Міністерство освіти і науки України
Чорноморський національний університет імені Петра Могили
Факультет комп'ютерних наук
Кафедра комп'ютерної інженерії

ДОПУЩЕНО ДО ЗАХИСТУ

Завідуючий кафедри,
канд. техн. наук, доцент

_____ Я. М. Крайник

« __ » _____ 2022 р.

МАРГІСТЕРСЬКА РОБОТА

Спеціальність: 123 Комп'ютерна інженерія

Тема: **Інтелектуальна система для дослідження моделей машинного навчання на базі платформи Jetson Nano**

Шифр: 123 – МР.ПЗ.00 – 605.21610506

Виконав:

Студент 6 курсу, групи 605,
спеціальності
123 Комп'ютерна інженерія
_____ Л. М. Горбуров

Керівник:

Завідуючий кафедри,
канд. техн. наук, доцент
_____ Я. М. Крайник

Миколаїв 2022

ЗАВДАННЯ

на виконання бакалаврської роботи

НЕ ВИДАЛЯТИ цю СТОРІНКУ з файлу !!!!!!!!!!!!!!!!!!

ЗАРЕЗЕРВОВАНА Сторінка 1

ця сторінка після друку буде замінена

ЗАВДАННЯ

на виконання бакалаврської роботи

НЕ ВИДАЛЯТИ цю СТОРІНКУ з файлу !!!!!!!!!!!!!!!!!!

ЗАРЕЗЕРВОВАНА Сторінка 2

ця сторінка після друку буде замінена

АНОТАЦІЯ

1 сторінка !!!!

НЕ ВИДАЛЯТИ цю СТОРІНКУ з файлу !!!!!!!!!!!!!!!!!

ЗАРЕЗЕРВОВАНА Сторінка 1

ця сторінка після друку буде замінена

ABSTRACT

1 сторінка !!!!

НЕ ВИДАЛЯТИ цю СТОРІНКУ з файлу !!!!!!!!!!!!!!!!

ЗАРЕЗЕРВОВАНА Сторінка 2

ця сторінка після друку буде замінена

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

BGR – Blue Green Red

CPU – Central Processor Unit

GPIO – General-Purpose Input/Output

GPU – Graphics Processing Unit

HSV – Hue Saturation Value

KNN – K-Nearest Neighbor

LDA – Linear Discriminant Analysis

UART – Universal Asynchronous Receiver/Transmitter

МН – Машинне навчання

ОЗП – Оперативний записуючий пристрій

ШІ – Штучний інтелект

ШІМ – Широтно-імпульсна модуляція

ВСТУП

Актуальність теми. Машинне навчання сьогодні приділяє всю необхідну увагу. Машинне навчання може автоматизувати багато завдань, особливо ті, які можуть виконувати лише люди зі своїм вродженим інтелектом. Відтворення цього інтелекту на машини можна досягти лише за допомогою машинного навчання.

За допомогою машинного навчання підприємства можуть автоматизувати рутинні завдання. Це також допомагає в автоматизації та швидкому створенні моделей для аналізу даних. Різні галузі залежать від величезної кількості даних для оптимізації своєї діяльності та прийняття розумних рішень. Машинне навчання допомагає створювати моделі, які можуть обробляти та аналізувати великі обсяги складних даних для отримання точних результатів. Ці моделі точні й масштабовані та функціонують із меншим часом виконання. Створюючи такі точні моделі машинного навчання, підприємства можуть використовувати прибуткові можливості та уникнути невідомих ризиків.

Найпоширенішою програмою машинного навчання є розпізнавання обличчя. Існує багато випадків використання розпізнавання обличчя, здебільшого з метою безпеки, ідентифікація злочинців, пошук зниклих безвісти, надання допомоги в судово-медичній експертизі. дослідження тощо. Інтелектуальний маркетинг, діагностика захворювань, відстеження відвідування шкіл це деякі інші види використання.

Розпізнавання зображень, генерація тексту та багато інших варіантів використання знаходять застосування в реальному світі. Це збільшує можливості експертів з машинного навчання показати себе як затребуваних професіоналів.

Мета: розробити інтелектуальну систему для дослідження моделей машинного навчання на базі платформи Jetson Nano.

Об’єкт: процес дослідження моделей машинного навчання та інтегрування цих моделей в інтелектуальну систему на платформі Jetson Nano.

Предмет: інтелектуальна систему для дослідження моделей машинного навчання на базі платформи Jetson Nano.

Для досягнення поставленої мети було поставлено такі завдання:

розглянути схожі інтелектуальні системі на платформі Jetson Nano

– розглянути, алгоритми машинного навчання для розпізнавання об’єктів та моделі міні комп’ютерів;

– розробити апаратну частину інтелектуальної системи з використання однопалатного комп’ютера;

– розробити програмну частину для інтелектуальної системи з використанням моделей машинного навчання;

– протестувати розроблений проект в реальних умовах.

Методи дослідження: абстрагування, аналіз і синтез, спостереження, порівняння.

Практичне значення роботи полягає в тому, що дана інтелектуальна система зможе замінити людину, а також автоматизувати процеси в будь-якій сфері діяльності завдяки моделей та алгоритмів машинного навчання.

Апробація результатів магістерської роботи відбулася в рамках Всеукраїнської науково-практичної конференції молодих вчених, аспіратнтів та студентів «Інформаційні технології та інженерія» (м. Миколаїв, ЧНУ ім. Петра Могили).

Публікації. За результатами магістерської роботи створено публікацію у збірнику матеріалів Всеукраїнської конференції “Інформаційні технології та інженерія” [1].

РОЗДІЛ 1
АНАЛІТИЧНИЙ ОГЛЯД

Магістерська робота за мету ставить розробку інтелектуальної системи для дослідження моделей машинного навчання на базі платформи Jetson Nano. Розглянемо класифікацію штучного інтелекту та моделі машинного навчання.

1.1. Дослідження класифікацій штучного інтелекту

Штучний інтелект є, мабуть, найскладнішим і найдивовижнішим творінням людства. І це не враховуючи той факт, що ця галузь залишається в основному невивченою.

Оскільки дослідження штучного інтелекту мають на меті змусити машини емулювати людське функціонування, ступінь, до якої система ШІ може відтворити людські можливості, використовується як критерій для визначення типів ШІ. Таким чином, залежно від того, як машина порівнюється з людиною з точки зору універсальності та продуктивності, ШІ можна віднести до одного, серед багатьох типів ШІ. У такій системі ШІ, який може виконувати більше людських функцій з еквівалентними рівнями кваліфікації, вважатиметься більш розвиненим типом ШІ, тоді як ШІ з обмеженою функціональністю та продуктивністю вважатиметься більш простим і менш розвиненим типом.

На основі цього критерію існує два способи класифікації ШІ. Один тип заснований на класифікації машин з штучним інтелектом і штучним інтелектом на основі їхньої схожості з людським розумом і їх здатності «думати» і, можливо, навіть «відчуватися» як люди. Згідно з цією системою класифікації, існує чотири типи ШІ або систем на основі ШІ: реактивні машини, машини з обмеженою пам’яттю, теорія розуму та самосвідомий ШІ. Розглянемо більш детальніше про ці типи в наступних розділах.

1.1.1 Реактивні машини

Основні типи систем штучного інтелекту є виключно реактивними і не мають здатності ні формувати спогади, ні використовувати минулий досвід для прийняття поточних рішень. Deep Blue, суперкомп'ютер IBM для гри в шахи, який наприкінці 1990-х переміг міжнародного гросмейстера Гаррі Каспарова, є ідеальним прикладом такого типу машин.

Deep Blue може визначити фігури на шаховій дошці та знати, як кожна рухається. Він може робити прогнози про те, які кроки можуть бути наступними для нього та його суперника. І він може вибрати найбільш оптимальні ходи з-поміж можливостей.

Але в ньому немає ні поняття про минуле, ні спогадів про те, що було раніше. Крім рідко використовуваного шахового правила проти повторення одного і того ж ходу тричі, Deep Blue ігнорує все до теперішнього моменту. Все, що він робить, — це дивитися на фігури на шаховій дошці, як вона стоїть зараз, і вибирати з можливих наступних ходів.

Цей тип інтелекту передбачає, що комп’ютер безпосередньо сприймає світ і діє відповідно до того, що він бачить. Він не спирається на внутрішню концепцію світу. У визначній роботі дослідник ШІ Родні Брукс стверджував, що ми повинні створювати лише машино подібних. Його головна причина полягала в тому, що люди не дуже добре вміють програмувати точні змодельовані світи для використання комп’ютерами, що в науках ШІ називають «уявленням» світу.

Нинішні розумні машини, або не мають такої концепції світу, або мають дуже обмежену та спеціалізовану машину для виконання своїх конкретних завдань. Інновація в дизайні Deep Blue не полягала в тому, щоб розширити діапазон можливих фільмів, які розглядав комп'ютер. Швидше, розробники знайшли спосіб звузити його погляд, припинити переслідувати деякі потенційні майбутні кроки, виходячи з того, як вони оцінювали їхній

результат. Без цієї здатності Deep Blue мав би бути ще потужнішим комп’ютером, щоб фактично перемогти Каспарова.

Аналогічно, AlphaGo від Google, який переміг провідних експертів Go, також не може оцінити всі потенційні майбутні кроки. Його метод аналізу є більш складним, ніж метод Deep Blue, який використовує нейронну мережу для оцінки розвитку гри.

Ці методи покращують здатність систем ШІ краще грати в певні ігри, але їх не можна легко змінити або застосувати до інших ситуацій. Ці комп’ютеризовані уяви не мають поняття про ширший світ, тобто вони не можуть функціонувати за межами конкретних завдань, які їм доручають, і їх легко обдурити [2].

Вони не можуть інтерактивно брати участь у світі. Натомість ці машини будуть вести себе точно так само щоразу, коли стикатимуться з такою ж ситуацією. Це може бути дуже добре для забезпечення надійності системи ШІ: щоб автономний автомобіль був надійним водієм. Але погано, якщо хотіти, щоб машини справді взаємодіяли зі світом і реагували на нього. Ці найпростіші системи штучного інтелекту ніколи не будуть нудьгувати, цікавити чи сумувати.

1.1.2 Обмежена пам'ять

Цей клас II типу містить машини, які можуть зазирнути в минуле. Автомобілі з автономним керуванням вже роблять це. Наприклад, вони спостерігають за швидкістю та напрямком інших автомобілів. Це не можна зробити за одну мить, а скоріше вимагає ідентифікації конкретних об’єктів та моніторингу їх у часі.

Ці спостереження додаються до попередньо запрограмованих уявлень про світ самокерованими автомобілями, які також включають розмітку смуги, світлофор та інші важливі елементи, як-от вигини на дорозі. Вони включаються, коли автомобіль вирішує, коли змінити смугу руху, щоб

уникнути зрізання іншого водія або збиття розташованого поблизу автомобіля.

Але ці прості відомості про минуле є лише тимчасовими. Вони не зберігаються як частина бібліотеки досвіду автомобіля, на якому він може вчитися, як люди-водії збирають досвід протягом багатьох років за кермом.

Тож як побудувати системи штучного інтелекту, які створюють повне уявлення, запам’ятовують їхній досвід і навчаються поводитися з новими ситуаціями? Брукс був правий у тому, що зробити це дуже важко. Мої власні дослідження методів, натхнених дарвінівською еволюцією, можуть почати компенсувати людські недоліки, дозволивши машинам створювати власні уявлення [3].

1.1.3 Теорія розуму

Можна було б зупинитися на цьому й назвати цей момент важливою різницею між машинами, які маємо, і машинами, які будемо створювати в майбутньому. Однак краще бути більш конкретним, щоб обговорити типи уявлень, які повинні сформувати машини, і про що вони повинні бути.

Машини майбутнього, більш просунутого класу не лише формують уявлення про світ, а й про інших агентів чи сутностей у світі. У психології це називається «теорією розуму» – розумінням того, що люди, істоти та предмети у світі можуть мати думки та емоції, які впливають на їх власну поведінку.

Це має вирішальне значення для того, як люди, формували суспільства, тому що вони дозволили нам мати соціальні взаємодії. Без розуміння мотивів і намірів один одного і без урахування того, що хтось знає про мене чи про оточення, працювати разом у кращому випадку важко, у гіршому неможливо.

Якщо системи штучного інтелекту дійсно коли-небудь будуть ходити серед нас, вони повинні мати можливість зрозуміти, що кожен з нас має думки, почуття та очікування щодо того, як з нами будуть поводитися. І їм доведеться відповідно скорегувати свою поведінку [4].

1.1.4 Самосвідомість

Останнім кроком розвитку ШІ є створення систем, які можуть формувати уявлення про себе. Зрештою, нам, дослідникам ШІ, доведеться не тільки зрозуміти свідомість, але й створити машини, які її мають.

Це, в певному сенсі, розширення «теорії розуму», якою володіють штучні інтелекти III типу. Свідомість недарма також називають «самосвідомістю». Свідомі істоти усвідомлюють себе, знають про свій внутрішній стан і здатні передбачати почуття інших. Ми припускаємо, що хтось, хто сигналить позаду нас у заторі, злий або нетерплячий, тому що саме так ми відчуваємо, коли сигналимо іншим. Без теорії розуму не можна було б зробити подібні висновки.

Навчання та здатності приймати рішення на основі минулого досвіду. Це важливий крок для розуміння людського інтелекту самостійно. І це важливо, якщо проектуємо чи розвиваємо машини, які більш ніж виняткові в класифікації того, що вони бачать перед собою [5].

1.1.5 Штучний вузький інтелект

Цей тип штучного інтелекту представляє весь існуючий ШІ, включаючи навіть найскладніший і найздатніший ШІ, який коли-небудь був створений на сьогоднішній день. Штучний вузький інтелект відноситься до систем ШІ, які можуть виконувати лише конкретне завдання автономно, використовуючи людські можливості [6]. Ці машини не можуть робити нічого більше, ніж те, на що вони запрограмовані, і, таким чином, мають дуже обмежений або вузький діапазон компетенцій. Відповідно до вищезгаданої системи класифікації, цим системам відповідають всі реактивні та обмежені пам'яті ШІ. Навіть найскладніший ШІ, який використовує машинне навчання та глибоке навчання для навчання, підпадає під ШТІ.

1.1.6 Штучний супер та загальний інтелект

Штучний загальний інтелект – це здатність агента ШІ вчитися, сприймати, розуміти та функціонувати повністю як людина. Ці системи зможуть самостійно створювати кілька компетенцій і формувати зв’язки та узагальнення між доменами, значно скорочуючи час, необхідний для навчання. Це зробить системи штучного інтелекту такими ж здатними, як і люди, завдяки відтворенню наших багатофункціональних можливостей [7].

Розвиток штучного супер інтелекту, ймовірно, стане вершиною досліджень ШІ, оскільки стане, безумовно, найздатнішою формою інтелекту на землі. ШСІ, на додаток до відтворення багатогранного інтелекту людей, буде надзвичайно кращим у всьому, що вони роблять, завдяки значно більшій пам’яті, швидшій обробці та аналізу даних, а також можливостям прийняття рішень. Розвиток ШЗІ та ШСІ призведе до сценарію, який найчастіше називають сингулярністю. І хоча потенціал наявності таких потужних машин у нашому розпорядженні здається привабливим, ці машини також можуть загрожувати нашому існуванню або, принаймні, нашому способу життя.

1.2. Дослідження моделей машинного навчання

На високому рівні машинне навчання – це просто вивчення навчання комп’ютерній програмі чи алгоритму, як поступово покращувати поставлене завдання. З точки зору досліджень, машинне навчання можна розглядати через призму теоретичного та математичного моделювання того, як працює цей процес. Однак більш практично це дослідження того, як створювати програми, які демонструють це ітераційне вдосконалення. Існує багато способів сформулювати цю ідею, але в основному існують три основні визнані категорії: навчання під наглядом, навчання без нагляду та навчання з підкріпленням.

1.2.1 Навчання під наглядом

Кероване навчання є одним з найпростіших видів машинного навчання. У цьому типі алгоритм машинного навчання навчається на позначених даних. Незважаючи на те, що дані повинні бути точно позначені, щоб цей метод працював, навчання з наглядом надзвичайно потужне, якщо використовується в правильних обставинах.

Під час навчання під керівництвом алгоритму надається невеликий набір навчальних даних для роботи. Цей навчальний набір даних є меншою частиною більшого набору даних і служить для того, щоб дати алгоритму основне уявлення про проблему, рішення та точки даних, з якими потрібно мати справу. Навчальний набір даних також дуже схожий на кінцевий набір даних за своїми характеристиками і надає алгоритм з позначеними параметрами, необхідними для вирішення проблеми (див. рис. 1.1).

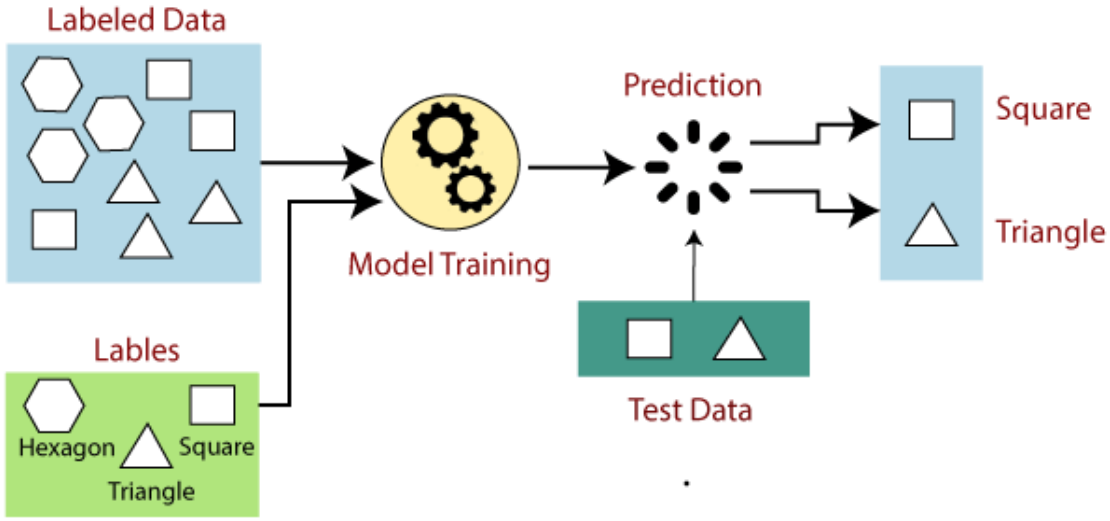

Рисунок 1.1 – Схема навчання під наглядом

Потім алгоритм знаходить зв’язки між заданими параметрами, по суті встановлюючи причинно-наслідковий зв’язок між змінними в наборі даних. Наприкінці навчання алгоритм має уявлення про те, як працюють дані та зв’язок між входом і виходом.

Потім це рішення розгортається для використання з остаточним набором даних, з якого воно вивчає так само, як і навчальний набір даних. Це

означає, що контрольовані алгоритми машинного навчання будуть продовжувати вдосконалюватися навіть після розгортання, відкриваючи нові закономірності та зв’язки, навчаючись на нових даних [8].

1.2.2 Навчання без нагляду

Перевага машинного навчання без нагляду полягає в тому, що він може працювати з немаркованими даними. Це означає, що людська праця не потрібна, щоб зробити набір даних машинозчитаним, що дозволяє програмі працювати з набагато більшими наборами даних.

Під час навчання під керівництвом мітки дозволяють алгоритму знайти точний характер зв’язку між будь-якими двома точками даних. Однак неконтрольоване навчання немає ярликів, які можна було б відпрацювати, що призводить до створення прихованих структур. Зв'язки між точками даних сприймаються алгоритмом абстрактна, без введення даних від людей (див. рис. 1.2).

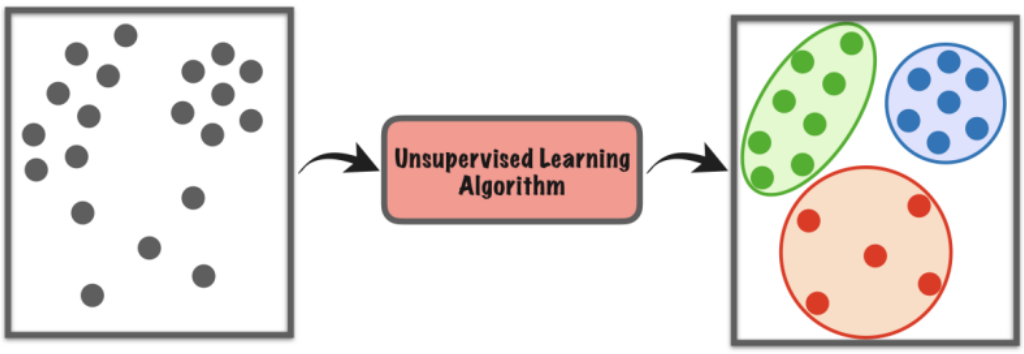

Рисунок 1.2 – Схема навчання без нагляду

Створення цих прихованих структур робить алгоритми навчання без нагляду універсальними. Замість визначеної та встановленої постановки проблеми алгоритми навчання без нагляду можуть адаптуватися до даних шляхом динамічної зміни прихованих структур. Це пропонує більше розвитку після розгортання, ніж алгоритми навчання під керівництвом [9].

1.2.3 Навчання з підкріпленням

Навчання з підкріпленням безпосередньо черпає натхнення з того, як люди навчаються на даних у своєму житті. У ньому є алгоритм, який вдосконалює себе і вивчає нові ситуації за допомогою методу проб і помилок. Сприятливі результати заохочуються або «підкріплюються», а несприятливі результати не заохочуються або «караються» [10].

На основі психологічної концепції кондиціонування навчання з підкріпленням працює шляхом розміщення алгоритму в робочому середовищі з перекладачем і системою винагород. На кожній ітерації алгоритму вихідний результат передається інтерпретатору, який вирішує, сприятливий результат чи ні.

Якщо програма знаходить правильне рішення, інтерпретатор підкріплює рішення, надаючи винагороду алгоритму. Якщо результат несприятливий, алгоритм змушений повторювати, поки не знайде кращий результат (див. рис. 1.3). У більшості випадків система винагород безпосередньо пов’язана з ефективністю результату.

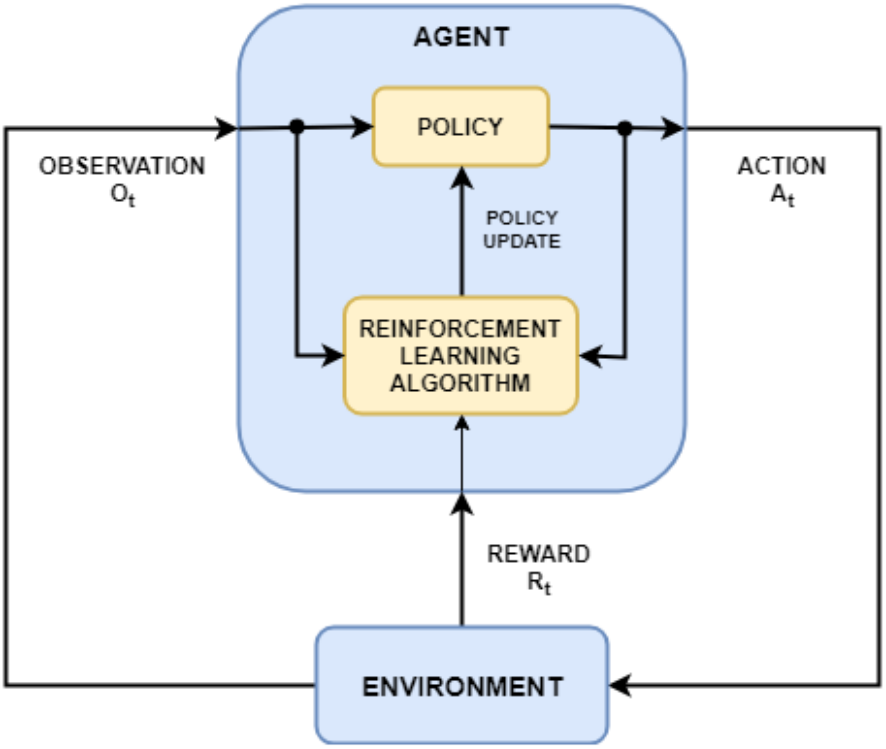

Рисунок 1.3 – Схема навчання без нагляду

У типових випадках використання навчання з підкріпленням, таких як пошук найкоротшого маршруту між двома точками на карті, рішення не є абсолютним значенням. Замість цього він отримує оцінку ефективності, виражену у відсотках. Чим вище це відсоткове значення, тим більше винагорода отримує алгоритм. Таким чином, програма навчена надавати найкраще можливе рішення за найкращу можливу винагороду.

Висновки до розділу 1

В розділі 1 проаналізовано моделі машинного навчання, а також класифікацію штучного інтелекту.

Сьогодні термін «штучний інтелект» використовується як зонтичний термін для позначення технології, яка має когнітивні характеристики, подібні до людини. Як правило, дослідження в області ШІ рухаються до більш узагальненої форми інтелекту, подібного до того, як малюки думають і сприймають навколишній світ. Це може ознаменувати еволюцію ШІ від програми, спеціально створеної для одного «вузького» завдання, до рішення, розгорнутого для «загальних» рішень; такого, якого ми можемо очікувати від людей.

Машинне навчання також включає глибоке навчання, спеціалізовану дисципліну, яка є ключем до майбутнього ШІ. Глибоке навчання містить нейронні мережі, тип алгоритму, який базується на фізичній структурі людського мозку. Нейронні мережі, здається, є найбільш продуктивним шляхом для досліджень ШІ, оскільки вони дозволяють набагато ближче імітувати людський мозок, ніж будь-коли раніше.

В результаті огляду моделей машинного навчання, для комплексу буде реалізовувано навчання за допомогою всіх моделей, а результаті дослідження буде проаналізовано кожну модель та обрана найшвидша та найточніша.

РОЗДІЛ 2
МОДЕЛЮВАННЯ ТА ПРОЕКТУВАННЯ ІНТЕЛЕКТУАЛЬНОЇ СИСТЕМИ

Не малу роль в розробці інтелектуальних систем відіграє проектування та моделювання за допомогою комп’ютерних технологій. Отож в цьому розділі змоделюємо інтелектуальну систему для дослідження моделей машинного навчання, підберемо оптимальний варіант живлення, а також розглянемо алгоритми машинного навчання.

2.1. Проектування інтелектуальної системи

Перед тим як реалізувати проект на реальних компонентах, спочатку потрібно спроектувати схему системи. Потужним програмним компонентом для проектування схем є Fritzing [11]. Отож спроектуємо систему.

В даному випадку є два види приєднання камери: за допомогою USB та за допомогою спеціального роз’єму для камери на платі. Перший є простим в з’єднанні, але бувають випадки, що камера може буди не сумісна, а саме не встановиться драйвер, тобто камера не буде працювати. Другий є більш складніший по з’єднанню, але швидший по передачі даний та зможе забезпечити передачу якісного зображення з камери.

Система складаються з однопалатного комп’ютера та камери. За допомогою шлейфового з’єднання камера підключається до міні-комп’ютера. Нижче наведено ілюстрація з’єднання компонентів (див. рис. 2.1).

Рисунок 2.1 Схема з’єднань компонентів

Правильне підібране живлення є запорукою швидкодії системи, адже чим більший вхідний струм, тим більше може використовуватися ядер центрального та відео процесору, а це прискорює обчислення. Особливо при обробці зображень з використанням нейронних мереж.

Є 3 варіанти живлення Jetson Nano:

− Через роз'єм micro USB на 5 В/2 А (10 Вт). За замовчуванням використовується лише 2 з 4 ядер ЦП (режим живлення 5 Вт).

− Використання гнізда Barrel на 5 В/4 А (20 Вт). За замовчуванням використовуються всі 4 ядра (режим максимальної потужності).

− Через роз’єм GPIO (рис. 2.2) на 5 В/6 А (30 Вт) для використання з периферійними пристроями більшої потужності (режим максимальної потужності).

Sysfs	Name	Pin	Pin	Name	Sysfs
	3.3V DC	1	2	5V DC	
	I2C_2_SDA	3	4	5V DC	
	I2C_2_SCL	5	6	GND	
gpio216	AUDIO_MCLK	7	8	UART_2_TX	
	GND	9	10	UART_2_RX	
gpio50	UART_2_RTS	11	12	I2S_4_CLK	gpio79
gpio14	SPI_2_SCK	13	14	GND	
gpio194	LCD_TE	15	16	SPI_2_CS1	gpio232
	3.3V DC	17	18	SPI_2_CS0	gpio15
gpio16	SPI_1_MOSI	19	20	GND	
gpio17	SPI_1_MISO	21	22	SPI_2_MISO	gpio13
gpio18	SPI_1_SCK	23	24	SPI_2_CS0	gpio19
	GND	25	26	SPI_2_CS1	gpio20
	IC2_1_SDA	27	28	I2C_1_SCL	
gpio149	CAM_AF_EN	29	30	GND	
gpio200	GPIO_PZO	31	32	LCD_BL_PWM	gpio168
gpio38	GPIO_PE6	33	34	GND	
gpio76	I2S_4_LRCK	35	36	UART_2_CTS	gpio51
gpio12	SPI_2_MOSI	37	38	I2S_4_SDIN	gpio77
	GND	39	40	I2S_4_SDOUT	gpio78

Рисунок 2.2 – Схема GPIO

Всі, крім контактів зв’язку I2C на заголовку, безпосередньо підключені до модуля Jetson Nano. Виводи I2C підключаються до перемикача

проміжного рівня для зміни напруги з 1,8 В на модулі до 3,3 В на стандартному інтерфейсі I2C. Це контакти 3 і 5 (виводи I2C SDA) і контакти 27 і 28 (контакти I2C SCL). Виводи 8 і 10 є контактами передавача (TX) і приймача (Rx) UART відповідно.

Плата також має два джерела живлення 3,3 В (виводи 1 і 17) і два джерела живлення 5 В (виводи 2 і 4). Розділ розширення також має кілька контактів заземлення.

Заголовок розширення має лише 2 канали ШІМ, безпосередньо підключені до апаратних ШІМ-контролерів. Він не підтримує програмне забезпечення ШІМ. На жаль, модуль Nano за своєю суттю не налаштований для визначення підключення до апаратного забезпечення ШІМ. Таким чином, щоб використовувати апаратне забезпечення ШІМ, системний PinMux має бути налаштований на забезпечення функціональних можливостей контактів ШІМ.

У цій роботі використовується роз’єм GPIO 5 В/6 А (30 Вт) – це є оптимальне рішення, адже обробка зображення потребує максимальної продуктивності системию.

2.2. Дослідження алгоритмів машинного навчання

2.2.1 Лінійна регресія

Лінійна регресія є, мабуть, одним із найбільш відомих і добре зрозумілих алгоритмів у статистиці та машинному навчанні.

Прогностичне моделювання в першу чергу спрямоване на мінімізацію помилок моделі або створення найбільш точних передбачень, які можна передбачити, за рахунок пояснення. Запозичувати, повторно використовувати та красти алгоритми з багатьох різних областей, включаючи статистику, і використовуватиме їх у цих цілях.

Подання лінійної регресії – це рівняння, яке описує лінію, яка найкраще відповідає відносинам між вхідними змінними (x) і вихідними

змінними (y), шляхом знаходження конкретних зважень для вхідних змінних (див. рис. 2.3), які називаються коефіцієнтами (B).

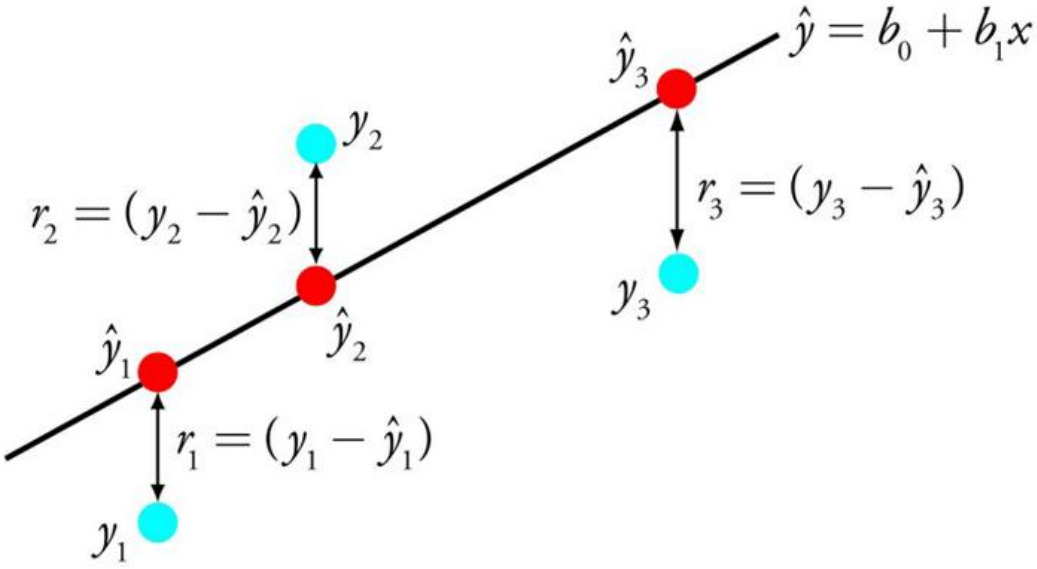

Рисунок 2.3 – Лінійна регресія

Для вивчення моделі лінійної регресії на основі даних можна використовувати різні методи, наприклад рішення лінійної алгебри для звичайних найменших квадратів та оптимізацію градієнтного спуску.

Лінійна регресія існує вже більше 200 років і широко вивчена. Деякі практичні правила під час використання цієї техніки полягають у тому, щоб видалити змінні, які дуже схожі (корельовані), і видалити шум із ваших даних, якщо це можливо. Це швидка та проста техніка та хороший перший алгоритм, який потрібно спробувати [12].

2.2.2 Логістична регресія

Логістична регресія – це ще один прийом, запозичений машинним навчанням зі сфери статистики. Це основний метод для задач бінарної класифікації (задачі з двома значеннями класу).

Логістична регресія подібна до лінійної регресії, оскільки мета полягає в тому, щоб знайти значення коефіцієнтів, які мають вагу для кожної вхідної змінної. На відміну від лінійної регресії, прогноз для результату

перетворюється за допомогою нелінійної функції, яка називається логістичною функцією.

Логістична функція виглядає як велике S і перетворює будь-яке значення в діапазон від 0 до 1 (див. рис. 2.4). Це корисно, оскільки можена застосувати правило до виводу логістичної функції, щоб прив’язати значення до 0 і 1 (наприклад, якщо менше 0,5, то вихід 1) і передбачити значення класу.

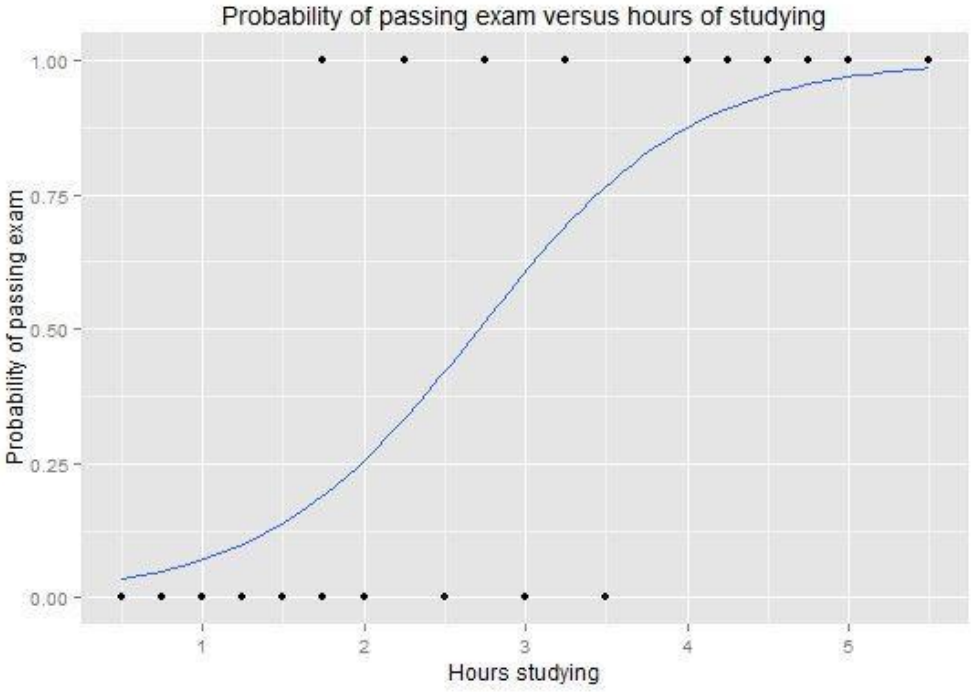

Рисунок 2.4 – Графік логістичної регресії

Завдяки тому, як вивчається модель, передбачення, зроблені за допомогою логістичної регресії, також можна використовувати як ймовірність того, що даний екземпляр даних належить до класу 0 або класу 1. Це може бути корисно для проблем, де потрібно надати більше обґрунтування для передбачення.

Як і лінійна регресія, логістична регресія працює краще, коли видаляють атрибути, які не пов’язані з вихідною змінною, а також атрибути, які дуже схожі один на одного. Це швидка модель для вивчення та ефективна у проблемах бінарної класифікації [13].

2.2.3 Лінійний дискримінантний аналіз

Логістична регресія – це алгоритм класифікації, який традиційно обмежується лише двокласовими задачами класифікації. Якщо більше двох класів, то найкращим методом лінійної класифікації є алгоритм лінійного дискримінантного аналізу.

Представництво LDA досить просте. Він складається зі статистичних властивостей ваших даних, розрахованих для кожного класу (див. рис. 2.5). Для однієї вхідної змінної це включає:

- Середнє значення для кожного класу.
- Дисперсія розрахована для всіх класів.

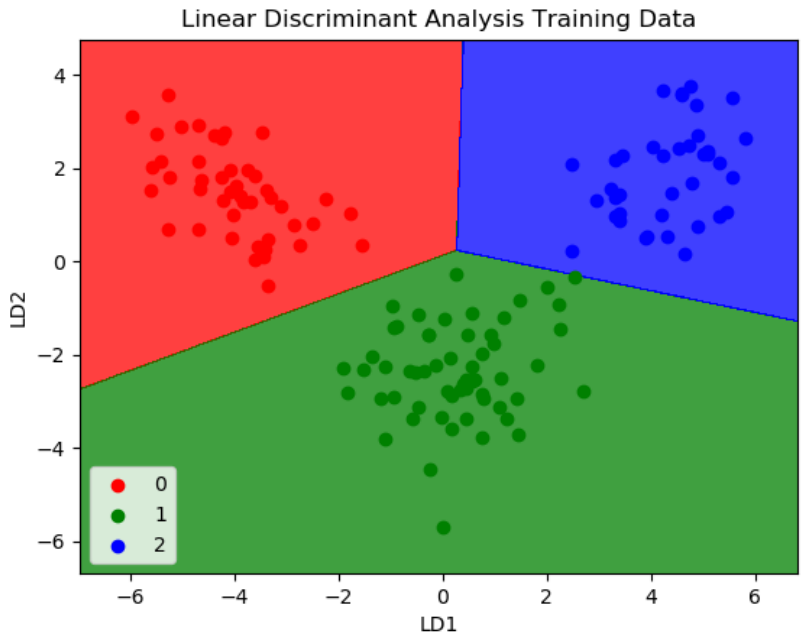

Рисунок 2.5 – Лінійний дискримінантний аналіз

Прогнози робляться шляхом обчислення дискримінаційного значення для кожного класу та створення прогнозу для класу з найбільшим значенням. Техніка передбачає, що дані мають гаусівський розподіл, тому доцільно попередньо видалити викиди з даних. Це простий і потужний метод для класифікації проблем прогнозного моделювання [14].

2.2.4 Дерева класифікації та регресії

Дерева рішень є важливим типом алгоритму для прогнозного моделювання машинного навчання.

Представленням моделі дерева рішень є бінарне дерево (див. рис. 2.6). Це двійкове дерево з алгоритмів і структур даних, нічого зайвого. Кожен вузол представляє одну вхідну змінну (x) і точку розділення цієї змінної (припускаючи, що змінна є числовою).

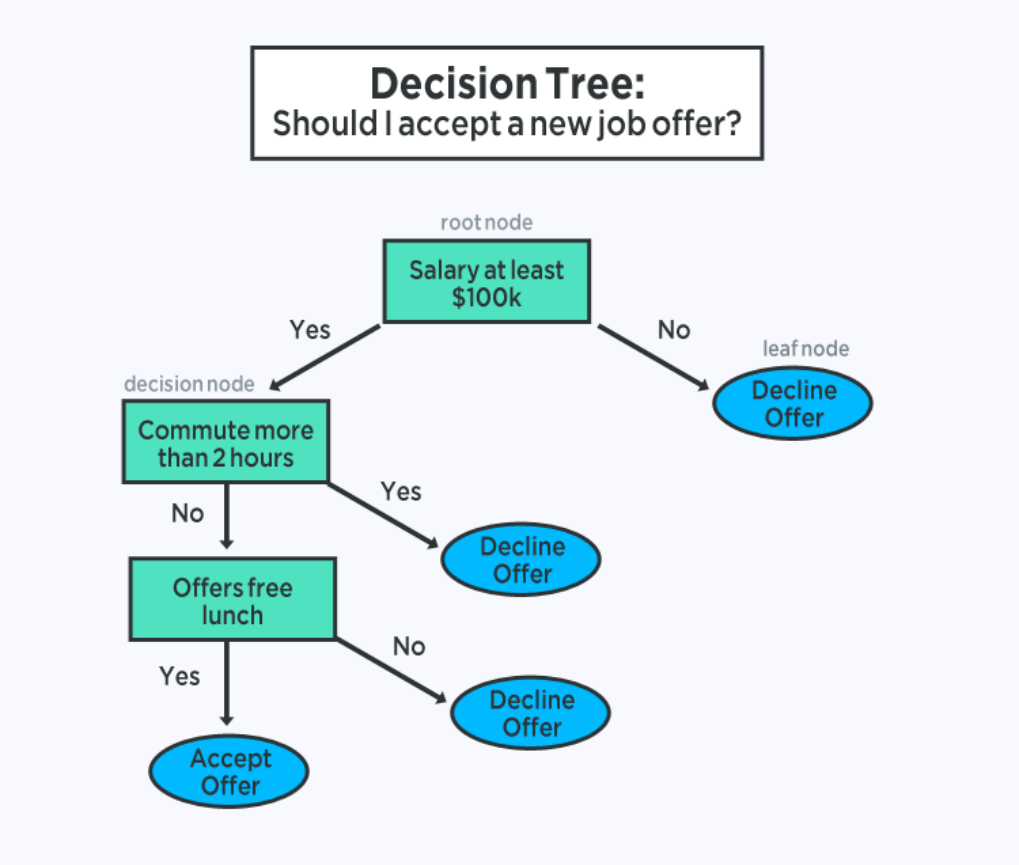

Рисунок 2.6 – Дерево рішень

Листові вузли дерева містять вихідну змінну (y), яка використовується для передбачення. Прогнози робляться шляхом проходження розколів дерева до прибуття до листкового вузла і виведення значення класу в цьому листковому вузлі.

Дерева швидко навчаються і дуже швидко роблять прогнози. Вони також часто точні для широкого кола проблем і не вимагають спеціальної підготовки для ваших даних.

2.2.5 Наївний Байєс

Наївний Байєс – це простий, але напрочуд потужний алгоритм прогнозного моделювання.

Модель складається з двох типів ймовірностей (див. рис. 2.7), які можна розрахувати безпосередньо з ваших навчальних даних:

1) ймовірність кожного класу;

2) умовна ймовірність для кожного класу з урахуванням кожного значення x.

Після обчислення модель ймовірності можна використовувати для прогнозування нових даних за допомогою теореми Байєса. Коли ваші дані мають реальні значення, зазвичай приймають гаусівський розподіл (дзвіночкова крива), щоб ви могли легко оцінити ці ймовірності.

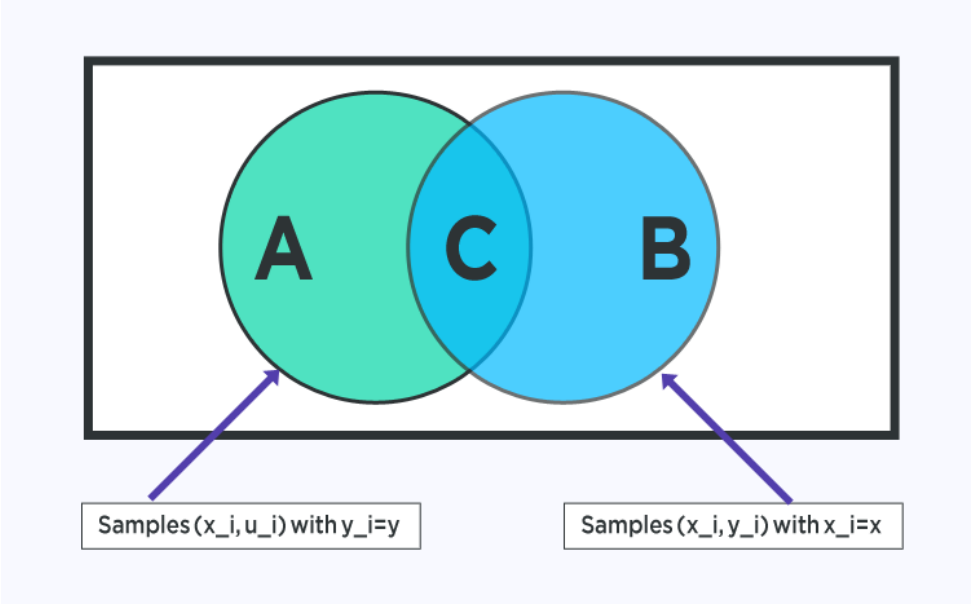

Рисунок 2.7 – Наївний Байєс

Наївний Байєс називається наївним, оскільки він передбачає, що кожна вхідна змінна є незалежною. Це сильне припущення і нереалістичне для реальних даних, тим не менш, методика дуже ефективна для широкого кола складних проблем [15].

2.2.6 K-найближчі сусіди

Алгоритм KNN дуже простий і дуже ефективний. Представлення моделі для KNN – це весь навчальний набір даних.

Прогнозування для нової точки даних здійснюється шляхом пошуку в усьому навчальному наборі (див. рис. 2.8) для K найбільш подібних екземплярів (сусідів) і підсумовування вихідної змінної для цих K екземплярів. Для задач регресії це може бути середня вихідна змінна, для задач класифікації це може бути значення класу режиму (або найпоширенішого).

Хитрість полягає в тому, як визначити схожість між екземплярами даних. Найпростіший прийом, якщо всі ваші атрибути мають однаковий масштаб (наприклад, усі в дюймах) – це використовувати евклідову відстань, число, яке ви можете обчислити безпосередньо на основі відмінностей між кожною вхідною змінною.

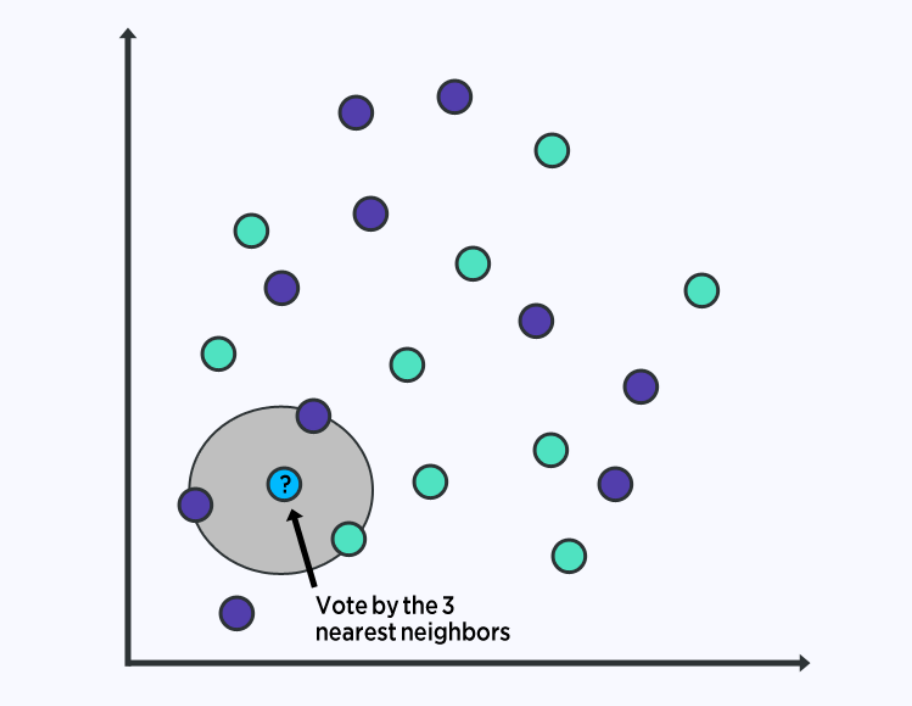

Рисунок 2.8 – K-найближчі сусіди

KNN може вимагати багато пам’яті або місця для зберігання всіх даних, але виконує обчислення (або навчання) лише тоді, коли потрібне

передбачення, точно вчасно. Ви також можете оновлювати та керувати свої навчальні екземпляри з часом, щоб прогнози були точними.

Ідея відстані чи близькості може розбитись на дуже великі розміри (багато вхідних змінних), що може негативно вплинути на продуктивність алгоритму для вашої проблеми. Це називається прокляттям розмірності. Він пропонує використовувати лише ті вхідні змінні, які є найбільш релевантними для прогнозування вихідної змінної [16].

2.3. Алгоритми оптимізації

Оптимізація – це проблема пошуку набору вхідних даних для цільової функції, що призводить до максимального або мінімального оцінювання функції.

Це складна проблема, яка лежить в основі багатьох алгоритмів машинного навчання, від підгонки моделей логістичної регресії до навчання штучних нейронних мереж.

Існують, можливо, сотні популярних алгоритмів оптимізації і, можливо, десятки алгоритмів на вибір у популярних наукових бібліотеках коду. Це може ускладнити визначення алгоритмів для даної задачі оптимізації.

2.3.1 Алгоритми брекетингу

Алгоритми оптимізації брекетингу призначені для задач оптимізації з однією вхідною змінною, де відомо, що оптимум існує в певному діапазоні.

Алгоритми брекетингу здатні ефективно орієнтуватися у відомому діапазоні та визначати місцезнаходження оптимуму, хоча вони припускають, що присутній лише один оптимум (званий унімодальними цільовими функціями) [17].

Деякі алгоритми брекетингу можна використовувати без похідної інформації, якщо вона недоступна.

Приклади алгоритмів брекетингу включають:

- Пошук Фібоначчі.
- Пошук золотого перетину.
- Метод бісекції.

2.3.2 Алгоритми локального спуску

Алгоритми оптимізації локального спуску призначені для задач оптимізації з більш ніж однією вхідною змінною та одним глобальним оптимумом (наприклад, унімодальна цільова функція).

Можливо, найпоширенішим прикладом алгоритму локального спуску є алгоритм пошуку рядків.

Існує багато варіантів пошуку рядків (наприклад, алгоритм Брента-Деккера), але процедура зазвичай включає вибір напрямку для переміщення в просторі пошуку, а потім виконання пошуку типу дужок у лінії або гіперплощині у вибраному напрямку [18].

Цей процес повторюється до тих пір, поки не буде внесено жодних покращень.

Обмеження полягає в тому, що оптимізація кожного спрямованого руху в просторі пошуку є дорогою з точки зору обчислень.

2.3.3 Алгоритми першого порядку

Алгоритми оптимізації першого порядку явно передбачають використання першої похідної (градієнта) для вибору напрямку руху в просторі пошуку.

Процедури передбачають спочатку обчислення градієнта функції, а потім слідування градієнту в протилежному напрямку (наприклад, спуск до мінімуму для задач мінімізації) з використанням розміру кроку (також називається швидкістю навчання).

Розмір кроку – це гіперпараметр, який контролює, як далеко рухатися в просторі пошуку, на відміну від «алгоритмів локального спуску», які виконують повний рядковий пошук для кожного напрямкового переміщення.

Занадто малий розмір кроку призводить до пошуку, який займає тривалий час і може застрягти, тоді як завеликий розмір кроку призведе до зигзагоподібного переміщення або підскаку в області пошуку, повністю пропустивши оптимум.

Алгоритми першого порядку зазвичай називають градієнтним спуском, а більш конкретні назви стосуються незначних розширень процедури, наприклад:

- Градієнтний спуск
- Імпульс
- Адаград
- RMSProp
- Адам

Алгоритм градієнтного спуску також надає шаблон для популярної стохастичної версії алгоритму під назвою Stochastic Gradient Descent (SGD), яка використовується для навчання моделей штучних нейронних мереж.

Важлива відмінність полягає в тому, що градієнт присвоюється, а не розраховується безпосередньо, використовуючи помилку прогнозування для навчальних даних, таких як одна вибірка, усі приклади або невелика підмножина навчальних даних.

Розширення, призначені для прискорення алгоритму градієнтного спуску (імпульсу тощо), можуть використовуватися та зазвичай використовуються з SGD.

- Стохастичний градієнтний спуск
- Пакетний градієнтний спуск
- Міні-пакетний градієнтний спуск

2.3.4 Алгоритми другого порядку

Алгоритми оптимізації другого порядку явно передбачають використання другої похідної (Гессена) для вибору напрямку руху в просторі пошуку.

Ці алгоритми підходять лише для тих цільових функцій, де матриця Гессе може бути обчислена або апроксимована.

Приклади алгоритмів оптимізації другого порядку для одновимірних цільових функцій включають:

- Метод Ньютона
- Метод січної
- Методи другого порядку для багатовимірних цільових функцій називаються квазіньютоновськими методами.
- Метод квазіньютона

Існує багато квазіньютоновських методів, і вони зазвичай називаються на честь розробників алгоритму, наприклад:

- Девідсон-Флетчер-Пауелл
- Бройден-Флетчер-Голдфарб-Шанно (BFGS)
- BFGS з обмеженою пам'яттю (L-BFGS)

Тепер, коли ми знайомі з так званими класичними алгоритмами оптимізації, давайте подивимося на алгоритми, які використовуються, коли цільова функція не диференційована [19].

2.3.5 Недиференційна цільова функція

Алгоритми оптимізації, які використовують похідну цільової функції, є швидкими та ефективними.

Тим не менш, існують цільові функції, де похідна не може бути обчислена, як правило, тому, що функція є складною з різних причин реального світу. Або похідну можна обчислити в деяких регіонах домену, але не у всіх, або не є хорошим орієнтиром.

Деякі труднощі щодо цільових функцій для класичних алгоритмів, описаних у попередньому розділі, включають:

- Немає аналітичного опису функції (наприклад, моделювання).
- Множинні глобальні оптимуми (наприклад, мультимодальні).
- Оцінка стохастичної функції (наприклад, шумова).

− Розривна цільова функція (наприклад, області з недійсними рішеннями).

Таким чином, існують алгоритми оптимізації, які не очікують, що похідні першого чи другого порядку будуть доступними.

Ці алгоритми іноді називають алгоритмами оптимізації чорного ящика, оскільки вони мало припускають або нічого (порівняно з класичними методами) щодо цільової функції.

2.5. Висновки до розділу 2

В розділі 2 було розглянуто алгоритми оптимізації та машинного навчання.

Загалом, існують дві основні категорії алгоритмів МН: контрольоване та неконтрольоване МН. Контрольовані алгоритми МН включають «навчання» машини виробляти результати на основі своїх навчальних даних, які вже позначені або структуровані. З іншого боку, неконтрольовані алгоритми ML працюють з неструктурованими даними – даними, які ще не були класифіковані чи позначені.

Також було зпроектовано та змодельовано інтелектуальну систему для дослідження моделей машинного навчання. За допомогою інструменту Fritzing було створена схема підключення апаратної складової проєкту. Розраховано живлення системи для коректної роботи однопалатного комп’ютера.

РОЗДІЛ 3
АПАРТНО-ПРОГРАМНЕ ЗАБЕПЕЧЕННЯ НТЕЛЕКТУАЛЬНОЇ СИСТЕМИ

3.1. Вибір одно платного комп’ютера для системи

Одноплатні комп'ютери (SBC) є універсальними інструментами для вирішення широкого спектра завдань. Вони можуть стати основою для систем числового програмного управління (ЧПУ), виступити в ролі сервера зберігання даних і навіть використовуватися як віддалена система збору даних. Вочевидь, що у кожному даному випадку вимоги до одно платним комп'ютерам різняться.

Сімейство «малинових» одноплатних комп'ютерів є своєрідним мірилом для всіх SBC. Дуже часто на різних інтернет майданчиках з'являються опитування та рейтинги під назвою «10 кращих альтернатив для Raspberry Pi». Це пов'язано із величезною популярністю Raspberry Pi. У багатьох спільнотах новачкові порадять почати знайомство із SBC саме з цих одноплатних комп'ютерів. При цьому основними перевагами будуть навіть не апаратні можливості, а наявність безлічі прикладів, безліч форумів і спільнот розробників, а також низька ціна.

Популярність Raspberry Pi пояснюється ще й тим, що сама плата спочатку створювалася як навчальна платформа для програмістів. Однак величезний інтерес та багатомільйонні продажі призвели до подальшого розвитку Raspberry Pi. Зараз на ринку присутні наступні модифікації Raspberry Pi: "A", "A+", "B", "B+", "2B", "Zero", "Zero W", "3B" та "3B+", "4".

Модель Raspberry Pi 4. стала однією з перших моделей сімейства, що отримала 64-бітний чотириядерний процесор Broadcom BCM2837B0 на базі ARM Cortex-A53 та робочою частотою 1,4 ГГц (див. рис. 3.1). Таким чином, для любителів ранніх моделей Raspberry Pi перехід на версію Raspberry Pi 4 може насамперед забезпечити приріст продуктивності.

Рисунок 3.1 – Raspberry Pi 4

Серед переваг Raspberry Pi 4 потрібно відзначити багатий набір комунікаційних інтерфейсів. По-перше, на платі є високошвидкісний порт Gigabit Ethernet. По-друге, для організації бездротових з'єднань відмінним рішенням стане використання вбудованого Wi-Fi 802.11b/g/n/ac. По-третє, для підключення мультимедійних та користувацьких пристроїв може використовуватись Bluetooth BLE 4.2 [20].

Основні характеристики Raspberry Pi 4 наведено нижче (див. табл. 3.1).

Таблиця 3.1 – Основні характеристики Raspberry Pi 4

GPU	Cortex-A72 @ 1.5GHz
CPU	Amlogic ARM® Cortex®-A53 (ARMv8) 1,5 Ghz quad core
Оперативна пам'ять	2Gbyte LPDDR4-2400 SDRAM
Сховище	microSD (Card not included)
Decode відео	H.265 4K/60FPS, H.264 1080/60FPS
Encode відео	H.264 1080/30FPS
Мережа	2.4 GHz and 5.0 GHz IEEE 802.11ac wireless, Bluetooth 5.0, BLE

Серед недоліків Raspberry Pi 4 слід відзначити відсутність вбудованої пам'яті – для роботи користувачеві знадобиться зовнішня картка пам'яті MicroSD. Обсяг ОЗП 1 Гбайт є досить скромним за мірками сучасних SBC, тому даний ком’ютер не підходить для реалізації інтелектуальної системи [21].

Найчастіше у різних оглядах плата ODROID-C2 протиставляється Raspberry Pi (див. рис. 3.2). Не важко помітити, що обидва одноплатні комп'ютери мають надзвичайно схожі характеристики та ідентичні розміри.

Рисунок 3.2 – Зовнішній вигляд одноплатного комп'ютера ODROID-C2

В ODROID-C2 використовується високопродуктивний 64-бітний чотириядерний процесор Amlogic S905 на базі ARM Cortex-A53 з робочою частотою 1,5 ГГц. Крім того, обсяг ОЗП у ODROID-C2 більший, ніж у Raspberry Pi, і становить 2 Гб. На борту у ODROID-C2 також, як і у Raspberry Pi, немає вбудованої Flash, проте можливе розширення пам'яті за рахунок MicroSD та eMMC.

Основні характеристики ODROID-C2 наведено нижче (див. табл. 3.2).

Таблиця 3.2 – Основні характеристики ODROID-C2

GPU	Mali ™ -450 GPU (3 Pixel-processors

	+ 2 Vertex shader processors)

Продовження таблиці 3.2

CPU	Amlogic ARM® Cortex®-A53 (ARMv8) 1,5 Ghz quad core
Оперативна пам'ять	2Gbyte DDR3 SDRAM
Сховище	microSD (Card not included)
Decode відео	H.265 4K/60FPS and H.264 4K/30FPS capable VPU
Encode відео	H.265 4K/60FPS and H.264 4K/30FPS capable VPU
Мережа	10/100/1000Mbps Ethernet with RJ-45

Багато користувачів відзначають, що порт Gigabit Ethernet у складі ODROID-C2 працює краще, ніж Raspberry Pi 4 Model. Однак у ODROID-C2 відсутня підтримка Wi-Fi та Bluetooth, тому це є не кращим варіантом для інтелектуальної системи.

NVIDIA Jetson Nano – мікрокомп'ютер, який буде корисний при розробці систем штучного інтелекту, оскільки забезпечує високу продуктивність (472 Гігафлопса) і компактність.

Рисунок 3.3 – NVIDIA Jetson Nano

Jetson Nano побудований на базі 64-розрядного чотирьох ядерного процесора Arm Cortex-A57, який працює на частоті 1,43 ГГц разом з графічним процесором NVIDIA Maxwell зі 128 ядрами CUDA, здатними виконувати 472 GFLOPs (FP16), і має 4 ГБ 64-розрядної оперативної пам’яті LPDDR4. з 16 Гб пам'яті eMMC і працює під керуванням Linux для Tegra . Модуль розміром 70×45 мм має 260-контактний роз’єм SODIMM, який роз’єднує інтерфейси, включаючи відео, аудіо, USB та мережу, і дозволяє підключати його до сумісної плати.

Основні характеристики Jetson Nano наведено нижче (див. табл. 3.3).

Таблиця 3.3 – Основні характеристики Jetson Nano

GPU	128-core NVIDIA Maxwell™ GPU
CPU	Quad-core ARM® A57 @ 1.43 GHz
Оперативна пам'ять	4 GB LPDDR4
Сховище	microSD (Card not included)
Decode відео	4K @ 30 | 4x 1080p @ 30 | 9x 720p @ 30 (H.264/H.265)

Encode відео	4K @ 60 | 2x 4K @ 30 | 8x 1080p @ 30 | 18x 720p @ 30 (H.264/H.265)
Мережа	Gigabit Ethernet, M.2 Key E

Серед іншого він підтримує Gigabit Ethernet, 4 × USB 3.0 роз’єми, порти HDMI і DisplayPort, роз’єм для камери MIPI-CSI, слот для карти micro SD і Power-over-Ethernet (PoE) [22].

Плата також пропонує GPIO, I2C, I2S, SPI, PWM і UART через 40-контактний блок заголовка GPIO, підозріло схожий на той, який використовується в Raspberry Pi.

«Комплект розробника Jetson Nano включає 40-контактний роз’єм, який з коробки сумісний з багатьма периферійними пристроями та іншими додатками. Налаштування GPIO за замовчуванням на Jetson Nano сумісні з багатьма додатковими модулями екосистеми та капелюхами введення в екосистему. Бібліотека Python Jetson GPIO дає вам повний контроль над заголовками вводу-виводу, щоб програмувати їх для різних модулів та інтерфейсів».

Очікується, що енергоспоживання плати становить від 5 до 10 Вт, і її можна живити від одного порту micro USB поруч із роз’ємом Ethernet.

NVIDIA рекомендує, щоб комплект Jetson Nano Dev Kit живився від джерела живлення мікро-USB від 5 В/2 А до 3,5 А з додатковою рекомендацією використовувати джерело живлення 5 В/4 А.

На щастя, на платі також є роз’єм, що означає, що ви можете живити її за допомогою «звичайного» джерела постійного струму і навіть не володіючи апаратним забезпеченням, я думаю, що це буде моєю рекомендацією. З огляду на всі проблеми з низькою напругою, які люди бачать у Raspberry Pi, ймовірно, не варто навіть намагатися йти шляхом мікро-USB.

Однією з цікавих речей про Jetson Nano є те, що це не скорочена версія інших плат NVIDIA і підтримує ряд різних фреймворків машинного навчання, включаючи TensorFlow, PyTorch, MxNet, Keras і Caffe.

Він також підтримує ту саму підтримку пакетів SDK NVIDIA JetPack і DeepStream, що й дорожчі – хоча й набагато більш потужні – плати TX та AGX Xavier.

Кінець закону Мура – це більше не просто передбачення , це факт, з яким галузь має справу вже багато років, і це спричинило зростання GPU загального призначення. Сьогодні GPU більше не призначений лише для графіки.

Можна сперечатися, що мирний дивіденд війн за смартфони був однією з речей, які спричинили рух виробників. Це новий дивіденд миру, наслідки битв навколо автономних транспортних засобів, які цілком могли б стимулювати машинне навчання на вбудованому апаратному забезпеченні та новий спеціальний кремній, який ми бачили від Intel і Google.

Порівняльна таблиця вищи представлених міні комп’ютерів наведена нижче (див. табл. 3.4).

Таблиця 3.4 – Порівняльна характеристика описаних міні комп’ютерів

	Jestson Nano	ODROID-C2	Raspberry Pi 4
GPU	128-core NVIDIA Maxwell™ GPU	Mali ™ -450 GPU (3 Pixel-processors + 2 Vertex shader processors)	Cortex-A72 @ 1.5GHz
CPU	Quad-core ARM® A57 @ 1.43 GHz	Amlogic ARM® Cortex®-A53 (ARMv8) 1,5 Ghz quad core	Amlogic ARM® Cortex®-A53 (ARMv8) 1,5 Ghz quad core
Оперативна пам'ять	4 GB LPDDR4	2Gbyte DDR3 SDRAM	2Gbyte LPDDR4-2400 SDRAM
Схови	microSD	microSD	microSD

ще	(Card not included)	(Card not included)	(Card not included)
Decode відео	4K @ 30 | 4x 1080p @ 30 | 9x 720p @ 30 (H.264/H.265)	H.265 4K/60FPS and H.264 4K/30FPS capable VPU	H.265 4K/60FPS, H.264 1080/60FPS
Encode відео	4K @ 60 | 2x 4K @ 30 | 8x 1080p @ 30 | 18x 720p @ 30 (H.264/H.265)	H.265 4K/60FPS and H.264 4K/30FPS capable VPU	H.264 1080/30FPS
Мережа	Gigabit Ethernet, M.2 Key E	10/100/1000 Mbps Ethernet with RJ-45	2.4 GHz and 5.0 GHz IEEE 802.11ac wireless, Bluetooth 5.0, BLE

Jetson Nano має більше оперативної пам’яті, кращий центральний процесор та дискретний відео процесор порівняно з конкурентами, тому він і буде використаний в проекті інтелектуальної системи для дослідження моделей машинного навчання.

3.2. Опис камери інтелектуальної системи

Модуль камери Raspberry Pi IMX219 (див. рис. 3.4) – це один із найпопулярніших модулів камер сімейства Raspberry Pi, на відміну від інших камер Raspberry Pi, які мають роздільну здатність 5 МП, цей модуль має 8 мега пікселів із сенсором зображення Sony IMX219 CMOS, якість зображення дуже висока та стабільна.

Рисунок 3.4 – Raspberry Pi IMX219

IMX219 – це CMOS-датчик зображення з діагоналлю 4,60 мм (тип 1/4,0) з активними пікселями з квадратним масивом пікселів і 8,08 млн ефективних пікселів. Цей чіп працює від трьох джерел живлення: аналогового 2,8 В, цифрового 1,2 В і ПЧ 1,8 В, а також має низьке енергоспоживання. Висока чутливість і відсутність мазка досягаються завдяки застосуванню R-пігментні мозаїчні фільтри основного кольору G і B. Цей чіп має електронний затвор із змінним накопичувачем заряду часу. Крім того, цей продукт призначений для використання в мобільних телефонах і планшетах.

Він здатний створювати статичні зображення 3280 x 2464 пікселів, а також підтримує відео 1080p@30fps, 720p@60fps і 640x480p@90fps. Він підключається за допомогою спеціального стандартного інтерфейсу CSI. Він є додатковим для офіційної камери Raspberry Pi, щоб задовольнити вимоги до різного кріплення об’єктива, поля зору (FOV) і глибини поля (DOF), а також моторизованого ІЧ-фільтра для денного і нічного бачення. Основні характеристики камери Raspberry Pi IMX219 представлені в таблиці 3.5.

Таблиця 3.5 – Основні характеристики камери Raspberry Pi IMX219

Модель	CMT-8MP-IMX219-M366
Датчик	1/4" SONY IMX219 CMOS
Кількість пікселів	8 мегапіксель

Найефективніші пікселі	3296 (В) x 2480 (В)
Розмір пікселя	1,12 x 1,12 мкм
Область зображення	5095 мкм (В) x 4930 мкм (В)
Тип фокусування	Фіксований фокус
Колір	RGB
Вхідна тактова частота	від 6 до 27 МГц
PGA	44,56 дБ (макс.)
Тип інтерфейсу	Стандартний інтерфейс CSI
Об'єктив FOV	70° і за бажанням

Камеру можна використовувати для багатьох додатків, таких як домашня безпека та спостереження, відеоспостереження, робототехніка, іграшки, система зору, детектор руху, уповільнена зйомка, IoT, широкомасштабна фотографія, сільське господарство NDVI та інші проекти камер Raspberry Pi тощо. Усе програмне забезпечення підтримується останньою версією операційної системи Linux [23].

3.3. Операційна система платформи

Ubuntu – це дистрибутив GNU/Linux, яким керує британська компанія Canonical, що базується на програмному забезпеченні. Сама Ubuntu базується на Debian, що означає, що він «будує на основі архітектури та інфраструктури Debian і широко співпрацює з розробниками Debian», згідно з веб-сайтом Ubuntu.

Окрім настільної версії (див. рис. 3.3), про яку йдеться в цьому огляді, Canonical також випускає версії Ubuntu [24] для хмарних, серверних та IoT-платформ. Ubuntu Touch, проект мобільної ОС з відкритим кодом, більше не управляється Ubuntu; спільнота UBports взяла на себе її розвиток.

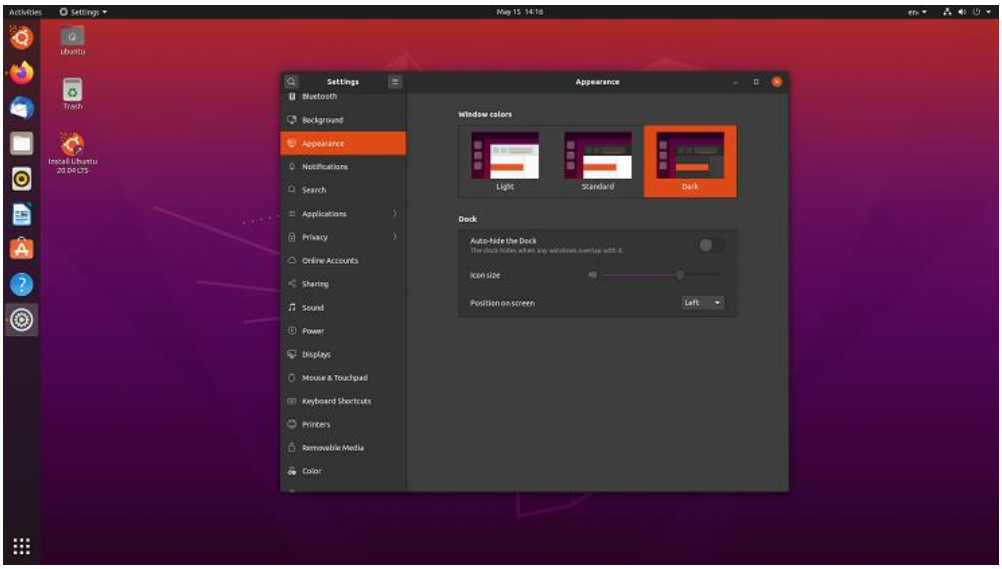

Рисунок 3.3 – Робочий стіл ОС Ubuntu

Для Ubuntu потрібен двоядерний процесор 2 ГГц або краще, 4 ГБ системної пам’яті (ОЗУ), 25 ГБ вільного місця на жорсткому диску, DVD-привід або порт USB для носія інсталятора та доступ до Інтернету (у більшості випадків). Це дещо змінилося в порівнянні з випуском 18.04, який вимагав лише 2 ГБ оперативної пам’яті. Існують полегшені версії ОС, які можна спробувати. У будь-якому випадку, ці вимоги не є необґрунтованими, і Ubuntu повинна нормально працювати навіть на недорогих машинах.

У більшості випадків ОС, яку використовують, пов’язана з вибраним обладнанням. Наприклад, щоб отримати macOS, потрібно купити комп’ютер Apple. Лише кілька пристроїв мають попередньо встановлену Ubuntu в порівнянні з іншими основними ОС. Одним з найкращих варіантів є Dell XPS 13 Developer Edition, але вона все ще поставляється з попередньо завантаженою версією 18.04. Dell також продає інші робочі станції та вежі з попередньо встановленим Ubuntu. Виробник System 76, орієнтований на Linux, продає моделі з попередньо встановленою останньою версією Ubuntu, а Think Penguin перераховує обладнання, яке підтримує оновлення 20.04. Також можна просто оновити Ubuntu до останньої версії на більшості комп’ютерного обладнання.

Безперечно, є перевага в тому, що драйвери програмного та апаратного забезпечення безперебійно працюють прямо з коробки. Пристрої Apple iMac і MacBook, лінійка Microsoft Surface і Pixelbook від Google – усі виграють від цієї тісної інтеграції. Хоча я не відчував жодних серйозних проблем під час використання 20.04 на моєму тестовому пристрої, ви повинні бути готові усунути неполадки та повозитися зі своєю системою на певному етапі процесу. Наприклад, периферійні пристрої можуть не працювати, обладнання може бути виявлено неправильно, або деякі файли можуть не відтворюватися за замовчуванням. Існують імовірні рішення для кожної проблеми, але найбільше розчаровує знайти це рішення.

Інші варіанти встановлення включають подвійне завантаження, запуск ОС на віртуальній машині (VM), наприклад, через програмне забезпечення VirtualBox від Oracle, або створення завантажувального USB-порту. Останній метод для людей, які хочуть випробувати ОС без її встановлення, оскільки він не залишає сліду в системі. Наприклад, цей метод не встановлює GRand Unified Bootloader (GRUB) назавжди у вашій системі. Крім того, Ubuntu надає чудове покрокове керівництво щодо створення завантажувального USB-порту з ОС.

Canonical також надає розробникам ще кілька способів використання інтерфейсу командного рядка Ubuntu. Одним з них є Multipass, попередньо запакована віртуальна машина, яку можна завантажити для Windows, macOS та Linux (у вигляді пакета Snap Package пакетного типу програми, який я обговорю пізніше). Інші два варіанти стосуються пристроїв з Windows 10. Можна інсталювати Ubuntu 20.04 з Microsoft Store або ввімкнути підсистему Windows для Linux. Усі ці параметри дають натовпу розробників легкодоступне середовище Ubuntu без необхідності постійно фіксувати системні ресурси, що вражає.

3.4. Підключення та налаштування камери

Сімейство Jetson завжди підтримувало камери MIPI-CSI. Цей протокол призначений для високошвидкісної передачі між камерами та хост-пристроями. В основному це прямий зв’язок процесора, там не так багато накладних витрат, як у випадку, наприклад, стека USB.

Однак для тих людей, які не є професійними розробниками апаратного та програмного забезпечення, отримати доступ до недорогих пристроїв обробки зображень через цей інтерфейс було, скажімо, складним завданням.

Це з кількох причин. По-перше, підключення камери та шлейфу здійснюється через роз’єм, до якого більшість любителів не мають належного доступу. Крім того, у ядрі Linux є багато змін із драйверами камери, а також маніпуляціями з деревом пристроїв, які мають відбутися до того, як станеться магія зображення.. Більшість людей йдуть шляхом найменшого опору і просто використовують USB-камеру.

У той же час однією з найпопулярніших камер CSI-2 є Raspberry Pi Camera. Камера має стрічковий роз'єм, який підключається до плати за допомогою простого роз'єму. По суті, камера RPi складається з фотокамери Sony IMX-219 і доступна в різних версіях з інфрачервоним фільтром і без нього.

Установка проста (див. рис. 3.4). На роз’ємі камери Jetson Nano J13 потрібно підняти шматок пластику, який утримує стрічковий кабель на місці. Після розслаблення потрібно вставити стрічковий кабель камери так, щоб контакти на кабелі були спрямовані всередину до модуля Nano. Потім натиснути на пластиковий язичок, щоб захопити стрічковий кабель.

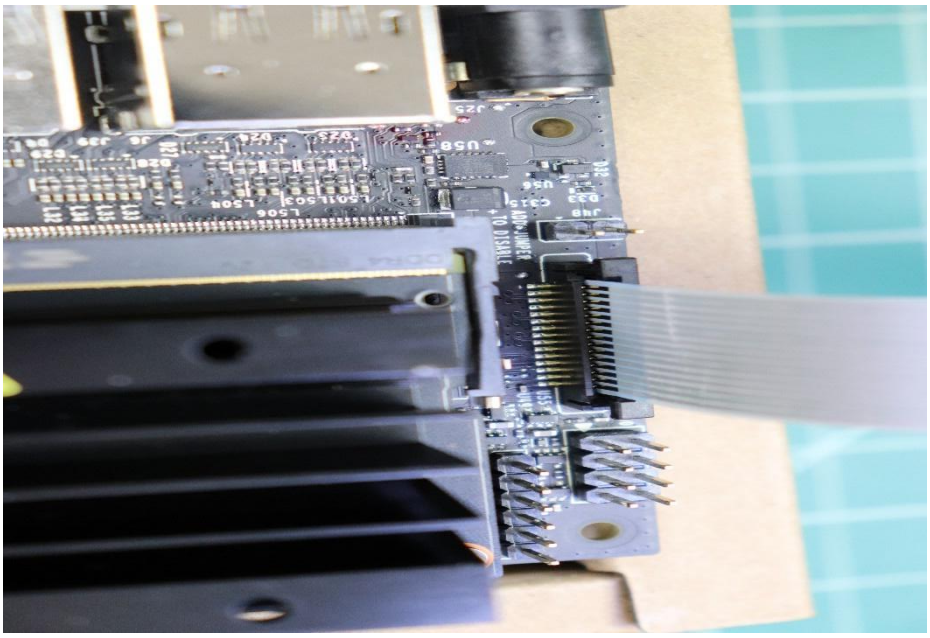

Рисунок 3.3 – Встановлений шлейф камери

Репозиторій CSI-Camera на Github містить деякий зразок коду для взаємодії з камерою. Після встановлення камера має відображатися на /dev/video0. У Jetson Nano GStreamer використовується для взаємодії з камерами. Ось простий командний рядок для перевірки камери:

```
$ gst-launch-1.0 nvarguscamerasrc ! nvoverlaysink
```

У новіших комплектах розробника Jetson Nano є два слоти для камер CSI. Можна використовувати атрибут sensor_mode з nvarguscamerasrc, щоб указати камеру. Допустимі значення — 0 або 1 (за замовчуванням — 0, якщо не вказано), тобто nvarguscamerasrc sensor_mode=0.

Більш конкретний приклад, який враховує фактичні режими роботи конкретного датчика:

$ gst-launch-1.0 nvarguscamerasrc sensor_mode=0 ! 'video/x-raw(memory:NVMM),width=3820, height=2464, framerate=21/1, format=NV12' ! nvvidconv flip-method=0 ! 'video/x-raw,width=960, height=616' ! nvvidconv ! nvegltransform ! nveglglessink -e

Це вимагає від GStreamer відкрити потік камери шириною 3820 пікселів і висотою 2464 при 21 кадрі на секунду і відобразити його у вікні

розміром 960 пікселів завширшки і 616 пікселів у висоту. «Метод перевертання» корисний, коли потрібно змінити орієнтацію камери.

3.5. Висновки до розділу 3

В розділі 3 проаналізовано 3 моделі однопалатних комп’ютерів, по всім характеристикам задовольняє для інтелектуальної системи міні-копютер від NVIDIA Jetson Nano – це невеликий потужний комп’ютер, який дозволяє запускати кілька нейронних мереж паралельно для таких програм, як класифікація зображень, виявлення об’єктів, сегментація та обробка мовлення. Все це на простій у використанні платформі, яка працює всього за 5 Вт.

Було описано характеристики використонаної камери Raspberry Pi IMX219 , дана камера є доволі потужною, що забезпечить вискоку роздільність зображення, а з’єднання за допомогою шлейфового роз’єму забезпечить швидку передачу зображення без значних затримок.

Також дану конфігурацію було підключено в єдине ціле. Встановлено операційну систему, а також налаштовано та перевірено працездатність камери.

РОЗДІЛ 4
ПРОГРАМНА РЕАЛІЗАЦІЯ ТА ТЕСТУВАННЯ

4.1. Програмні технології для інтегрування машинного навчання

4.1.1 Мова програмування Python

Великий вибір бібліотек є однією з головних причин, чому Python (див. рис. 4.1) є найпопулярнішою мовою програмування, що використовується для ШІ. Бібліотека – це модуль або група модулів, опублікованих різними джерелами, такими як PyPi, які містять попередньо написаний фрагмент коду, який дозволяє користувачам отримувати певні функції або виконувати різні дії. Бібліотеки Python надають елементи базового рівня, тому розробникам не доводиться кожен раз кодувати їх із самого початку.

Рисунок 4.1 – Логотип мови програмування Python

ML вимагає безперервної обробки даних, а бібліотеки Python дозволяють отримати доступ, обробляти та перетворювати дані. Ось деякі з найбільш поширених бібліотек, які ви можете використовувати для машинного навчання та AI:

− Scikit-learn для роботи з основними алгоритмами машинного навчання, такими як кластеризація, лінійні та логістичні регресії, регресія, класифікація та інші.

− Pandas для високорівневих структур даних і аналізу. Він дозволяє об’єднувати та фільтрувати дані, а також збирати їх із інших зовнішніх джерел, наприклад, Excel.

− Keras для глибокого навчання. Він дозволяє швидкі обчислення та створення прототипів, оскільки він використовує графічний процесор на додаток до ЦП комп’ютера.

− TensorFlow для роботи з глибоким навчанням шляхом налаштування, навчання та використання штучних нейронних мереж з масивними наборами даних.

− Matplotlib для створення двовимірних графіків, гістограм, діаграм та інших форм візуалізації.

− NLTK для роботи з обчислювальною лінгвістикою, розпізнавання природної мови та обробки.

− Scikit-image для обробки зображень.

− PyBrain для нейронних мереж, навчання без нагляду та навчання з підкріпленням.

− Caffe для глибокого навчання, що дозволяє перемикатися між процесором і графічним процесором і обробляти понад 60 мільйонів зображень на день за допомогою одного GPU NVIDIA K40.

− StatsModels для статистичних алгоритмів і дослідження даних.

У репозиторії PyPI можна знайти та порівняти більше бібліотек Python.

Python для машинного навчання – чудовий вибір, оскільки ця мова дуже гнучка:

− Пропонує можливість вибрати або використовувати ООП або сценарії.

− Також немає необхідності перекомпілювати вихідний код, розробники можуть впровадити будь-які зміни та швидко побачити результати.

− Програмісти можуть комбінувати Python та інші мови для досягнення своїх цілей.

Більше того, гнучкість дозволяє розробникам вибирати стилі програмування, які їм цілком зручні, або навіть комбінувати ці стилі для вирішення різних типів проблем найефективнішим способом.

− Імперативний стиль складається з команд, які описують, як комп’ютер повинен виконувати ці команди. За допомогою цього стилю визначають послідовність обчислень, які відбуваються як зміна стану програми.

− Функціональний стиль також називають декларативним, оскільки він оголошує, які операції слід виконувати. Він не враховує стан програми, порівняно з імперативним стилем, він оголошує висловлювання у вигляді математичних рівнянь.

− Об’єктно-орієнтований стиль ґрунтується на двох поняттях: клас і об’єкт, де подібні об’єкти утворюють класи. Цей стиль не повністю підтримується Python, оскільки він не може повністю виконати інкапсуляцію, але розробники все ще можуть використовувати цей стиль до певної міри.

− Процедурний стиль є найпоширенішим серед новачків, оскільки виконує завдання в покроковому форматі. Він часто використовується для послідовності, ітерації, модульності та вибору.

Фактор гнучкості зменшує ймовірність помилок, оскільки програмісти мають шанс взяти ситуацію під контроль і працювати в комфортних умовах.

Python не тільки зручний у використанні та простий у освоєнні, але й дуже універсальний. Ми маємо на увазі, що Python для розробки машинного навчання може працювати на будь-якій платформі, включаючи Windows, MacOS, Linux, Unix та ще двадцять одну. Щоб перенести процес з однієї платформи на іншу, розробникам необхідно внести кілька невеликих змін і змінити деякі рядки коду, щоб створити виконувану форму коду для вибраної платформи. Розробники можуть використовувати такі пакети, як PyInstaller , щоб підготувати свій код до роботи на різних платформах.

Знову ж таки, це економить час і гроші на тестування на різних платформах і робить загальний процес більш простим і зручним.

Python дуже легко читається, тому кожен розробник Python може зрозуміти код своїх однолітків і змінити, копіювати або поділитися ним. Немає плутанини, помилок чи суперечливих парадигм, і це веде до більш ефективного обміну алгоритмами, ідеями та інструментами між професіоналами AI та ML.

Доступні також такі інструменти, як IPython, який є інтерактивною оболонкою, яка надає додаткові функції, такі як тестування, налагодження, завершення вкладок тощо, і полегшує робочий процес.

Python пропонує багато функцій, які є корисними, зокрема, для AI та ML, і це робить його найкращою мовою для цих цілей. Не дивно, що різні галузі використовують Python для передбачень та інших завдань машинного навчання [25].

4.1.2 Бібліотека для роботи із зображенням OpenCV

OpenCV – чудовий інструмент для обробки зображень і виконання завдань комп’ютерного зору (див. рис. 4.2). Це бібліотека з відкритим кодом, яку можна використовувати для виконання таких завдань, як виявлення обличчя, відстеження заперечень, виявлення орієнтирів та багато іншого. Вона підтримує кілька мов, включаючи python, java C++.

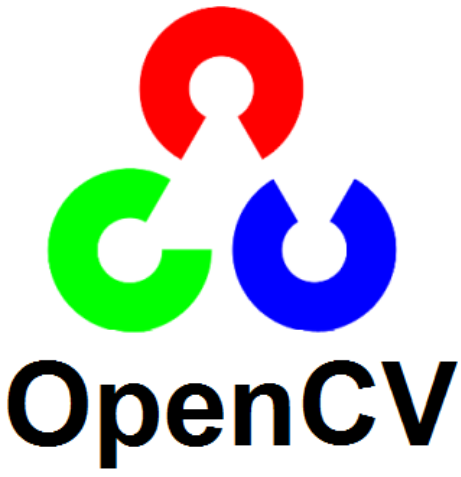

Рисунок 4.2 – Логотип бібліотеки OpenCV

Бібліотека оснащена сотнями корисних функцій і алгоритмів, які є у вільному доступі. Деякі з цих функцій дійсно поширені і використовуються майже в кожній задачі комп’ютерного зору. Тоді як багато функцій досі не вивчені і їм ще не приділено особливої уваги.

Водяні знаки можуть бути дійсно дратівливими, і в житті кожного настає момент, коли шукаємо по всьому Інтернету, щоб видалити цей маленький шматочок тексту. На щастя, на цьому пошуки закінчуються. Використовуючи OpenCV, можна виконати це завдання менш ніж 10 рядками коду.

Починаємо з імпортування необхідних бібліотек (тобто OpenCV і Numpy) і читання зображення, що містить водяний знак. Потім змінюємо колірний простір з BGR на HSV. Використовуємо колірний формат HSV, оскільки він допомагає отримати краще зображення маски, яке створимо на наступному кроці. Маска – це зображення, яке має лише два значення – чорне та біле. Призначення маски – застосувати зміни лише до окремих частин зображення. Для нашої мети зображення маски повинно містити білі значення всюди, де є водяний знак, і чорні скрізь.

Для створення зображення маски будемо використовувати порогове значення, яке вже виконав для цього зображення. Порогове значення визначає, чи знаходиться значення в заданому діапазоні чи ні. Якщо хочете використовувати інше зображення, можна знайти власні граничні значення, використовуючи цей фрагмент коду. Тепер є зображення маски, можемо викликати головну функцію, необхідну для виконання цього завдання. Будемо використовувати функцію cv2.inpaint() на нашому зображенні маски, щоб видалити водяний знак. Насправді функція дозволяє видалити невеликі шуми із зображення за допомогою методу, який називається inpainting.

Функцію можна використовувати з двома алгоритмами. Один заснований на маршовому методі, а інший - на механіці рідини. На щастя, не

потрібно писати ці алгоритми самостійно. Можемо включити об’єкт прапора всередині функції cv2.inpaint() і вибрати будь-який алгоритм, який хочемо використовувати.

Рисунок 4.3 – Створення зображення маски OpenCV

Перевірявши результати на зображенні вище (див. рис. 4.3). Функція малювання спрацювала добре. Якщо вважаєте, що потрібні кращі результати, можна спробувати налаштувати порогові значення. Також можна використовувати такі методи, як розмиття за Гауссом, щоб підвищити якість зображення маски.

Під час роботи із зображеннями фоновий шум дуже поширений. Використовуючи цю функцію OpenCV [26], можна витягувати з зображення лише корисну інформацію та відкидати все, що є у фоновому режимі. Добре те, що він дає змогу ввести форму виходу кінцевого зображення, щоб не доводилося здогадуватися, який буде розмір зображення.

Завдання – видалити фон і витягти передній план (блокнот) із зображення (див. рис. 4.4). Функція, яку будемо використовувати, вимагає двох наборів точок. Перший набір міститиме 4 точки, які охоплюватимуть область переднього плану. На зображенні вище, якщо ви придивитесь, можна побачити 4 точки. Перший набір буде координатами цих точок. Інший набір точок буде межами нашого вихідного зображення. Після цього ми скористаємося функцією cv2.getPerspective(), яка наддасть вихідну матрицю, а функція cv2.warpPerspective перетворить цю матрицю в потрібну форму.

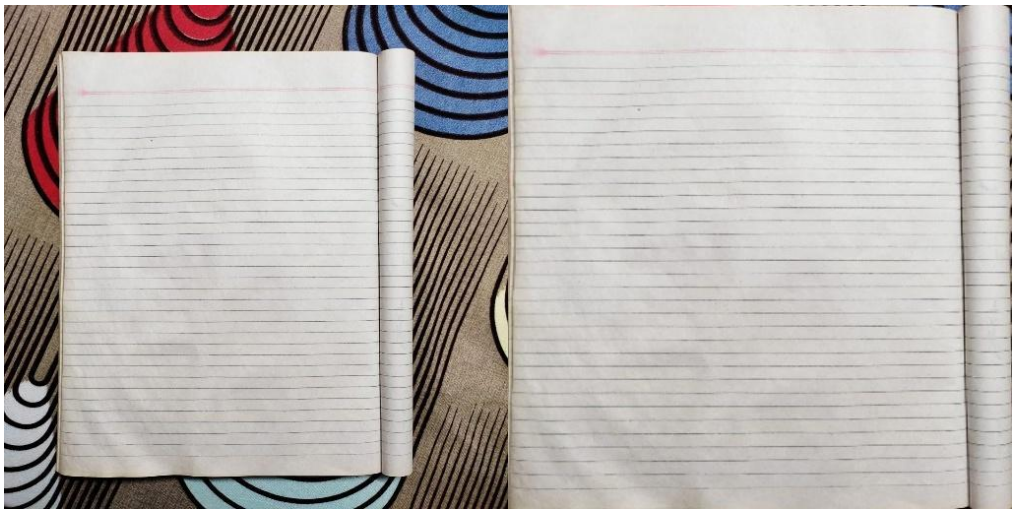

Рисунок 4.4 – Видалення фону за допомогою OpenCV

На відміну від текстових даних, зображення можна використовувати для вилучення кількох функцій. Ці особливості підкреслюють властивості зображення. Можна використовувати ці функції для розмиття зображення, підвищення різкості зображення, виділення країв із зображення та багато іншого. Для виконання цього завдання будемо використовувати поняття згортки. Згортка – це в основному процес застосування ядра до кожного пікселя на зображенні. Ядро – це невелика матриця, що складається з чисел, які зазвичай доводять до одиниці. Залежно від значень ядра можна витягти кілька функцій із зображення.

У цьому завданні будемо просто розмивати зображення, використовуючи згортки. Будемо використовувати фільтр (ядро). Це ядро також називають фільтром усереднення. Як випливає з назви, він усереднює навколишні пікселі та призначає значення центральному пікселю. Навіть можна вибрати інший набір значень, який до вподоби, просто переконатися, що сума всіх значень дорівнює 1.

Після прочитання зображення створимо наше ядро і створимо з нього масив NumPy. Потім включимо функцію cv2.filter2d(), яка візьме вхідне зображення, глибину та ядро.

OpenCV є важливою частиною спільноти комп’ютерного зору, і за допомогою нього ми можемо створювати тисячі дивовижних програм. Деякі з цих програм використовуються нами в повсякденному житті.

4.1.3 Бібліотека NumPy

NumPy, що розшифровується як Numerical Python (див. рис. 4.5), є бібліотекою, що складається з багатовимірних об’єктів масиву та набору підпрограм для обробки цих масивів. За допомогою NumPy можна виконувати математичні та логічні операції над масивами [27].

Рисунок 4.5 – Логотип бібліотеки NumPy

Numeric , предок NumPy, був розроблений Джимом Хугуніном. Також був розроблений інший пакет Numarray, який має деякі додаткові функції. У 2005 році Тревіс Оліфант створив пакет NumPy, включивши функції Numarray у пакет Numeric.

Використовуючи NumPy, розробник може виконувати такі операції:

− Математичні та логічні операції над масивами.

− Перетворення Фур’є та підпрограми для маніпуляції з формою.

− Операції, пов’язані з лінійною алгеброю. NumPy має вбудовані функції для лінійної алгебри та генерації випадкових чисел.

NumPy часто використовується разом із такими пакетами, як SciPy (Scientific Python) і Matplotlib (бібліотека графіків). Ця комбінація широко використовується як заміна MatLab, популярної платформи для технічних

обчислень. Однак Python, альтернатива MatLab, тепер розглядається як більш сучасна і повна мова програмування.

Найважливішим об’єктом, визначеним у NumPy, є N-вимірний тип масиву, який називається ndarray. Він описує колекцію предметів одного типу. Доступ до елементів колекції можна отримати за допомогою індексу з нульовим значенням.

Кожен елемент у ndarray має той самий розмір, що й блок у пам’яті. Кожен елемент у ndarray є об’єктом об’єкта типу даних (званого dtype).

Будь-який елемент, витягнутий з об’єкта ndarray (зрізом), представлений об’єктом Python одного із скалярних типів масиву.

Примірник класу ndarray можна створити за допомогою різних процедур створення масивів. Базовий ndarray створюється за допомогою функції масиву в NumPy наступним чином: numpy.array

Він створює ndarray з будь-якого об’єкта, який відкриває інтерфейс масиву, або з будь-якого методу, який повертає масив.

numpy.array(object, dtype = немає, copy = True, order = None, subok = False, ndmin = 0)

Об'єкт ndarray складається з безперервного одновимірного сегмента пам'яті комп'ютера, об'єднаного зі схемою індексування, яка відображає кожен елемент у місце в блоці пам'яті. Блок пам'яті містить елементи в порядку рядків (стиль C) або великого порядку стовпців (стиль FORTRAN або MatLab).

Схожою по логіці робити є бібліотека pandas. Нижче наведено порівняльну характеристику цих бібліотек (див. табл. 4.1).

Таблиця 4.1 – Порівняльна характеристика NumPy і Pandas

NumPy	Pandas
Numpy створює n-вимірний об'єкт масиву.	Pandas створюють DataFrame і Series.
Масив Numpy містить	Pandas добре підходить для

дані однакових типів	табличних даних

Продовження таблиці 4.1.

Numpy вимагає менше пам'яті	Pandas вимагало більше пам’яті в порівнянні з NumPy
NumPy підтримує багатовимірні масиви.	Pandas підтримують двовимірні масиви

4.2. Реалізація моделей машинного навчання на базі платформи Jetson Nano

Розпізнавання обличь – це процес виявлення обличь із зображення чи відео, який не має значення. Програма не робить нічого іншого, крім пошуку облич. Але з іншого боку, розпізнавання обличь, програма, яка знаходить обличчя, а також може визначити, яке обличчя кому належить. Тому це більше інформації, ніж просто їх виявлення.

Щоб написати код, який розпізнає обличчя, потрібні деякі навчальні дані, навчити машину, щоб вона знала обличчя та хто вони. У цьому розділі реалізуємо дві моделі машинного навчання під наглядом і без нагляду. В навчанні з наглядом програм буде розпізнавати заданні обличчя, а без нагляду буде шукати обличчя на відео з камера, таким чином навчатись, щоб розпізнати модель.

Отож реалізуємо ці дві моделі мишиного навчання на платформі від компанії NVIDIA Jetson Nano.

4.2.1 Реалізація машинного навчання з наглядом

Для цього проекту використаємо два основних модулі, які називаються Face Recognition і OpenCV. OpenCV – це високо оптимізована бібліотека з акцентом на додатках реального часу.

На початку потрібно встановити деякі бібліотеки, щоб програма працювала. Ось список бібліотек, які встановимо: cmake, face_recognition,

numpy, opencv-python. Cmake є обов’язковою бібліотекою, щоб установка бібліотеки розпізнавання обличчя не давала помилок.

Їх можна встановити в один рядок за допомогою менеджера бібліотек PIP:

pip install cmake face_recognition numpy opencv-python.

Після завершення інсталяції бібліотек імпортуємо їх у наш редактор коду. Деякі з цих бібліотек включені в Python, тому можна імпортувати їх, не встановлюючи їх (див. рис. 4.6).

```
import face_recognition
import cv2
import numpy as np
import time
```

Рисунок 4.6 – Імпортування бібліотек

Потрібно створити проект за допомогою компілятора коду (IDE) Pycharm для мови програграмування Python.

Тепер переходимо до наступного кроку, де будемо імпортувати зображення та використовувати їх для навчання нашої програми В цьому проекті створити директорію для зберігання зображень (див. рис. 4.7), які будуть використовуватися в якості моделей розпізнавання обличчя.

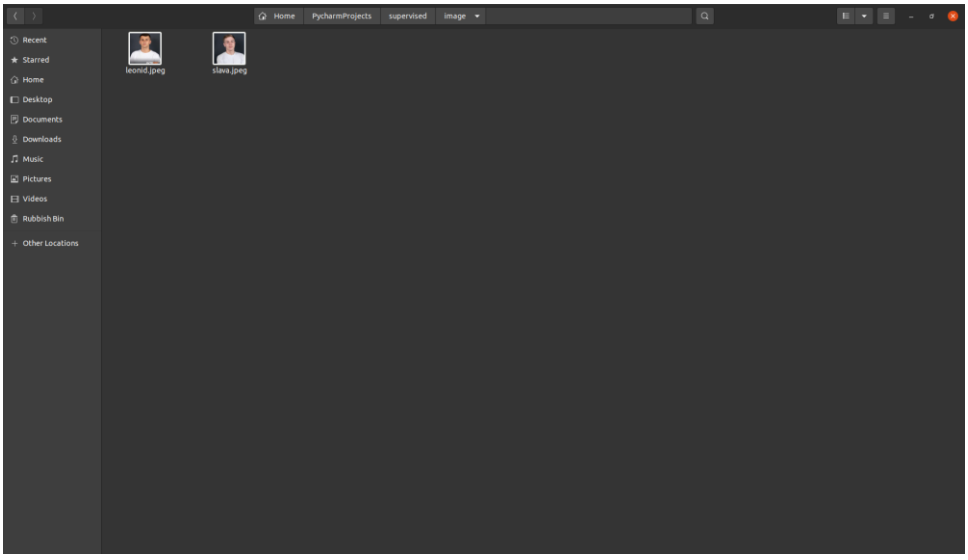

Рисунок 4.7 – Директорія для зберігання зображень

Тепер перейдемо до наступного кроку, де імпортуємо зображення та використаємо їх для навчання програми (див. рис. 4.8).

```
slava_image = face_recognition.load_image_file("image/slava.jpeg")
slava_face_encoding = face_recognition.face_encodings(slava_image)[0]
leonid_image = face_recognition.load_image_file("image/leonid.jpeg")
leonid_face_encoding = face_recognition.face_encodings(leonid_image)[0]
known_face_encodings = [
    slava_face_encoding,
    leonid_face_encoding
]
known_face_names = [
    "Slava Osipov",
    "Leonid Horburov"
]
```

Рисунок 4.8 – Імпорт зображень

Сутність наведених вище рядків:

– Усі зображення знаходяться в одній папці під назвою «image».

– Назви файлів зображень мають відповідати імені людини на зображенні. (наприклад: leonid.jpeg).

– Назви файлів перераховуються та призначаються змінній «known_face_names».

– Типи файлів мають бути однаковими. У даному випадку викоритаэмо формат «jpeg».

Тепер потрібно почати тренування програми вхідними зображеннями. Після цього за допомогою бібліотеки openCV взяти потік зображення з камери виконавши заданий код: video_capture = cv2.VideoCapture(0).

Далі програма буду порівнювати енкодоване зображення імпортоване з директорії та вхідне зображення в режимі реального часу:

matches = face_recognition.compare_faces(known_face_encodings, face_encoding).

Порівнявши ці зображення з вхідним масивом програма виведе на екран ім’я та ініціали в червоному квадраті навколо обличчя (див. рис. 4.9).

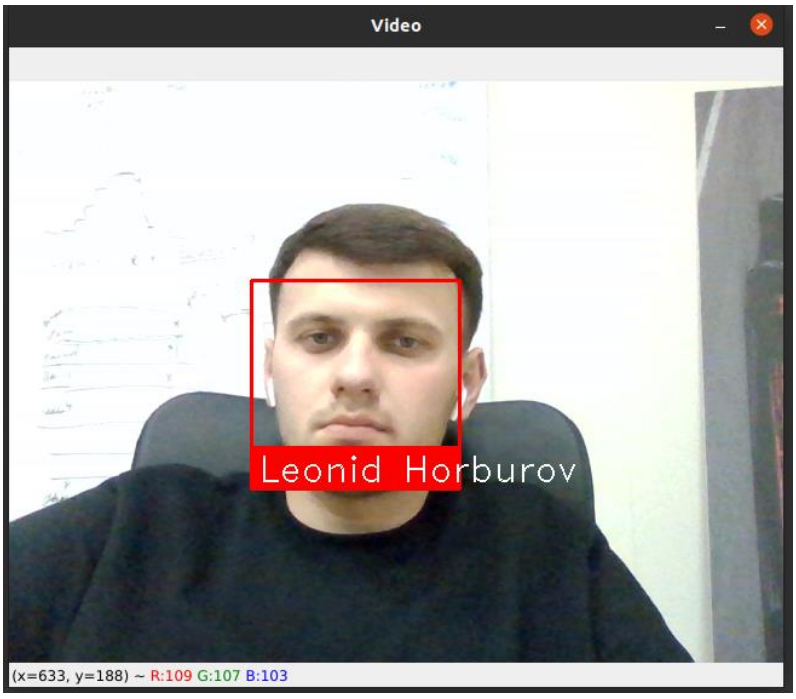

Рисунок 4.9 – Співпадіння обличчя з імпортованим зображенням

Представивши на вхідному зображенні з камери обличчя іншої людина програма повинна в червоному квадраті вказати, що це обличчя є невідомим, а саме вивести на екран “Unknown”, що і наведено на рисунку 4.10. Також програма може одразу розпізнавати декілька обличь на одному вхідному зображенні з камери.

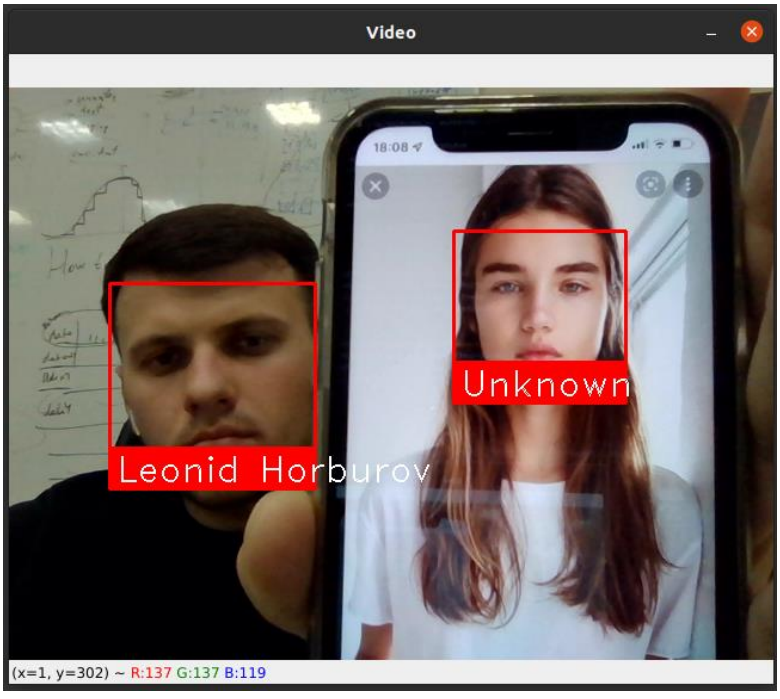

Рисунок 4.10 – Демонстрація реакції на невідоме обличчя

Отож було створено програму, яка розпізнає обличчя на зображенні. Тепер маємо уявлення про те, як використовувати комп’ютерний зір у реальному проекті.

4.2.2 Реалізація машинного навчання без нагляду

Алгоритм k-середніх є одним з найбільш базових алгоритмів класифікації в машинному навчанні. Він відноситься до категорії машинного навчання з навчанням без нагляду. k-середніх часто використовується в пошукових програмах, де шукають «подібні» елементи. Вимірюють подібність шляхом створення векторного представлення елементів, а потім порівнюють вектори за допомогою відповідної метрики відстані (наприклад, евклідову відстань).

Зазвичай він використовується для аналізу даних, розпізнавання образів, рекомендаційних систем і виявлення вторгнень.

Реалізуємо модель машинного навчання без нагляду за допомогою програмних ресурсів мови програмування Python.

Для початку імпортуємо бібліотеки які знадобляться в цьому проекті:

1. OpenCV2;
2. Pandas;
3. Numpy;
4. Scikit-learn.

Scikit-learn надає цілий ряд контрольованих і неконтрольованих алгоритмів навчання через узгоджений інтерфейс на Python. Ця бібліотека побудована на основі SciPy, який має бути встановлений на присторях, щоб використовувати scikit_learn.

В проекті використаємо набір даних, який легко доступний в Інтернет імпортуємо його в проект:

classifier = cv2.CascadeClassifier("people_face.xml").

Запустивши код який наведено в додадку Б. Потрібно зпочатку в консолі ввести ім’я персони яка буде користуватися додатком. Далі з’явиться

два вікна з повноцінним зображенням з камери, а також вікно із зображенням обличчя знайденому на основному екрані програми (див. рис. 4.11).

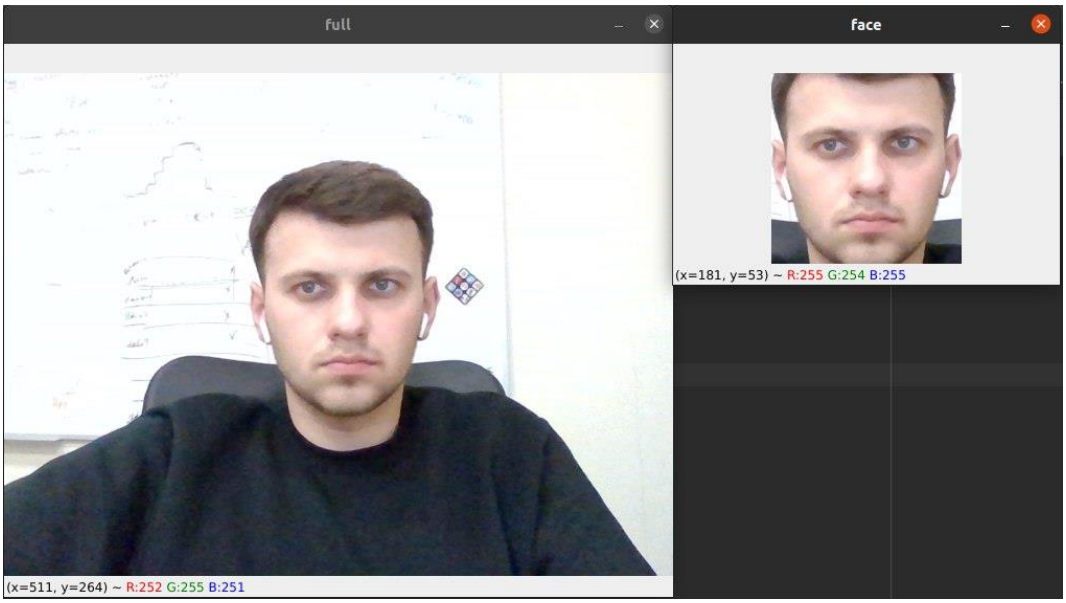

Рисунок 4.11 – Демонстрація розпізнавання образу обличчя

У разі якщо на вхідному зображені з камери відсутнє обличчя, відповідно в консоль буде виведено “Not Found”.

Після кожного завершення роботи програми всі зчитані моделі зберігаються в файлі з розширенням .csv (див. рис. 4.12).

Рисунок 4.12 – Збережені дані моделей обличчя в форматі .csv

Дана конфігурація дасть змогу в подальшому більш швидше розпізнавати образ обличчя, тобто модель машинного навчання буде навчатися сама з власне отриманих даних.4.3. Тестування та порівняння методів машинного навчання

4.3. Тестування та порівняння методів машинного навчання

Навчання з наглядом вивчає навчальний набір даних шляхом ітераційного прогнозування даних і коригування для правильної відповіді. Контрольовані методи мають справу з позначеними даними, де шаблони вихідних даних відомі системі.

Це робить моделі навчання з наглядом точнішими, ніж моделі навчання без нагляду, оскільки очікуваний результат відомий заздалегідь.

Моделі навчання без нагляду працюють самостійно, щоб виявити притаманну структуру даних без міток. Алгоритм навчання без нагляду працює з немаркованими даними, вихідні дані яких базуються виключно на зборі сприйняття.

Головним завдання магістерської роботи було проаналізувати та протестувати ефективність роботи кожної моделі машинного навчання.

За допомогою бібліотеки Python під назвою time заміряємо швидкість розпізнавання обличчя. Для цього потрібно вставити наступні рядки на початку алгоритму розпізнавання: current_time = round(time.time() * 1000) та в кінці, коли програма розпізнала обличчя: print(round(time.time() * 1000) - current_time, "ms").

Тепер запустимо програму в якій використовується модель машиного навчання з наглядом та в консолі (див. рис. 4.13) режимі реального часу побачимо скільки затрачено часу на розпізнавання образу обличчя.

Рисунок 4.13 – Вивід в консоль часу розпізнавання обличчя з наглядом

Як бачимо для розпізнавання моделі в режимі реального часу з камери потрібно в середньому 81 мс.

Далі запестимо програму з моделлю машинного навчання з без нагляду. Отримаємо результат в консоль (див. рис. 4.14).

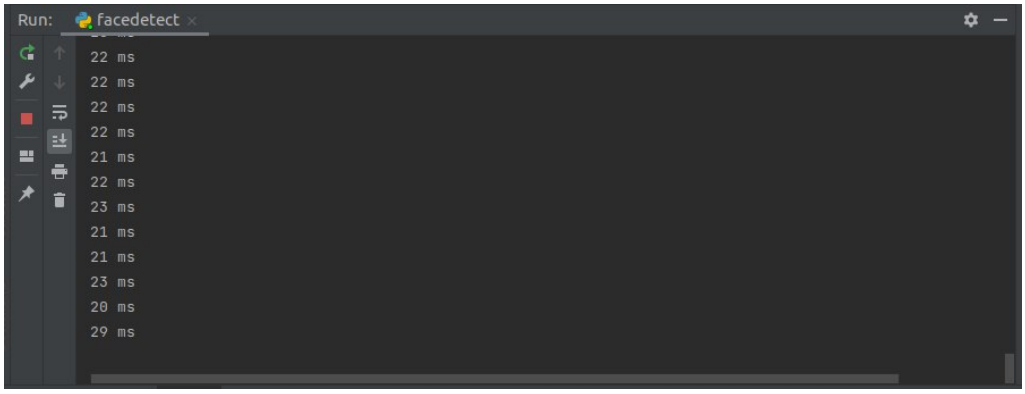

Рисунок 4.13 – Вивід в консоль часу розпізнавання обличчя без нагляду

В середньому для розпізнавання образу обличчя з моделлю машинного навчання без нагляду потрібно 22 мс.

На основі цих результатів можна зробити висновки:

− Контрольоване навчання працює за допомогою добре позначеного набору даних, у якому цільовий результат добре відомий.

− Контрольоване навчання має механізм зворотного зв’язку.

− Кероване навчання можна далі розділити на задачі класифікації та проблеми регресії. У класифікації вихідна змінна є категоричною, тоді як для регресії вихідна змінна є дійсним або безперервним значенням.

4.4. Висновки до розділу 4

В розділі 4 було розглянуто програмні компоненти для машинного навчання. Мова програмування Python є найбільш розвинутою, а також адаптованою завдяки внутрішніх та сторонніх бібліотек, тому вона була використана в проекті. Також для обробки зображення використано бібліотек openCV, дана бібліотека має багато функція для обробки зображення. В якості пришвидшення розрахунків було викристано бібліотеку Numpy.

Реалізовано розпізнавання обличь з використанням моделей машинного навчанням з наглядом там без нагляду. Продемонстровано роботу програм, а також описані основні нюанси налаштування та її роботи.

Протестовано розроблені моделі, шляхом заміру швидкості роботи алгоритмів розпізнання в режимі реального часу.

Отож модель навчання з наглядом є біль простіша в реалізації, але програє по часу розпізнаванню моделі без нагляду. Загалом ці моделі доречно використовувати в конкретних цілях, але ні як в однакових.

ВИСНОВКИ

В ході виконання магістерської роботи було реалізовано інтелектуальну систему для дослідження моделей машинного навчання на базі платформи Jetson Nano. Апаратною частиною є одно платний комп’ютер NVIDIA Jetson Nano та камера Raspberry Pi IMX219. Це обумовлено тим що машине навачання є доволі складним процесом та потребує великих ресурсів для обробки зображення.

Розглянуто моделі машинного навчання, а також реалізовано на базі платформи Jetson Nano розпізнавання обличь з наглядом та безнагляду.

Створено програмно частину за допомогою мови програмування Python, а також бібліотеки для обробки зображення openCV та бібліотеки для роботи з багатовимірними масивами NumPy.

В ході роботи створенні моделі машинного навчання були протестовані на швидкість розпізнавання образів. Як показали дослідження модель машинного навчання без нагляду працює, більш швидше та оптимізованіше, завдяки алгоритмів кластиризації, а саме на 50 мс порівняно з моделлю без нагляду. Це зумовлено тим, що модель постійно навчається за допомогою аналізу даних насамперед не з статичних, а з динамічних, які постійно оновлюються, шляхом отримання даних з камери.

Розроблена інтелектуальна система зможе замінити людину, а також автоматизувати процеси в будь-якій сфері діяльності завдяки моделей та алгоритмів машинного навчання.

Також розроблено питання з охорони праці. Проаналізовано умови праці в лабораторії кафедри ЧНУ ім. Петра Могили, в якому відбувалося тестування програмно-апаратного комплексу. Задля покращення умов праці в приміщенні, було запропоновано встановити лампи більшої потужності.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

Л. М. Горбуров, «Інформаційні технології та,» в *Особливості проектування інтелектуальних систем для дослідження нейронних мереж*, Миколаїв, 2022.

[1]

«Machine Learning, ML,» 2019. [Онлайновий]. Available: https://www.it.ua/knowledge-base/technology-innovation/machine-learning. [Дата звернення: 07 02 2022].

[2]

«Класичне машинне навчання, або Classical Machine Learning,» 2020. [Онлайновий]. Available: https://evergreens.com.ua/ua/articles/classical-machine-learning.html. [Дата звернення: 07 02 2022].

[3]

В. М.В., «Навчання машин та штучний інтелект,» *ДВНЗ «ПДТУ»,* рр. 39-41, 2019.

[4]

«Свідомість і мова. Проблема штучного інтелекту.,» 2021. [Онлайновий]. Available: https://studopedia.com.ua/1_10071_svidomist-i-mova-problema-shtuchnogo-intelektu.html. [Дата звернення: 07 02 2022].

[5]

«Штучний інтелект — революція, надія чи утопія?,» 2019. [Онлайновий]. Available: https://www.imena.ua/blog/ai-revolution/. [Дата звернення: 07 02 2022].

[6]

«Existential risk from artificial general intelligence,» 2017. [Онлайновий]. Available: https://uk.wikisla.ru/wiki/Existential_risk_from_artificial_general_intelligence. [Дата звернення: 07 02 2022].

[7]

«Types Machine Learning,» 2022. [Онлайновий]. Available: https://uk.myservername.com/types-machine-learning. [Дата звернення: 07 02 2022].

[8]

«Unsupervised learning,» 2019. [Онлайновий]. Available:

9] https://www.techtarget.com/searchenterpriseai/definition/unsupervised-learning. [Дата звернення: 07 02 2022].

«Навчання з підкріпленням,» 2016. [Онлайновий]. Available:
10] https://znaimo.com.ua. [Дата звернення: 07 02 2022].

«About Fritzing,» 2022. [Онлайновий]. Available: https://fritzing.org/.
11] [Дата звернення: 07 02 2022].

«Лінійне регресійне моделювання,» 2022. [Онлайновий]. Available:
12] https://uk.education-wiki.com/8949275-linear-regression-modeling. [Дата звернення: 07 02 2022].

«Logistic Regression,» 2017. [Онлайновий]. Available:
13] https://wiki.loginom.ru/articles/logistic-regression.html. [Дата звернення: 07 02 2022].

«ОБЗОР ИНСТРУМЕНТА ДЛЯ АВТОМАТИЗАЦИИ ЛИНЕЙНО-
14] ДИСКРИМИНАНТНОГО АНАЛИЗА,» 2017. [Онлайновий]. Available: https://www.sciencehunter.net/Handbook/LDA. [Дата звернення: 07 02 2022].

«Naive Bayes Classifier From Scratch in Python,» 2019. [Онлайновий].
15] Available: https://machinelearningmastery.com/naive-bayes-classifier-scratch-python/. [Дата звернення: 07 02 2022].

«K-Means Clustering Algorithm,» 2019. [Онлайновий]. Available:
16] https://www.javatpoint.com/k-means-clustering-algorithm-in-machine-learning. [Дата звернення: 07 02 2022].

«Root-finding algorithms,» 2019. [Онлайновий]. Available:
17] https://uk.wikinoe.ru/wiki/root-finding_algorithms. [Дата звернення: 07 02 2022].

«МЕТОД ЛОКАЛЬНОГО СПУСКА,» 2019. [Онлайновий].
18] Available: https://studref.com/549943/informatika/metod_lokalnogo_spuska. [Дата звернення: 07 02 2018].

«Класифікація методів оптимізації,» 2015. [Онлайновий]. Available: 19] https://dl.khadi.kharkov.ua/mod/page/view.php?id=106601. [Дата звернення: 07 02 2022].

«Raspberry Pi 4,» 2022. [Онлайновий]. Available: 20] https://www.raspberrypi.com/products/raspberry-pi-4-model-b/. [Дата звернення: 07 02 2022].

«ODROID-C2,» 2019. [Онлайновий]. Available: 21] https://wiki.odroid.com/odroid-c2/odroid-c2. [Дата звернення: 07 02 2022].

«NVIDIA JETSON NANO,» 2022. [Онлайновий]. Available: 22] https://www.nvidia.com/ru-ru/autonomous-machines/embedded-systems/jetson-nano/. [Дата звернення: 07 02 2022].

«8MP Raspberry Pi Camera Module with SONY IMX219 sensor,» 2020. 23] [Онлайновий]. Available: http://www.camera-module.com/product/raspberrypicameramod/8mp-raspberry-pi-camera-module-sony-imx219-sensor.html. [Дата звернення: 07 02 2022].

«Ubuntu,» 2022. [Онлайновий]. Available: https://ubuntu.com/. [Дата 24] звернення: 07 02 2022].

«Python For Beginners,» 2022. [Онлайновий]. Available: 25] https://www.python.org/about/gettingstarted/. [Дата звернення: 07 02 2022].

«OpenCV,» 2022. [Онлайновий]. Available: https://opencv.org/. [Дата 26] звернення: 07 02 2022].

«NumPy Introduction,» 2022. [Онлайновий]. Available: 27] https://www.w3schools.com/python/numpy/numpy_intro.asp. [Дата звернення: 07 02 2022].

ДОДАТОК А

Лістинг коду навчання під наглядом:

```
import face_recognition
import cv2
import numpy as np
import time
video_capture = cv2.VideoCapture(0)
slava_image = face_recognition.load_image_file("image/slava.jpeg")
slava_face_encoding = face_recognition.face_encodings(slava_image)[0]
leonid_image = face_recognition.load_image_file("image/leonid.jpeg")
leonid_face_encoding = face_recognition.face_encodings(leonid_image)[0]
known_face_encodings = [
    slava_face_encoding,
    leonid_face_encoding
]
known_face_names = [
    "Slava Osipov",
    "Leonid Horburov"
]
face_locations = []
face_encodings = []
face_names = []
process_this_frame = True
while True:
    ret, frame = video_capture.read()
    small_frame = cv2.resize(frame, (0, 0), fx=0.25, fy=0.25)
    rgb_small_frame = small_frame[:, :, ::-1]
    if process_this_frame:
```

```
            current_time = round(time.time() * 1000)
            face_locations = face_recognition.face_locations(rgb_small_frame)
            face_encodings = face_recognition.face_encodings(rgb_small_frame, face_locations)
            face_names = []
            for face_encoding in face_encodings:
                matches = face_recognition.compare_faces(known_face_encodings, face_encoding)
                name = "Unknown"
                face_distances = face_recognition.face_distance(known_face_encodings, face_encoding)
                best_match_index = np.argmin(face_distances)
                if matches[best_match_index]:
                    name = known_face_names[best_match_index]
                face_names.append(name)
                print(round(time.time() * 1000) - current_time, "ms")
        process_this_frame = not process_this_frame
        for (top, right, bottom, left), name in zip(face_locations, face_names):
            top *= 4
            right *= 4
            bottom *= 4
            left *= 4
            cv2.rectangle(frame, (left, top), (right, bottom), (0, 0, 255), 2)
            cv2.rectangle(frame, (left, bottom - 35), (right, bottom), (0, 0, 255), cv2.FILLED)
            font = cv2.FONT_HERSHEY_DUPLEX
            cv2.putText(frame, name, (left + 6, bottom - 6), font, 1.0, (255, 255, 255), 1)
        cv2.imshow('Video', frame)if cv2.waitKey(1) & 0xFF == ord('q'):
            break
```

```
video_capture.release()
cv2.destroyAllWindows()
```

ДОДАТОК Б

Лістинг коду навчання без нагляду:

```
import cv2
import numpy as np
import npwriter

name = input("Enter your name: ")
cap = cv2.VideoCapture(0)

classifier = cv2.CascadeClassifier("face.xml")
  f_list = []

while True:
    ret, frame = cap.read()
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    faces = classifier.detectMultiScale(gray, 1.5, 5)
    faces = sorted(faces, key = lambda x: x[2]*x[3],
                                      reverse = True)
    faces = faces[:1]
    if len(faces) == 1:
        face = faces[0]
        x, y, w, h = face
        im_face = frame[y:y + h, x:x + w]

        cv2.imshow("face", im_face)
    if not ret:
        continue

    cv2.imshow("full", frame)
```

```
            key = cv2.waitKey(1)
            if key & 0xFF == ord('q'):
                break
            elif key & 0xFF == ord('c'):
                if len(faces) == 1:
                    gray_face = cv2.cvtColor(im_face, cv2.COLOR_BGR2GRAY)
                    gray_face = cv2.resize(gray_face, (100, 100))
                    print(len(f_list), type(gray_face), gray_face.shape)
                    f_list.append(gray_face.reshape(-1))
                else:
                    print("face not found")
                if len(f_list) == 10:
                    break
        npwriter.write(name, np.array(f_list))
        cap.release()
        cv2.destroyAllWindows()
import pandas as pd
        import numpy as np
        import os.path
        f_name = "face_data.csv"

        # storing the data into a csv file
        def write(name, data):

            if os.path.isfile(f_name):

                df = pd.read_csv(f_name, index_col = 0)

                latest   =   pd.DataFrame(data,   columns   =   map(str,
range(10000)))
                latest["name"] = name

                df = pd.concat((df, latest), ignore_index = True, sort =
False)
```

```
    else:
        df = pd.DataFrame(data, columns = map(str, range(10000)))
        df["name"] = name

    df.to_csv(f_name)
import cv2
import numpy as np
import pandas as pd

from npwriter import f_name
from sklearn.neighbors import KNeighborsClassifier
data = pd.read_csv(f_name).values

# data partition
X, Y = data[:, 1:-1], data[:, -1]

print(X, Y)
model = KNeighborsClassifier(n_neighbors = 5)

# fdtraining of model
model.fit(X, Y)

cap = cv2.VideoCapture(0)

classifier = cv2.CascadeClassifier("face.xml")
f_list = []
while True:
    ret, frame = cap.read()
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    faces = classifier.detectMultiScale(gray, 1.5, 5)
    X_test = []
    for face in faces:
        x, y, w, h = face
```

```
            im_face = gray[y:y + h, x:x + w]
            im_face = cv2.resize(im_face, (100, 100))
            X_test.append(im_face.reshape(-1))

        if len(faces)>0:
            response = model.predict(np.array(X_test))
            for i, face in enumerate(faces):
                x, y, w, h = face
                cv2.rectangle(frame, (x, y), (x + w, y + h),
                                            (255, 0, 0), 3)
                cv2.putText(frame, response[i], (x-50, y-50),
                                    cv2.FONT_HERSHEY_DUPLEX, 2,
                                            (0, 255, 0), 3)
        cv2.imshow("full", frame)
        key = cv2.waitKey(1)

        if key & 0xFF == ord("q") :
            break
    cap.release()
    cv2.destroyAllWindows()
```