. Flask надає інструменти, бібліотеки та механізми, які дозволяють створювати веб-додатки без накладення залежностей або жорстких обмежень на те, як має виглядати ваш проект. Виявляється, ви можете використовувати JavaScript у поєднанні з Flask, щоб створити зовнішню частину веб-сайту вашого проекту. Головною перевагою JavaScript є можливість динамічно створювати веб-сторінки.

Проведено огляд процесу генерації даних для навчання нейронних мереж. Щоб навчити та протестувати модель нейронної мережі, нам потрібно розділити дані на набори: навчання та перевірка. Ми рекомендуємо використовувати GPU під час навчання нейронних мереж. Центральний процесор - це послідовний пристрій, який виконує кілька завдань по одному, тоді як графічний процесор є багатопоточним. Тому принцип роботи полягає в паралельній обробці даних

великою кількістю малопотужних ядер. Графічні процесори використовуються для машинного навчання, оскільки нейронні мережі є паралельними алгоритмами.

У третьому розділі ми розробили веб-додаток, який оцінює 3D пози людини, а також якість та кількість рухів тіла на основі технології штучних нейронних мереж. Програмний код розробляється у редакторі вихідного коду PyCharm. Для розробки програми використовували мову програмування Python та бібліотеку TensorFlow. Бібліотека TensorFlow значно розширила можливості мови програмування Python у задачі застосування моделей згорткових нейронних мереж, прискорюючи розробку та реалізацію проектів. Ми реалізували серверну частину веб-програми з використанням можливостей фреймворку Flask.

Flask – це зовнішній шлюз веб-сервера та платформа веб-додатків, призначена для полегшення веб-розробки. Користувальницький інтерфейс веб-програми реалізований за допомогою функцій мови сценаріїв JavaScript з використанням технологій HTML та CSS.

НЕЙРОННІ МЕРЕЖІ, 3D ФОРМИ І ПОЗИ ЛЮДИНИ, PYTHON, РОЗПІЗНАВАННЯ ЗОБРАЖЕНЬ, РОЗПІЗНАВАННЯ ОБ’ЄКТІВ, МАШИННЕ НАВЧАННЯ

Перелік публікацій керівника та співробітників кафедри, використаних в роботі:

1. Rudenko O. et al. Developing a Multi-Step Recurrent Algorithm to Maximize the Criteria of Correntropy. *Eastern-European Journal of Enterprise Technologies*. Vol. 1. No. 4. 2021. 109.

2. Rudenko O. et al. Robust identification of non-stationary objects with nongaussian interference. *Eastern-european Journal of enterprise Technologies.* Vol. 5. No. 4. 2019. 44-52.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

VR - віртуальнареальність (VirtualReality).

AI - штучний інтелект (Artificial intelligence);

API - програмний інтерфейс додатка (Application Programming Interface)

MR - змішана реальність (Mixed Reality);

AR - доповнена реальність (Augmented Reality);

CNN - згорткова нейронна мережа (Convolutional Neural Network);

CPU - центральный процессор (Central Processing Unit);

GPU - графічнийпроцесор (Graphics Processing Unit);

HTML - мова розмітки гіпертексту (HyperText Markup Language);

CSS - каскадні таблиці стилів (Cascading Style Sheets);

ML - машинне навчання (Machine Learning);

NaN - особливий стан числа (Not a Number);

SDK - комплект для розробки програмного забезпечення (Software Development Kit);

ВСТУП

Сфера штучного інтелекту дедалі більше втручається у наше особисте, соціальне та професійне життя. У зв'язку з розвитком напряму штучного інтелекту вимоги і очікування людини ставали дедалі конкретнішими, що призводило до прагнення використовувати штучний інтелект зниження людських помилок і робочої навантаження у більшості галузей людської діяльності. У цих областях моделі нейронних мереж в основному застосовуються для навчання мозку та парадигм функціональних обчислень.

Нейронна мережа - це обчислювальна структура, що з набору однотипних елементів. Ці елементи виконують прості функції, і всі процеси, що відбуваються в штучних нейронних мережах, можна віднести до процесів, що відбуваються в нервовій системі живих організмів. Нейронні мережі за своєю суттю є нелінійними і явно залежними, що робить розроблену технологію доступною. Протягом багатьох років лінійне моделювання було домінуючим методом моделювання, оскільки процедури оптимізації добре розроблені.

Штучні нейронні мережі - це моделі нейронної структури мозку, здатні розпізнавати, обробляти, зберігати та генерувати інформацію, представлену зображеннями. Мозок здатний на самонавчання, він може навчатися на власному досвід, то є його особливістю. Адаптивні системи на основі штучних нейронних мереж можуть успішно вирішувати задачі розпізнавання образів, прогнозування, оптимізації, асоціативної пам’яті, управління та інші інтелектуальні завдання без використання традиційного програмування.

Однак стало зрозуміло, що лише використання людських парадигм функціонування та навчання недостатньо для того, щоб комп’ютери демонстрували автономну поведінку.

Існує три ключові питання, які необхідно вирішити для реалізації автономних систем на основі нейронних мереж. Вони виконують автономні інтелектуальні дії людини. Це передбачає вибір характеристик даних, які

відповідають проблемі, що аналізується. Друга проблема стосується збору баз даних для перевірки результативності системи та навчання. І, звичайно, третє - це обрання схеми класифікації. Ця схема може вплинути на досягнуті результати. Усе це необхідно оптимальної продуктивності з погляду обчислювальної ефективності.

Алгоритмом навчання називається процедура, за допомогою якої здійснюється процес навчання. Основою його функціоналу є те, щоб періодично змінювати синаптичні ваги мережі для досягнення бажаної мети дизайну.

Традиційний метод проектування нейронних мереж забезпечує активна зміна синаптичних коефіцієнтів. Цей підхід найбільш близький до теорії лінійного адаптивного фільтра, яка вже перевірена і використовується в різних галузях промисловості. Однак нейронні мережі можуть змінювати власну топологію, мотивуючи це тим, що нейрони в мозку людини гинуть і можуть створювати нові синаптичні зв’язки.

Нейронні мережі явно отримують вигоду від обчислювальних переваг завдяки своїм масивно паралельним розподіленим структурам і здатності навчатися з різними вагами. Зміна ваги є нормою. Це стосується отримання хороших виходів для вхідних даних, з якими нейронна мережа не стикається під час навчання. Нейронні мережі не можуть надати рішення, які працюють самі по собі. Навпаки, його слід інтегрувати в послідовний підхід до системної інженерії. Зокрема, складну задачу, що представляє інтерес, розбивають на кілька відносно простих завдань, а нейронним мережам призначають підмножину завдань, які відповідають унікальним функціям. Але важливо розуміти, що нам ще належить пройти довгий шлях до створення комп’ютерної архітектури, яка імітує людський мозок.

1 АКТУАЛЬНІСТЬ ТА АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Аналіз основних технологій нейронних мереж

Алгоритми нейронної мережі машинного навчання засновані на архітектурі нейронної мережі [1] та динаміці мозку [2]. Використовуйте дуже ідеалізовану модель нейрона. Проте, основи чи основні принципи залишаються незмінними.

Штучні нейронні мережі мають здатність навчатися, змінюючи зв'язок між нейронами [3], тобто ваги [4]. Штучні нейронні мережі ще досягли рівня складності мозку (рис. 1.1). Однак між біологічними та штучними нейронними мережами є дві важливі подібності. По-перше, базовими будівельними блоками обох мереж є прості обчислювальні пристрої з високим ступенем взаємозв'язку (хоча штучні нейрони набагато простіші за біологічні нейрони). Зв'язки між нейронами визначають функціональність мережі [5].

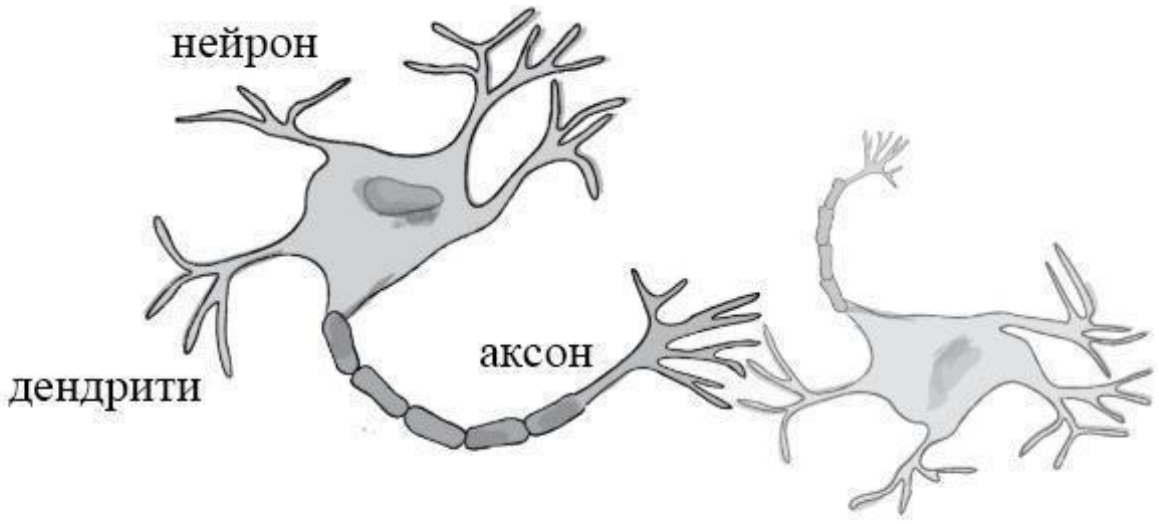

Рисунок 1.1 – Зображення структури нейрона людини

Моделі штучних нейронних мереж використовуються з середини 20 століття [6]. Проте нинішня хвиля популярності нейронних мереж та їх різновидів, нейронних мереж глибокого навчання, припадає на кінець 2000-х років [7]. Спільною рисою багатьох типів керованих і неконтрольованих моделей глибокого навчання ми маємо те, що ці моделі навчаються на багатьох шарах прихованих нейронів у поєднанні зі зворотним поширенням і градієнтами помилок стохастичного градієнтного спаду [8].

Прикладом є те , що нейронні мережі можуть певною мірою узагальнювати інформацію і навчатися розпізнавати подібності в наборах даних[9]. Це основна перевага нейронних мереж перед стандартними підходами до комп’ютерних алгоритмів. Регулюючи міцність зв’язків між нейронами, можна навчити штучні нейронні мережі точно класифікувати дані. Зміна значення вагового коефіцієнта [10]. Якщо нові дані не відрізняються від даних навчання, то результати також можна узагальнити на інші набори даних. Вирішити цю проблему можна з розпізнаванням об'єктів зображення. Самокерований автомобіль. Недавній сплеск інтересу до машинного навчання значною мірою пояснюється успіхом нейронних мереж для візуального розпізнавання об’єктів.

Іншою сферою, де нейронні мережі процвітають і активно використовуються, є машинний переклад [11]. Як правило, для цього використовуються мережі з повторюваною архітектурою. Оператори виходять на вхід цього типу мережі. Якщо ви надсилаєте слово за словом, мережа покаже вам слова перекладеного речення. Рекурентні мережі можна ефективно навчити на великому наборі вхідних речень та їх перекладів. Рекурентні мережі також були успішними у передбаченні хаотичної динаміки [12].

Навчальний набір (рис. 1.2) містить список цільових значень або міток і відповідний список вхідних шблонів. Вони кодують властивості вхідних шаблонів та навчаються мережею.

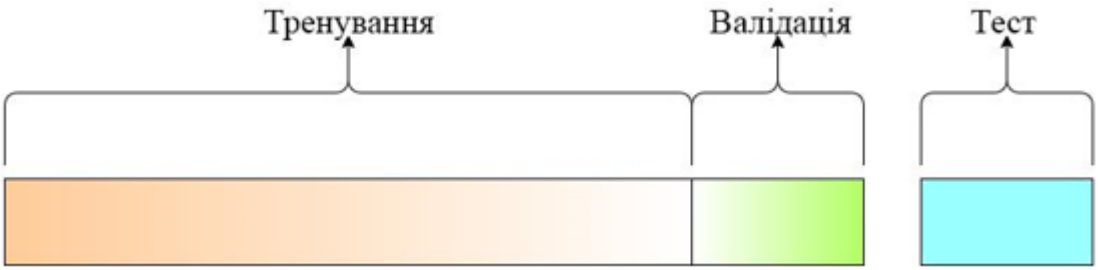

Рисунок 1.2– Схема розподілу даних для навчальної вибірки

Фінансова торгівля, планування підприємства, бізнес-аналітика, додатки для обслуговування продуктів – в усіх цих галузях широко використовуються нейронні мережі [13]. А також використовуються в бізнес-додатках, наприклад, рішення для прогнозування та дослідження ринку, пошук шахрайства та оцінки ризиків.

1.2 Основні принципи розпізнавання зображень

1.2.1 Аналіз сфер використання технології для розпізнавання зображень

Категорія штучного інтелекту та комп’ютерного зору, розпізнавання зображень – це набір методів виявлення та аналізу зображень для автоматизації конкретних завдань. Це метод, який дозволяє нам ідентифікувати місця, людей, об’єкти та багато інших типів елементів на зображеннях і аналізувати їх, щоб робити висновки (рис. 1.4).

Рисунок 1.4 – Схема розпізнавання об'єктів бібліотекою OpenCV

В залежності від типу необхідної інформації, розпізнавання відео- та фотоматеріалів може здійснюватися з різним ступенем точності. Модель або алгоритм можуть виявляти конкретні елементи та класифікувати зображення за широкими категоріями на основі їхніх характеристик. Завдання, які можна виконувати за допомогою розпізнавання зображень, включають:

- виявлення - це необхідно для пошуку об'єктів на зображеннях. Зазвичай після виявлення елемента навколо нього міститься обмежувальна рамка;
- сегментація - це теж питання дослідження. Сегментація може визначити положення елементів зображення з точністю до пікселя. Іноді потрібно бути дуже точним. Наприклад, розробка безпілотних автомобілів;
- класифікація - це ідентифікація класу, до якої належить зображення. Зображення може мати лише один класс;
- маркування - це теж класифікаційне завдання, але з більшою точністю. Нейронні мережі можуть розпізнавати наявність кількох понять або фігур на зображенні. Тому цьому зображенню можна надати кілька тегів.

1.2.2 Аналіз процесу розпізнавання зображень

Розпізнавання зображень - це технологія, яка дає нам змогу ідентифікувати об’єкти, людей, сутності. У наш час користувачі, через програми, соціальні мережі та веб-сайти, обмінюються величезною кількістю даних. Також, зростання кількості смартфонів, оснащених камерами високої роздільної здатності, створює багато цифрових відео і фото. Отже, галузі використовують величезний обсяг цифрових даних для надання кращих та інноваційних послуг. також - це підкатегорія технології комп’ютерного зору та процес, який допомагає ідентифікувати об’єкт або атрибут у цифрових зображеннях або відео. Однак комп’ютерний зір - це ширша команда, що включає різні методи збору, обробки та аналізу даних з реального світу. Оскільки дані є багатовимірними, вони створюють числову та символьну інформацію у формі рішень. Окрім розпізнавання зображень, комп’ютерний зір також включає в себе розпізнавання об’єктів, реконструкцію зображення, виявлення подій і відстеження відео також

на ньому базується техніка глибокого навчання. Глибоке навчання є під категорією машинного навчання і відноситься до набору методів прийомів автоматичного навчання на основі штучних нейронних мереж [17]–[20]

Щоб нейронна мережа розпізнала одне чи кілька понять на зображенні, вона має пройти навчання. Для цього нам необхідно зібрати набір візуальних даних (зазвичай у вигляді зображень) і створити навчальну вибірку [21]. Розпізнавання зображень - це технологія, яка дозволяє нам ідентифікувати об'єкти, людей, об'єкти та інші змінні на зображеннях. У сучасному світі користувачі обмінюються великими обсягами даних через програми, соціальні мережі та веб-сайти. Крім того, число смартфонів з камерами високої чіткості, що росте, буде генерувати безліч цифрових зображень і відео. Тому галузь використовує великі обсяги цифрових даних для надання якісніших та інноваційних послуг.

Після створення набору даних важливо анотувати його. Іншими словами, важливо повідомити моделі, чи містить зображення елемент, який шукає нейронну мережу, і де цей елемент розташований. Існують різні типи міток залежно від обраного завдання.

Після процесу анотування всього набору даних ми можемо перейти навчання. Подібно до людського мозку, нейронні мережі повинні бути навчені розпізнавати поняття, показуючи їм різні приклади.

Кінцева мета навчання дозволити алгоритму робити прогнози після аналізу зображень. Тобто алгоритм має вміти присвоювати зображенню клас або вказувати, чи існує той чи інший елемент.

Системи або платформи розпізнавання зображень можна використовувати для автоматизації бізнес-процесів та підвищення продуктивності. Коли модель розпізнає елемент зображення, його можна запрограмувати виконання певних дій. Варіанти використання цієї автоматизації широко використовуються у різних галузях та секторах.

Наприклад, відділ телекомунікацій запровадив рішення для автоматизації контролю за якістю. Фактично виїзні техніки використовують системи розпізнавання зображень контролю якості устаткування.

Іншим прикладом є інтелектуальна система відеоспостереження на основі розпізнавання зображень, яка може повідомляти про незвичайну поведінку та ситуації на парковках.

Таким чином, розпізнавання зображень може бути використане не тільки в сферах зв’язку та відеоспостереження (рис. 1.6), але також у будівельній та фармацевтичній промисловості.

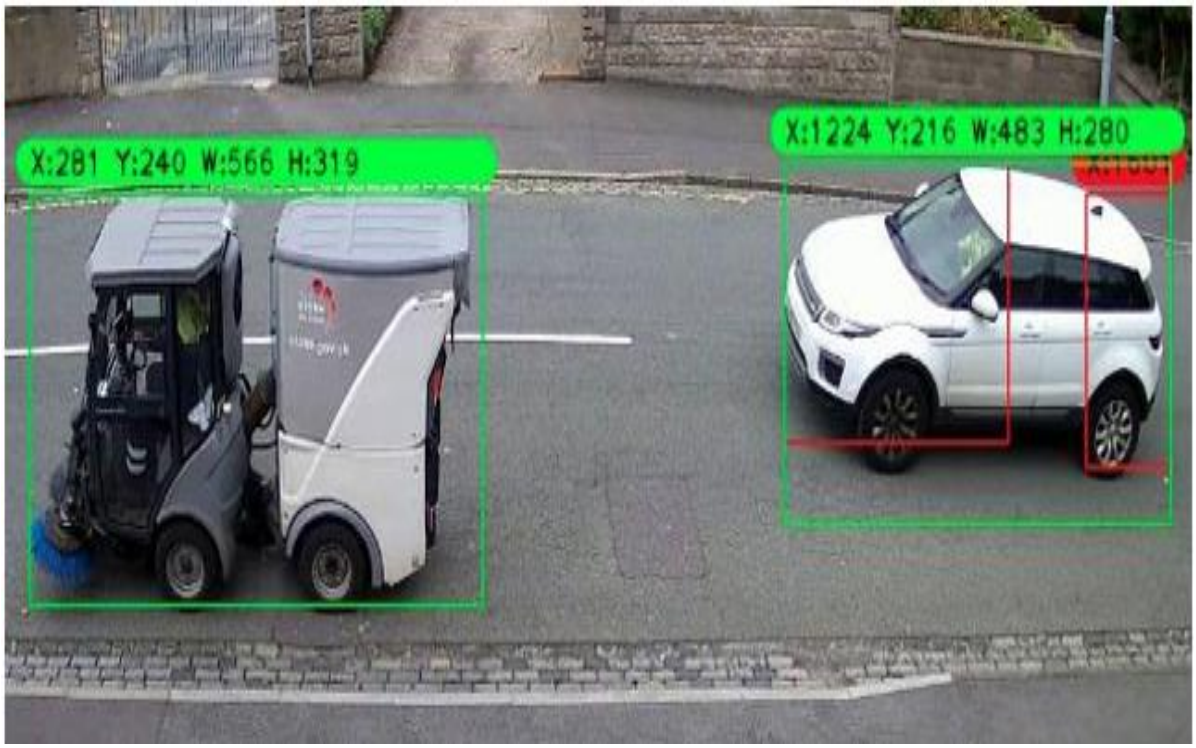

Рисунок 1.6– На зображенні приклад розпізнавання автомобілів камерою зовнішнього спостереження

Цифрове зображення є матрицею чисел. Це число надає дані, які пов'язані з піксельним зображенням. Різні інтенсивності пікселів формують середнє значення та відображаються у вигляді матриці (рис. 1.7).

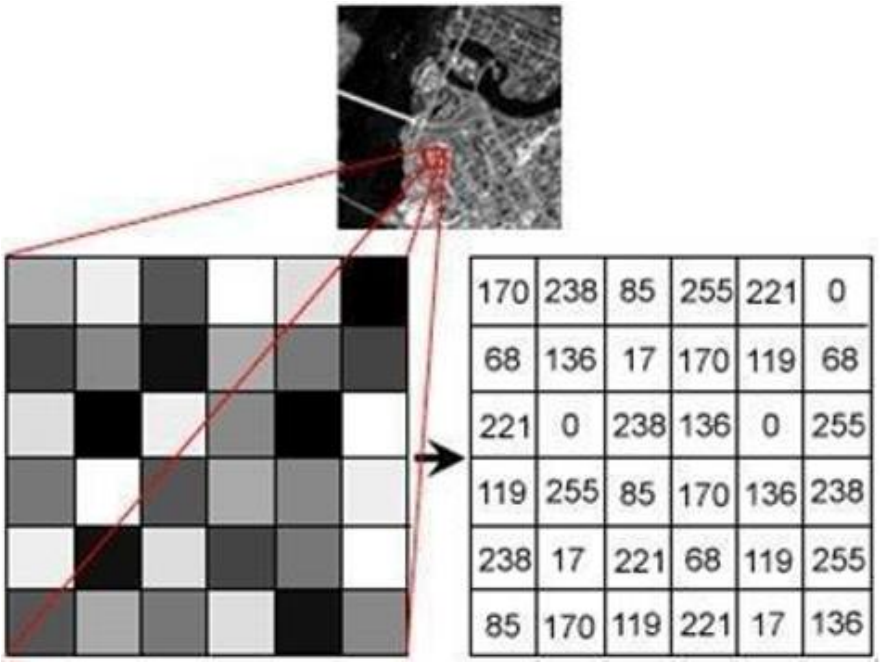

Рисунок 1.7– Матриця значення інтенсивності світла для окремих пікселів чорно-білого зображення

Дані, що відправляються в систему розпізнавання, в першу чергу є інтенсивність та положення різних пікселів на зображенні. Таким чином, систему можна навчити використовувати цю інформацію для демонстрації закономірностей та взаємозв'язків між декількома зображеннями.

Після того, як завершиться процес навчання, можна проаналізувати результативність нашої системи на основі тестових даних. Вага нейронної мережі оновлюється для підвищення точності системи та отримання більш точних результатів розпізнавання зображень. Нейронні мережі використовують алгоритми глибокого навчання для перебору цих чисел та порівняння їх із конкретними параметрами для отримання бажаних результатів.

1.3 Основні види нейронних мереж

1.3.1 Нейронні мережі прямого поширення

Як випливає з назви, дані передаються безпосередньо по мережі. Кожен прихований шар отримує вхідні дані, обробляє їх відповідно до своєї функції

активації і передає наступному шару. Наприклад, персептрон (рис. 1.8). У ньому набір вхідних даних після входу до шару множиться на коефіцієнт (вага). Потім додайте кожне значення, щоб отримати виважену суму вхідних значень. Якщо менше настроєного порогу, вихідне значення дорівнює 0.

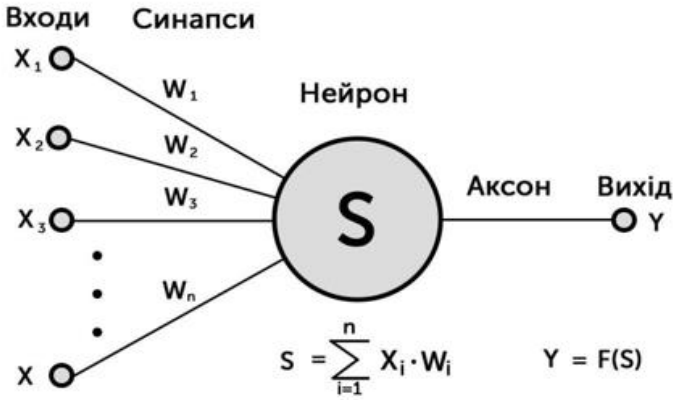

Рисунок 1.8– Одношаровий перцептрон

Для розуміння одношарового перцептрона важливо розуміти штучні нейронні мережі. Штучні нейронні мережі - це система обробки інформації, механізм якої натхненний функціональністю біологічних нейронних ланцюгів. Штучна нейронна мережа має багато процесорних блоків, пов'язаних один з одним. Одношаровий персептрон є першою запропонованою нейронною моделлю. Зміст локальної пам'яті нейрона наповнений вектором ваг. Обчислення одношарового персептрона виконується через обчислення суми вхідного вектора кожного із значенням, помноженим на відповідний елемент вектора ваг. Вихідним сигналом функції активації і буде значення, що відображується у вихідних даних.

У нейронної мережі прямого поширенням (рис. 1.9) інформація поширюється лише одному напрямку від вхідного рівня через приховані верстви до вихідного рівня [22]. Інформація проходить по мережі безпосередньо, ніколи не проходячи через один і той самий вузол двічі.

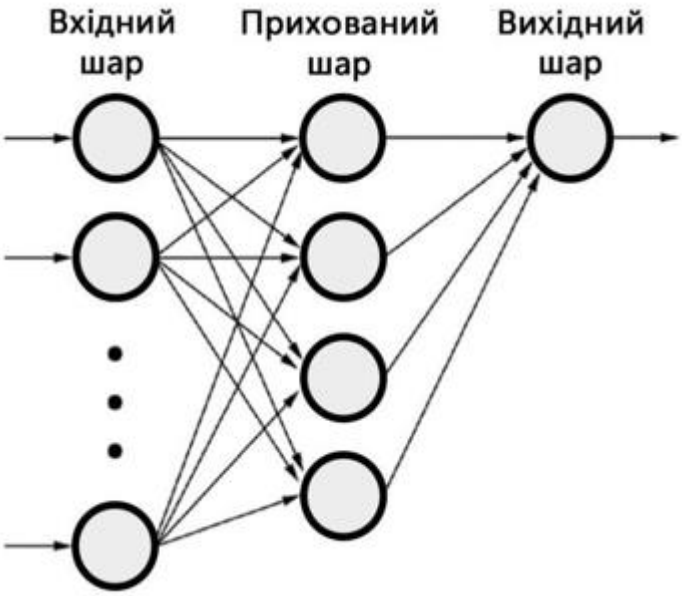

Рисунок 1.9– Схема зображення нейронної мережі прямого поширення

Нейронна мережа з прямим зв'язком використовується для вивчення взаємозв'язку між незалежною змінною, яка служить входом в мережу, і залежною змінною, яка позначається як вихід мережі. Навчання відбувається, коли мережі надається набір зразків «навчального набору» з відомими мітками класів, а ваги мережі встановлюються так, щоб мінімізувати різницю між виходом мережі та відомим правильним виходом. Після коригування ваги з використанням прикладів з навчальної вибірки мережу можна використовуватиме прогнозування належності невідомих вибірок до класу.

1.3.2 Згорткові нейронні мережі

Штучний інтелект був свідком монументального зростання в подоланні розриву між можливостями людей і машин. Дослідники та ентузіасти працюють над багатьма аспектами галузі, щоб створювати дивовижні речі. Однією з багатьох таких сфер є комп’ютерне бачення. Порядок денний для цієї сфери полягає в тому, щоб дозволити машинам дивитися на світ так само, як люди, сприймати його подібним чином і навіть використовувати знання для багатьох завдань, таких як розпізнавання фото- і відеозображень, аналіз і класифікація

зображень, відтворення медіа, рекомендації. Системи, обробка природної мови тощо. Удосконалення комп'ютерного бачення з глибоким навчанням були створені та вдосконалені з часом, головним чином за допомогою одного конкретного алгоритму - згорткової нейронної мережі [26].

Згорткова нейронна мережа (CNN) - спеціальна архітектура штучних нейронних мереж, націлена на ефективне розпізнавання образів, входить до складу технологій глибокого навчання, який може сприймати вхідне зображення, призначати важливість різним об'єктам зображення та має можливість відрізнити один від іншого [27] також вона використовує спеціальну техніку (рис. 1.11).

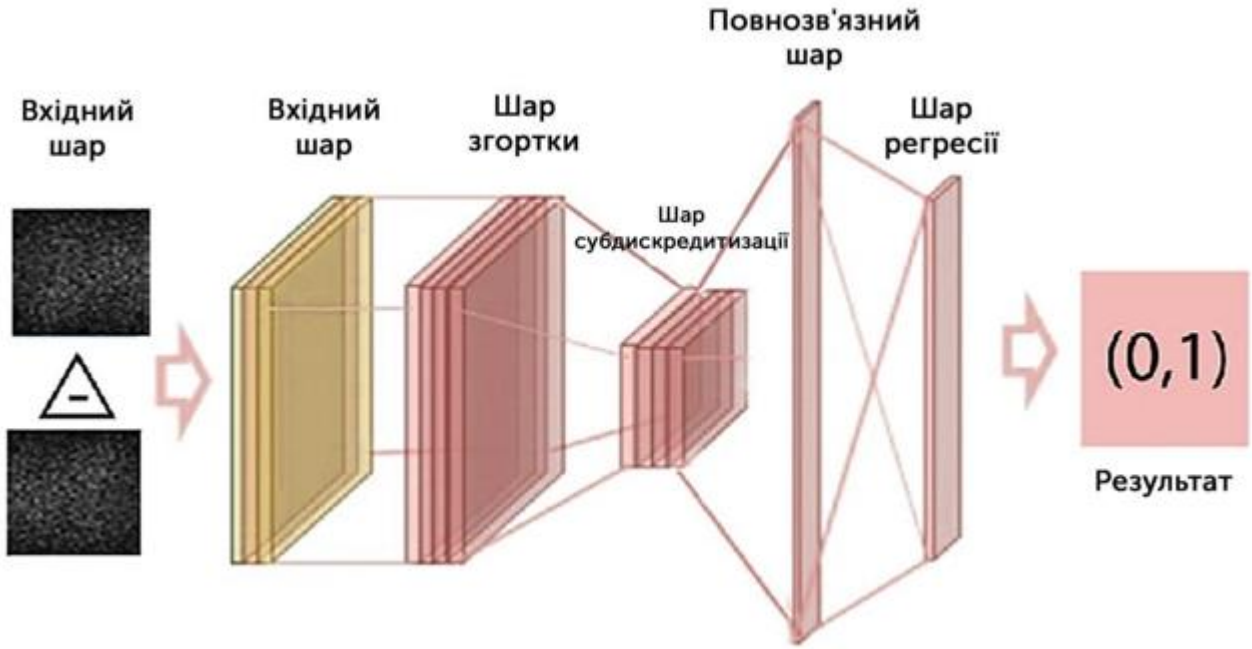

Рисунок 1.11 – Схема зображення архітектури згорткової нейронної мережі

Згорткові шари відіграють важливу роль у CNN. Він складається з особливого типу лінійної арифметичної операції. У цифрових зображеннях значення пікселів зберігаються в двомірній (2D) сітці, яка є масивом чисел, а невелика сітка параметрів, звана ядром (оптимізованим екстрактором ознак), застосовується до кожного розташування зображення, тому CNN високоефективні. Для обробки зображення. Видалені об'єкти можуть ставати ієрархічними і все більш складними в міру того, як один шар передає свої вихідні

дані наступному шару [28]. Кожна підобласть аналізує нейрони для попередньої обробки невеликої кількості інформації. Це називається обробкою згортки.

Архітектура згортання нейронних мереж формується послідовністю будівельних блоків для вибору ознак, що відрізняють відповідні класи зображень від інших класів [29]. Будівельний блок складається з однієї або декількох частин:

− об’єднання шарів (POOL), яке стискає інформацію шляхом зменшення розміру проміжного зображення (часто шляхом підвибірки);

− корекційний шар, який часто називають «ReLU» з посиланням на функцію активації (Rectified Linear Unit);

− згортковий шар (CONV), який обробляє дані рецепторного поля.

Будівельні блоки чергуються на останньому рівні мережі для виконання класифікації зображень та обчислення помилки між прогнозованими та цільовими значеннями:

− шар втрат (LOSS);

− повністю зв'язаний (FC) шар – є персептроноподібним шаром.

Характерною рисою архітектури мережі є те, як згорткові шари, модифікації та об’єднання відрізняються один від одного в межах будівельних блоків і як самі будівельні блоки відрізняються один від одного (рис. 1.12). Алгоритм роботи архітектури CNN дуже простий:

− розрізання на підобласті, які називаються плитками;

− аналіз згортковим ядром.

1.3.3 Рекурентні нейронні мережі

Рекурентна (повторювана) нейронна мережа (RNN) - це вид штучної нейронної мережі, яка використовує послідовні дані або тимчасові ряди. У RNN інформація проходить через повторювальні цикли. Коли система приймає рішення, вона враховує поточну інформацію, і ща враховує е, що вона дізналася з раніше отриманої інформації[23]. RNN і нейронні мережі прямого розповсюдження назвали так через спосіб передачі інформації (рис. 1.10). Ці

алгоритми глибокого навчання зазвичай використовуються для вирішення проблем часу або послідовності, таких як переклад мови, обробка природної мови (NLP), розпізнавання мовлення та підписи до зображень. Їх можна знайти в таких популярних програмах, як Siri, Google Translate і в «Голосовий пошук». Подібно до нейронних мереж прямого зв'язку та згорткових нейронних мереж (CNN), рекурентні нейронні мережі використовують навчальні дані для навчання. Вони відрізняються своєю пам'яттю, оскільки вони беруть інформацію з попередніх вхідних даних, щоб впливати на їхні поточні вхідні та вихідні дані. Хоча традиційні глибокі нейронні мережі припускають, що їхні входи та виходи незалежні один від одного, вихід рекурентних нейронних мереж залежить від попереднього елемента в послідовності. Майбутні події також можуть допомогти визначити результат послідовності, але односпрямовані рекурентні нейронні мережі не можуть пояснити ці події у своїх прогнозах.

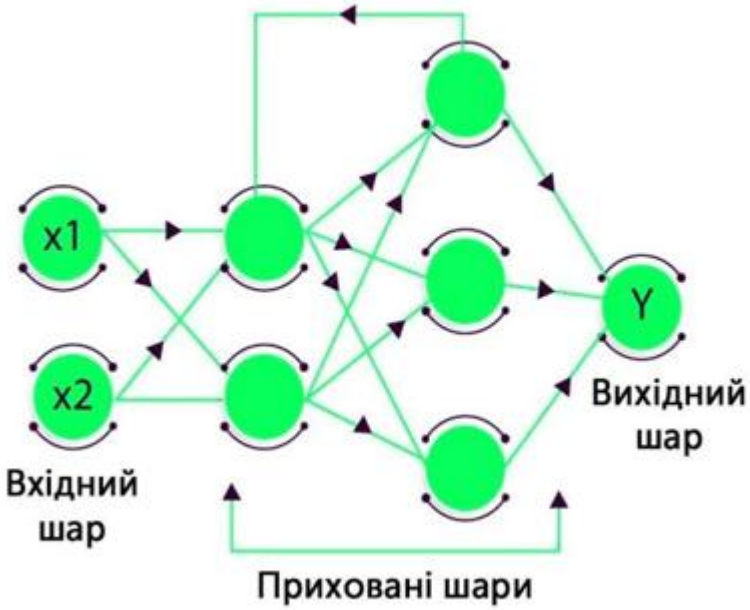

Рисунок 1.10 – Схема архітектури рекурентної нейронної мережі (RNN)

Двонаправлені рекурентні нейронні мережі (BRNN) - це варіант мережевої архітектури RNN. Тоді коли однонаправлені RNN можуть використовувати лише попередні вхідні дані для прогнозування поточного стану, двонаправлені RNN

залучають майбутні дані для підвищення їх точності. Якщо ми повернемося до прикладу «відчуття погоди» раніше в цій статті, модель зможе краще передбачити, що другим словом у цій фразі є «під», якщо вона знала, що останнє слово в послідовності - «погода».

Довга короткочасна пам’ять (LSTM) - це популярна архітектура RNN, яку представили Зепп Гохрайтер і Юрген Шмідхубер як рішення проблеми зникнення градієнта. У своїй статті вони працюють над вирішенням проблеми довгострокових залежностей. Тобто, якщо попередній стан, який впливає на поточний прогноз, не належить до недавнього минулого, модель RNN може бути не в змозі точно передбачити поточний стан. Як приклад, скажімо, ми хотіли передбачити наступні слова, виділені курсивом, «В Аліси алергія на горіхи». Вона не може їсти арахісове масло». Контекст алергії на горіхи може допомогти нам передбачити, що їжа, яку не можна їсти, містить горіхи. Однак, якби цей контекст містився кількома реченнями раніше, RNN ускладнив би або навіть унеможливив зв’язок інформації. Щоб виправити це, LSTM мають «клітини» в прихованих шарах нейронної мережі, які мають три ворота: вхідний, вихідний і забутий. Ці ворота контролюють потік інформації, який необхідний для прогнозування виходу в мережі. Наприклад, якщо займенники статі, такі як «вона», повторюються кілька разів у попередніх реченнях, ви можете виключити це зі стану клітинки.

Gated recurrent unit (GRU) - цей тип RNN схожий на LSTM, бо він також працює для вирішення проблеми короткочасної пам’яті моделей RNN. Однак, використання інформації для регулювання «стану комірки» він використовує приховані стани, і замість трьох воріт він має два - шлюз скидання та шлюз оновлення. Подібно до шлюзів у LSTM, шлюзи скидання та оновлення контролюють, скільки та яку інформацію зберігати.

Традиційні глибокі нейронні мережі працюють у схемах, у яких входи та виходи не залежать один від одного, тоді як вихідні дані рекурентних нейронних мереж залежать від попереднього елемента в послідовності. Майбутні події також

допомагають визначити результат послідовності, але односторонні рекурентні нейронні мережі не можуть пояснити ці події у своїх прогнозах.

Ще одна характеристика рекурентних мереж у тому, кожен шар мережі має загальні параметри. Мережа з прямим зв'язком має різні ваги в кожному вузлі, тоді як рекурентна нейронна мережа має той самий ваговий параметр на кожному рівні мережі. Ці ваги коригуються з використанням процесів зворотного розповсюдження та градієнтного спуску, щоб полегшити навчання із підкріпленням.

Рекурентні нейронні мережі використовують алгоритм зворотного розповсюдження часу (BPTT) визначення градієнтів [25]. Відрізняється від традиційного зворотного поширення тим, що є специфічним для послідовності даних. Принцип BPTT такий самий, як у класичного зворотного розповсюдження. Модель навчається шляхом обчислення помилки від вихідного до вхідного шару. Розрахунок даних про помилки допомагає правильно налаштувати та вибрати параметри моделі. BPTT відрізняється від традиційних підходів тим, що він підсумовує помилки на кожному тимчасовому кроці, але оскільки мережі прямого розповсюдження не мають загальних параметрів на кожному рівні, немає необхідності підсумовувати помилки.

За допомогою цього процесу RNN зазвичай вирішують дві проблеми, відомі як пакетні градієнти та зникаючі градієнти. Ці проблеми визначаються розміром градієнта, який є градієнтом функції втрат уздовж кривої помилок. Якщо градієнт занадто малий, продовжуйте оновлення, зменшуючи параметр ваги, доки він не стане незначним. 0. Алгоритм припиняє навчання, коли це відбувається. Якщо градієнт занадто великий, градієнт буде нестабільним і створюватиме нестабільний візерунок. У цьому випадку вагові коефіцієнти в моделі стають дуже великими і зрештою відображаються як NaN. Рішенням цих проблем є зменшення кількості прихованих шарів у нейронній мережі, щоб усунути частину складності моделі RNN.

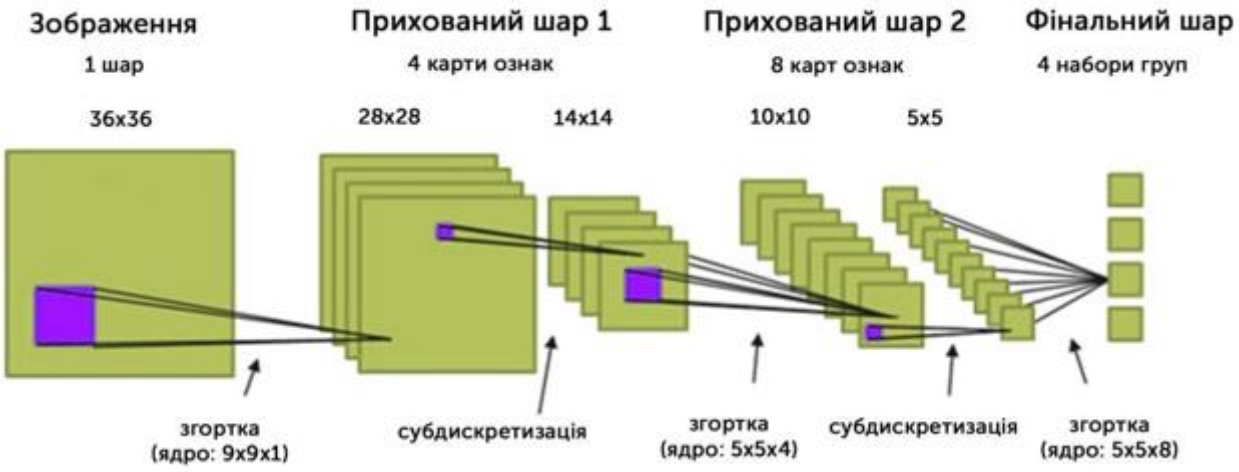

Рисунок 1.12 – Робота згорткової штучної нейронної мережі на участку зображення

Згорткове ядро - це розмір тайла, часто 3 * 3 або 5 * 5. Область аналізу (рецептивне поле) трохи більше ядра через додавання сходів, отже рецептивні поля перекриваються. Цей метод призводить до кращого рендерингу зображення та більш послідовної обробки.

Процес оптимізації ядра називається навчанням. Це робиться для мінімізації різниці між результатом та базовою міткою за допомогою зворотного розповсюдження помилки та алгоритму оптимізації, що називається градієнтним спуском.

Особливості зображення ядра згортки - це операції фільтрації з вагами, пов'язаними з кожним пікселем. Застосування фільтра до зображення називається пакунком.

Після згортки ми отримуємо карту об'єктів, яка є абстрактним представленням зображення. Його значення залежить від параметрів застосованого ядра згортки та значень пікселів вхідного зображення.

Декілька ядер згортки проходять через зображення, внаслідок чого виходить кілька вихідних карток об'єктів. Кожне ядро згортки має параметри, специфічні для інформації на зображенні (наприклад, ядро згортки типу фільтра Собеля має параметри для пошуку контурів на зображенні).

Фільтр менше, ніж вхід, і тип множення, що застосовується між вхідним блоком розміру фільтра та фільтром, є скалярним твором. Скалярне твір - це елементи фрагмента розміру вхідного фільтра, помножені на фільтр і сумовані до значення. Цю операцію часто називають "скалярним твором", оскільки вона повертає одне значення.

Використання меншого фільтра, ніж вхідні дані, є навмисним. Це пов'язано з тим, що той самий фільтр (набір ваг) можна багаторазово множити на різні вхідні точки у вхідному масиві. Зокрема, фільтри систематично застосовуються ліворуч і зверху вниз шляхом перекриття частин або зрізів розміру вхідного фільтра.

Вибір параметрів ядра згортки залежить від розв'язуваного завдання. У методах глибокого навчання ці параметри автоматично навчаються з даних з використанням алгоритмів (рис. 1.13). Зокрема, завдяки техніці зворотного розповсюдження градієнта, що дозволяє коригувати параметри залежно від значення градієнта втрат. Функція втрат рахує помилку між прогнозованим значенням та цільовим значенням.

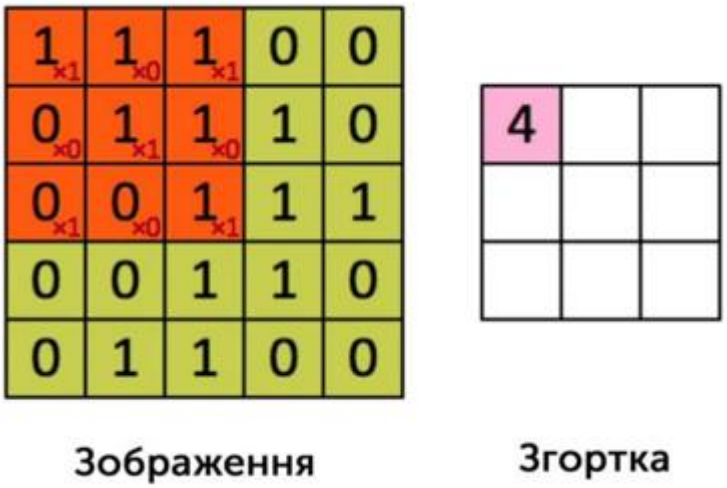

Рисунок 1.13 – Принцип роботи згортки

Основна ідея полягає в систематичному застосуванні тих самих фільтрів до ваших зображень. Якщо фільтр призначений для виявлення об'єкта певного типу

у вхідних даних, систематичне застосування цього фільтра до вхідного зображення дозволить фільтру виявити об'єкт у будь-якому місці зображення.

Результатом одного множення фільтра на вхідний масив є значення. Фільтр застосовується кілька разів до вхідного масиву, тому результатом є двовимірний масив вихідних значень, що представляє фільтрацію вхідних даних. Тому двовимірний вихідний масив цієї операції називається картою об'єктів.

Після створення карти об'єктів ви можете передати кожне значення карти об'єктів через нелінійність, таку як ReLU, так само, як і для виведення пов'язаного повно шару.

1.4 Основні моделі розпізнавання 3D поз людини

1.4.1 Оцінки 3D поз людини

Метод комп'ютерного зору, який визначає положення об'єкта або людини на зображенні або відео, називається оцінкою пози. Цей процес здійснюється шляхом перегляду комбінації орієнтацій та поз конкретного об'єкта чи людини. Грунтуючись на орієнтації точок, ви можете порівнювати різні моменти та пози людей та об'єктів, щоб генерувати ідеї.

Оцінка пози переважно виконується шляхом визначення ключових точок об'єкта/людини або шляхом визначення місцезнаходження.

- Для об'єктів: ключовими точками є кути чи краї об'єкта.
- Для зображень: зображення людей із ключовими точками, такими як руки, плечі, голова та ноги.

1.4.2 Застосування оцінки пози

Сьогодні існує багато рішень з використанням технологій комп’ютерного зору. Ефективні системи відстеження та вимірювання оцінки ставлення становлять великий інтерес для методів оцінки ставлення.

1.4.3 Причини використання оцінки пози

Виявлення людей відіграє велику роль у процесі розпізнавання. Нещодавні розробки в алгоритмах машинного навчання (ML) значно спростили використання виявлення та відстеження поз. У традиційних методах, таких як виявлення об'єктів, розпізнані об'єкти зазвичай виділяються лише їх рамками, що обмежують.

Поліпшене визначення та відстеження поз допомагає машинам вивчати мову людського тіла. Оцінка пози дозволяє відстежувати об'єкти з вищим рівнем деталізації. Ці потужні методи пропонують широкий спектр можливостей для практичного застосування.

Для відстеження рухів та дій людини оцінка пози має декілька додатків, таких як доповнена реальність, охорона здоров'я та робототехніка. Оцінку пози людини тепер можна використовувати різними способами, наприклад, підтримки соціального дистанціювання в чергах у банках, комбінуючи оцінку пози людини з проекцією відстані. Це допомагає людям дотримуватись належних правил і норм гігієни в банках та підтримувати фізичне дистанціювання у місцях масового скупчення людей.

Ще один приклад здійсненності та ефективності відстеження та оцінки поз – безпілотні автомобілі. Багато аварій за участю безпілотних автомобілів трапляються, коли транспортний засіб не розуміє поведінку пішоходів. Використовуючи методи оцінки пози, модель може краще вчитися.

1.4.4 Методи оцінки поз кількох осіб

Для оцінки пози використовуються два поширені методи:

1.4.4.1 Підхід «зверху вниз»

Алгоритм починає виявляти людей та одночасно створює рамку навколо кожної розпізнаної людини, далі він оцінює кожну частину тіла опізнанної людини.

1.4.4.2 Підхід «знизу вгору»

Попереше у білшости випадків цей підхід займає більше часу ніж «зверху вниз». Алгоритм починає виявляти частини зображення, а частини, що належать людині, зв’язуються/групуються [30].

1.4.5 Розповсюджені моделі для оцінки пози

В останні роки швидкий прогрес у розробці рішень глибокого навчання випередив деякі підходи комп’ютерного зору для багатьох завдань, таких як оцінка пози, включаючи сегментацію зображення та виявлення об’єктів.

Існує кілька моделей для оцінки постави. Вибір моделі залежить від вимог завдання. При виборі моделі також слід враховувати багато факторів. Вимоги включають час виконання, розмір моделі тощо.

Нижче наведено найпопулярніші загальнодоступні бібліотеки оцінки пози. Легко налаштовується для вашого випадку використання: регіональна оцінка пози кількох людей, HRNet, openPose

1.4.6.1 HRNet

Нейронна мережа HRNet використовується саме для оцінки пози людини. HRNet – це найсучасніший алгоритм в області оцінки пози людини. HRNet широко використовується для виявлення пози людини в телевізійних видах спорту.

HRNet використовує згорточні нейронні мережі (CNN). Це дозволяє масштабувати великі набори даних зображень, на відміну від традиційних нейронних мереж. Традиційні нейронні мережі дуже повільні, оскільки всі приховані шари взаємопов’язані, що уповільнює виконання моделі. Наприклад,

під час обробки зображення розміром 32323 пікселів використовується вага 3072, що є занадто великим і його важко обчислити. Це сповільнює роботу нейронної мережі під час передачі цих даних.

1.4.6.2 AlphaPose

Alpha Pose - це дуже точна система оцінки пози кількох людей у реальному часі. Це перша система з відкритим кодом, яка може досягти 70+ mAP (72,3 mAP) на наборі даних COCO та 80+ mAP (82,1 mAP) на наборі даних MPII. Щоб зв'язати пози, які вказують на ту саму особу в різних кадрах, ми також надаємо ефективний онлайн-трекер пози під назвою Pose Flow. Це також перший онлайн-трекер пози з відкритим кодом, який може задовольнити як 60+ mAP (66,5 mAP), так і 50+ MOTA (58,3 MOTA) у наборі даних PoseTrack Challenge. Він використовує трекер пози під назвою PoseFlow і також є відкритим кодом.

1.4.6.3 OpenPose

OpenPose - це бібліотека для визначення пози людини в режимі реального часу, яка вперше показала можливість спільного виявлення людського тіла, стопи, руки та ключових точок обличчя на окремих зображеннях. OpenPose здатний виявляти загалом 135 ключових точок. Цей метод є переможцем COCO 2016 Keypoints Challenge і популярний завдяки своїй гідній якості та надійності в умовах, де працюють кілька людей. Бібліотека OpenPose спочатку витягує функції із зображення за допомогою перших кількох шарів. Витягнуті ознаки потім вводяться в два паралельні відділи шарів згорткової мережі. Перший розділ передбачає набір із 18 карт впевненості, кожна з яких позначає певну частину скелета пози людини. Наступна гілка передбачає ще один набір із 38 полів спорідненості частин (PAF), які позначають рівень асоціації між частинами. Наступні етапи використовуються для очищення прогнозів, зроблених гілками. За допомогою довірчих карт будуються дводольні графи між парами частин. Завдяки значенням PAF слабші ланки відсікаються в дводольних графах. Тепер,

застосовуючи всі наведені кроки, скелети людської пози можна оцінити та призначити кожній людині на знімку (рис. 1.14).

Рисунок 1.14 – Алгоритм роботи OPenPose

1.4.7 Категоризація оцінки пози

Загалом оцінки пози діляться на дві групи:

1) оцінка однієї пози: виявлення окремого об’єкта чи людини на зображенні чи відео;

2) оцінка кількох поз: виявлення кількох об’єктів або людей на зображенні відео.

Також, існують різні види оцінки пози:

- 2D оцінка пози;
- 3D оцінка пози;
- оцінка пози голови;
- оцінка пози руки.

1.4.7.1 Оцінка пози людини.

Оцінку пози можна зробити як у 2D, так і в 3D. Оцінка двовимірної пози передбачає ключові точки зображення через значення пікселів. У той час як оцінка 3D пози стосується прогнозування тривимірного просторового розташування ключових точок як результат (рис 1.15).

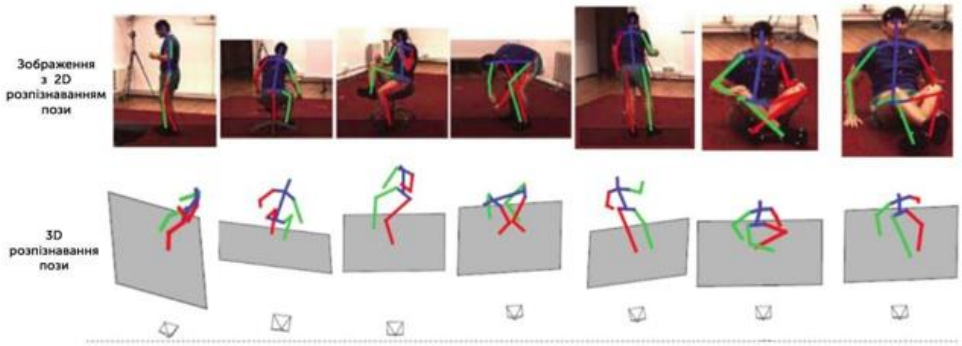

Рисунок 1.15 – Схема роботи розпізнавання пози людини у 2D та 3D

1.4.7.2 Оцінка 2D пози

Цей метод оцінює ключові моменти зображення з точки зору рівня пікселів. 2D-оцінка оцінює ключові місця у просторі вхідних 2D-даних. Розташування ключових точок позначено X і Y.

1.4.7.3 Оцінка 3D пози

Кінцевий результат містить тривимірне просторове розташування всіх об’єктів або людей. Об’єкти перетворюються з 2D на 3D зображення за допомогою додаткової осі z для прогнозування результату.

1.4.7.4 Оцінка розпізнавання пози голови

Знаходження положення голови вже давно здобуло популярності у нашому часі. Ми можемо зустріти ци технології наприклад у SnapChat або Instagram, коли ми використовуємо маски, або ж наприклад FaceID.

Одним з найпростіших методів оцінки пози голови є бібліотеки OpenCV та MediaPipe. MediaPipe виявляє 468 3D орієнтирів на геометрії обличчя. Ці координати нормалізуються до значень у діапазоні від 0,0 до 1,0 (рис. 1.16).

Рисунок 1.16 – Розпізнавання положення рук та голови за допомогою OpenCV та MediaPipe

1.4.7.5 Оцінка розпізнавання пози руки

Оцінка положення руки спрямована на прогнозування положення суглобів на руці за вхідним зображенням, і вона стала популярною через появу технологій віртуальної реальності. Після виявлення долоні на всьому зображенні наша подальша модель орієнтира руки виконує точну локалізацію ключової точки 21 3D-координати руки й пальця всередині виявлених областей руки за допомогою регресії, яка є прямим прогнозуванням координат. Модель вивчає послідовне внутрішнє представлення пози руки та є надійною навіть до частково видимих рук і самоокклюзій.

Щоб отримати наземні правдиві дані, ми вручну анотували ~30 тис. зображень реального світу з 21 3D-координатою, як показано нижче (ми беремо Z-значення з карти глибини зображення, якщо воно існує для відповідної координати). Щоб краще охопити можливі пози рук і забезпечити додатковий нагляд за природою геометрії руки, ми також візуалізуємо високоякісну синтетичну модель руки на різних фонах і наносимо її на відповідні 3D-координати (рис. 1.17).

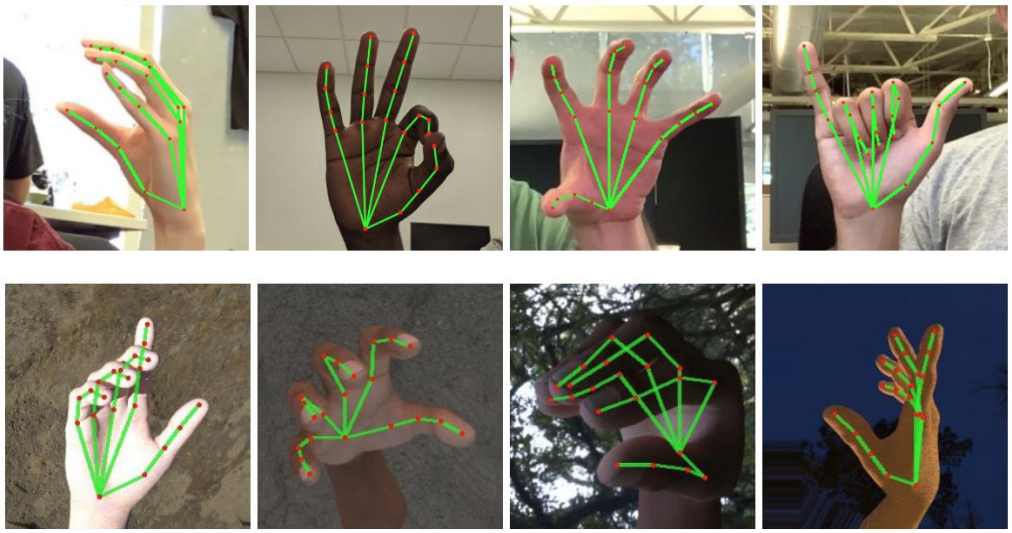

Рисунок 1.17 – Розпізнавання пози руки людини за допомогою MediaPipe

1.4.8 Дані для оцінки 3D пози людини

Існує багато різни наборів даних для оцінки 3D пози людини але я хочу відмітити лишне три найпопулярніші з них: MPII, COCO, HumanEva.

1.4.8.1 MPII

Набір даних MPII Human Pose - це сучасний еталон для оцінки артикульованої пози людини. Набір даних включає близько 25 тисяч зображень, що містять понад 40 тисяч людей із анотованими суглобами тіла. Зображення були систематично зібрані з використанням встановленої таксономії повсякденної діяльності людини. Загалом набір даних охоплює 410 дій людини, і кожне зображення має мітку активності.

Дотримуючись найкращих практик для тестів оцінки продуктивності в літературі, ми не надаємо анотації тестів, щоб запобігти переобладнанню та налаштуванню тестового набору. Ми працюємо над автоматичним сервером оцінки та інструментами аналізу ефективності на основі багатих анотацій набору тестів (рис. 1.18).

Рисунок 1.18 – Розпізнавання пози людини за допомогою набору даних MPII

MPII була першою базою даних, представленою в 2014 році, а також першим набором даних, який охоплює різні пози. Дані включали зображення людської діяльності, зібрані з відео YouTube, камер спостереження тощо. 410 зображень людської діяльності анотовано та відображено в одній добірці. Кожне зображення витягується з відео YouTube, і надаються кадри до та після без анотацій. Крім того, для тестового набору були представлені більш детальні анотації, такі як оклюзія частин тіла та 3D орієнтація тулуба та голови.

1.4.8.2 COCO

Набір даних COCO - це великомасштабний набір даних виявлення об’єктів, сегментації та субтитрів, опублікований Microsoft. Інженери машинного навчання та комп’ютерного зору широко використовують набір даних COCO для різних проектів комп’ютерного бачення. Розуміння візуальних сцен є основною метою комп’ютерного зору; це передбачає розпізнавання наявних об’єктів, локалізацію об’єктів у 2D та 3D, визначення атрибутів об’єкта та характеристику зв’язку між

об’єктами. Таким чином, алгоритми для виявлення об’єктів і класифікації об’єктів можна навчити за допомогою набору даних. (рис. 1.19).

Рисунок 1.19 – Розпізнавання 3D пози за допомогою набору даних COCO

1.4.8.3 HumanEva

Набір даних HumanEva містить 7 відкаліброваних відеопослідовностей (4 градації сірого та 3 кольорові), які синхронізовані з 3D-позами тіла, отриманими з системи захоплення руху. База даних містить 4 суб’єкти, які виконують 6 типових дій (наприклад, ходьба, біг, жестикуляція тощо). Учасникам надаються показники помилок для обчислення помилок у 2D і 3D позі. Набір даних містить набори для навчання, перевірки та тестування (з прихованою базовою правдою) (рис 1.20).

Рисунок 1.12 – Приклад роботи HumanEva

2 ЗОВНІШНЄ ТА ЛОГІЧНЕ ПРОЕКТУВАННЯ

2.1 Вибір мови програмування

При виборі мови програмування необхідно проаналізувати завдання, які стоять перед розробниками, і проблеми, які можуть виникнути в процесі розробки. Відповідно до вимог, програма має бути веб-додатком. Це пов’язано з тим, що веб-інтерфейс сьогодні є одним із найпоширеніших і найпростіших. Ви повинні вибрати мову програмування, яка повністю підтримує цю систему. Після аналізу наведених особливостей як мову програмування було обрано Python. Такі мови, як Python, можуть добре спілкуватися з нейронними мережами. IDLE було обрано для розробки Python з PyCharm.

2.2 Мова програмування «Python»

Для даного проекту рекомендую використовувати мову програмування Python для розробки рішень щодо розпізнавання об'єктів. Python – це мова програмування, яка використовується для створення програмного забезпечення та веб-сайтів, автоматизації та аналізу даних. Python - це мова загального призначення, яку можна використовувати для написання різних програм. Python не призначений для будь-якої конкретної задачі. Ця універсальність і простота використання для програмістів-початківців зробили його сьогодні однією з найпопулярніших мов програмування. Python використовується багатьма фінансистами та вченими для вирішення різних повсякденних завдань [31]–[33].

Деякі зі сфер використання Python:

- розробка веб-сайтів, веб-додатків, тощо
- написання тестів для програмного забезпечення
- аналіз даних
- автоматизація або написання сценаріїв
- машинне навчання

Використовуючи бібліотеку Python, ви можете створювати різні візуалізації даних, такі як гістограми, кругові діаграми, тривимірні діаграми та багато іншого. Python також дозволяє програмістам швидше та ефективніше писати програми для аналізу даних та машинного навчання, використовуючи такі інструменти, як TensorFlow.

Python часто використовується для розробки внутрішньої функціональності веб-сайту або програми. Python у веб-розробці може виконувати сервісні функції, такі як відправлення даних на сервер, відправлення даних із сервера користувачеві, обробка даних, обмін даними з базою даних, маршрутизація URL-адрес та забезпечення безпеки. Python пропонує кілька середовищ для веб-розробки, включаючи Django та Flask.

Веб-розробка з дисциплінами Python включає Backend Engineers, Full-Stack Engineers, Python Developers, Software Engineers і DevOps Engineers.

2.3 Фреймворк для «Python» - Flask

Flask - це мікрофреймоврк для Python, призначена для полегшення веб-розробки. Ця платформа допомагає розробникам досягти бажаних результатів, спрощює роботу тому що має встроєнні функції та масштабованість, ефективність, надаючи багаторазовий код або розширення для звичайних операцій. Flask має вбудовану підтримку масштабування що дозволяє його використовувати у складних проектах також на сьогоднішній день він одним із найпопулярніших фреймворків Python.

Flask пропонує розробникам безліч варіантів розробки веб-додатків. Він надає інструменти, бібліотеки та механізми, які дозволяють створювати веб-програми без накладання залежностей або жорстких обмежень на зовнішній вигляд вашого проекту.

Веб-додаток, розроблений з використанням мікрофреймоврку Flask, може мати практичну спрямованість, наприклад, блог, комерційний веб-сайт або веб-

сторінку. Цей фреймворк допомагає розробникам використовувати кілька розширень, наданих спільнотою.

Це дозволяє вам додати більше функціональності до вашого веб-додаток. Як згадувалося раніше, Flask належить до категорії мікрофреймворків. Мікрофреймворк зазвичай є фреймворк, який залежить від зовнішніх бібліотек.

Перевага використання Flask як платформа для розробки веб-додатків полягає в тому, що вона менше залежить від оновлень та пошуку помилок безпеки.

Недоліком використання Flask і те, що список залежностей зростає у міру додавання розширень. Flask заснований на службовій бібліотеці WSGI Werkzeug та механізмі шаблонів Jinja2.

Ця структура веб-додатків дозволяє компілювати модулі та бібліотеки, які також допомагають створювати веб-програми без написання низькорівневого коду, такого як потоки та протоколи. Щоб використовувати Flask, вам потрібно створити веб-додаток та підключити його до вашого HTML.

Щоразу, коли користувач надсилає інформацію в Інтернеті або переходить до рядка пошуку, HTML виступає посередником для використання Flask. Платформа Flask шукає HTML-файли (шаблони) у папці Templates. Код Python, який впроваджує змінні та код, запускається перед відправкою шаблону.

Його також можна використовувати для створення соціальних мереж, платформ для блогів та сайтів із звичайним контентом. Однак, якщо ваше завдання - створити великомасштабну веб-додаток, Flask - це мікрофреймворк, тому вибір іншого веб-фреймворку має сенс.

2.4 Бібліотека для «Python» - BlazePose

BlazePose - бібліотека для оцінки пози людини, яка адаптована для висновків у режимі реального часу на мобільних пристроях або у веб-додатках. Під час висновків мережа створює 33 ключові точки тіла для однієї людини та працює зі швидкістю понад 30 кадрів на секунду. Це робить його особливо

придатним для використання в режимі реального часу, як-от відстеження фізичної активності та розпізнавання мови жестів. Наші основні внески включають нове рішення для відстеження пози тіла та легку нейронну мережу для оцінки пози тіла, яка використовує як теплові карти, так і регресію до координат ключових точок. [34], [35].

BlazePose використовує полегшену архітектуру CNN для виявлення 33 ключових точок і оптимізовано для висновків у реальному часі. Крім того, це частина рішення Google MediaPipe Pose ML, яке забезпечує зручний і добре задокументований спосіб застосування моделі до проекту.

Завдяки збільшеній кількості ключових точок BlazePose надає повнішу інформацію про позу людини, тому він був сильним кандидатом для проекту (рис. 2.1):

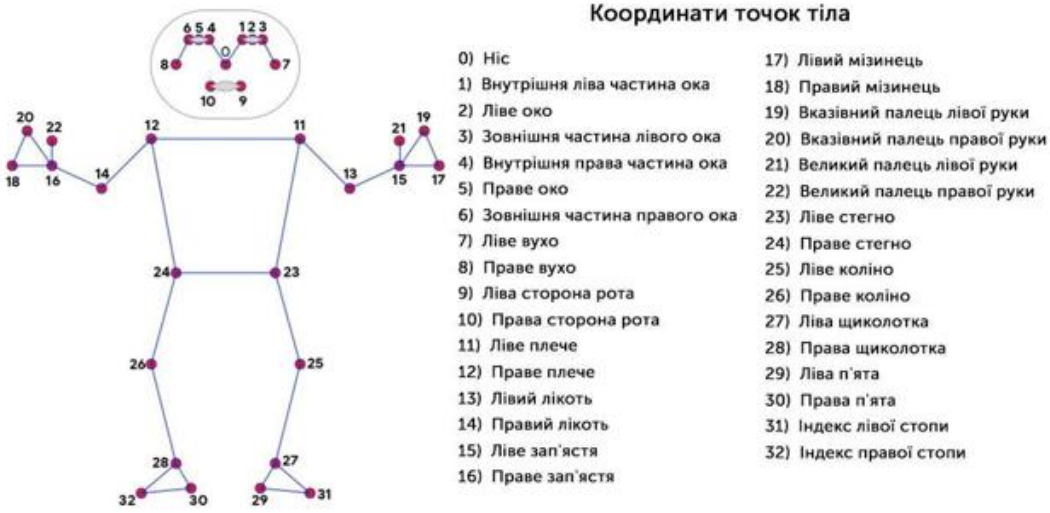

Рисунок 2.1 – Список координат пози людини моделі BlazePose

Детектор є архітектурою одноразового детектора (SSD). Враховуючи зображення розміру вхідного масиву, детектор виводить рамки, що обмежують, і оцінки точності. 12 елементів обмежуючої рамки мають вигляд (x, y, w, h, kp1x, kp1y, kp4x, kp4y). де від kp1x до kp4y – додаткові ключові точки. Кожен з цих елементів має власне ім'я.

Є два способи використання детектора: у прямокутному режимі рамка, що обмежує, визначається своїм положенням (x, y) і розміром (w, h). У режимі вирівнювання масштаб і кут визначаються (kp1x, kp1y) та (kp2x, kp2y). Це дозволяє передбачити рамку, що обмежує, що містить перекіс об'єкта.

Оцінювач використовує теплові картки для навчання, але обчислює ключові точки без теплових карток, щоб прискорити більш оптимізовані прогнози. Архітектура мережі стеження показано малюнку 2.2.

Рисунок 2.2 – Архітектура мережі відстеження BlazePose

Перший вихід оцінювача – 1195 орієнтирів. Орієнтири складаються з 165 елементів значень x, y, z, видимості та присутності для кожної з 33 ключових точок.

Видимість і присутність зберігаються в min_float і max_float і перетворюються на значення ймовірності за допомогою сигмоїдної функції. Значення видимості повертає ймовірність знаходження характерної точки, яка знаходиться всередині кадру і не закрита іншими об’єктами. значення повертає ймовірність того, що ключова точка існує у кадрі.

BlazePose використовує двоступеневий конвеєр відстеження детектора машинного навчання, який довів ефективність у рішеннях MediaPipe Hands і MediaPipe Face Mesh. Використовуючи детектор, конвеєр спочатку визначає позу людини/об’єкта (ROI) у кадрі. На наступному етапі трекер передбачає орієнтири пози та маски сегментації для цікавих регіонів. При використанні відеодетектори викликаються лише за потреби. Тобто на першому кадрі, коли трекер більше не може визначити існування пози тіла попереднього кадру.

2.5 HTML та CSS

2.5.1 HTML

HTML розшифровується як "мова розмітки гіпертексту". Тобто, це мова розмітки або інший спосіб зберігання інформації. Він використовує HTML для маркування тексту та повідомляє веб-браузерам, як розуміти виділений текст. Робиться це лише за допомогою якого інформація зберігається в блоках, кластерах, секторах та доріжках, як і на жорстких дисках. Певна структура повідомляє комп'ютер, що читати, а що ні.

Плюси HTML:

a) широко вживана мова з великою кількістю ресурсів та величезною спільнотою;

b) запускається в кожному веб-браузері;

c) з відкритим кодом і абсолютно безкоштовна;

d) чиста і послідовна розмітка;

e) легко інтегрується з серверними мовами, такими як php, node.js, python (з фреймворком) та інші.

Мінуси HTML:

a) в основному використовується для статичних веб-сторінок. для динамічної функціональності вам може знадобитися використовувати javascript або серверну мову, таку як php;

b) це не дозволяє користувачеві реалізовувати логіку. як результат, усі веб-сторінки потрібно створювати окремо, навіть якщо вони використовують однакові елементи, наприклад, верхній та нижній колонтитули;

c) деякі браузери повільно застосовують нові функції.

2.5.2 CSS

CSS (скорочення від Cascading Style Sheets) - мова розмітки даних, яка використовується для опису зовнішнього вигляду документів (як і де відображати елементи веб-сторінки), це спеціальна мова (мова стилів). CSS в основному використовується для документів, розмічених за допомогою HTML, XHTML та XML.

Що дає використання CSS:

a) відображати один і той же документ в різних стилях;

b) декілька дизайнів сторінки для різних пристроїв. наприклад, на екрані дизайн буде розрахований на велику ширину, під час друку меню не поводитиметься, а на смартфоні меню буде внизу, під вмістом;

c) зменшення часу завантаження сторінок сайту за рахунок перенесення правил відображення в окремий css-файл. в цьому випадку браузер завантажує тільки структуру документа і дані, що зберігаються на сторінці, а стильові правила цих даних завантажуються браузером тільки один раз і кешуються;

d) простота подальшої зміни дизайну. не потрібно правити кожну сторінку, а лише змінити css-файл;

e) додаткові можливості оформлення. наприклад, за допомогою css-розмітки можна зробити так, щоб меню було завжди видно при скролінгу сторінки, або прибрати підкреслення у посилань;

f) дозволяє створювати складну і пропрацьовану техніку дизайну.

2.6 Тренування нейронних мереж

Навчання нейронної мережі полягає у пошуку оптимальних значень вагових коефіцієнтів нейронних зв'язків завдяки петлям зворотного зв'язку. Це також називається градієнтним зворотним розповсюдженням. Основна мета поділу даних на набори - запобігти перенавченню моделі (рис. 2.3).

Рисунок 2.3 – Процес використання навчальної вибірки для навчання нейронної мережі

Щоб навчити та протестувати модель нейронної мережі, нам потрібно розділити дані на три набори: навчання, навчання та перевірка.

Навчальний набір повинен містити в собі досить різноманітний набір даних, щоб модель могла вчитися на багатьох можливих сценаріях та прогнозувати нові вибірки даних у майбутньому. Перевірочний набір - це окремий набір даних із

навчального набору. Його завдання – перевірити валідність моделі під час навчання. Процес перевірки надає дані, які допомагають точно настроїти коефіцієнти та конфігурацію моделі. Алгоритми навчання дуже примітивні. Модель навчається на тренувальному наборі і після кожної епохи навчання модель оцінюється на тестовому наборі. На рис. 2.4 показані три варіанти коефіцієнтів поділу навчальних даних у навчальній вибірці.

Рисунок 2.4 – Варіанту пропорцій поділу навчальних даних у навчальному наборі

2.7 Аналіз та вибір процесора для навчання нейронних мереж

Завдання навчання нейронної мережі найкраще виконувати на графіку, а чи не на процесорі. Це правило продиктовано згаданими вище відмінностями у робочій архітектурі (рис. 2.5).

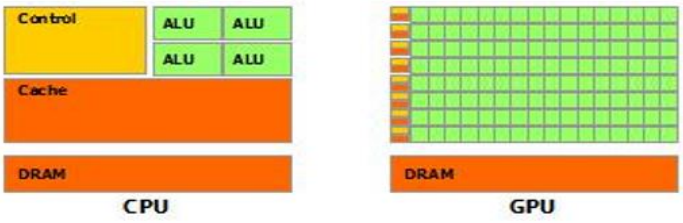

Рисунок 2.5 – Порівняння архітектури CPU та GPU

GPU мають більшу кількість арифметичних та логічних пристроїв для обробки даних у порівнянні з CPU. Графічні процесори є багатопотоковими, а CPU - це послідовні пристрої, які виконують ряд завдань по одному за раз.

Графічні процесори використовуються для машинного навчання, оскільки нейронні мережі є паралельними алгоритмами. Ще однією перевагою графічних процесорів є те, що нейронні мережі оптимізовані для матричних операцій – основного виду операцій, необхідні отримання результатів.

Щоб виміряти продуктивність графічного процесора порівняно з ЦП, ми рекомендуємо використовувати параметр операцій із плаваючою комою на секунду. (рис. 2.6).

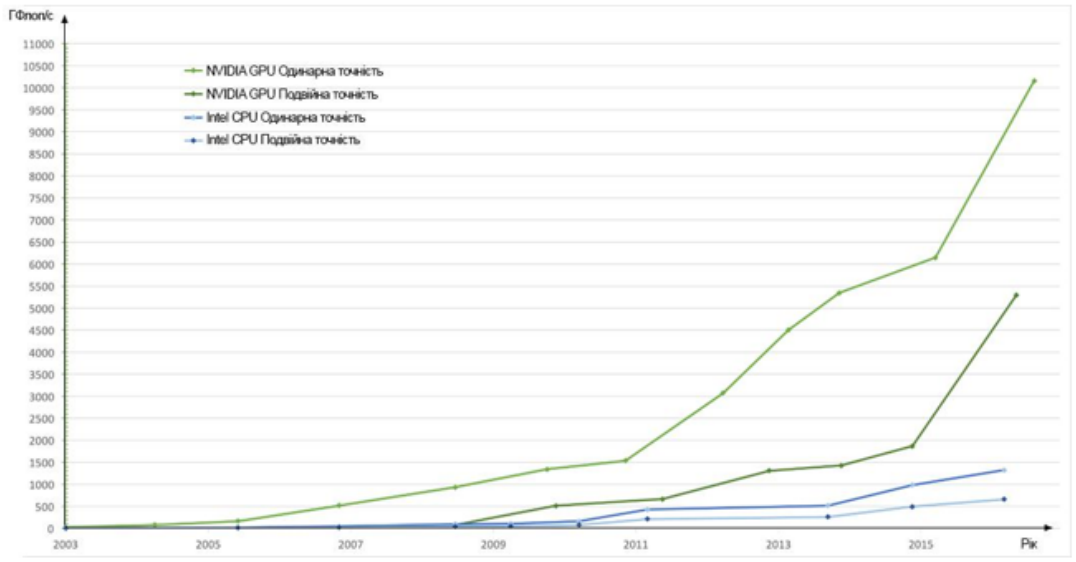

Рисунок 2.6 – Швидкість процесора та графічного процесора з плаваючою комою на секунду

Навчання багатошарової нейронної мережі на процесорі може зайняти кілька тижнів, але на графічному процесорі ту саму операцію можна виконати протягом дня. У той самий час перерахунок економить ресурси графічного процесора і робить обчислення ефективнішими.

3 РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Розробка веб-застосунку

Розробіть свій проект у редакторі вихідного коду PyCharm. Використовуйте мову програмування Python для написання програмного коду. Для завдань розробки проекту використовувався Python, оскільки він пропонує можливість писати ефективний та оптимізований код.

Веб-сайт або його розмітка створюється за допомогою мови гіпертекстової розмітки HTML із додаванням стилів у каскадних таблицях стилів CSS.

Також використовуємо фреймворк Flask, легкий інтерфейс шлюзу веб-сервера, щоб об’єднати веб-частини з кодом, написаним на мові програмування Python. Фреймворки надають інструменти, бібліотеки та механізми, які дозволяють створювати веб-програми, але без необхідності встановлювати залежності чи жорсткі обмеження на те, як має виглядати ваш проект, що ускладнює розробку цього проекту.

Для завдання розпізнавання положення тіла людини в реальному часі ми використовуємо модель BlazePose, яка може обчислити координати ключових точок в 3D-просторі.

Згідно з попередніми тестами, коли значення параметрів «min_detection_confidence» і «min_tracking_confidence» дорівнюють «0,45», модель може дуже точно й оптимально розпізнавати присутність і позу людського тіла при маніпулюванні зображеннями. Ok з веб-камери. Тому ми використовуємо «min_detection_confidence=0,45», «min_tracking_confidence=0,45» як параметри для моделі BlazePose.

Для успішного розпізнавання 3D пози людини вам потрібно вибрати правильну модель для використання у веб-додатку. Розробляючи веб-програми, ви повинні мати можливість запускати свою програму на великій кількості пристроїв. Але в той же час модель повинна бути достатньо точною, щоб точно сприймати позу людини в будь-який момент. Таким чином, попереднє тестування

показало, що модель BlazePose GHUM Full є найкращим варіантом для використання у веб-додатках, які ми розробляємо. Він досить легкий, щоб безперебійно працювати на більшості сучасних пристроїв, забезпечуючи при цьому високий рівень точності, що відповідає потребам реалізації цієї веб-програми.

Для розробки та тестування веб-застосунку використано ноутбук Macbook Pro 14 з характеристиками, наведеними у табл. 3.1.

Таблиця 3.1 – Технічні характеристики Macbook Pro 14

Центральний процесор	M1
ОЗП	16Gb
ОС	MacOS

3.2 Встановлення фреймворку

3.2.1 Встановлення фреймворку «Flask»

Щоб використовувати фреймворк Flask, нам потрібно встановити його. Для цього використовуйте вбудований у Python менеджер пакетів pip. За допомогою команди «pip install Flask» pip автоматично встановить фреймворк Flask для вашого проекту (рис. 3.1).

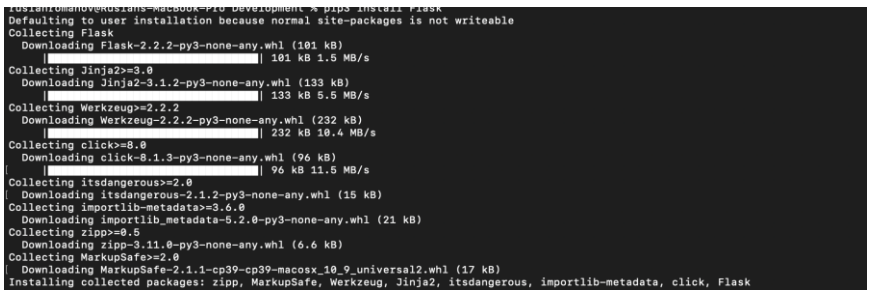

Рисунок 3.1 – Результат запуску команди

3.2.2 Початок роботи з Flask

Для початку роботи з фреймоврком Flask створюємо файл, у який далі будемо імпортувати функції фреймворка Flask – «main.py» (рис. 3.2).

```
1  import cv2
2  import mediapipe as media_pipe
3  import numpy as num_py
4  import counting
5  import staging
6
```

Рисунок 3.2 – Імпортування бібліотек необхідних для роботи

Далі ми імпортуємо 3 необхідні функції для роботи з Flask у данному проекті. Функція Training з файла training.py буде використана далі для оновлення зображення з веб-камери (рис. 3.3).

```
8  from flask import Flask, render_template, Response
9  from training import Training
```

Рисунок 3.3 – Імпортування функцій Flask

Розпочинаємо з шаблоного старту Flask. Невеликий опис: «@app.route('/')», це є коренем сайту, тобто перша сторінка котру ми бачимо при піднятті веб-серверу. Вона поверта відренерений темплейт файлу «index.html» (рис. 3.4).

```
app = Flask(__name__)

@app.route('/')
def index():
return render_template('index.html')
```

Рисунок 3.4 – Робота з API-адреса Flask

Потім скористаємося конструкцією while у «app.controller», щоб викликати функцію get.frame із файлу training.py для постійного рефрешування зображення з веб-камери. Використовуємо методи Flask для маніпулювання зображенням (рис. 3.5).

Рисунок 3.5 – Використання while

Створюємо новий API-шлях «@app.route('/training')» у якому буде тільки функции для обробки зображення з веб-камери. Також цей шлях допоможе нам у подальшій відладці проекту (рис. 3.6).

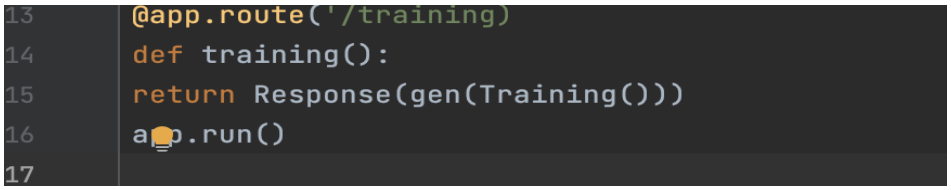

Рисунок 3.6 – Новий API-шлях «/training»

Також якщо треба вімкнути режим налагодження треба, передати функції run параметр «debug=TRUE» (рис. 3.7).

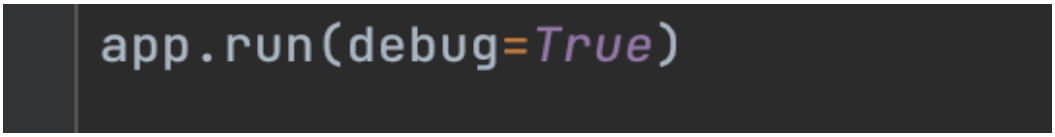

Рисунок 3.7 – Режим налагодження

3.3 Створення HTML - розмітки

Також для коректного відображення нам треба створити HTML файл - index.html з розміткою та потрібними мета-тегами. (рис. 3.8).

Рисунок 3.8 – Метадані у файлі index.html

Також додаємо панель навігації вгорі сторінки. Це залишається у верхній частині сторінки, коли ви прокручуєте сторінку вниз. Для цього оголосіть відповідний клас. Наступні кроки додамо елемент <img>, який містить посилання на логотип програми, який відображатиметься на панелі навігації шириною та висотою 24 пікселі. Також введіть назву «Training». Це буде назва програми, яка відображається на нав барі. (рис. 3.9)

Рисунок 3.9 – HTML розмітка для панелі навігації

Далі створюємо тег-заголовок <h1> та в ньому вказуємо назву проекту. Також створюємо блок та центруємо його у цей блок розміщуємо елемент <img>з посиланням «url_for('training')» воно буде автоматично розпізнано Flask (рис. 3.10).

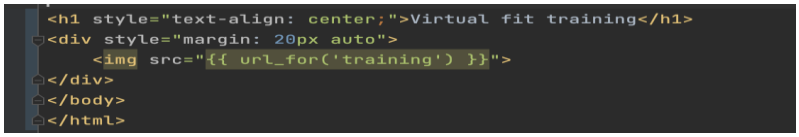

Рисунок 3.10 – Створення заголовка

3.4 Написання скрипту

3.4.1 Імпортування потрібних для проекту бібліотек

Для даного проекту нам потрібні 3 бібліотеки: mediapipe - оптимізація використання ресурсів для побудови 3D моделі, cv2 - надає доступ до веб-камери пристрою та також розпізнає вхідне зображення, numpy - бібліотека с формулами, наприклад розрахунку кута та інші (рис. 3.11):

```
1  import cv2
2  import mediapipe as media_pipe
3  import numpy as num_py
4  import counting
5  import staging
6
```

Рисунок 3.11 – Імпорт потрібних бібліотек у файл training.py

Ініціалізуємо функції з файлів counting та staging (рис.3.12):

```
counting.init()
staging.init()
```

Рисунок 3.12 – Ініціалізація функцій counting та staging

Беремо 2 функції з бібліотеки mediapipe для відображення 3D частин тіла (рис. 3.13):

```
mp_draw = media_pipe.solutions.drawing_utils
mp_pose = media_pipe.solutions.pose
```

Рисунок 3.13 – Додавання функцій с бібліотеки mediapipe для відображення частин тіла

3.4.2 Розробка основного класу Training

Оголошуємо клас Training з шаблонними функціями класу для Flask: init(), del() та get_frame()(рис. 3.14).

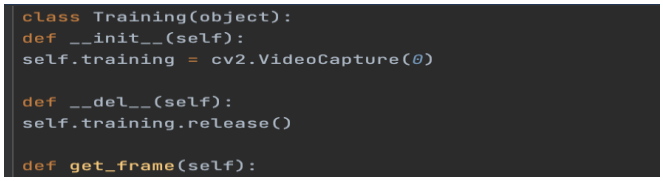

Рисунок 3.14 – Будова класу Training

Використовуємо mediapipe з налаштуваннями min_detection_confidence=0.45 та min_tracking_confidence=0.45. Параметри відповідають за відстеження положення тіла (рис. 3.15).

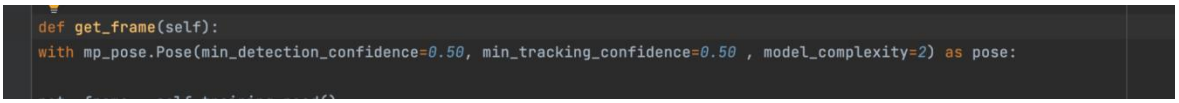

Рисунок 3.15 – Відстеження положення тіла

3.4.3 Алгоритм розпізнавання

Метод try використовується, щоб уникнути переривання роботи алгоритму, оскільки розпізнавання може відбуватися не в кожному кадрі, і записує масив розпізнаних точок тіла в змінну орієнтира. Якщо метод try не зміг обчислити кут і змінити значення змінної обчислення через відсутність даних - алгоритм пропускає цей кадр. (рис. 3.16)

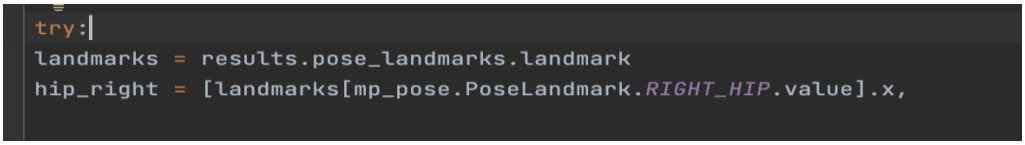

Рисунок 3.16 – Обробка вхідного зображення за допомогою try

Далі виберемо необхідні координати відповідної частини тіла для розпізнавання необхідних контрольних точок тіла. (рис. 3.17).

```
landmarks[mp_pose.PoseLandmark.RIGHT_HIP.value].y]

shoulder_right = [landmarks[mp_pose.PoseLandmark.RIGHT_SHOULDER.value].x,

landmarks[mp_pose.PoseLandmark.RIGHT_SHOULDER.value].y]

elbow_right = [landmarks[mp_pose.PoseLandmark.RIGHT_ELBOW.value].x,

landmarks[mp_pose.PoseLandmark.RIGHT_ELBOW.value].y]

hip_left = [landmarks[mp_pose.PoseLandmark.LEFT_HIP.value].x,

landmarks[mp_pose.PoseLandmark.LEFT_HIP.value].y]

shoulder_left = [landmarks[mp_pose.PoseLandmark.LEFT_SHOULDER.value].x,

landmarks[mp_pose.PoseLandmark.LEFT_SHOULDER.value].y]

elbow_left = [landmarks[mp_pose.PoseLandmark.LEFT_ELBOW.value].x,

landmarks[mp_pose.PoseLandmark.LEFT_ELBOW.value].y]
```

Рисунок 3.17 – Розпізнавання необхідних контрольних точок тіла

Далі створюємо функцію angle_between_three_points для підрахунку кута між трьома точками та змінні start, middle та end, для запису координат точок частин тіла (рис. 3.18).

```
import numpy as num_py
def angle_between_three_points(start, mid, end): = num_py.array(start) num_py.array(mid)num_py.array(end)
```

Рисунок 3.18 – Оголошення функції angle_between_three_points і змінні

Завдяки num_py.abs у змінну result ми отримаємо градуси (рис. 3.19)

```
result = num_py.abs(radians*180.0/np.pi)
```

Рисунок 3.19 – Отримання градусів для кута

Отримавши масив даних про положення потрібної точки на тілі, використовуйте функцію calculateAngle для обчислення значень кутів кожного разу, коли викликаються навчальні класи лівої та правої руки. Щоб використовувати функцію angle_between_three_points, у верхній частині основного файлу training.py було додано рядок, що містить імпорт цієї функції (рис. 3.21).

```
17    left = calculateAngle(hip_left, step_left, culbit_left)
18    right = calculateAngle(hip_right, step_right, culbit_right)
19
```

Рисунок 3.20 – Підрахування кутів

Для налагодження використовуйте cv2.putText. Аргументи функції включають змінну зображення, що містить зображення, отримане з веб-камери, значення кутів, знайдені завдяки функції angle_between_three_points і перетворені в рядковий формат. Встановлює відображення кута для обох рук (рис. 3.21).

```
cv2.putText(image, str(left),
tuple(num_py.multiply(
step_left, [640, 480]).astype(int)),
cv2.FONT_HERSHEY_SCRIPT_SIMPLEX, 0.7, (
255, 255, 255), 2, cv2.LINE_AA
)

cv2.putText(image, str(right),
tuple(num_py.multiply(
step_right, [640, 480]).astype(int)),
cv2.FONT_HERSHEY_SCRIPT_SIMPLEX, 0.7, (
255, 255, 255), 2, cv2.LINE_AA
)
```

Рисунок 3.21 – Візуалізація значення кута

Створюємо логіку, яка підраховує кількість виконань. Для тестування було обрано махи обома руками тулуба в різні боки (можливі варіації з гантелями). За замовчуванням алгоритм передбачає, що рука опущена, тому кут між рукою та тілом менше 30 градусів. Якщо умова виконана, для проміжної змінної файлу staging.py буде встановлено значення «ВНИЗ». Якщо обидві руки піднято (тобто обидві руки зафіксовано у піднятому положенні, а кут між руками та тілом більше 90 градусів), алгоритм розглядає це як повторення однієї вправи. У цьому випадку проміжній змінній у файлі staging.py надається значення «UP», а змінна лічильника, що дорівнює 0, у файлі counting.py збільшується на 1. Оновлене значення відображається на екран (рис. 3.22).

```
if left < 30 and right < 30:
staging.staging = "DOWN"

if left > 90 and right > 90 and staging.staging == 'DOWN':
staging.staging = "UP"

count.count += 1

print(count.count)
```

Рисунок 3.22 – Алгоритм підрахунку виконаних разів

3.4.4 Реалізація відображення статистики вправ

Ми будемо використовувати функції прямокутника та putText із вбудованої бібліотеки OpenCV, щоб вказати тривалість вправи та поточне положення руки. Функції Rectangle передаємо змінну image, що містить зображення з веб-камери, а також вказуємо початкову та кінцеву точки прямокутника, який відображатиметься поверх зображення з веб-камери. Встановлює білий колір. У колірній схемі RGB значення пікселів дорівнюють (255 255 255). Також вказуємо значення -1 щоб прямокутник був залитий цим кольором (рис. 3.23).

```
cv2.rectangle(image, (0, 0), (425, 95), (255, 255, 255), -1)
```

Рисунок 3.23 – Відображення результатів

Для відрисовки кількісткості та стану виконань використали функцію putText. У цю функцію передається змінна зображення, і результатом є накладений зображення графік з необхідними значеннями. (рис. 3.24).

```
cv2.putText(image, 'Counting', (24, 10), cv2.FONT_HERSHEY_COMPLEX, 1, (0, 0, 0), 1, cv2.LINE_AA)
```

Рисунок 3.24 – Додавання строкового значення для змінної counting

Далі нам потрібно додати поле для відображення статусу положення руки. Для цього, як і в попередній дії, використовуємо функцію putText.

Задайте параметри для рядка «STATUS»: Початкові координати (415,14) шрифтом Hershey Complex і чорним кольором шириною 1 піксель. Також задайте параметри відображення проміжних змінних: Початкові координати (375.50) шрифтом Hershey Complex і чорним кольором шириною 1 піксель (рис 3.25).

```
cv2.putText(image, str(count.count), (25, 75), cv2.FONT_HERSHEY_COMPLEX, 2, (0, 0, 0), 2, cv2.LINE_AA)
cv2.putText(image, 'STATUS', (415, 14), cv2.FONT_HERSHEY_COMPLEX, 0.5, (0, 0, 0), 1, cv2.LINE_AA)
cv2.putText(image, staging.staging, (375, 50), cv2.FONT_HERSHEY_COMPLEX, 2, (0, 0, 0), 2, cv2.LINE_AA)
```

Рисунок 3.25 – Налаштування параметрів відображення для проміжних змінних

3.4.5 Реалізація відображення статистики графіків

Додаємо відображення для розпізнаних точок тіла за допомогою функції draw_landmarks mideapipe. Передайте змінюване зображення функції та додайте

параметри. Колір за схемою RGB, значення (255, 255, 0), товщина 5. Ця функція дозволяє налагодити процес виправлення помилок розпізнавання. Потім скористаємося функцією imencode, щоб перетворити зображення з веб-камери у формат jpg і записати його в змінну jpg. Клас Training має повернути побітово закодоване зображення у форматі jpg (рис. 3.26).

```
mp_draw.draw_landmarks(image,
 results.pose_landmarks,
 mp_pose.POSE_CONNECTIONS,
 mp_draw.DrawingSpec(color=(0,  0,  255),   thickness= 5, circle_radius=3),
mp_draw.DrawingSpec(color=(255, 255,    0), thickness= 5, circle_radius=3)
)

ret, jpg = cv2.imencode('.jpg', image)

return jpg.tobytes()
```

Рисунок 3.26 – Процес додавання зображення розпізнаного людського скелета та перетворення зображення на побітовий формат jpg.

3.5 Огляд результату роботи

Результатом розробки є веб-сторінка, яка містить додаток, який може розрахувати кількість і точність виконання одного виду фізичних вправ. Тобто розведіть руки в сторони, з додатковою вагою або без.

Як видно з опису програми ми реалізували програму яка має можливість підраховувати кількість виконань і статус позиції руки у реальному часі за допомогою Python.

Кількість виконання розраховується за наступним алгоритмом: За замовчуванням, коли програма запускається, значення підрахунку виконання дорівнює 0, а значення статусу позиції - «DOWN». Кожен кадр, отриманий від веб-камери, проходить процес розпізнавання зображення за допомогою моделі BlazePose за допомогою MediaPipe. Після оновлення інформації отримують миттєве значення кута між рукою та тілом. Частина тіла, для якої розраховується кут, залежить від обраної вправи. У цьому випадку вправа розводить руки в сторони. Ці миттєві значення кута розраховуються для правої та лівої сторін тіла.

Якщо значення кута менше 45 градусів, стан положення руки під час вправи матиме значення «DOWN».

Оскільки правильним виконанням вправи є розведення рук і підйом їх вище рівня плечей, алгоритм програми виконує тільки одне повторення, якщо кут між тулубом і руками перевищує 90 або 45 градусів, враховуючи яку вправу ви обрали, а також якщо ця умова вірна на обох руках одночасно.

Для того щоб програма зарахувала виконання вправи треба - підняти обидві руки під кутом 90 градусів. Якщо ви піднімите одну руку, або обидві руки, але нижче ніж 90 градусів - програма не зарахує це.

ВИСНОВКИ

В результаті виконання магістерської роботи було веб-застосунок написаний на мові Python з мікрофреймворк Flask та бібліотеками BlazePose для оцінки 3D пози і форми людини за допомогою штучних нейронних.

Під час аналізу предметної області були освітлені основні проблеми використання штучних нейронних мереж.

В роботі виконано аналіз існуючих принципів розпізнавання зображень за допомогою навчання штучного інтелекту його точність розпізнавання фотографій та відео залежить від типу необхідної інформації. Модель або алгоритм можуть виявляти певні елементи так, як зображення можуть бути віднесені до великих категорій на основі їх відповідних характеристик. Також був проведений аналіз основи побудови нейромережевих систем. Проаналізовані рекурентні нейронні мережі та згорткові, та їх важлива роль у будові нейронних мереж. Розглянуто набір мереж для розпізнавання пози людини. Найчастіше використовувані з них - HRNet, OpenPose і BlazePose. Було представлено огляд засобів розробки та навчання нейронних мереж. Мова програмування Python рекомендується для розробки рішень щодо розпізнавання об'єктів це бібліотека TensorFlow для Python.

В кінцевому результаті був отриманий веб-застосунок для оцінки 3D пози і форми людини за допомогою штучних нейронних мереж та підрахунок кількості та якості виконанних вправ, який задовольняє сучасні потреби, користувальницький інтерфейс веб-програми відображає кількість виконаних повторень і положення рук під час виконання вправи. Додано режим налагодження для відображення розпізнаних частин тіла для кращої видимості та відстеження помилок розпізнавання.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Aggarwal, Charu C., et al. Neural networks and deep learning. *Springer*. 2018. Vol. 10. No. 3. 978.

2. Kukreja, Harsh, et al. An introduction to artificial neural network. *Int J Adv Res Innov Ideas Educ*. 2016. Vol.1. 27-30.

3. Raschka, S., Liu, Y. H., Mirjalili, V., & Dzhulgakov, D. Machine Learning with PyTorch and Scikit-Learn: Develop machine learning and deep learning models with Python. *Packt Publishing Ltd*. 2022.

4. Christodoulou, Christos; GEORGIOPOULOS, Michael. Applications of neural networks in electromagnetics. *Artech House, Inc*. 2000.

5. Ramgulan, Asha; ERTEKIN, Turgay; FLEMINGS, Peter B. An artificial neural network utility for the optimization of history matching processes. In: Latin American & Caribbean Petroleum Engineering Conference. *OnePetro*. 2007.

6. Rawat, Waseem; WANG, Zenghui. Deep convolutional neural networks for image classification: A comprehensive review. *Neural computation*. 2017. Vol. 29. No. 9. 2352-2449.

7. Lopez, Adam. Statistical machine translation. *ACM Computing Surveys (CSUR)*. Vol. 40. No. 3. 2008. 1-49.

8. Wolf, Lior; HASSNER, Tal; MAOZ, Italy. Face recognition in unconstrained videos with matched background similarity. *CVPR 2011. IEEE*. 2011. 529-534.

9. Davies, E. Roy. Computer vision: principles, algorithms, applications, learning. *Academic Press*. 2017.

10. Prince, Simon JD. Computer vision: models, learning, and inference. *Cambridge University Press*. 2012.

11. Shirley, Peter; ASHIKHMIN, Michael; MARSCHNER, Steve. Fundamentals of computer graphics. *AK Peters/CRC Press*. 2009.

12. De Marchi, Leonardo, and Laura Mitchell. Hands-On Neural Networks: Learn how to build and train your first neural network model using Python. *Packt Publishing Ltd*. 2019.

13. Singh, Pradeep, ed. Fundamentals and Methods of Machine and Deep Learning: Algorithms, Tools, and Applications. *John Wiley & Sons*. 2022.

14. Kleppmann, Martin. Designing data-intensive applications: The big ideas behind reliable, scalable, and maintainable systems. *O'Reilly Media, Inc*. 2017.

15. Goldberg, Yoav. Neural network methods for natural language processing. Synthesis lectures on human language technologies. *O'Reilly Media, Inc*. Vol. 10. No. 1. 2017. 1-309.

16. Kostadinov, Simeon. Recurrent Neural Networks with Python Quick Start Guide: Sequential learning and language modeling with TensorFlow. *Packt Publishing Ltd*. 2018.

17. Datta, Debajit, et al. Neural machine translation using recurrent neural network. *International Journal of Engineering and Advanced Technology*. Vol. 9. No. 4. 2020. 1395-1400.

18. Raschka, S., Liu, Y. H., Mirjalili, V., & Dzhulgakov, D. Machine Learning with PyTorch and Scikit-Learn: Develop machine learning and deep learning models with Python. *Packt Publishing Ltd.* 2022.

19. AGGARWAL, Charu C. Deep Reinforcement Learning. Neural Networks and Deep Learning. *Springer, Cham*. 2018. 373-417.

20. JAVADIHA, Mohammadreza, et al. Estimating Player Positions from Padel High-Angle Videos: Accuracy Comparison of Recent Computer Vision Methods. *Sensors*. 2021. Vol. 21. No. 10. 3368.

21. Choi, Brendan. Introduction to Python Network Automation. In: Introduction to Python Network Automation. *Apress, Berkeley, CA*. 2021. 1-21.

22. Buelta, Jaime. Python Automation Cookbook: 75 Python automation ideas for web scraping, data wrangling, and processing Excel, reports, emails, and more. *Packt Publishing Ltd*. 2020.

23. Gundecha, U. Learning Selenium testing tools with Python. *Packt Publishing Ltd*. 2014.

24. Kapoor, Amita, et al. Deep Learning with TensorFlow and Keras: Build and deploy supervised, unsupervised, deep, and reinforcement learning models. Packt *Publishing Ltd*. 2022.

25. Kapoor, Amita, et al. Deep Learning with TensorFlow and Keras: Build and deploy supervised, unsupervised, deep, and reinforcement learning models. Packt *Publishing Ltd*. 2022.

26. Reddi, Vijay Janapa, et al. Mlperf inference benchmark. *ACM/IEEE 47th Annual International Symposium on Computer Architecture (ISCA)*. IEEE. 2020. 446-459.

27. Zhang, Fan, et al. Mediapipe hands: On-device real-time hand tracking. *arXiv preprint*. 2020.

28. Bulat, Adrian; TZIMIROPOULOS, Georgios. How far are we from solving the 2d & 3d face alignment problem. *Proceedings of the IEEE International Conference on Computer Vision*. 2017. 1021-1030.

29. Anilmukar, Ardra, et al. Pose Estimated Yoga Monitoring System. *Available at SSRN 3882498*. 2021.

30. SWEIGART, Al. Beyond the Basic Stuff with Python: Best Practices for Writing Clean Code. *No Starch Press*. 2020.

31. WOLF, Lior; HASSNER, Tal; MAOZ, Itay. Face recognition in unconstrained videos with matched background similarity. *CVPR 2011*. *IEEE*. 2011. 529-534.

32. Rudenko O. et al. Developing a Multi-Step Recurrent Algorithm to Maximize the Criteria of Correntropy. *Eastern-European Journal of Enterprise Technologies*. Vol. 1. No. 4. 2021. 109.

33. Rudenko O. et al. Robust identification of non-stationary objects with nongaussian interference. *Eastern-european Journal of enterprise Technologies*. Vol. 5. No. 4. 2019. 44-52.