Вінницький національний технічний університет
(повне найменування вищого навчального закладу)

Факультет інтелектуальних інформаційних технологій та автоматизації
(повне найменування інституту, назва факультету (відділення))

Кафедра комп'ютерних наук
(повна назва кафедри (предметної, циклової комісії))

МАГІСТЕРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему:

«Інформаційна технологія розпізнавання об'єктів у відеопотоці»

Виконала: студентка 2-го курсу, групи 2КН-22м
спеціальності 122 «Комп'ютерні науки»
(шифр і назва напряму підготовки, спеціальності)
Левчук Я.К.
(прізвище та ініціали)

Керівник: к.т.н., доц. каф. КН
Озеранський В.С.
(прізвище та ініціали)
«7» 12 2023 р.

Опонент: к.т.н., доц. каф. АІІТ
Барабан М.С.
(прізвище та ініціали)
«7» 12 2023 р.

Допущено до захисту
Завідувач кафедри КН
д.т.н., проф. Яровий А.А.
(прізвище та ініціали)
«8. 12» 2023 р.

Вінниця ВНТУ - 2023 рік

Вінницький національний технічний університет
Факультет інтелектуальних інформаційних технологій та автоматизації
Кафедра комп'ютерних наук
Рівень вищої освіти ІІ-й (магістерський)
Галузь знань – 12 – Інформаційні технології
Спеціальність – 122 – Комп'ютерні науки
Освітньо-професійна програма – Системи штучного інтелекту

ЗАТВЕРДЖУЮ
Завідувач кафедри КН
д.т.н., проф. Яровий А.А.
(підпис)
"29" 08 2023 року

З А В Д А Н Н Я
НА МАГІСТЕРСЬКУ КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТЦІ
Левчук Ярославі Костянтинівні

1 Тема роботи: «Інформаційна технологія розпізнавання об'єктів у відеопотоці».
Керівник роботи: Озеранський Володимир Сергійович, к.т.н., доцент.
затверджені наказом вищого навчального закладу «18» 09 2023 року № 247

2 Строк подання студентом роботи 13.11.2023

3 Вихідні дані до роботи: Мова програмування - обєктно-орієнтована; формат відео - mkv, mp4, avi, mov, wmv; якість відеопотоку - не менше 720р.

4 Зміст пояснювальної записки (перелік питань, які потрібно розробити): вступ; аналіз сучасних технологій розпізнавання об'єктів у відеопотоці; моделювання інформаційної технології розпізнавання об'єктів у відеопотоці; програмна реалізація інформаційної технології розпізнавання об'єктів у відеопотоці; тестування та аналіз результатів роботи розробленої програми; економічна частина; висновки; перелік використаних джерел; додатки.

5 Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень): схема алгоритму функціонування інформаційної технології розпізнавання об'єктів у відеопотоці; вигляд інтерфейсу користувача; вікно введення основних даних для розпізнавання об'єктів у відеопотоці; приклади роботи програм

6 Консультанти розділів проекту (роботи)

Консультанти розділів роботи в таблиці 1.

Таблиця 1 - Консультанти роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1-3	Озеранський В. С., к.т.н., доц. каф. КН	29.08.23	10.11.23
4	Ратушняк О.Г., к.т.н., доц. каф. ЕПВМ	5.11.23	10.11.23

7 Дата видачі завдання 29.08.23

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломної роботи	Строк виконання етапів роботи	Примітка
1	аналіз сучасних технологій розпізнавання об'єктів у відеопотоці	01.09.23 – 07.09.23	
2	моделювання інформаційної технології розпізнавання об'єктів у відеопотоці	8.09.23 – 24.09.23	
3	програмна реалізація технології розпізнавання об'єктів у відеопотоці	25.09.23 – 15.10.23	
4	тестування та аналіз результатів роботи розробленої програми	16.10.23 – 20.10.23	
5	Розробка інструкції користувача	21.10.23 – 5.11.23	
6	Оформлення матеріалів до захисту МКР	6.11.23 – 10.11.23	

Студентка _____ (підпис) Левчук Я.К. (прізвище та ініціали)

Керівник роботи _____ (підпис) Озеранський В. С. (прізвище та ініціали)

АНОТАЦІЯ

УДК 621.374.415

Левчук Я.К. Інформаційна технологія розпізнавання об'єктів у відеопотоці. Магістерська кваліфікаційна робота зі спеціальності 122 – комп’ютерні науки, освітня програма - Системи штучного інтелекту. Вінниця: ВНТУ, 2023. 111с.

На укр. мові. Бібліогр.: 20 назв; рис.: 35; табл. 16.

У даній магістерській кваліфікаційній роботі досліджено основні підходи щодо розв’язання задачі створення інформаційної технології розпізнавання об'єктів у відеопотоці. Виконано аналіз аналогів та розглянуто технології, що використовуються для вирішення подібних задач.

Проведено обґрунтування вибору моделі інформаційної технології розпізнавання об'єктів у відеопотоці. Для розв’язання даної задачі було використано штучні згорткові нейронні мережі на основі перцептронів.

В розділі економічної частини проведено оцінювання комерційного потенціалу розробки інформаційної технології розпізнавання об’єктів у відеопотоці, спрогнозовано витрати на виконання наукової роботи та впровадження результатів, які склали 279581,79 грн, розраховано період окупності – 2,1 року.

Ключові слова: Нейромережева технологія, розпізнавання, відеопоток.

ABSTRACT

Levchuk Ya.K. Information technology for object recognition in a video stream. Master's thesis on specialty 122 - computer science, educational program - Artificial intelligence systems. Vinnytsia: VNTU, 2023. 111 p.

In Ukrainian speech Bibliography: 20 titles; Fig.: 35; table 14.

In this master's qualification work, the main approaches to solving the problem of creating information technology for object recognition in a video stream are investigated. An analysis of analogues was performed and the methods and technologies used to solve similar problems were considered.

The selection of the information technology model for object recognition in the video stream is justified. Artificial convolutional neural networks based on perceptrons were used to solve this problem.

In the section of the economic part, an assessment of the commercial potential of the development of information technology for object recognition in the video stream was carried out, the costs of performing scientific work and implementing the results were predicted, which amounted to UAH 279,581.79, and the payback period was calculated - 2.1 years.

Keywords: Neural network technology, recognition, video stream.

ЗМІСТ

ВСТУП

Актуальність теми. На сьогоднішній день, завдяки стрімкому розвитку і вдосконаленню цифрових технологій, безліч завдань можна виконувати автоматизовано. Такий підхід, при достатній кваліфікації виконавця, допоможе зберегти час, знизити витрати, уникнути людський фактор, несвоєчасне звітування, порушення дисципліни та багатьох інших речей, що глобально впливають на результат. Делегація рутинних процесів інструментам автоматизації уже охопила сфери фінансів, бізнесу, медицини, військової справи, прогнозування, комунікації, обчислення, класифікації, розпізнавання образів та ін.

Інформаційна технологія розпізнавання об'єктів у відеопотоці може виступити основою систем відеонагляду, збору статистики по завантаженості торговельних систем, систем аналізу пасажиропотоку. Підрахунок відвідувачів на торговельних майданчиках необхідний для обчислення долі покупців серед великої кількості відвідувачів магазину. Це дуже важливий показник, який реально демонструє ефективність роботи магазину.

У місцях масового обслуговування потреба в персоналі прямо пропорційна кількості клієнтів, тому точний підрахунок відвідувачів буде корисним для складання оптимального розкладу роботи співробітників. Реалізація цієї функції може допомагти бібліотекам, паркам, музеям, національним пам'ятників і аналогічним місцям точно вимірювати кількість відвідувачів.

Статистика також може бути використана для того, щоб мінімізувати незручності самої організації, та її відвідувачів - такі заходи як прибирання приміщення чи інші технічні роботи повинні проводитись під час мінімального потоку відвідувачів.

Звідси можна зробити висновок, що створення інформаційної технології розпізнавання об'єктів у відеопотоці в сучасних реаліях є досить актуальною.

Зв'язок роботи з науковими програмами, планами, темами. Магістерська кваліфікаційна робота виконана відповідно до напряму наукових досліджень кафедри комп’ютерних наук Вінницького національного технічного університету 22 К1

«Моделі, методи, технології та пристрої інтелектуальних інформаційних систем управління, економіки, навчання та комунікацій» та плану наукової та навчально-методичної роботи кафедри.

Мета і завдання дослідження.

Метою магістерського дослідженння є підвищення достовірності розпізнавання об’єктів у відеопотоці.

Для досягнення поставленої мети необхідно розв’язати такі задачі:

- здійснити аналіз сучасних технологій розпізнавання об’єктів у відеопотоці;
- здійснити порівняльний аналіз програм аналогів розпізнавання об’єктів у відеопотоці;
- здійснити моделювання інформаційної технології розпізнавання об’єктів у відеопотоці;
- удосконалити математичну модель розпізнавання об’єктів у відеопотоці;
- розробити алгоритм розпізнавання об’єктів у відеопотоці;
- здійснити програмну реалізацію запропонованої інформаційної технології;
- провести тестування програмного засобу та проаналізувати отримані результати;
- здійснити економічні розрахунки доцільності розробки нової інформаційної технології.

Об’єкт дослідження − процес розпізнавання об’єктів у відеопотоці.

Предметом дослідження є програмні засоби розпізнавання об’єктів у відеопотоці.

Методи дослідження. У роботі використані такі методи наукових досліджень: методи та моделі подання знань, методи математичного моделювання, методи автоматизованого тестування програмних засобів, зокрема, функціональне тесування та тестування методом «чорного ящика», методи та техніки тест-дизайну, методи об’єктно-орієнтованого програмування.

Наукова новизна одержаних результатів.

Набула подальшого розвитку інформаційна технологія розпізнавання об’єктів у відеопотоці, яка відрізняється від існуючих використанням удосконаленої

математичної моделі розпізнавання об’єктів, що дозволило підвищити достовірність розпізнавання об’єктів у відеопотоці.

Практичне значення одержаних результатів полягає в тому, що на основі проведених досліджень розроблено програмне забезпечення для розпізнавання об’єктів у відеопотоці.

Запропонована інформаційна технологія сприяє підвищенню достовірності розпізнавання об’єктів у відеопотоці:

- розроблено алгоритм функціонування інформаційної технології розпізнавання об’єктів у відеопотоці;
- розроблено програмні засоби для розпізнавання об’єктів у відеопотоці.

Достовірність теоретичних положень магістерської кваліфікаційної роботи підтверджується коректністю постановки завдання, коректністю використання математичного апарату методів дослідження, експериментальними дослідженнями тестування програмної реалізації інформаційної технології розпізнавання об’єктів у відеопотоці. Адекватність розроблених математичних моделей підтверджується результатами експериментальних досліджень.

Особистий внесок здобувача. Усі результати, що наведені у магістерській кваліфікаційній роботі, отримані самостійно. У працях, які написано у співавторстві, здобувачу належать: особливості розпізнавання об’єктів у відеопотоці [1].

Апробація результатів роботи. Результати роботи були апробовані на конференції «Молодь в науці: дослідження, проблеми, перспективи (МН-2023)» (м. Вінниця, Україна, 2023 р.) [1]. Результати дослідження впроваджено в роботу ТОВ «ІТІ».

Публікації. За результатами досліджень опубліковано тези доповіді на науково-технічній конференції [1].

1 АНАЛІЗ СУЧАСНИХ ТЕХНОЛОГІЙ РОЗПІЗНАВАННЯ ОБ'ЄКТІВ У ВІДЕОПОТОЦІ

1.1Аналіз сучасних технологій комп’ютерного зору

Комп'ютерний зір – це технологія, за допомогою якої машини можуть знаходити, відстежувати, класифікувати та ідентифікувати об'єкти, витягуючи дані з зображень та аналізуючи отриману інформацію [3].

Комп'ютерний зір застосовується для розпізнавання об'єктів, відеоаналітики, опису зображень та відео, розпізнавання жестів та рукописного введення, а також для інтелектуальної обробки зображень.

Машинний зір дещо відрізняється від комп’ютерного зору – він використовує аналіз зображень для того, щоб вирішувати промислові завдання.

Початківцям може здатися, що це різні назви однієї й тієї ж технології, але це не так, оскільки комп'ютерний зір – це загальна назва набору технологій, а машинний зір – сфера застосування.

Строго кажучи, експерти з комп'ютерного зору «створюють алгоритми та системи для автоматичного аналізу зображень та отримання інформації з видимого світу». З погляду непрофесіонала вони створюють машини, які здатні бачити. Для таких пристроїв, як персональні роботи, безпілотні автомобілі та дрони, з якими ми стикаємося все частіше у повсякденному житті, зір дуже важливий.

Машинний зір дозволяє відмовитися від ручної праці, адже контролювати збирання виробів, рахувати та вимірювати об'єкти, читати текст, цифри та ідентифікувати об'єкти може робототехнічна система.

Машинний зір використовується у різних галузях. У медицині - для того, щоб точніше ставити діагноз, у промисловості - для зниження собівартості товарів за рахунок автоматизації. В автомобільній індустрії - для навігації безпілотників, а в економіці - для зчитування штрих-кодів або підрахунку відвідувачів.

Оскільки машинний зір використовується для вирішення різних промислових завдань, то залежно від того, яку саме завдання треба вирішити, створюються спеціальні системи машинного зору. Типові системи машинного зору складаються з

камер, програмного забезпечення, процесорів, джерел світла та різних датчиків. Наприклад, певний датчик визначив, що деталь на конвеєрі потрібно перевірити, автомат запустив камеру та зробив знімок цієї деталі. Після цього зображення відправляється в комп'ютер, де програмне забезпечення машинного зору обробляє отриману картинку. Після того, як зображення оброблено, залежно від стану деталі програма дозволяє або блокує пересування деталі далі по конвеєру. Тобто, якщо деталь пошкоджена - програмне забезпечення сформує сигнал для її відхилення, зупинить виробництво або попередить людину-оператора про те, що є деталь з дефектом.

Машинний зір зосереджується, в основному, на застосуванні в промисловості, наприклад, автономні роботи та системи візуальної перевірки та вимірювань. Це означає, що технології датчиків зображення та теорії управління пов'язані з обробкою відеоданих для управління роботом та обробка отриманих даних у реальному часі здійснюється програмно чи апаратно.

Обробка зображень та аналіз зображень в основному зосереджені на роботі із 2D зображеннями, тобто. як перетворити одне зображення на інше. Наприклад, попіксельні операції збільшення контрастності, операції з виділення країв, усунення шумів або геометричні перетворення, такі як обертання зображення. Дані операції припускають, що обробка/аналіз зображення діють незалежно від самих зображень.

Комп'ютерний зір зосереджується на обробці тривимірних сцен, спроектованих на одне чи кілька зображень. Наприклад, відновлення структури або іншої інформації про 3D сцену за одним або декількома зображеннями. Комп'ютерний зір часто залежить від більш складних припущень щодо того, що представлено на зображеннях.

Також існує область, яка називається візуалізацією. Вона спочатку була пов'язана з процесом створення зображень, але іноді має справу з обробкою та аналізом наявного зображення. Наприклад, рентгенографія працює з аналізом відеоданих медичного застосування.

Нарешті, розпізнавання образів є областю, яка використовує різні методи отримання інформації з відеоданих, які переважно, засновані на статистичному

підході. Значна частина цієї галузі присвячена практичному застосуванню цих методів.

Задачі, які виконує машинний зір:

- Розпізнавання об’єктів - класичне завдання в комп'ютерному зорі, обробці зображень і машинному зорі це визначення чи містять відео певний характерний об'єкт, особливість чи активність. Це завдання може бути достовірно і легко вирішене людиною, але досі не вирішене з високою точністю у комп'ютерному зорі;
- ідентифікація об’єктів – завдання розпізнавання індивідуального екземпляру об'єкта, що належить до будь-якого класу. Наприклад, ідентифікація певної людини або відбитка пальців або автомобіля;
- виявлення об’єктів у фото чи відео даних – завдання, засноване на відносно простих і швидких обчисленнях, іноді використовується для знаходження невеликих ділянок в аналізованому зображенні, які потім аналізуються за допомогою прийомів, більш вимогливих до ресурсів, для отримання правильної інтерпретації.
- розпізнавання тексту - знаходження всіх зображень у великому наборі даних, які мають певне різними шляхами зміст, визначення положення або орієнтації певного об'єкта щодо камери, оптичне розпізнавання символів на зображеннях друкованого або рукописного тексту (зазвичай для перекладу текстового формату в найбільш зручний для редагування або індексації – електронний;
- відновлення 3D форми об’єктів за 2D зображенням здійснюється за допомогою стереореконструкції карти глибини, реконструкції поля нормалей та карти глибини за забарвленням напівтонового зображення, реконструкції карти глибини за текстурою та визначення форми по переміщенню;
- відновлення відео-сцен – це завдання відтворити тривимірну модель сцени. У найпростішому випадку моделлю може бути набір точок тривимірного простору. Більш складні методи відтворюють повну тривимірну модель;

- відновлення зображень за фрагментами. Найбільш простим підходом до вирішення цієї задачі є різні типи фільтрів, таких як нижніх або середніх фільтри частот. Вищий рівень видалення шумів досягається в ході початкового аналізу відеоданих на наявність різних структур, таких як лінії або межі, а потім управління процесом фільтрації на основі цих даних;
- виділення на зображенні структур певного виду, сегментація зображень;
- аналіз оптичного потоку.

Алгоритми комп'ютерного зору на даний момент є однією з найбільш трансформаційних і потужних систем штучного інтелекту у світі. Системи комп'ютерного зору використовуються в автономних транспортних засобах, роботів-навігаціях, системах розпізнавання облич тощо. Щоб повністю зрозуміти, як працюють системи комп'ютерного зору, давайте спочатку обговоримо, як люди розпізнають об'єкти. Найкращим поясненням того, як люди розпізнають об'єкти, є модель, яка описує початкову фазу розпізнавання об'єктів як таку, коли основні компоненти об'єктів, такі як форма, колір і глибина, спочатку інтерпретуються мозком. Сигнали від ока, які надходять у мозок, аналізуються, щоб спочатку витягнути краї об'єкта, і ці краї об'єднуються в більш складне уявлення, яке завершує форму об'єкта.

Системи комп'ютерного зору працюють дуже подібно до зорової системи людини: спочатку розрізняють краї об'єкта, а потім з'єднують ці краї разом у форму об'єкта. Велика відмінність полягає в тому, що, оскільки комп'ютери інтерпретують зображення як числа, системі комп'ютерного зору потрібен спосіб інтерпретації окремих пікселів, які складають зображення. Система комп'ютерного зору призначатиме значення пікселям на зображенні, і, досліджуючи різницю значень між однією областю пікселів та іншою областю пікселів, комп'ютер може розрізняти краї. Наприклад, якщо розглянуте зображення має відтінки сірого, то значення будуть варіюватися від чорного (відображається 0) до білого (представленого 255). Раптова зміна діапазону значень пікселів поруч один з одним буде вказувати на край.

Цей основний принцип порівняння значень пікселів також можна виконати з кольоровими зображеннями, коли комп'ютер порівнює відмінності між різними

каналами кольору RGB. Отже, система комп’ютерного зору вивчає значення пікселів для інтерпретації зображення, а звідси архітектура системи комп’ютерного зору матиме наступну структуру яку наведемо на прикладі згорткової нейронної мережі.

Згорткові нейронні мережі (CNN). Основним типом штучного інтелекту, який використовується в задачах комп'ютерного зору, є той, що базується на згорткових нейронних мережах. Згортки – це математичні процеси, які мережа використовує для визначення різниці значень між пікселями. Якщо уявити сітку значень пікселів, то це ще менша сітка яка переміщується по цій головній сітці. Значення під другою сіткою аналізуються мережею, тому мережа розглядає лише кілька пікселів за раз. Це часто називають технікою «розсувних вікон». Значення, що аналізуються за допомогою розсувного вікна, підсумовуються мережею, що допомагає зменшити складність зображення та полегшити для мережі вилучення шаблонів.

Згорткові нейронні мережі поділяються на дві різні секції:

- згорткові;
- повністю зв’язані.

Згорткові шари мережі – це екстрактори ознак, завдання яких – аналізувати пікселі в зображенні та формувати їх уявлення, з яких щільно пов’язані шари нейронної мережі можуть вивчати шаблони. Згорткові шари починаються з простого вивчення пікселів і виділення низькорівневих ознак зображення, таких як краї. Пізніші згорткові шари з’єднують краї разом у більш складні форми. Відповідно в кінці мережа матиме представлення країв і деталей зображення, які вона зможе передати повністю пов’язаним шарам.

При роботі з згортковими нейронними мережами слід виділити ряд ключових етапі які необхідно виділити для успішного виконання завдання. Перелік етапів для успішної роботи з згортковими нейронними мережами наведено на рисунку 1.1.

Рисунок 1.1 – Ключові етапи при роботі з нейронними мережами

Згортка виконується на зображенні для визначення певних елементів зображення. Згортка допомагає розмити, збільшити різкість, виявити краї, зменшити шум тощо на зображенні, що може допомогти машині дізнатися конкретні характеристики зображення.

Об'єднання. Закручене зображення може бути занадто великим, і тому його потрібно зменшити. Об'єднання в основному виконується для зменшення зображення без втрати функцій або візерунків.

Згладжування. Згладжування перетворює двовимірну матрицю ознак у вектор ознак, який можна подати в нейронну мережу або класифікатор.

Повне підключення. Повне підключення просто відноситься до процесу подачі сплощеного зображення в нейронну мережу.

У той час як згорткова нейронна мережа може витягувати шаблони із зображень сама по собі, точність системи комп'ютерного зору можна значно підвищити, додавши до зображень анотації.

Анотація зображення – це процес додавання метаданих до зображення, які допомагають класифікатору виявляти важливі об'єкти на зображенні. Використання анотації зображень важливо, коли системи комп'ютерного зору повинні бути високоточними, наприклад, під час керування автономним транспортним засобом або роботом.

Існують різні способи анотування зображень для покращення продуктивності класифікатора комп'ютерного зору. Анотація зображень часто виконується за

допомогою обмежувальних рамок, блоку, який оточує краї цільового об’єкта та наказує комп’ютеру зосередити свою увагу всередині рамки.

Семантична сегментація – це інший тип анотації зображення, який діє шляхом присвоєння класу зображення кожному пікселю в зображенні. Іншими словами, кожен піксель, який можна вважати «травою» або «деревою», буде позначено як належний до цих класів. Ця техніка забезпечує точність на рівні пікселів, але створення анотацій семантичної сегментації є більш складним і трудомістким, ніж створення простих обмежувальних рамок. Існують також інші методи анотації, такі як лінії та точки.

Машинне навчання є однією з найбільш швидкозростаючих технологічних галузей, але незважаючи на те, як часто вживаються слова «машинне навчання», може бути важко зрозуміти, що саме таке машинне навчання.

Машинне навчання не відноситься до однієї речі, це загальний термін, який можна застосувати до багатьох різних концепцій і технік. Розуміння машинного навчання означає знайомство з різними формами аналізу моделі, змінними та алгоритмами. Хоча термін машинне навчання можна застосувати до багатьох різних речей, загалом цей термін відноситься до дозволу комп’ютеру виконувати завдання, не отримуючи явних порядкових інструкцій для цього. Спеціалісту з машинного навчання не потрібно писати всі кроки, необхідні для вирішення проблеми, оскільки комп’ютер здатний «навчатися», аналізуючи закономірності в даних і узагальнюючи ці закономірності на нові дані.

Системи машинного навчання складаються з трьох основних частин:

- входи;
- алгоритми;
- виходи.

Вхідні дані – це дані, які подаються в систему машинного навчання, а вхідні дані можна розділити на мітки та функції. Характеристики – це відповідні змінні, змінні, які будуть проаналізовані, щоб дізнатися закономірності та зробити висновки. Тим часом мітки — це класи/описи, надані окремим екземплярам даних.

Функції та мітки можна використовувати в двох різних типах машинного навчання: навчання з наглядом і навчання без нагляду (рис. 1.2).

Рисунок 1.2 – Класифікація методів машинного навчання

Навчання без нагляду і контрольоване навчання. При навчанні з наглядом вхідні дані супроводжуються основною істиною. Проблеми з навчанням з наглядом мають правильні вихідні значення як частину набору даних, тому очікувані класи відомі заздалегідь. Це дає змогу спеціалісту з даних перевірити продуктивність алгоритму, перевіривши дані в тестовому наборі даних і побачивши, який відсоток елементів було правильно класифіковано.

Навпаки, проблеми з навчанням без нагляду не мають основних позначок істини. Алгоритм машинного навчання, навчений виконувати завдання навчання без нагляду, повинен мати можливість самостійно виводити відповідні закономірності в даних.

Алгоритми навчання з наглядом зазвичай використовуються для задач класифікації, де є великий набір даних, заповнений екземплярами, які необхідно відсортувати в один із багатьох різних класів. Іншим типом навчання з керівництвом є завдання регресії, де значення, виведене алгоритмом, є безперервним, а не категоричним.

Тим часом алгоритми навчання без нагляду використовуються для таких завдань, як оцінка щільності, кластеризація та навчання репрезентації. Ці три завдання потребують моделі машинного навчання, щоб зробити висновок про структуру даних, для моделі немає попередньо визначених класів.

Коротко розглянемо деякі з найпоширеніших алгоритмів, які використовуються як у навчанні без нагляду, так і в навчанні з наглядом.

Поширені алгоритми навчання під наглядом включають:

- наївний Байєс;
- опорні векторні машини;
- логістична регресія;
- випадковий ліс;
- штучні нейронні мережі.

Машини опорних векторів - це алгоритми, які поділяють набір даних на різні класи. Точки даних групуються в кластери шляхом малювання ліній, що відокремлюють класи один від одного. Точки, знайдені з одного боку від прямої, будуть належати до одного класу, а точки по інший бік лінії - до іншого класу. Машини опорних векторів мають на меті максимізувати відстань між лінією та точками, знайденими по обидва боки від лінії, і чим більше відстань, тим впевненіше класифікатор, що точка належить одному класу, а не іншому.

Логістична регресія – це алгоритм, який використовується в завданнях двійкової класифікації, коли точки даних необхідно класифікувати як належні до одного з двох класів. Логістична регресія працює, позначаючи точку даних 1 або 0. Якщо сприйняте значення точки даних становить 0,49 або нижче, воно класифікується як 0, а якщо воно дорівнює 0,5 або вище, воно класифікується як 1.

Алгоритми дерева рішень діють шляхом поділу наборів даних на все менші і менші фрагменти. Точні критерії, які використовуються для поділу даних, залежить від інженера з машинного навчання, але мета полягає в тому, щоб остаточно розділити дані на окремі точки даних, які потім будуть класифіковані за допомогою ключа.

Алгоритм Random Forest - це, по суті, багато окремих класифікаторів дерева рішень, пов’язаних разом у більш потужний класифікатор.

Наївний байесовский класифікатор обчислює ймовірність того, що дана точка даних сталася на основі ймовірності попереднього події, що сталася. Він заснований на теоремі Байєса і розділяє точки даних у класи на основі їх обчисленої ймовірності. При реалізації наївного байєсового класифікатора передбачається, що всі предиктори мають однаковий вплив на результат класу.

Штучна нейронна мережа, або багатошаровий персептрон, алгоритми машинного навчання, натхненні структури і функціями людського мозку. Штучні нейронні мережі отримали свою назву через те, що вони складаються з багатьох вузлів/нейронів, пов’язаних між собою. Кожен нейрон маніпулює даними за допомогою математичної функції. У штучних нейронних мережах є вхідні шари, приховані шари та вихідні шари.

Прихований шар нейронної мережі – це місце, де дані фактично інтерпретуються та аналізуються на наявність шаблонів. Іншими словами, це те, де вчиться алгоритм. Більше нейронів, об’єднаних разом, створюють складніші мережі, здатні вивчати більш складні шаблони.

Типи неконтрольованого навчання. Алгоритми навчання без нагляду включають:

- K-means кластеризацію;
- автокодери;
- аналіз основних компонентів.

Кластеризація K-середніх – це методика класифікації без нагляду, яка працює шляхом поділу точок даних на кластери або групи на основі їхніх особливостей. Кластеризація K-середніх аналізує ознаки, знайдені в точках даних, і розрізняє в них закономірності, які роблять точки даних, знайдені в кластері даного класу, більш схожими один на одного, ніж вони є на кластери, що містять інші точки даних. Це досягається шляхом розміщення можливих центрів для кластера, або центроїдів, на графіку даних і перепризначення положення центроїда, доки не буде знайдено

положення, яке мінімізує відстань між центроїдом і точками, які належать до класу цього центроїда. Дослідник може вказати потрібну кількість кластерів.

Аналіз основних компонентів – це метод, який зменшує велику кількість функцій/змінних до меншого простору/меншої кількості функцій. «Основні компоненти» точок даних вибираються для збереження, тоді як інші ознаки стискаються в менше представлення. Зв'язок між вихідними зіллями даних зберігається, але оскільки складність точок даних простіша, дані легше кількісно оцінити та описати.

Автокодери – це версії нейронних мереж, які можна застосувати до завдань навчання без нагляду. Автокодери здатні приймати дані довільної форми без міток і перетворювати їх у дані, які може використовувати нейронна мережа, створюючи в основному власні мічені навчальні дані. Мета автокодера полягає в тому, щоб перетворити вхідні дані та перебудувати їх якомога точніше, тому мережа спонукає визначати, які функції є найважливішими та витягувати їх.

1.2 Особливості розпізнавання об’єктів у відеопотоці

На сьогоднішній день, підрахунок відвідувачів є надзвичайно поширеним явищем, яке уже набрало велику популярність. Основні причини такої тенденції - введення чисельних методів у розвиток торгівлі, карантині обмеження, необхідність отримання даних для покращення організації пасажироперевезення та боротьба з тероризмом та багато інших.

Для автоматизованого підрахунку людей необхідно створити систему підрахунку відвідувачів, яка забезпечить облік кількості людей, що пройшли через певний прохід за певний проміжок часу. З їх допомогою також можна визначити напрям руху, але найчастіше використовується поділ на два класи: вхідні та вихідні.

Системи розпізнавання та підрахунку відвідувачів, як правило, встановлюються на вході в приміщення, дозволяють слідкувати за загальним числом відвідувачів і інтерпретують людей в зрозумілий для електронно-обчислювальної машини формат для подальшого їх автоматизованого підрахунку [3].

Найдоцільнішим підходом до автоматизованого підрахунку людей у відеопотоці є нейронні мережі, оскільки реалізації методами без них, наприклад, за допомогою Optical flow, дають гірші результати.

Загальний процес реалізації являється багатоетапним. Перш за все необхідно визначити на кадрі людей, ця задача називається “Object Detection”, ця задача добре вивчена та має цілу низку якісних реалізацій. ЇЇ концепцією являється те, що кожен клас об’єктів має свої власні особливості, які допомагають класифікувати об’єкт - наприклад, усі кола круглі. Розпізнавання класів об’єктів використовує ці спеціальні функції. Наприклад, під час пошуку кіл шукаються предмети, які знаходяться на певній відстані від точки (тобто центру). Так само, шукаючи квадрати, потрібні предмети, які перпендикулярні по кутах і мають однакову довжину сторін. Подібний підхід застосовується для ідентифікації обличчя, де можна знайти очі, ніс та губи, а також такі функції, як колір шкіри та відстань між очима. Розглянемо найпопулярніші реалізації “Object Detection”.

Ряд архітектур на основі R-CNN (регіони зі згортковими нейронними мережами): R-CNN, Fast R-CNN, Faster R-CNN, Mask R-CNN (рис. 1.3).

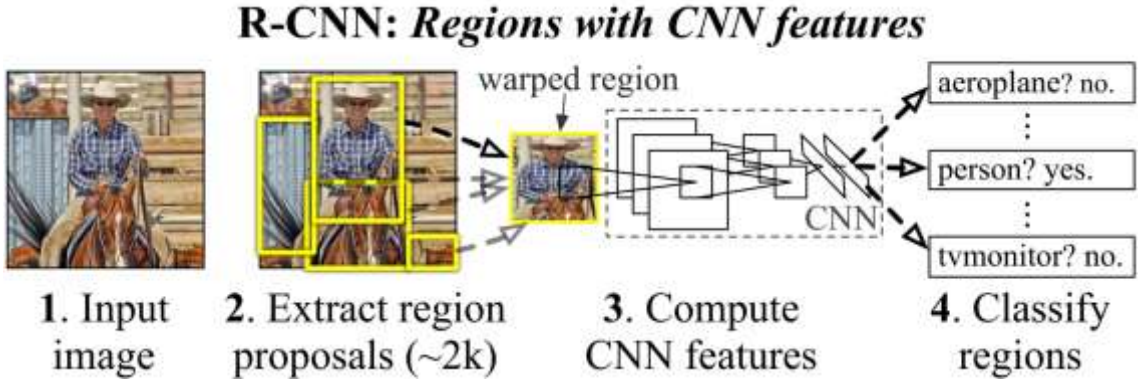


Рисунок 1.3 –Архітектура нейромережі R-CNN

Для виявлення об’єкта на зображенні за допомогою механізму мережі регіональних пропозицій (RPN) вибираються обмежені області (межі). Спочатку замість RPN використовувався повільніший механізм вибіркового пошуку. Потім витягнуті обмежені області подаються на вхід звичайної нейронної мережі для класифікації. В архітектурі R-CNN є явні цикли «for», що дозволяють ітерацію в

обмежених регіонах, до 2000 проходів у внутрішній мережі AlexNet. Явні цикли for уповільнюють обробку зображень. Кількість явних циклів, що проходять через внутрішню нейронну мережу, зменшується з кожною новою версією архітектури, і вносяться десятки інших змін, щоб збільшити швидкість і замінити задачу виявлення об’єктів на сегментацію об’єктів у Mask R-CNN, але архітектура все ще залишається незмінною. досить повільно

SSD (Single Shot MultiBox Detector). Відмінна риса: розрізняйте об’єкти за один прохід за допомогою віконної сітки за замовчуванням (рамка за замовчуванням) на піраміді зображення. Піраміда зображення кодується в тензорах згортки з подальшими операціями згортки та об’єднання (рис. 1.4). Таким чином за один прохід мережі визначають як великі, так і малі об'єкти. Така нейронна мережа має надзвичайно високу швидкодію, проте її точність залишає бажати кращого. Така мережа є хорошим вибором для програмних застосунків для мобільних пристроїв.

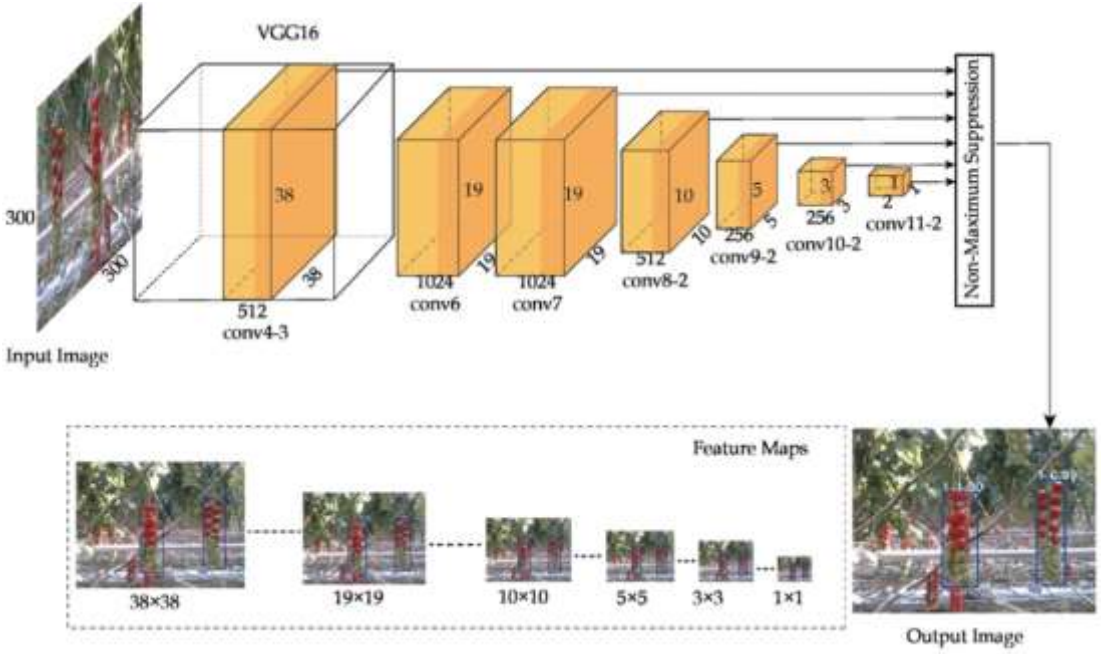


Рисунок 1.4 –Архітектура нейромережі SSD

YOLO (You Only Look Once) - перша нейронна мережа, яка розпізнає об’єкти в режимі реального часу на мобільних пристроях. Характерна ознака: розрізнення предметів за один прохід (достатньо одного погляду). Це означає, що в архітектурі

YOLO немає явних циклів "for", що забезпечує швидку роботу мережі. Наприклад, наступна аналогія: в операціях NumPy над масивами немає явних циклів «for», які реалізовані в NumPy на нижчих рівнях архітектури через мову програмування C. YOLO використовує сітку попередньо визначених вікон. Щоб запобігти повторному визначенню одного й того ж об’єкта, використовується коефіцієнт перетину через об’єднання (IoU). Ця архітектура працює в широкому діапазоні і характеризується високою продуктивністю (рис. 1.5).

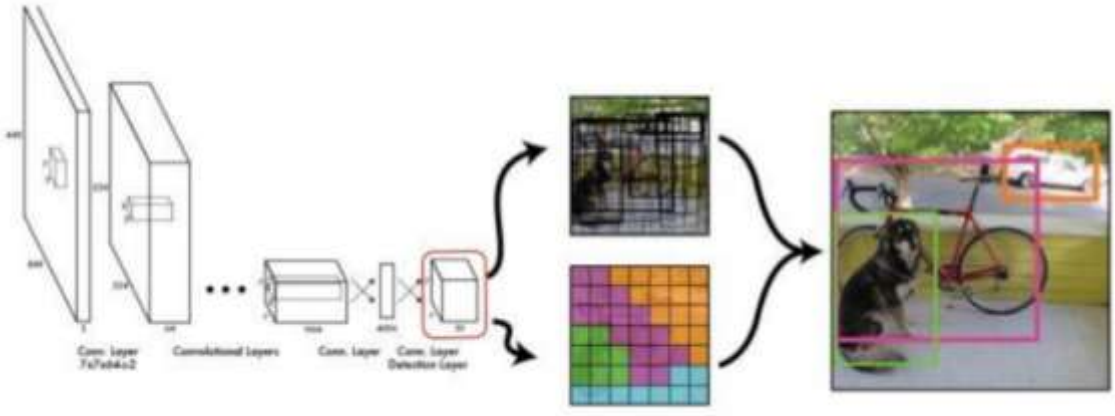


Рисунок 1.5 –Архітектура нейромережі YOLO

Створення власного детектора є надзвичайно часо- та ресурсозатратною задачею, у зв’язку із чим більш доцільно обрати уже готову реалізацію. Архітектура YOLO має найкращий баланс між швидкодією та точністю, тому і буде обраною у даній роботі.

1.3 Основні технології реідентифікації об’єктів у відеопотоці

Наступним етапом обробки відеопотоку для розпізнавання та автоматизованого підрахунку людей являється задача реідентифікації людини, для якої не існує універсального готового рішення, тому буде створено власну нейронну мережу. Для побудови якої можна використати одну із наступних архітектур: глибокі згорткові нейронні мережі (CNN), рекурентні згорткові нейромережі (RCNN) та автоенкодери (AutoEncoder).

Згорткова нейронна мережа (CNN, або ConvNet) - це клас глибокої нейронної мережі, який найчастіше застосовується для аналізу візуальних зображень. Вони також відомі як інваріантні до зсуву або просторово-інваріантні штучні нейронні мережі (SIANN), засновані на архітектурі спільної ваги ядер згортки або фільтрів, які ковзають уздовж вхідних функцій і забезпечують еквівалентні відповіді на трансляцію, відомі як карти функцій. На противагу інтуїції, більшість згорткових нейронних мереж еквівалентні лише, на відміну від інваріантних, перекладу. Вони мають програми в розпізнаванні зображень та відео. Використовуються у системах: класифікації зображень, сегментації зображень, медичного аналізу зображень, обробки природної мови, інтерфейси мозок-комп’ютер, та фінансові часові ряди. CNN - це регуляризовані версії багатошарових персептронів. Багатошарові перцептрони зазвичай означають повністю пов'язані мережі, тобто кожен нейрон в одному шарі пов'язаний з усіма нейронами наступного шару. «Повна зв'язність» цих мереж робить їх схильними до перенавчання даних. Типові способи регуляризації або запобігання перенавчання включають: штрафні параметри під час навчання (наприклад, зниження ваги) або обмеження можливості підключення (пропущені з'єднання, випадання тощо). CNN використовують інший підхід до регуляризації: вони використовують переваги ієрархічного шаблону в даних і збирають шаблони зростаючої складності, використовуючи більш дрібні і прості шаблони, видавлені в їх фільтрах. Отже, за шкалою зв'язності і складності CNN знаходяться на нижній межі. Згорткові мережі були натхненні біологічними в тому сенсі, що патерн зв'язку між нейронами нагадує організацію зорової кори головного мозку тварин. Окремі нейрони кори відповідають на стимули тільки в обмеженій області поля зору, відомої як рецептивної полі. Чутливі поля різних нейронів частково перекриваються, так що вони покривають все поле зору. CNN використовують відносно невелику попередню обробку в порівнянні з іншими алгоритмами класифікації зображень. Це означає, що мережа вчиться оптимізувати фільтри (або ядра) за допомогою автоматичного навчання, тоді як в традиційних алгоритмах ці фільтри створюються вручну. Ця незалежність від попередніх знань і втручання людини в витяг ознак є великою перевагою.

Згорткові рекурентні нейронні мережі - це комбінація двох найбільш відомих нейронних мереж. CRNN включає CNN, за якою слідує RNN. Пропонована мережа схожа на CRNN, але дає кращі або оптимальні результати, особливо щодо обробки аудіосигналів. Мережа починається з традиційної двовимірної згорткової нейронної мережі, за якої слідує пакетна нормалізація, активація ELU, максимальне об'єднання і випадання з частотою випадання 50%. Три таких згортальних шари розміщуються послідовно з відповідними активаціями. За згортковими шарами слідують шар перестановки і зміни форми, який дуже необхідний для CRNN, оскільки форма вектора ознак відрізняється від CNN до RNN. Згорткові шари розробляються на основі тривимірних векторів ознак, тоді як рекурентні нейронні мережі розробляються на основі двовимірних векторів ознак. Шари перестановки змінюють напрямок осей векторів ознак, за якими слідують шари зміни форми, які перетворять вектор ознак в двовимірний вектор ознак. RNN сумісна з двовимірними векторами ознак.

Автоенкодер - це тип штучної нейронної мережі, яка використовується для навчання ефективному кодуванню даних неконтрольованим чином. Завдання автоенкодера - вивчити уявлення (кодування) набору даних, зазвичай для зменшення розмірності, шляхом навчання мережі ігнорування «шуму» сигналу. Поряд зі стороною скорочення вивчається сторона відновлення, де автоенкодер намагається згенерувати з скороченим кодування уявлення, максимально наближене до вихідного входу, звідси і його назва. Існують варіанти, спрямовані на те, щоб вивчені уявлення придбали корисні властивості. Прикладами є регуляризовані автоенкодери (розріджені, шумоподавляючі і звужуючі), які ефективні при навчанні уявлень для наступних завдань класифікації, і варіаційні автоенкодери з додатками як у генеративних моделей. Автоенкодери застосовуються для вирішення багатьох завдань, від розпізнавання осіб до отримання семантичного значення слів. Автоенкодер - це нейронна мережа, яка вчиться копіювати свій вхід на свій вихід. Вона має внутрішній (прихований) шар, який описує код, який використовується для представлення введення, і складається з двох основних частин: кодувальника, який відображає введення в код, і декодера, який відображає код на реконструкцію

введення. Бездоганне виконання завдання копіювання просто дублює сигнал, і саме тому автокодери зазвичай обмежені способами, які змушують їх приблизно реконструювати введення, зберігаючи тільки найбільш важливі аспекти даних в копії.

1.4 Наявні набори даних

Towncentre. Набір даних Oxford Town Center – це відео із камер відеоспостереження за пішоходами, який використовується для досліджень та розробки систем розпізнавання діяльності та розпізнавання облич. Він був використаний у дослідженнях щодо алгоритмів соціального дистанціювання та класифікації статі людини. З моменту публікації у 2009 році цей набір даних використовувався у понад 60 перевірених дослідницьких проектах, включаючи комерційні дослідження Amazon, Disney, OSRAM та Huawei; та академічні дослідження в Китаї, Ізраїлі, Росії, Сінгапурі, США та Німеччині серед десятків інших (рис. 1.6).

Рисунок 1.6 – Набір даних Oxford Town Centre

Набір даних Oxford Town Centre унікальний тим, що він використовує кадри з камери громадського спостереження. Пропонується повністю автоматизована система класифікації для розпізнавання статі. Їх дослідження базується на більшому наборі

зображень, який називається PETA, та містить 10 піднаборів даних. Для підмножин з 3DPes, SARC3D та i-LID точність була відносно низькою, близько 50%. Однак на підмножинах зображень Oxford Town Centre точність дуже висока, понад 98% . Загалом вони заявляють приблизно 77% точності класифікації для всіх підмножин, будучи першими, хто застосував підхід глибокого навчання лише для розпізнавання статі людини. щодо 2D оцінок ключових точок. Принцип роботи детектора OpenPose зображено на (рис. 1.7).

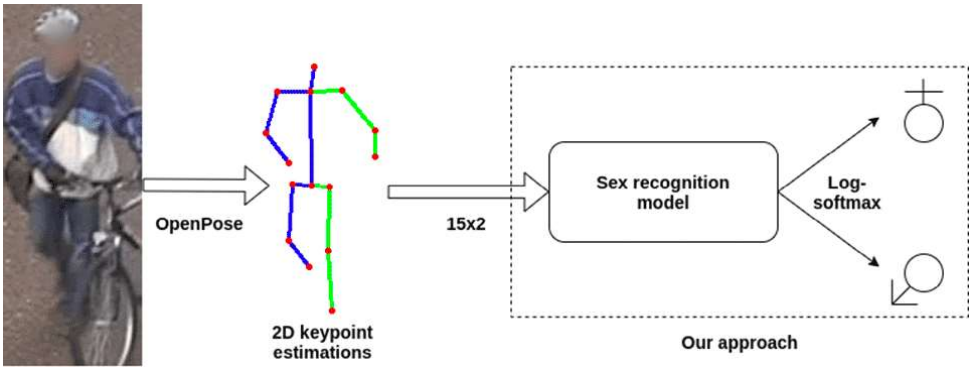


Рисунок 1.7 – Принцип роботи детектора OpenPose

Також використання оригінального набору даних CCTV Oxford Town Center призначене для розробки алгоритмів соціального дистанціювання. У квітні 2020 року засновник Landing.ai написав у Twitter, що його компанія розробила інструмент ШІ для моніторингу співробітників та забезпечення соціального дистанціювання на робочому місці. Його компанія опублікувала на YouTube відео, на якому показано, як працює алгоритм, використовуючи набір даних Oxford Town Center в рекламній демонстрації та в розробці алгоритму. Оскільки Landing.ai розміщував інформацію про свій інструмент ШІ, численні інші розробники та компанії застосовували той самий підхід для створення власної версії. Пошук варіантів соціальних мереж Oxford Town Center на YouTube дав кілька подібних демо-версій, серед яких кілька інших комерційних організацій, включаючи Airpix та JoltVision.

Duke MTMC (Multi-Target, Multi-Camera) – це набір відеозаписів із камер відеоспостереження, зроблених у кампусі університету Дюка у 2014 році, і

використовується для досліджень та розробки систем відеоспостереження, повторної ідентифікації людей та розпізнавання обличчя з низькою роздільною здатністю.

Набір даних містить більше 14 годин синхронізованого відеоспостереження з 8 камер при 1080p та 60 кадрів в секунду, з понад 2 мільйонами кадрів з 2000 студентів, що йдуть до та з класів. 8 камер спостереження, розміщених у студентському містечку, були спеціально встановлені для знімання студентів у періоди між лекціями, коли пішохідний рух інтенсивний (рис. 1.8).

Існують наступні розширення набору даних Duke MTMC: Duke MTMC Re-ID, Duke MTMC Video Re-ID, Duke MTMC Groups, Duke MTMC Attribute, набір даних розпізнавання обличчя з низькою роздільною здатністю під назвою QMUL-SurvFace та ін.

Рисунок 1.8 – Зображення з набору даних Duke MTMC

Набір даних Wildtrack – це також набір відеозаписів із камер відеоспостереження студентів, записаний біля головної будівлі університету ETH в Цюріху. Загалом це датасет із семи 35-хвилинних відео, що містять тисячі студентів, які були записані та зроблені загальнодоступними для будь-якого типу досліджень.

Через велику кількість студентів у відеороликах, анотування одного кадру займало в середньому 10 хвилин. Потім ці анотації використовувались для

дослідницьких робіт The WILDTRACK Multi-Camera Person Data Data і WILDTRACK: A Multi-camera HD Dataset for Dense Unscripted Pedestrian Detection. Призначений для багатокамерного виявлення пішоходів програмами для безпеки, спостереження, ідентифікації осіб, робототехніки, автономного водіння та краудсорсингу. Набір даних Wildtrack зображено на (рис 1.9).

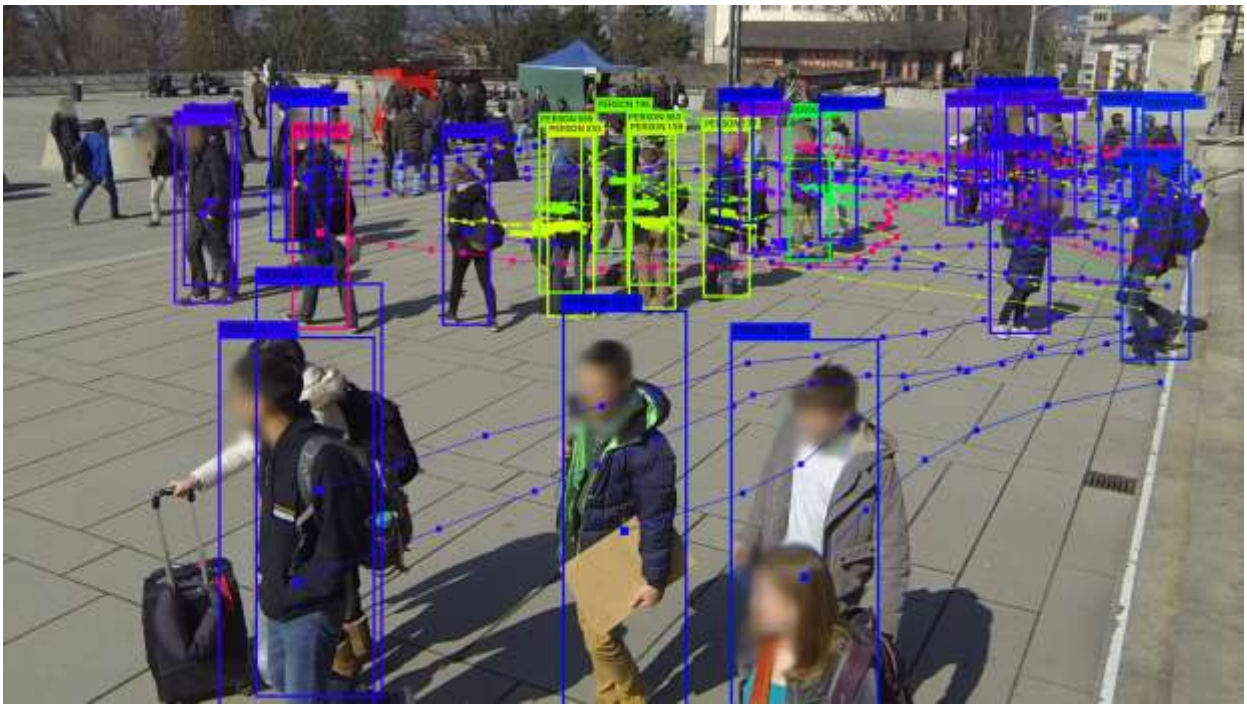


Рисунок 1.9 – Зображення з набору даних Wildtrack

1.5 Аналіз програм аналогів розпізнавання об’єктів у відеопотоці

В якості системи-аналогу вибрано PyImageSearch People Counter. PyImageSearch - це веб-ресурс для розробників, які працюють у сферах комп'ютерного зору, машинного навчання та обробки зображень. Створений Адріаном Россборо, PyImageSearch надає різноманітні матеріали.

PyImageSearch містить потужну базу знань, де розглядаються теоретичні концепції, приклади коду та різноманітні завдання у сферах обробки зображень, комп'ютерного зору та машинного навчання. Веб-ресурс веде активний блог, де публікуються новини, тенденції та інструкції. Крім того, PyImageSearch сприяє формуванню спільноти розробників, які обмінюються ідеями та рішеннями.

Сайт також містить приклади реальних проектів та додатків, які можуть бути корисними для розробників, які хочуть застосовувати свої навички в практичних завданнях. PyImageSearch допомагає розробникам удосконалювати свої навички та розуміння в галузі комп'ютерного зору, надаючи високоякісний контент та ресурси для самостійного вивчення та розвитку.

Програма об'єднує концепцію виявлення об'єктів та відстеження об'єктів в єдиний алгоритм, який зазвичай поділяється на дві фази:

Фаза 1 – Виявлення: Під час фази виявлення ми запускаємо наш відстежувач об’єктів, щоб виявити, чи з’явилися нові об’єкти в нашому поданні, і перевірити, чи можемо ми знайти об’єкти, які були „загублені” на етапі відстеження . Для кожного виявленого об’єкта ми створюємо або оновлюємо відстежувач об’єктів з новими координатами обмежувального вікна.

Фаза 2 - Відстеження: Коли ми не перебуваємо у фазі “виявлення”, ми перебуваємо у фазі “відстеження”. Для кожного з виявлених об’єктів ми створюємо інструмент відстеження об’єктів для відстеження об’єкта при його переміщенні по кадру. Наш трекер об’єктів повинен бути швидшим та ефективнішим, ніж детектор об’єктів, щоб продуктивність системи була високою.

Перевага цього гібридного підходу полягає в тому, що ми можемо застосовувати високоточні методи виявлення об'єктів заощаджуючи машинний час на етапі відслідковування, проте сам цей етап працює не найкращим чином. Перша фаза програми-аналогу зображена на (рис 1.10).

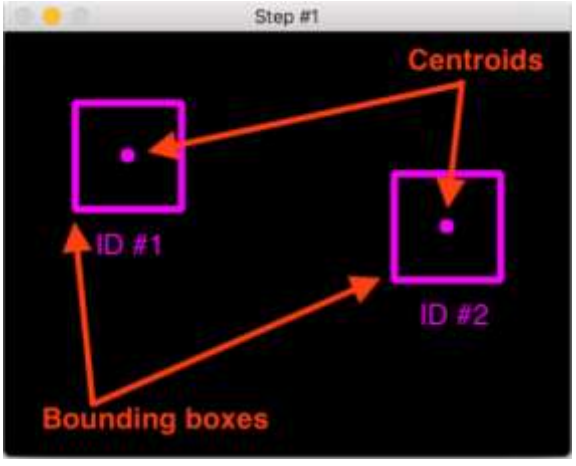


Рисунок 1.10 – Перша фаза програми-аналогу

Друга фаза програми-аналогу зображена на (рис 1.11).

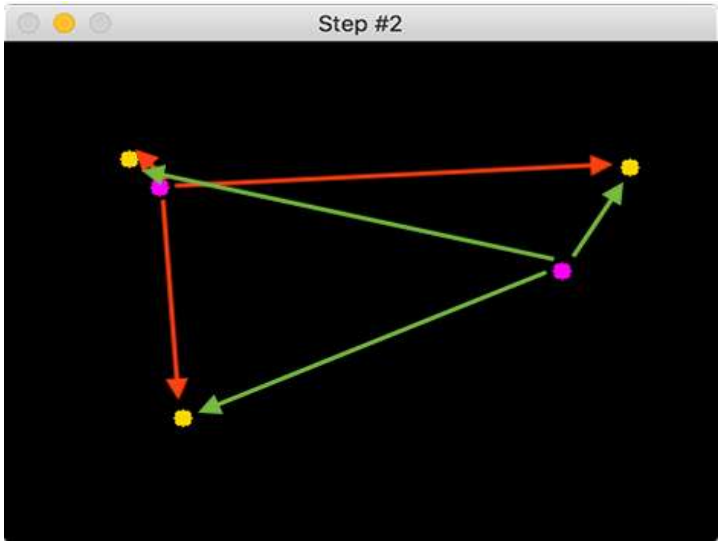


Рисунок 1.11 – Друга фаза програми-аналогу

Вікно вибору файлу для обробки зображено на (рис 1.12).

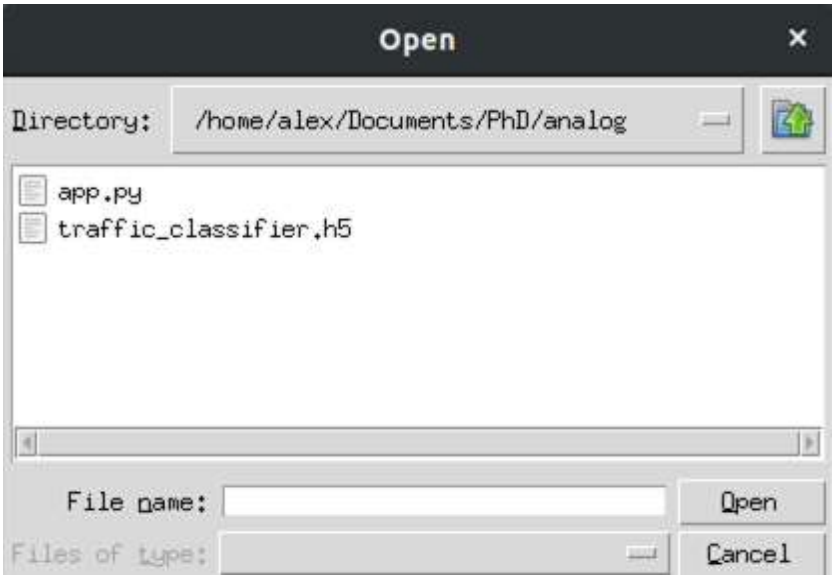


Рисунок 1.12 – Вибір файлу для обробки

Результат роботи програми аналога зображено на рис. 1.13.

Рисунок 1.13 – Результат роботи програми аналога

Розробники стверджують про 82% точності у даної розробки.

1.6 Висновки до розділу 1

Таким чином у цьому розділі було виконано дослідження найкращих підходів вирішення задачі автоматизованого розпізнавання об’єктів у відеопотоці. Досліджено архітектури CNN, RNN, LSTM, їх переваги і недоліки. Було описано відкриті набори даних та вибраний датасет Duke MTMC для навчання нейромережі для розпізнавання людей у відеопотоці. В якості системи-аналогу вибрано веб-ресурс PyImageSearch People Counter.

2 МОДЕЛЮВАННЯ ІНФОРМАЦІЙНОЇ ТЕХНОЛОГІЇ РОЗПІЗНАВАННЯ ОБ'ЄКТІВ У ВІДЕОПОТОЦІ

2.1 Вибір базової архітектури нейронної мережі

Для досягнення поставлених цілей в даній роботі використовується моделі нейромереж, які навчаються за принципом supervised learning (навчання з учителем). В процесі навчання таких нейронних мереж використовується розмічений набір даних, що називається датасетом. Він складається із зображень і відповідного їм масиву лейблів, що вказують на клас об’єкта, що знаходиться на зображенні. В рамках цієї роботи це буде ідентифікатор певної людини.

Проте перед тим як подати класи в нейромережу, потрібно форматувати мітки із текстового формату в чисельний, проте не можна поступити найпростішим шляхом і замінити мітку на її номер за яким вона знаходиться в списку класів, оскільки нейромережа в процесі навчанні буде сприймати, що мітка N якісно перевершує собою мітку N-1. Для усунення цієї проблеми буде використано метод OneHotEncoder, в процесі якого мітки виражаються у вигляді одиничних векторів базису векторного простору із розмірністю рівній кількості координат, і подається в нейромережу у вигляді матриці.

При supervised learning масив даних рекомендується розділяти на три частини – навчальну, тестову і валідаційну вибірку.

Навчальна вибірка (training sample) – вибірка, за якої відбувається налаштування (оптимізація параметрів) моделі залежності. Якщо модель залежності побудована за навчальною вибіркою X^m, то оцінка якості цієї моделі, зроблена по тій же вибірці X^m виявляється, як правило, оптимістично зміщеною. Це небажане явище називають перенавчанням (overfiting). На практиці воно зустрічається дуже часто. Хорошу емпіричну оцінку якості побудованої моделі дає її перевірка на незалежних даних, які не використовувалися для навчання.

Тестова (або контрольна) вибірка (test sample) – вибірка, за якої оцінюється якість побудованої моделі. Якщо навчальна і тестова вибірки незалежні, то оцінка, зроблена за тестовою вибіркою, є не зміщеною. Оцінку якості, зроблену за тестовою вибіркою, можна застосувати для вибору найкращої моделі. Однак тоді вона знову опиниться оптимістично зміщеною. Для отримання не зміщеної оцінки обраної моделі доводиться виділяти третю вибірку.

Перевірочна вибірка (validation sample) – вибірка, за якою здійснюється вибір найкращої моделі з безлічі моделей, побудованих за навчальною вибіркою.

В процесі навчання, за допомогою специфічного для конкретного алгоритму правила, параметри нейронної мережі налаштовуються з використанням навчальної вибірки таким чином, щоб отримавши в якості вхідних даних зображення, модель на виході виробляла б мітку відповідного класу (рис. 2.1).

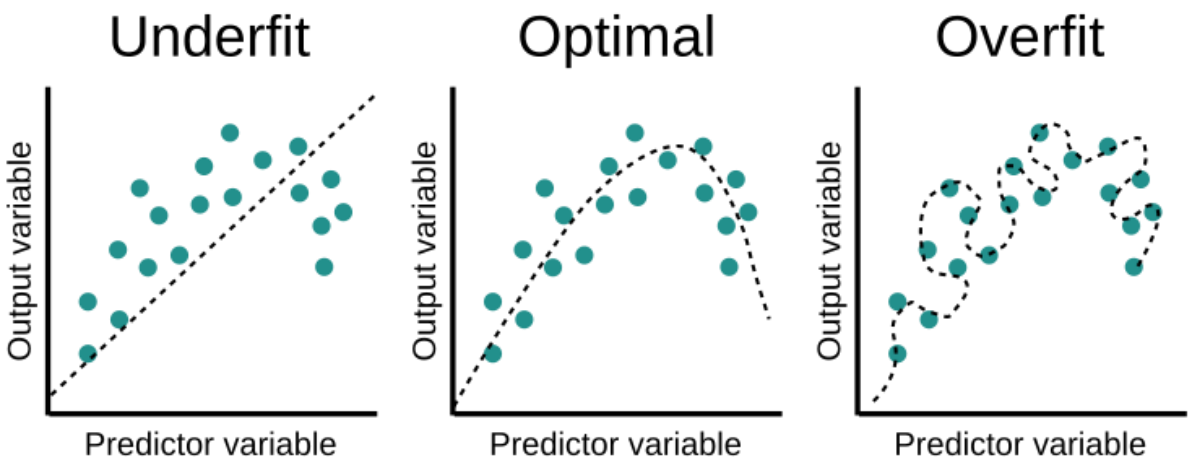


Рисунок 2.1 – Апроксимація функції нейромережі

Безліч моделей використовують даний підхід, наприклад штучна нейронна мережа (багатошаровий перцептрон), регресивна модель, дерева прийняття рішень, метод опорних векторів і моделі-ансамблі, що представляють собою поєднання деяких перерахованих моделей. В даній роботі буде використано штучну нейронну мережу на основі згорткових нейронів.

VGG та ResNet є найпоширенішими архітектурами штучних нейронних мереж на основі згорткових нейронів. Вони і будуть використані в даній роботі для вирішення задачі реідентифікації людей.

VGG – це згорткової нейронна мережа з глибиною 19 шарів (рис. 2.2). Вона була побудована і навчена К. Симоняном і А. Зіссерманом в Оксфордському університеті в 2014 році. Мережа VGG-19 навчена з використанням більш одного мільйона зображень з бази даних ImageNet. Вона навчалася на кольорових зображеннях розміром 224x224 пікселів. Звичайно, ви можете імпортувати модель ImageNet з уже навченими вагами. Ця попередньо навчена мережа може класифікувати до тисячі об'єктів. Модель досягає точності 92.7% – топ-5, при тестуванні на ImageNet в задачі розпізнавання об'єктів на зображенні. VGG16 – одна з найзнаменитіших моделей, відправлених на змагання ILSVRC-2014. Вона є покращеною версією AlexNet, в якій замінені великі фільтри (розміру 11 і 5 в першому і другому згорткові шари, відповідно) на кілька фільтрів розміру 3x3, наступних один за іншим.

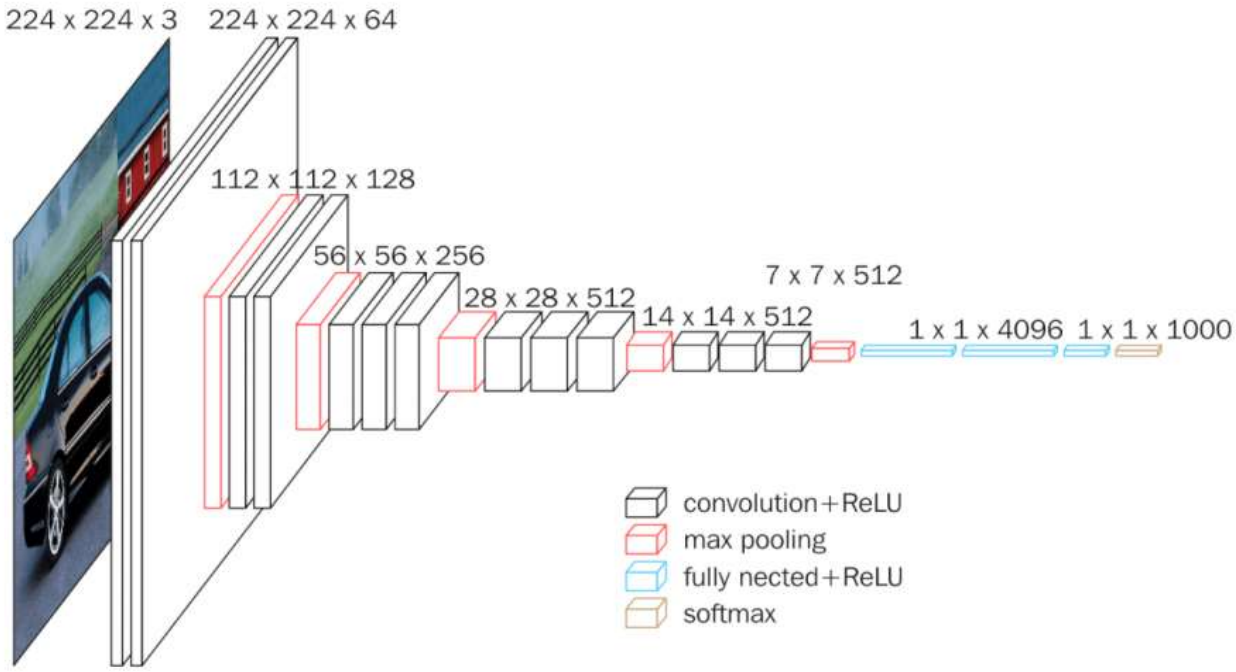


Рисунок 2.2 – Архітектура VGG

Недоліки VGG:

- повільний процес навчання.
- з'являються проблеми з диском і пропускною спроможністю, оскільки ваги мережі важать надто багато.
- при обмежених ресурсах рекомендується використовувати менші конфігурації нейронної мережі, наприклад VGG8.
- на сьогодні дана архітектура нейромережі не використовується в задачах розпізнавання, бо являється застарілою, проте вона є чудовим вибором для створення автоенкодерів, тому VGG все ще актуальний для розгляду у завдані реідентифікації, що і буде продемонстровано далі в даній роботі.

Штучні нейронні мережі, які навчаються за допомогою механізму back propagation, витіснили будь які інші підходи в багатьох задачах штучного інтелекту, зокрема розпізнавання та оцінки параметрів. Але такі нейромережі мають певні недоліки, які здаються непереборними:

- складність багатокомпонентного алгоритму, тобто складно локалізувати джерело проблеми, або локально довчити нейрмережу на одному з рівнів (доводиться просто перенавчати всю нейромережу після додавання нових даних в навчальну вибірку), тобто проблема «чорного ящика»;
- нестійкість при обробці вхідних даних;
- ігнорується статистична природа спостережуваних даних.

Функція розподілу випадкових величин або щільність розподілу займає дуже важливе місце в теорії прийняття рішень. Необхідно оцінити функцій розподілу для розрахунку апостеріорного ризику.

Автоенкодер – це інструмент для відтворення введення. Наприклад, машина робить зображення і може створити тісно пов’язану картину. Вхід у цю нейронну мережу не є маркованим, тобто мережа здатна навчатися без учителя. Отже, вхід кодується мережею, щоб зосередити увагу лише на найважливішій функції. Це одна з причин, чому автоенкодер популярний для зменшення розмірності. Крім того, автоенкодери можуть бути використані для створення генеративних моделей

навчання. Наприклад, нейронну мережу можна навчити з набором облич, а потім можна створити нові обличчя [7].

Мета автоенкодера – виробляти приближення введення, зосереджуючись лише на суттєвих ознаках. Можемо подумати над тим, щоб навчитися копіювати і вставляти дані для отримання результату. Фактично, автоенкодер – це набір обмежень, які змушують мережу вивчати нові способи подання даних, відмінні від простого копіювання результатів [7].

У сумі можна зазначити, що підхід автоенкодерів чудово підходить для вирішення поставленої задачі. І, хоча час на оптимізацію параметрів мережі витрачається значно більше, автоенкодери володіють однією важливою перевагою: створювати якісні ембедінги для нових даних, в тому числі і для помилкових даних, наприклад «не людей». Строго кажучи, реконструювати можна зображення із будь яких даних, але що робити в такій ситуації буде показано далі.

Оскільки в даній роботі практично відсутня можливість створювати або хоча б доповнювати існуючий набір даних, то автоенкодер являється оптимальним рішенням для розпізнавання об’єктів в відеопотоці, тому що дозволяє згладити недоліки датасету.

Алгоритмічно автоенкодер - це послідовне застосування функції енкодера $z = g(x)$ і декодера $\hat{x} = f(z)$, де x- вхідний вектор, а z- латентне уявлення. В певній підмножині вхідних даних, яка як правило, дуже близька до навчальної, $\hat{x} = x + n = f(g(x))$, де n – це нев’язка, тобто Гаусовий шум, його параметри можна оцінити після навчання автоенкодера. Важливим фактором є те, що на сам автоенкодер не накладається ніяких обмежень. А далі можна отримати якісну оцінку щільності ймовірності p(x), на підставі якої можна охарактеризувати якість нейромережі при навчані.

Такий підхід дозволяє якісно покращити роботу нейромережі та покращує вихідні ембедінги, а також зменшує вимоги до набору навчальних даних.

2.2 Використання згорткової нейронної мережі для розпізнавання об’єктів

На протязі останніх років штучні нейронні мережі стають все більш популярними і отримали величезний розвиток, що призвело до вагомих результатів у вирішенні багатьох проблем, що постають перед людством, наприклад, таких як розпізнавання голосу та комп’ютерний зір. Одним із факторів, що призвели до цього, є нейронна мережа, яка називається згортковою. Якщо найпростіше описати CNN, то це мережа, яка використовує багато копій одного нейрона та дозволяє працювати з великими моделями і зберігати однакову кількість параметрів.

Згортка - це операція, яку можна застосувати до двох послідовностей (наприклад, f і g), щоб створити третю послідовність:

$$(f * g)(c_1, c_2) = \sum f(a_1, a_2)\, g(c_1 - a_1, c_2 - a_2)$$

де f – функція визначення яскравості пікселя,

g – функція нового стану пікселя,

a_1, a_2, c_1, c_2 – коефіцієнти.

Згортокова нейронна мережа - одна з архітектур штучних нейронних мереж, створена для аналізу та розпізнавання зображень. Спочатку використовуються згорткові шари, потім RELU, шари об’єднання та вихідні шари (рис. 2.3).

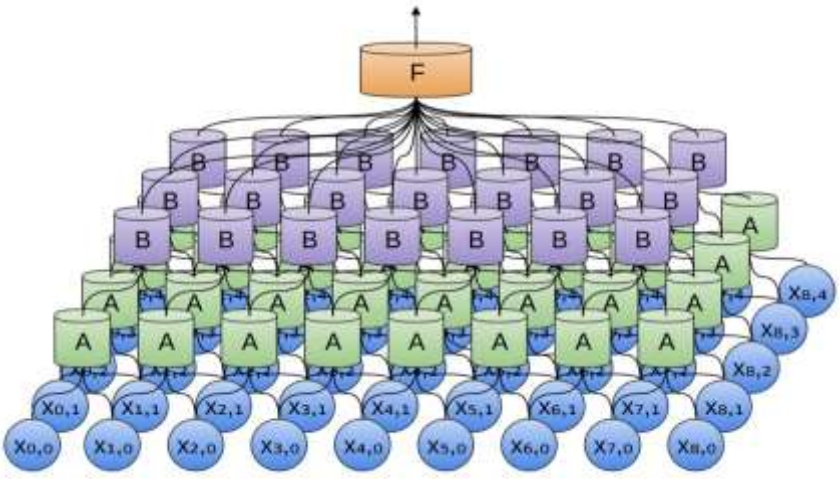


Рисунок 2.3 – 2D зображення CNN

Нейрони згорткової НМ розташовані в трьох вимірах, як показано на одному з шарів. У цьому прикладі червоний вхідний шар містить зображення, тому його ширина та висота визначаються розміром зображення, а глибина має значення 3 (рис. 2.4).

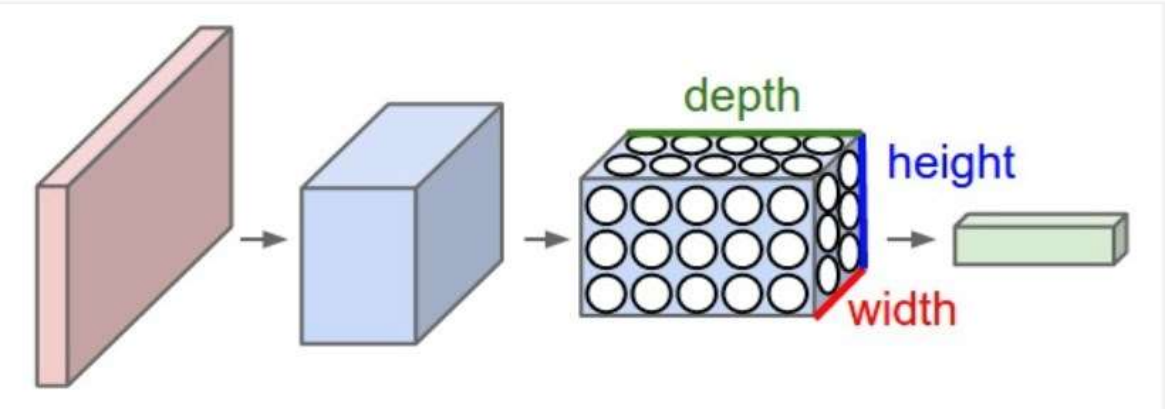


Рисунок 2.4 – 3D згортка

Для того, щоб краще зрозуміти, як це працює, розглянемо одновимірний згортковий шар з кількістю входів n і одним виходом, який має назву F (хоча тут можуть бути вихідні нейрони кількістю n), як зображено на рисунку 2.5.

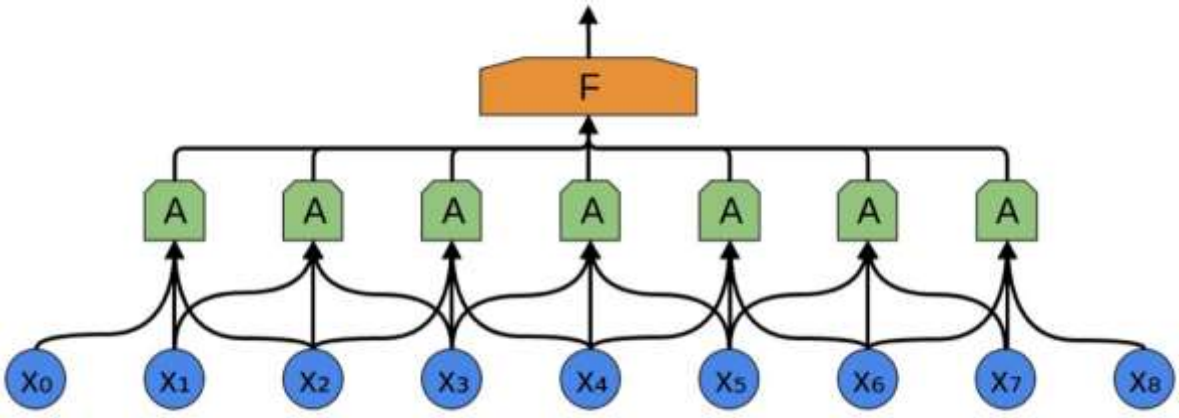


Рисунок 2.5 – Одновимірний вид шару згорткової нейронної мережі

Раніше ми говорили, що в середині згорткового шару є багато копій того самого нейрона, тому результат може мати однакові ваги для деяких у

$$Y_1 = f(W_0x_0+W_1x_1+W_2x_2+b)$$
$$Y_2 = f(W_1x_1+W_2x_2+W_3x_3+b)$$

де W – значення ваг,

x – значення вхідних даних.

Вагова матриця з’єднує кожен вхід із кожним нейроном. Матриця згорткового шару має властивість, що в деяких місцях вона може дорівнювати 0, оскільки нейрони не підключені до всіх можливих входів, і ваги в матриці можна знайти кілька разів. Операція згортки використовує невелику вагову матрицю, яка рухається вздовж шару, який ми обробляємо, а потім створює сигнал активації для нейрона наступного шару. Цю матрицю називають ядром (kernel), і для її характеристики також використовують термін розмір ядра (kernel size) - він характеризує розмірність матриці. У випадку, коли ми маємо один канал, терміни фільтр і ядро мають однакове значення, але коли ми розглядаємо, наприклад, кольорове зображення, де є більше ніж один колірний канал, фільтр є набором ядер, кожне з яких відповідає канал зображення.

У процесі виконання операції згортки кожен фрагмент, накладений на ядро, множиться елемент за елементом матриці згортки, а потім ці значення додаються і зберігаються у відповідній позиції. Отриманий шар вказує, чи існує дана функція. У CNN існує багато наборів ваг, які створюються під час навчання нейронної мережі. В результаті після проходження через ядро створюється карта ознак і створюється певна їх кількість, в результаті ми маємо багато карт ознак на одному згортковому шарі і таким чином нейронна мережа стає багатоканальною.

Поряд із шарами згортки розміщується шар об’єднання (шар pooling), який служить для зменшення розміру карт ознак, що дозволяє зменшити кількість обчислень, параметрів мережі, а також контролювати та запобігати повторному навчанню. Цей шар просторово масштабує обсяг за допомогою функції максимуму. Береться фільтр певного розміру, наприклад 2x2 і крок 2, і аналізуються 4 числа, з яких вибирається найбільше (рис. 2.6).

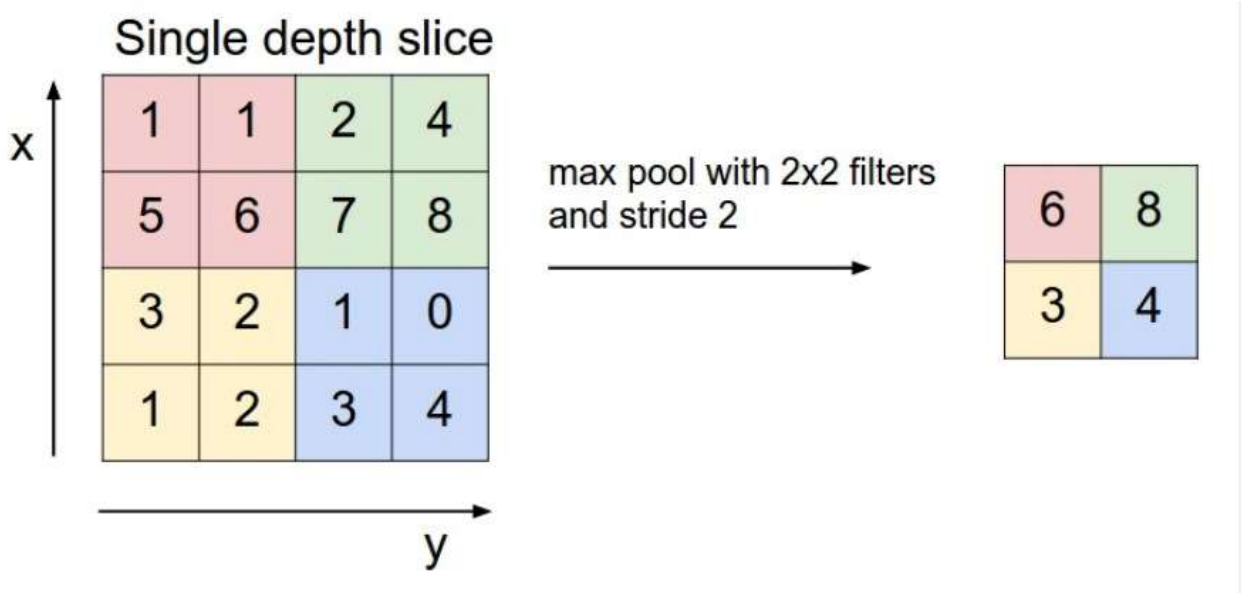


Рисунок 2.6 – Приклад роботи шару об’єднання (pooling шару)

Пройшовши певну кількість шарів, карта ознак витягується у вектор або навіть скаляр, але таких карт досить багато, тому вони потім переносяться на кілька рівнів нейронної мережі, таких як наприклад, Perceptron.

2.3 Математична модель оптимізації навчання нейронної мережі

Штучний інтелект (ШІ) став невід’ємною частиною нашого повсякденного життя, забезпечуючи роботу різноманітних додатків, від віртуальних помічників до безпілотних автомобілів. Одним із найбільш значних досягнень ШІ є розробка моделей глибокого навчання, які продемонстрували надзвичайні можливості у вирішенні складних проблем. Однак навчання цих моделей може бути дорогим з точки зору обчислень і займати багато часу, що робить важливим пошук ефективних алгоритмів оптимізації, які можуть прискорити процес навчання. Одним із таких алгоритмів, який набув широкої популярності в останні роки, є оптимізатор Adaptive Moment Estimation (Adam).

У 2014 році Diederik P. Kingma та Jimmy Lei Ba запропонували метод адаптивної оцінки моментів Adam [10]. За словами авторів, метод поєднує переваги методів AdaGrad та RMSProp і добре працює при on-line навчанні. Даний метод

показує стабільно позитивні результати на різних вибірках та останнім часом рекомендується до застосування за замовчуванням у різних пакетах.

В основі оптимізатора лежить розрахунок експоненціальної середньої градієнта m та експоненціальної середньої квадратів градієнта v. Кожна середня експоненціальна має свій гіперпараметр ß, що визначає період усереднення.

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t$$
$$v_t = \beta_2 v_{t-1} + (1 - \beta_2) g_t^2 \quad (2.1)$$

Пропонується використовувати за замовчуванням ß1 на рівні 0,9 та ß2 на рівні 0,999. При цьому m0 та v0 приймають нульові значення. З такими параметрами формули 2.1 на початку навчання повертають значення близькі до "0" і, як наслідок, ми отримуємо низьку швидкість навчання на початковому етапі. Для прискорення навчання пропонується скоригувати отримані моменти.

$$\hat{m} = {}^{m_t}/_{1 - \beta_1^t}$$
$$\hat{v} = {}^{v_t}/_{1 - \beta_2^t} \quad (2.2)$$

Оновлення параметрів здійснюється шляхом коригування відношення скоригованого моменту градієнта m до кореня квадратного з скоригованого моменту квадрата градієнта v. Для виключення ділення на нуль до знаменника додають константу Ɛ, близьку до "0". Отримане відношення коригується коефіцієнтом навчання α, який у даному випадку виступає верхньою межею кроку навчання. За замовченням пропонується використовувати α на рівні 0,001.

$$w_{t+1} = w_t - \alpha \frac{\hat{m}}{\sqrt{\hat{v}} + \varepsilon} \quad (2.3)$$

Порівняння різних технологій оцінки роботи нейромережі зображено на рисунку 2.7.

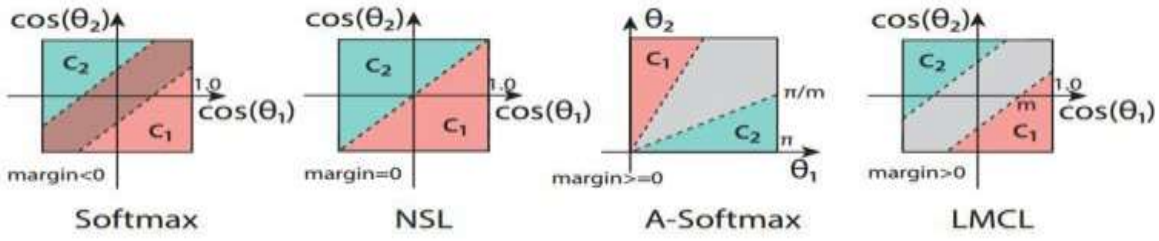


Рисунок 2.7 - Порівняння різних технологій оцінки роботи нейромережі

Одною з останніх розробок в напрямку нейромереж автноенкодерів являється технологія Large Margin Cosine Loss. Це підхід, при якому вектор виходу нейромережі проектується на сферу описану у векторному просторі R^N, де N - це задана величина вектора.

На відміну від звичного підходу, що тільки нормалізує вагові вектори, метод LMCL використовує схему нормалізації для отримання рецептури косинуса помилки та усунення варіацій у радіальних напрямках. LMCL нормалізує і вагові вектори і функціональні вектори одночасно. В результаті вектори функцій розподіляються по гіперсфері, де параметр масштабування s контролює величину радіуса гіперсфери.

Початкові втрати без нормування функції неявно засвоюють як евклідову норму (L2-норму) функціональних векторів, так і значення косинуса кута. Норма L2 адаптивно засвоюється для мінімізації загальних втрат, що призводить до відносно слабкого косинусного обмеження. Зокрема, адаптивна швидкість L2 для зразків світла стає набагато вищою, ніж тверді зразки для корекції нижчих показників косинусів. Навпаки, LMCL вимагає, щоб весь набір функціональних векторів мав однакову норму L2, так що навчання залежить лише від значень косинусів для розвитку дискримінантної сили [12]. Характеристики векторів з тих самих класів згруповані, а ті, що знаходяться в різних класах, розділені на поверхню гіперсфери, як показано на рисунку 2.8.

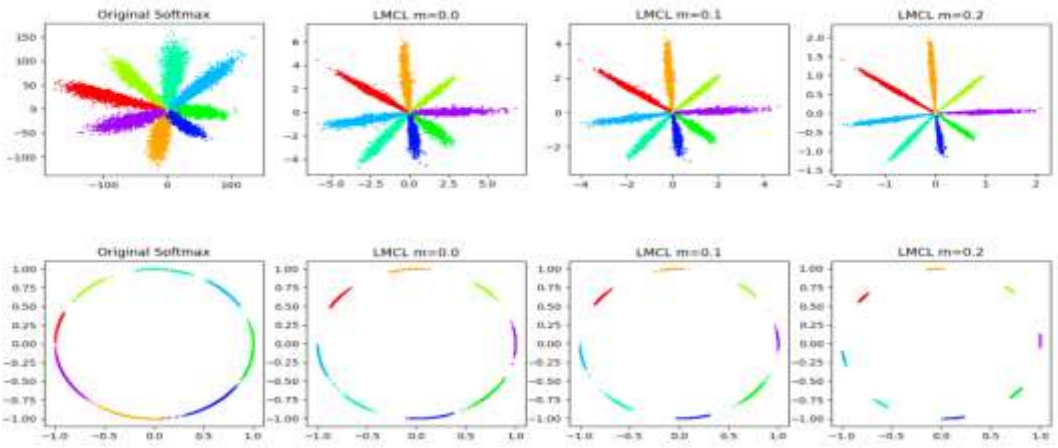


Рисунок 2.8 — Результат роботи LCML

На рисунку 2.9 зображено геометричний зміст LCML.

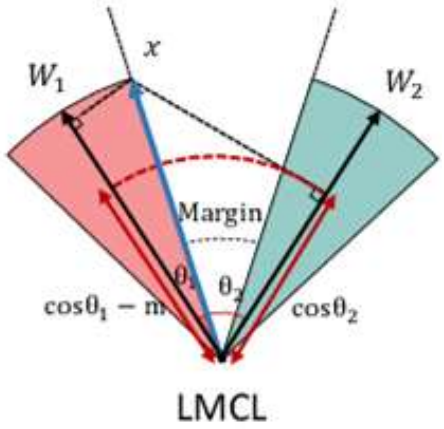


Рисунок 2.9 - Геометричний зміст LCML

В подальшому косинусна функція може приймати вигляд інших тригонометричних виразів. Із ефективних варіацій існують методи CosFace, ArcFace та SphereFace [13]. Варіації імплементації LCML (рис 2.10)

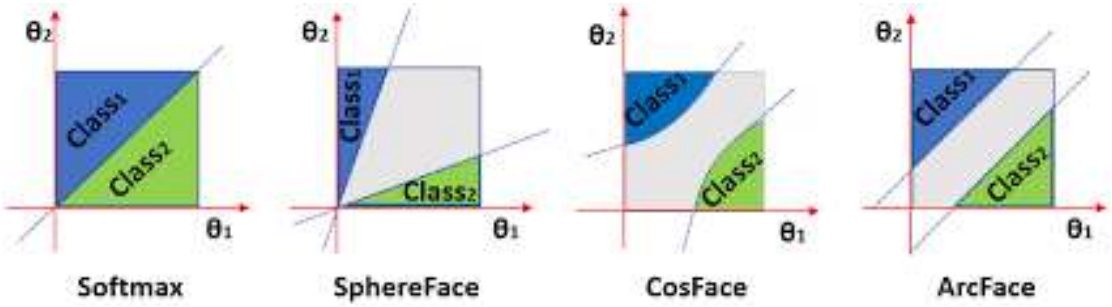


Рисунок 2.10 - Варіації імплементації LCML

Adam (Adaptive Moment Estimation) визначається своєю здатністю адаптивно налаштовувати швидкість навчання для кожного параметра окремо, що робить його потужним інструментом у контексті стохастичного градієнтного спуску[14].

Однією з ключових особливостей Adam є використання моменту, який дозволяє враховувати не лише поточний градієнт, а й інформацію про минулі градієнти. Це дозволяє алгоритму швидше пристосовуватися до змін у градієнтах та підвищує його стабільність під час оптимізації.

Adam також включає в себе елемент адаптивності швидкості навчання, що дозволяє кожному параметру мати власний рівень апдейту. Це особливо корисно у випадках, коли різні параметри мережі мають різні масштаби або знаходяться на різних стадіях навчання.

Загалом, Adam вважається потужним інструментом для оптимізації нейронних мереж і здатен прискорити збіжність під час тренування, забезпечуючи ефективне та стабільне навчання моделей глибокого навчання.

Adam – адаптивна оцінка моменту, це алгоритм оптимізації, який поєднує ідею накопичення руху та ідею слабшого відновлення ваги для типових особливостей. Від інших методів оптимізації Adam відрізняється тим, що накопичує не $\Delta\theta$, а значення градієнта. Крім того, метод контролює, як часто змінюється градієнт. Автори алгоритму запропонували оцінити середню нецентровану дисперсію. Від інших методів Adam відрізняється невибагливістю і простотою у налаштуванні. Для налаштування методу, досить задати початковий імпульс. Тому чудово підходить для використання у задачах реідентифікації. Порівняння різних оптимізаційних методів приведено на рисунках нижче. Робота методів оптимізації на поверхні “Басейни та стіни” (рис 2.11).

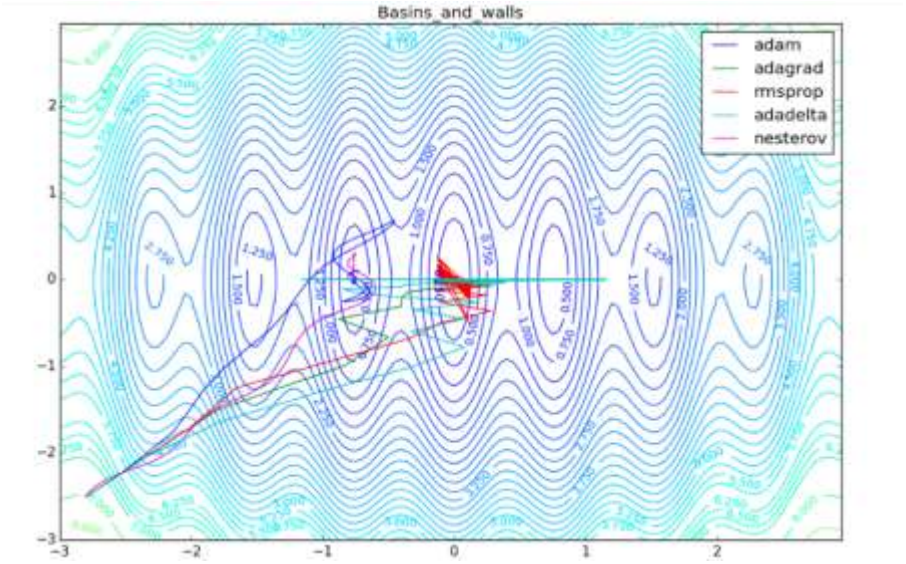


Рисунок 2.11 – Робота методів оптимізації на поверхні “Басейни та стіни”

Метод “Пагорби та каньйони” (рис. 2.12).

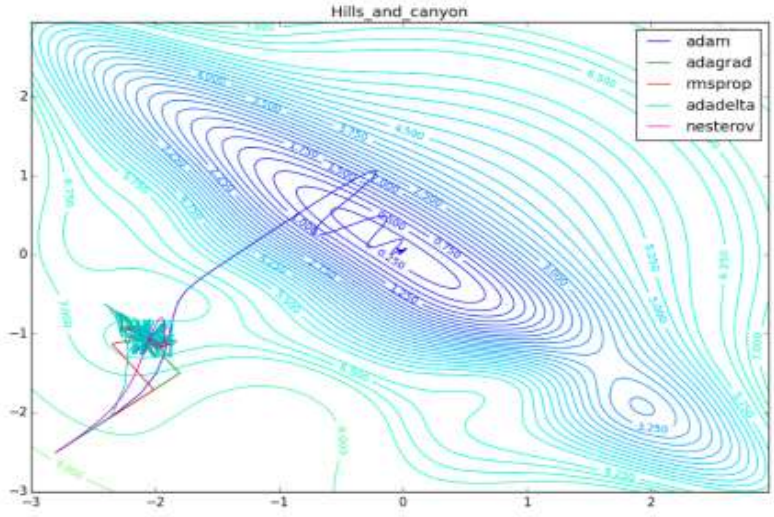


Рисунок 2.12 – Робота методів оптимізації на поверхні “Пагорби та каньйони”

Таким чином метод оптимізації Adam, являється оптимальним рішенням для навчання нейромережі, тому і буде використаний далі.

2.4 Розробка алгоритму функціонування інформаційної технології розпізнавання об’єктів у відеопотоці

Робота інформаційної технології автоматизованого підрахунку людей у відеопотоці є багатоступеневим процесом. Крім того, він повинен бути універсальним для різних типів вхідних даних

Вхід для читання вимагає обробки багатьох відеоформатів. Незважаючи на свою складність, це завдання можна повністю залишити сучасним бібліотекам для роботи з комп'ютерним зором, таким як OpenCV та PIL. Однак зчитувані дані можуть мати будь-який розмір, який може не збігатися з розміром нейронної мережі, тому зображення потрібно відрегулювати до бажаного розміру. Ця операція може знизити якість модуля, тому рекомендується використовувати збір даних потрібного розміру, але за відсутності альтернатив найкращим рішенням є зміна розміру. Також, залежно від мережі, може знадобитися змінити канал зображення або його нормалізацію.

Перевизначення досягається за допомогою нейронних мереж сімейств R-CNN, YOLO та SSD. Цей етап досить ресурсозатратний, оскільки вимагає значних обчислювальних потужностей, але в цій роботі цей етап не буде реалізований.

Потім результат роботи нейронної мережі детектора розрізається на окремі частини і подається в нейронну мережу–реідентифйікатор, але така частина також повинна бути оброблена перед розпізнаванням. Типовими операціями є нормалізація, зміна розміру, зміна колірної схеми. При використанні автокодерів нейромережевий класифікатор ділиться на два етапи:

- Виділення знаків
- Фактична повторна ідентифікація

Відповідна схема алгоритму показана на рисунку 2.9.

Рисунок 2.9 – Алгоритм розпізнавання об’єктів у відеопотоці

2.5 Висновки до розділу 2

У даному розділі було визначено архітектуру нейромережі, а саме автоенкодер на основі технології VGG, визначена математична модель для її навчання, а також метод оптимізації ADAM та критерій LCMC. Було розроблено алгоритм розпізнавання об’єктів у відеопотоці.

3 ПРОГРАМНА РЕАЛІЗАЦІЯ ІНФОРМАЦІЙНЇ ТЕХНОЛОГІЇ РОЗПІЗНАВАННЯ ОБ'ЄКТІВ У ВІДЕОПОТОЦІ

3.1 Обґрунтування вибору мови, середовища та бібліотек програмування

Мова програмування – це сукупність лексичних абстракцій, що забезпечує зручний опис конкретних проблем, які формуються людиною і потребують вирішення за допомогою комп'ютера. Мова програмування – це програма, яка дозволяє комп'ютеру отримувати конкретні результати. Мови програмування мають дві складові: синтаксис та семантику [10].

Ця нейронна мережа написана мовою програмування Python 3. Python – це високорівнева мова програмування загального призначення, який використовується в тому числі і для розробки веб-додатків. Мова орієнтований на підвищення продуктивності розробника і читання коду.

Правильне українська вимова назви мови програмування – Пайтон, але частіше використовується спотворене – Пітон.

Python підтримує кілька парадигм програмування: структурний, об'єктно-орієнтоване, функціональне, імперативне і аспектно-орієнтоване. У мові застосовується автоматичне керування пам'яттю та динамічна типізація даних, інтроспекція та механізм обробки різноманітних виключень, а також підтримка багатопотокових обчислень і зручні високорівневі структури даних. Програмний код на Python організовується у функції та класи, які можуть об'єднуватися в модулі, а вони в свою чергу можуть бути об'єднані в пакети. Python зазвичай використовується як інтерпретується, але може бути скомпільовано в байт-код Java і в MSIL (в рамках платформ .NET).

Розробники мови програмування Python дотримуються такої філософії програмування, яку називають «Джен пітона» («The Zen of Python»):

- красиве краще, ніж негарне;
- явне краще, ніж приховане;

- простіше краще, ніж складне;
- складніше краще, ніж заплутане;
- прямота краще, ніж вкладеність;
- тонке краще, ніж товсте;
- читабельність має значення;
- особливі випадки не настільки виняткові, щоб порушувати правила;
- практичність важливіша за досконалість;
- помилки ніколи не слід замовчувати;
- Коли є двозначністю, треба утриматися від спокуси вгадування;
- має бути один і тільки один очевидний спосіб зробити це;
- краще зараз, ніж взягалі ніколи;
- хоча інколи ніколи, краще, ніж зараз;
- якщо реалізацію важко пояснити, ідея погана;
- якщо реалізацію легко пояснити, ідея, ймовірно, хороша;
- просторові імена – це чудово!

Порівняльний опис мов програмування Python 3, Julia та R наведено в таблиці 3.1.

Таблиця 3.1 – Порівняльна характеристика мов програмування

		Java	

Отже, Python 3 можна вибрати для реалізації автоматизованих модулів підрахунку у відеопотоках, оскільки ця мова більше підходить для розробки нейронних мереж, має різноманітні засоби обробки даних і є простішою для розуміння та дуже проста у використанні.

Для розробки цієї системи було використано два середовища розробки: Microsoft Visual Studio Code та Jupyer Notebook. Microsoft Visual Studio Code – це набір продуктів Microsoft, що включає інтегроване середовище розробки програмного забезпечення та багато інших інструментів. Ці продукти дозволяють розробляти як консольні програми, так і програми, що використовують графічні інтерфейси, а також розробляти веб-сайти, веб-додатки, веб-сервіси для всіх відповідних платформ [18].

3.2 Вибір програмних інструментів

Для реалізації програмного забезпечення розпізнавання об’єктів у відеопотоці було обрано інструменти: Pandas, Numpy та PyTorch.

Pandas — це бібліотека програмування Python для обробки й аналізу даних. Обробка даних pandas покладається на бібліотеку NumPy, яка є інструментом нижчого рівня. Надає спеціальні структури даних і операції для роботи з числовими таблицями та часовими рядами. Назва бібліотеки походить від економетричного терміну «панельні дані», який використовується для опису багатовимірних, структурованих наборів інформації. pandas розповсюджується за новою ліцензією BSD.

Основна область застосування полягає в тому, щоб дозволити роботу в середовищі Python не тільки для збору та очищення даних, але й для завдань аналізу даних і моделювання без необхідності переходу на більш детальні статистичні мови (такі як R і Octave).

До основних можливостей бібліотеки можна віднести:

- об’єкт DataFrame для роботи з індексованими масивами двовимірних даних;
- засоби для обміну даними між структурами в пам'яті і файлами різних форматів;
- способи об'єднання даних і методи обробки відсутньої інформації;
- переформатування наборів даних, включаючи створення зведених таблиць;
- поділ даних за значеннями індексу, розширені можливості індексування, вибірка з великих наборів даних;

- вставлення та видалення стовпців даних;
- можливості групування дозволяють виконувати триетапну операцію розділення-застосування-злиття;
- комбінування та комбінування наборів даних;
- ієрархічне індексування дозволяє працювати з даними великої розмірності в менших розмірних структурах;
- робота з часовими рядами: створення часових періодів і зміна часових інтервалів.

Бібліотека добре оптимізована для високопродуктивної роботи, найбільш важливі частини програмного коду написані на Cython і мові C. Pandas – це бібліотека Python, що надає широкі можливості для аналізу даних. Дані часто зберігаються в формі таблиць – наприклад, в форматах .csv, .tsv або .xlsx. За допомогою бібліотеки Pandas такі табличні дані дуже зручно завантажувати, обробляти і аналізувати за допомогою SQL-подібних запитів. А в зв'язці з бібліотеками Matplotlib і Seaborn Pandas надає широкі можливості візуального аналізу табличних даних.

NumPy це open-source модуль для Python, який вміщує в собі загальні математичні і числові операції у вигляді скомпільованих функцій, що робить їх дуже швидкими. Вони об'єднуються в високорівневі пакети, які покривають більшу частину потреб в математичних розрахунків. Numpy забезпечує функціонал, який схожий з функціоналом MatLab та надає методи для роботи з великими масивами і матрицями.

Pandas – це бібліотека Python з відкритим вихідним кодом, що надає високопродуктивний інструмент для обробки та аналізу даних із використанням його потужних структур даних. Назва Pandas походить від слова Panel Data – економетрика із багатовимірних даних.

У 2008 році розробник Уес МакКінні розпочав розробку панд, коли їм потрібен високопродуктивний, гнучкий інструмент для аналізу даних.

До Pandas Python в основному використовувався для збору та підготовки даних. Це мало дуже невеликий внесок у аналіз даних. Панди вирішили цю проблему. Використовуючи Pandas, ми можемо виконати п'ять типових кроків для обробки та

аналізу даних, незалежно від походження даних – завантажити, підготувати, маніпулювати, моделювати та аналізувати.

Python з Pandas використовується у широкому спектрі областей, включаючи академічні та комерційні галузі, включаючи фінанси, економіку, статистику, аналітику тощо.

NumPy — одна з найпотужніших бібліотек у Python. Тож давайте розглянемо основні фічі бібліотеки і поширені математичні функції. Виклад у статті буде простим і ця інфомрація стане у пригоді для кращого розуміння Python.

NumPy — бібліотека обчислень для Python з відкритим сирцевим кодом. Містить такі структури даних, як багатовимірні масиви та матриці. Може слугувати для виконання певних математичних операцій над масивами, на зразок тригонометричних, статистичних, алгебраїчних обчислень.

NumPy — це розширення Numeric та Numarray. Бібліотека містить багато математичних, алгебраїчних функцій та функцій перетворення. Також є генератори випадкових чисел. NumPy — обгортка над бібліотекою на C. Об'єкти Pandas дуже залежать від об'єктів NumPy. Pandas розширює NumPy.

NumPy - це бібліотека Python, яку застосовують для математичних обчислень: починаючи з базових функцій і закінчуючи лінійною алгеброю. Повна назва бібліотеки — Numerical Python extensions, або Числові розширення Python.

Ця бібліотека має кілька важливих особливостей, які зробили її популярним інструментом. Вихідний код у вільному доступі зберігається на GitHub, тому NumPy називають open-source модулем для Python. Бібліотека написана мовами C та Fortran. Це компілювані мови (мови програмування, текст яких перетворюється на машинний код — набір інструкцій для конкретного типу процесора. Перетворення відбувається за допомогою спеціальної програми-компілятора, завдяки якому обчислення компілюваними мовами відбуваються швидше), на яких обчислення виробляються набагато швидше і ефективніше, ніж мовами, що інтерпретуються (мови програмування, які не заточені під конкретний тип процесора і можуть бути запущені на різних типах пристроїв). До цих мов належить і сам Python.

PyTorch – це Python бібліотека для глибинного навчання з використанням GPU використовує Torch. Вона була розроблена Адамом Паске, Семом Гроссом, Сумітом Чінтала та Грегорі Шананом (дослідницька група AI у Facebook) з 2016 року.

PyTorch написана на Python, C та CUDA. Бібліотека підтримує бібліотеки прискорення, такі як Intel MKL та NVIDIA (CuDNN, NCCL). В своїй основі, вона використовує CPU і GPU Tensor та NN-пакети (TH, THC, THNN, THCUNN), записані як незалежні бібліотеки на API C99. PyTorch підтримує тензорні обчислення з сильним прискоренням графічного процесора (забезпечує Tensors, який може працювати або на CPU, або на GPU, прискорюючи обчислення), а також глибинні нейронні мережі, побудовані в системі автоградів на основі стрічки. Як правило, змінити поведінку мережі означає написати її з нуля. PyTorch використовує техніку, звану автодиференціацією у зворотному режимі, яка дозволяє змінювати спосіб поведінки мережі з прикладанням відносно невеликих зусиль (тобто динамічний обчислювальний граф або DCG). Бібліотека знаходиться у вільному доступі за ліцензією BSD, і її підтримують Facebook, Twitter, NVidia та багато інших організацій.

PyTorch – це бібліотека машинного навчання з відкритим кодом, заснована на бібліотеці Torch, яка використовується для таких задач, як комп’ютерний зір та обробка природних мов [6], розроблена переважно дослідницькою лабораторією AI Facebook (FAIR). Це безкоштовне програмне забезпечення з відкритим кодом, випущене під модифікованою ліцензією BSD. Незважаючи на те, що інтерфейс Python є головним напрямком розвитку, PyTorch також має інтерфейс C ++. На PyTorch побудовано кілька програм глибокого навчання, зокрема автопілот Tesla, Pyro Uber, Трансформери HuggingFace, PyTorch Lightning, і Catalyst. PyTorch виконує дві функції високого рівня:

- тензорні обчислення (як NumPy) з сильним прискоренням за допомогою графічних процесорів (GPU)
- роботу з глибокими нейронними мережами

Тобто, дані інструментидобр підходять для реалізації модуля автоматизованого підрахунку людей у відеопотоці.

3.3 Навчання нейромережі для розпізнавання об’єктів

Навчання нейромережі для розпізнавання об'єктів — це процес, під час якого штучна нейромережа "навчається" розпізнавати певні об'єкти або класи об'єктів на основі вхідних даних. Цей процес є ключовим у галузі машинного навчання, зокрема в глибокому навчанні, де використовуються глибокі нейронні мережі, такі як згорткові нейронні мережі (CNN) або рекурентні нейронні мережі (RNN).

Вибірка даних має включати зображення об'єктів, які потрібно розпізнавати. Кожне зображення повинно бути позначене (анотоване) інформацією про класи об'єктів, які містяться на зображенні.

Важливо відзначити, що ефективність моделі може залежати від якості та кількості тренувальних даних, а також від правильного вибору гіперпараметрів та налаштувань. Також може бути важливо враховувати проблеми, такі як перенавчання (overfitting) та недонавчання (underfitting).

Перш за все виберемо оптимальний критерій із Softmax, CosFace, ArcFace та SphereFace при латентному векторі розмірності 3, щоб мати можливість візуалізації. Для вибору оптимального критерію реалізуємо навчання VGG8 з однаковими параметрами. Навчання буде проводитися на датасеті Duke MTMC. Графік навчання із критерієм Softmax зображено на рисунку 3.1.

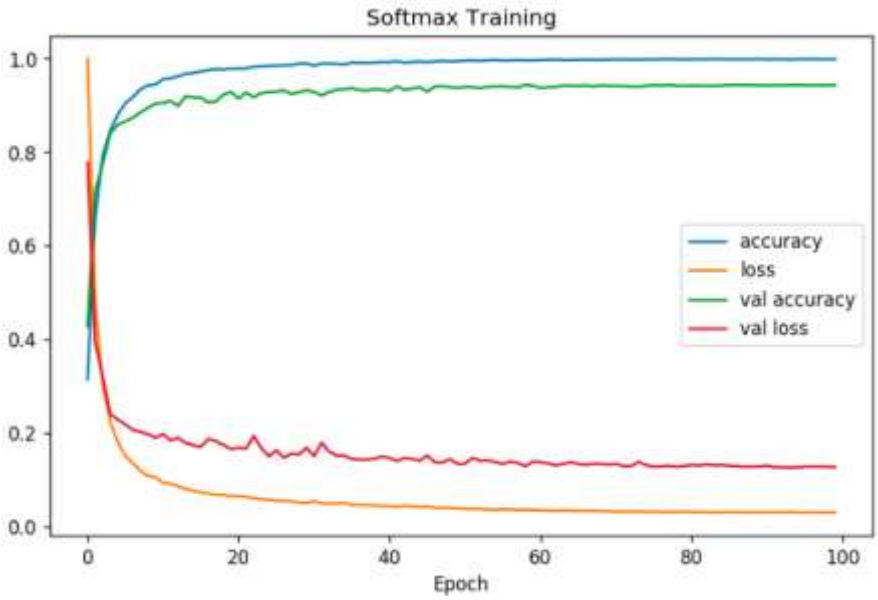


Рисунок 3.1 - Графік навчання нейромережі з критерієм Softmax

Графік навчання із критерієм CosFace зображено на рисунку 3.2.

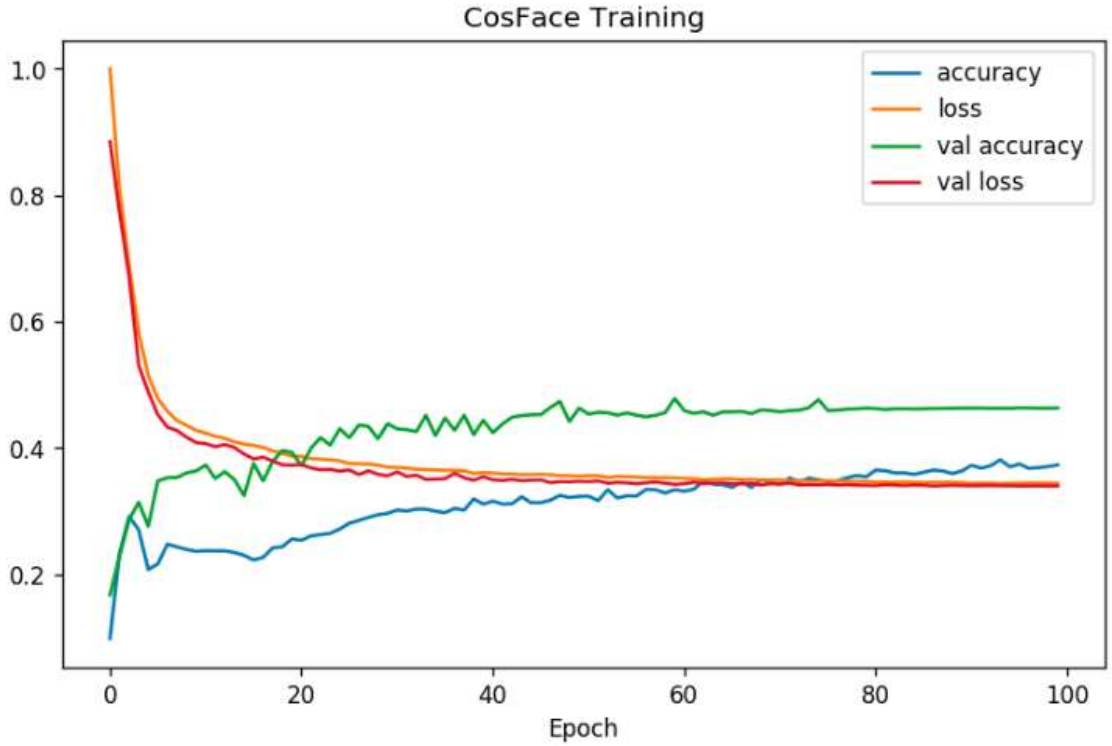


Рисунок 3.2 - Графік навчання нейромережі з критерієм CosFace, латентність – 3

Графік навчання із критерієм ArcFace зображено на рисунку 3.2.

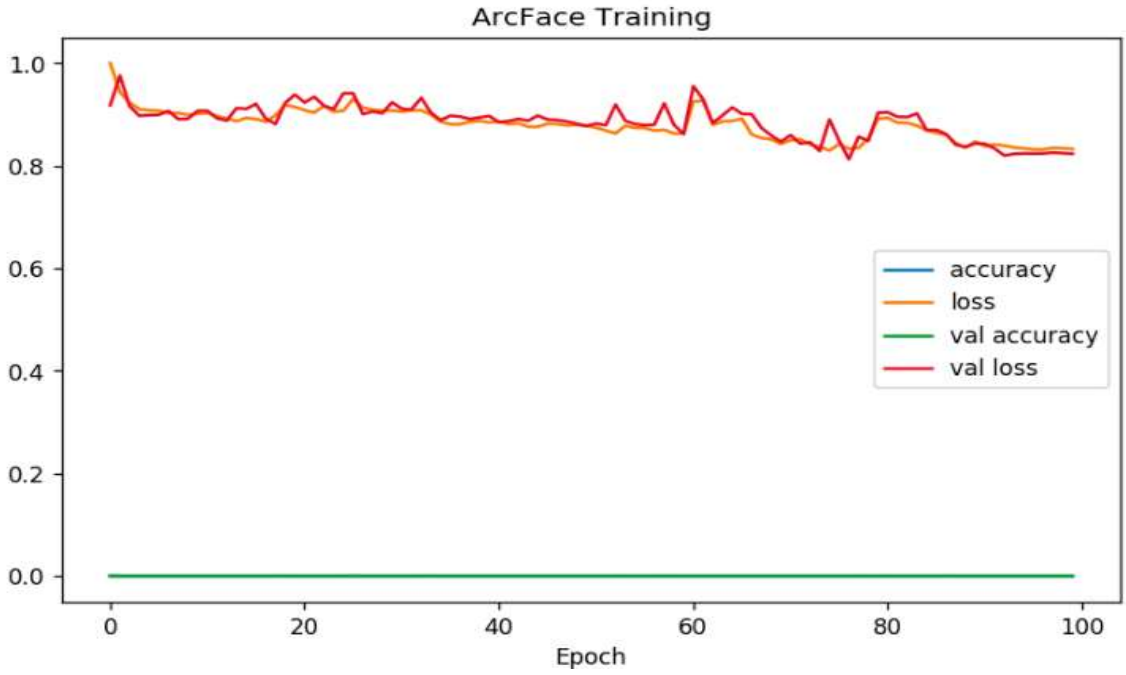


Рисунок 3.3 - Графік навчання нейромережі з критерієм ArcFace, латентність – 3

Графік навчання із критерієм SpereFace зображено на рисунку 3.4.

Рисунок 3.4 - Графік навчання нейромережі з критерієм SpereFace, латентність – 3

3.4 Тестування роботи програмного забезпечення розпізнавання об’єктів у відеопотоці та аналіз результатів

Розроблене програмне забезпечення розпізнавання об’єктів у відеопотоці було протестоване, що підтвердило коректність його роботи. Було проведено 100 запусків програмного додатку, протестовано можливості його роботи, що дало можливість адекватно оцінити його роботу. Після запуску програмного додатку відкривається вікно початкової активності (рис. 3.5).

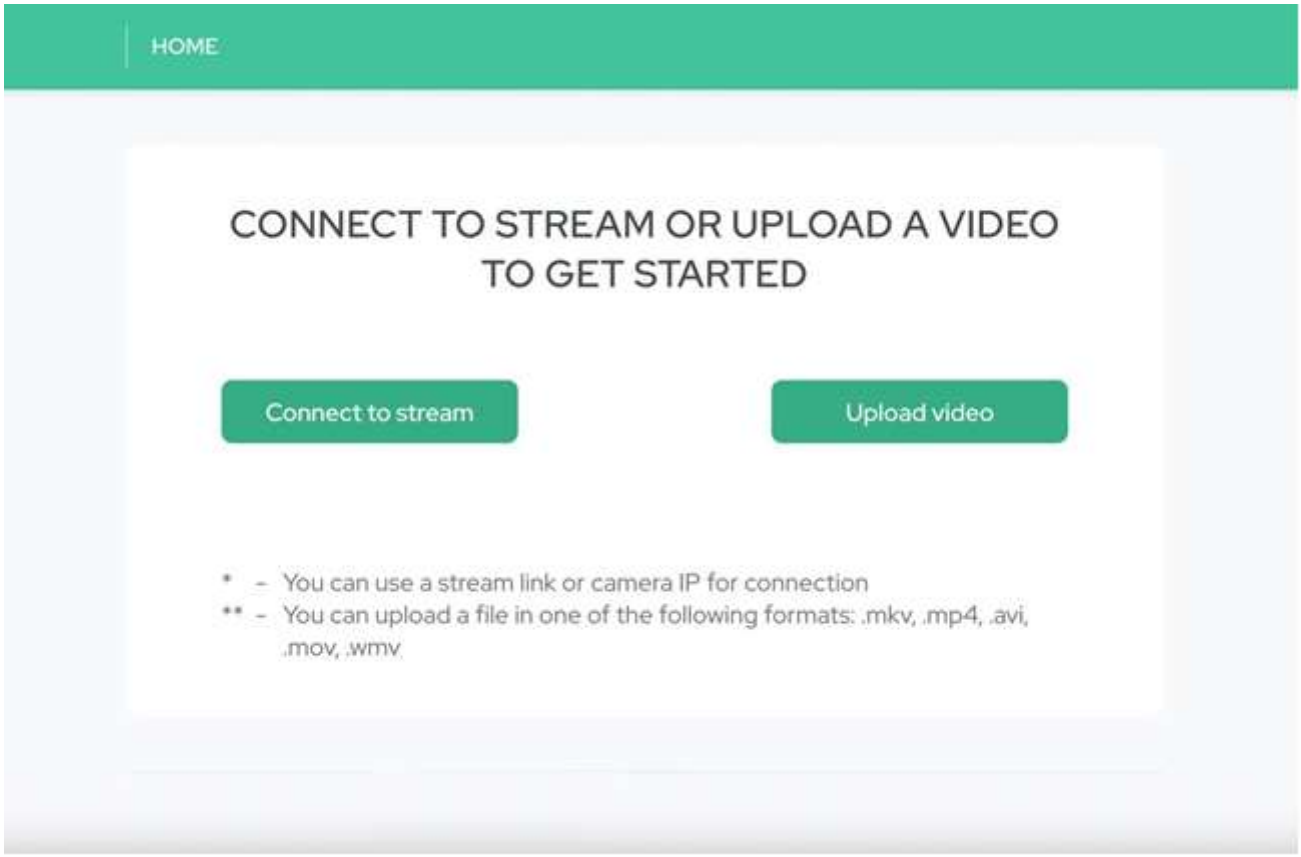


Рисунок 3.5 – Вікно початкової активності

Користувачу необхідно обрати спосіб для введення в програмне забезпечення початкових даних. Наявні можливості як підключення до камери , так і завантаження файлу.

Для наведеного далі тестування був обраний шлях завантаження підготовлених відео. Після натискання на кнопку «Upload video» відкривається стандартне вікно вибору файлу для завантаження. Одномоментно дозволяється обрати лише один файл у форматах mkv, mp4, avi, mov, wmv (рис. 3.6).

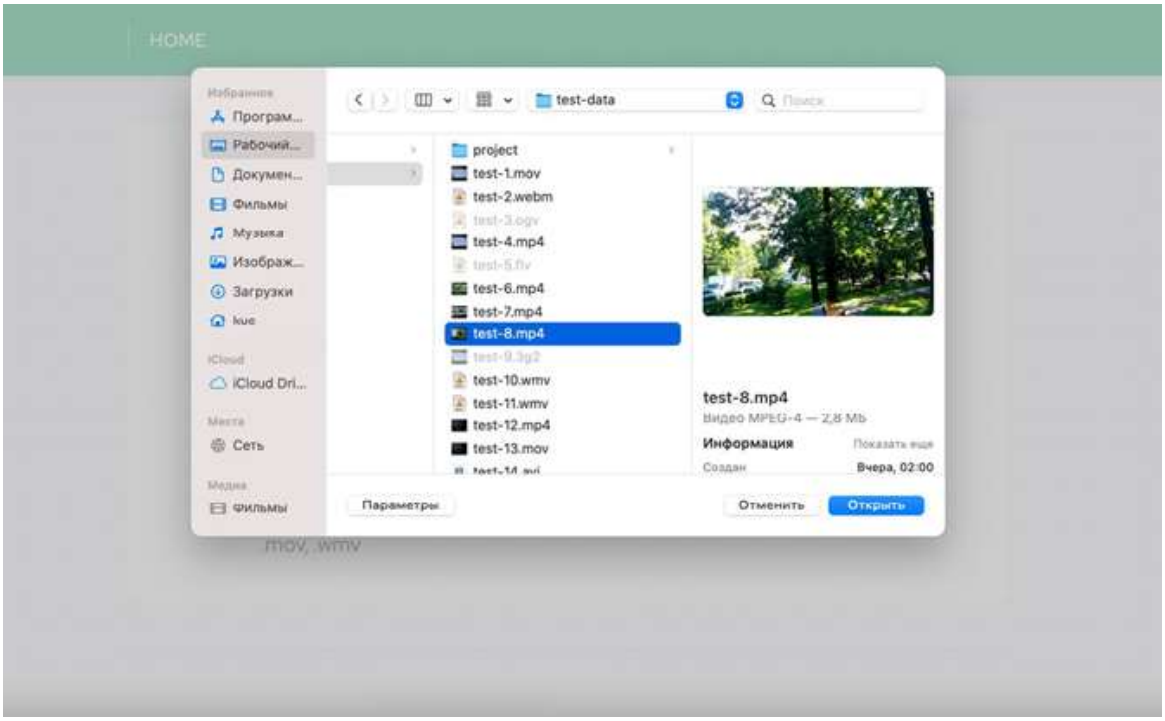


Рисунок 3.6 – Вікно вибору відео файлу

Оскільки відеофайли можуть бути великих розмірів, впроваджено лоадер для повідомлення користувача про процес завантаження відео (рис. 3.7).

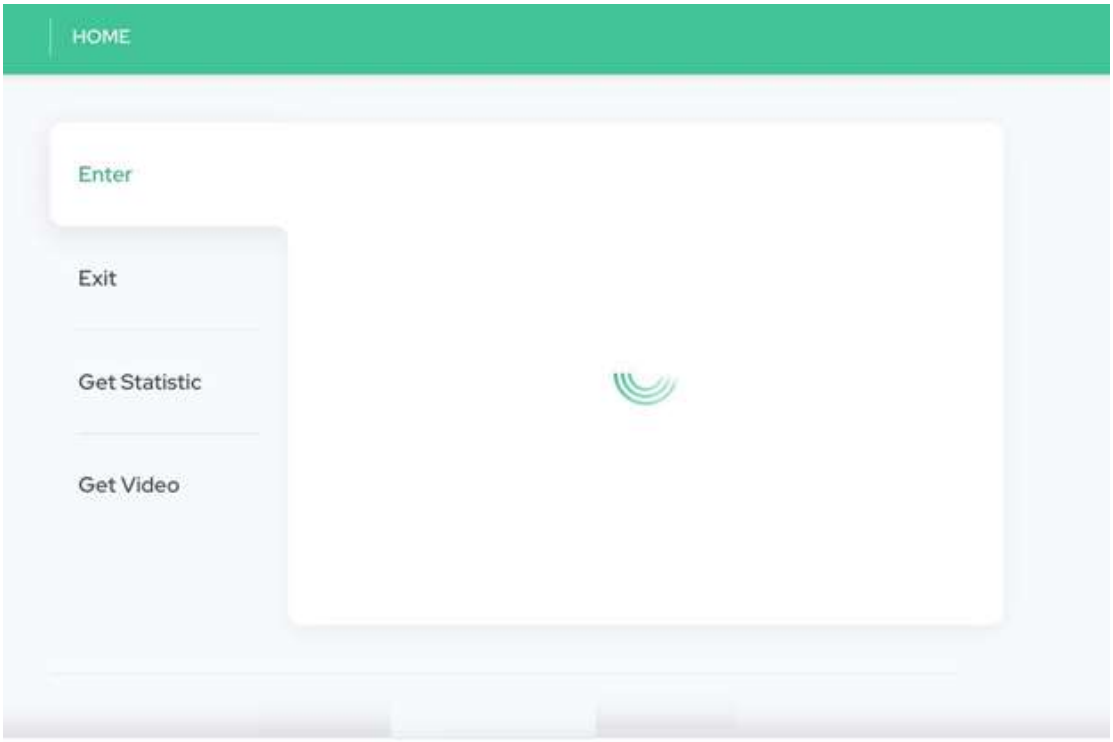


Рисунок 3.7 – Вікно процесу завантаження відео

Після успішного завантаженню файлу, програмний модуль повертає перший кадр для нанесення розмітки полігонів для опису зон переходів. Вікно нанесення розмітки полігону для опису зони входу зображено на рисунку 3.8.

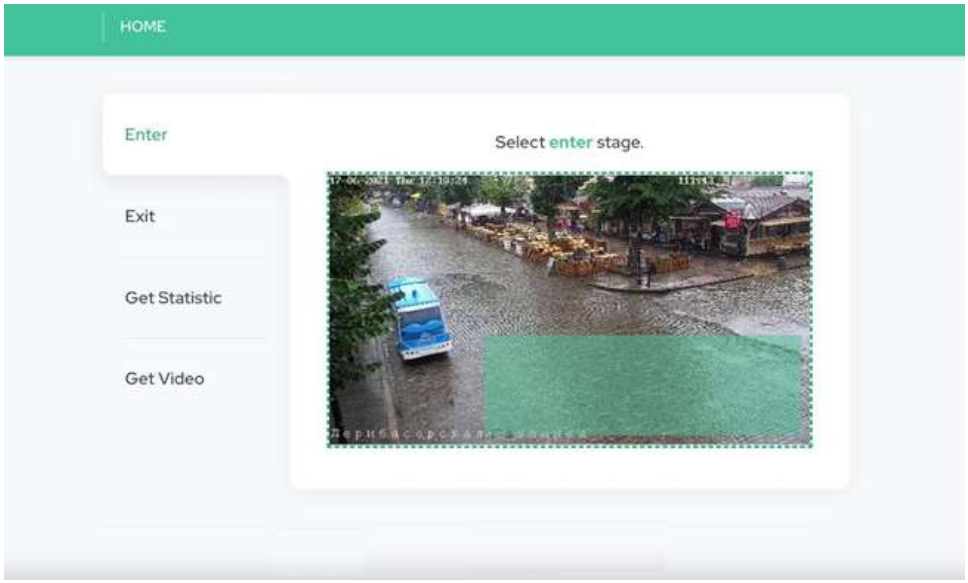


Рисунок 3.8 – Вікно нанесення розмітки полігону для опису зони входу

Вікно нанесення розмітки полігону для опису зони виходу зображено на рисунку 3.9.

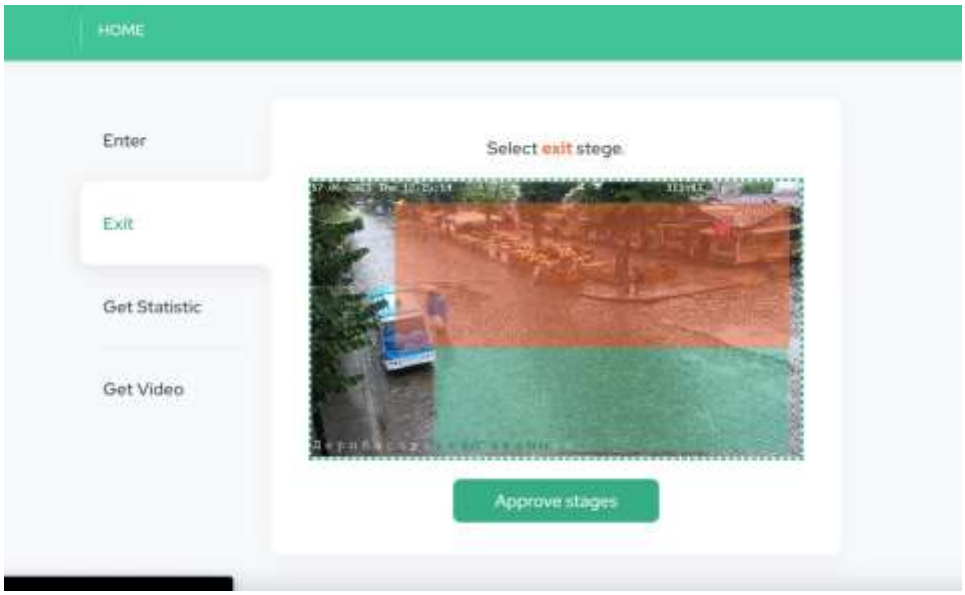


Рисунок 3.9 – Вікно нанесення розмітки полігону для опису зони виходу

Програмне забезпечення фіксує пересування об’єктів, в якості яких виступають люди, відносно цих полігонів та оновлює статистичні дані, що далі надсилаються користувачеві та виводяться на екран (рис 3.10).

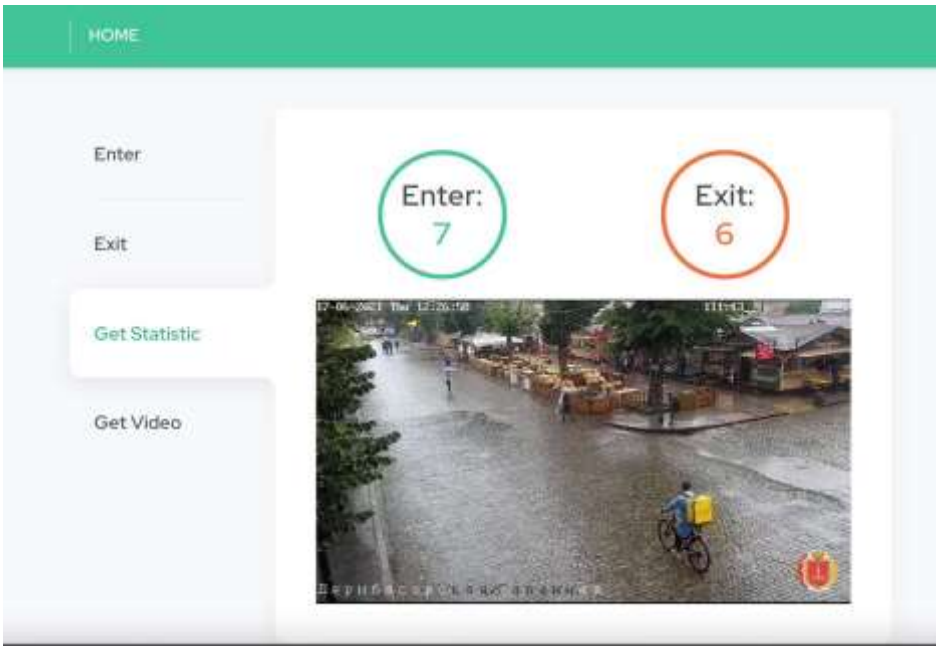


Рисунок 3.10 – Вікно виведення результатів

У програмного застосунка є можливість завантажити відео, до якого було застосовано алгоритм автоматизованого розпізнавання об’єктів та їх підрахунку у відеопотоці. При натисненні на кнопку «Get video» розпочинається процес завантаження опрацьованого файлу, який в подальшому відкривається користувачем (рис. 3.11).

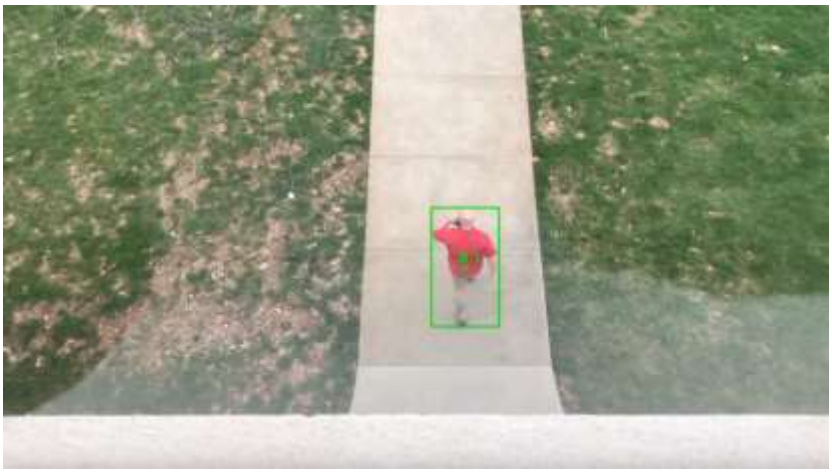


Рисунок 3.11 – Приклад опрацьованого відео

Для порівняння результатів роботи програми з програмою-аналогом PyImageSearch було проведено тестування 80 відео довжиною до 10хв з середнім трафіком людей близько 100 чол. При автоматизованому розпізнаванні та підрахунку людей розроблений додаток показав кращий результат відносно програми аналогу принаймні на 2,67%. В таблиці 3.2 наведено вибірку тестування.

Таблиця 3.2 – Порівняння достовірності розробленого програмного засобу та аналогу

Набір відео	Розроблена програма	Р у	Ручний підрахунок	Покращення

3.5 Висновки до розділу 3

У процесі порівняння можливих мов для програмної реалізації інформаційної технології розпізнавання об’єктів у відеопотоці було обрано мову програмування Python. Перевагами якої є динамічна типізація, простота розроби та обширний набір інструментів для роботи з нейромережами та матиматичними алгоритмати.

Для розробки нейронної мережі вибрано пакети Pandas, NumPy та PyTorch. В сукупності вони дозволяють розробляти рішення великої складності.

Таким чином у цьому розділі було розроблено та протестовано програмне забезпечення розпізнавання об’єктів у відеопотоці, на основі попередніх досліджень, яке цілком справляється з поставленою задачею. При розробці нейромережі підібрані оптимальні параметри, що робить програмний продукт готовим до застосовування на практиці. Для порівняння результатів роботи програми з програмою-аналогом PyImageSearch було проведено тестування 80 відео. При автоматизованому розпізнаванні та підрахунку людей розроблений додаток показав кращий результат відносно програми аналогу принаймні на 2,67%.

4 ЕКОНОМІЧНА ЧАСТИНА

Для успішного впровадження науково-технічної розробки важливо, щоб вона відповідала актуальним вимогам науково-технічного прогресу та ураховувала економічні аспекти. Надання оцінки економічної ефективності результатів науково-дослідної роботи є невід'ємною частиною цього процесу. Проведене дослідження, представлене у магістерській роботі і присвячене розробці та вивченню "Інформаційна технологія розпізнавання об'єктів у відеопотоці", відноситься до науково-технічних робіт, які спрямовані на виведення на ринок. Вирішення питання щодо комерціалізації розробки може відбутися під час проведення самої роботи, що створює можливість подальшого виведення на ринок. Цей напрямок розглядається як пріоритетний, оскільки розроблені результати можуть бути корисними для різних зацікавлених сторін і приносити економічні вигоди. Проте для успішної реалізації цього процесу важливо здобути зацікавленого інвестора, який виявив би інтерес до втілення даного проекту, і переконати його в обґрунтованості вкладання інвестицій у цю розробку. Для досягнення цієї мети передбачені наступні етапи виконання робіт:

1. Проведення комерційного аудиту науково-технічної розробки, включаючи визначення науково-технічного рівня та комерційного потенціалу.
2. Розрахунок витрат на реалізацію науково-технічної розробки.
3. Проведення розрахунку економічної ефективності впровадження та комерціалізації науково-технічної розробки для потенційного інвестора, а також обґрунтування економічної доцільності комерціалізації з точки зору інвестора.

4.1 Проведення комерційного та технологічного аудиту науково-технічної розробки

Метою проведення комерційного і технологічного аудиту дослідження за темою «Інформаційна технологія розпізнавання об'єктів у відеопотоці» є розширення функціональних можливостей програмного забезпечення для захисту файлів.

Оцінювання науково-технічного рівня розробки та її комерційного потенціалу рекомендується здійснювати із застосуванням 5-ти бальної системи оцінювання за 12-ма критеріями, наведеними в табл. 4.1 [20].

Таблиця 4.1 – Рекомендовані критерії оцінювання науково-технічного рівня і комерційного потенціалу розробки та бальна оцінка

Бали (за 5-ти бальною шкалою)					
	0	1	2	3	4
Технічна здійсненність концепції					
1	Достовірність концепції не підтверджена	Концепція підтверджена експертними висновками	Концепція підтверджена розрахунками	Концепція перевірена на практиці	Перевірено працездатність продукту в реальних умовах
Ринкові переваги (недоліки)					
2	Багато аналогів на малому ринку	Мало аналогів на малому ринку	Кілька аналогів на великому ринку	Один аналог на великому ринку	Продукт не має аналогів на великому ринку
3	Ціна продукту значно вища за ціни аналогів	Ціна продукту дещо вища за ціни аналогів	Ціна продукту приблизно дорівнює цінам аналогів	Ціна продукту дещо нижче за ціни аналогів	Ціна продукту значно нижче за ціни аналогів
4	Технічні та споживчі властивості продукту значно гірші, ніж в	Технічні та споживчі влас-тивості продукту трохи гірші, ніж в аналогів	Технічні та споживчі властивості продукту на рівні аналогів	Технічні та споживчі властивості продукту трохи кращі, ніж в	Технічні та споживчі властивості продукту значно кращі, ніж в
5	Експлуатаційні витрати значно вищі, ніж в аналогів	Експлуатаційні витрати дещо вищі, ніж в аналогів	Експлуатаційні витрати на рівні експлуатаційних витрат аналогів	Експлуатаційні витрати трохи нижчі, ніж в аналогів	Експлуатаційні витрати значно нижчі, ніж в аналогів
Ринкові перспективи					
6	Ринок малий і не має позитивної динаміки	Ринок малий, але має позитивну динаміку	Середній ринок з позитивною динамікою	Великий стабільний ринок	Великий ринок з позитивною динамікою
7	Активна конкуренція великих компаній на ринку	Активна конкуренція	Помірна конкуренція	Незначна конкуренція	Конкурентів немає

Продовження таблиці 4.1

Практична здійсненність					
8	Відсутні фахівці як з технічної, так і з комерційної реалізації ідеї	Необхідно наймати фахівців або витрачати значні кошти та час на навчання наявних фахівців	Необхідне незначне навчання фахівців та збільшення їх штату	Необхідне незначне навчання фахівців	Є фахівці з питань як з технічної, так і з комерційної реалізації ідеї
9	Потрібні значні фінансові ресурси, які відсутні. Джерела фінансування ідеї відсутні	Потрібні незначні фінансові ресурси. Джерела фінансування відсутні	Потрібні значні фінансові ресурси. Джерела фінансування є	Потрібні незначні фінансові ресурси. Джерела фінансування є	Не потребує додаткового фінансування
10	Необхідна розробка нових матеріалів	Потрібні матеріали, що використовуються у військово промисловому комплексі	Потрібні дорогі матеріали	Потрібні досяжні та дешеві матеріали	Всі матеріали для реалізації ідеї відомі та давно використовуються у виробництві
11	Термін реалізації ідеї більший за 10 років	Термін реалізації ідеї більший за 5 років. Термін окупності інвестицій більше 10-ти років	Термін реалізації ідеї від 3-х до 5-ти років. Термін окупності інвестицій більше 5-ти років	Термін реалізації ідеї менше 3-х років. Термін окупності інвестицій від 3-х до 5-ти років	Термін реалізації ідеї менше 3-х років. Термін окупності інвестицій менше 3-х років
12	Необхідна розробка регламентних документів та отримання великої кількості дозвільних документів на виробництво та реалізацію продукту	Необхідно отримання великої кількості дозвільних документів на виробництво та реалізацію продукту, що вимагає значних коштів та часу	Процедура отримання дозвільних документів для виробництва та реалізації продукту вимагає незначних коштів та часу	Необхідно тільки повідомлення відповідним органам про виробництво та реалізацію продукту	Відсутні будь- які регламентні обмеження на виробництво та реалізацію продукту

Результати оцінювання науково-технічного рівня та комерційного потенціалу науково-технічної розробки потрібно звести до таблиці. Для оцінки науково-технічного рівня і комерційного потенціалу розробки експертами було запрошено трьох незалежних експертів: Озеранський В.С. доцент кафедри «Комп’ютерних наук» Вінницького національного технічного університету, Яровий А. А. професор кафедри

«Комп'ютерних наук» Вінницького національного технічного університету, Нікітчук В. В. директор ТОВ «Символ».

Таблиця 4.2 – Результати оцінювання науково-технічного рівня і комерційного потенціалу розробки експертами

Критерії	Експерт (ПІБ, посада)		
	Озеранський В.С.	Яровий А.А.	Нікітчук В.В.
	Бали, виставлені експертами:		
1. Технічна здійсненність концепції	4	5	4
2. Ринкові переваги (наявність аналогів)	3	3	3
3. Ринкові переваги (ціна продукту)	4	4	3
4. Ринкові переваги (технічні властивості)	3	3	4
5. Ринкові переваги (експлуатаційні витрати)	2	2	3
6. Ринкові перспективи (розмір ринку)	3	3	3
7. Ринкові перспективи (конкуренція)	2	2	2
8. Практична здійсненність (наявність фахівців)	5	5	5
9. Практична здійсненність (наявність фінансів)	2	3	2
10. Практична здійсненність (необхідність нових матеріалів)	4	5	5
11. Практична здійсненність (термін реалізації)	3	4	5
12. Практична здійсненність (розробка документів)	4	5	4
Сума балів	$СБ_1$=39	$СБ_2$=44	$СБ_3$=43
Середньоарифметична сума балів $СБ_с$	42,0		

За результатами розрахунків, наведених в таблиці 4.2, зробимо висновок щодо науково-технічного рівня і рівня комерційного потенціалу розробки. При цьому використаємо рекомендації, наведені в табл. 4.3 [20].

Таблиця 4.3 – Науково-технічні рівні та комерційні потенціали розробки

Середньоарифметична сума балів СБ, розрахована на основі висновків експертів	Науково-технічний рівень та комерційний потенціал розробки
41…48	Високий
31…40	Вище середнього
21…30	Середній
11…20	Нижче середнього
0…10	Низький

Згідно проведених досліджень рівень комерційного потенціалу розробки за темою «Інформаційна технологія розпізнавання об'єктів у відеопотоці» становить 42 бали, що, відповідно до таблиці 4.3 рівень комерційного потенціалу розробки високий, що свідчить про комерційну важливість проведення даних досліджень.

Магістерська кваліфікаційна робота «Інформаційна технологія розпізнавання об'єктів у відеопотоці» відноситься до науково-технічних робіт, які орієнтовані на виведення на ринок, тобто при цьому відбувається комерціалізація науково-технічної розробки. Цей напрямок є для нас пріоритетним, оскільки результатами розробки можуть користуватися не тільки самі розробники, а й інші споживачі, отримуючи при цьому суттєвий економічний ефект.

4.2 Визначення рівня конкурентоспроможності розробки

В процесі визначення економічної ефективності науково-технічної розробки також доцільно провести прогноз рівня її конкурентоспроможності за сукупністю параметрів, що підлягають оцінюванню.

В якості системи-аналогу вибрано Pyimageserach People Counter. Основними недоліками аналога є: невисока точність детекції та розпізнавання рухомих об’єктів; невисока швидкодія процесу розпізнавання об’єктів у відеопотоці.

У розробці дана проблема вирішується покращеним алгоритмом попередньої обробки зображення перед процесом розпізнавання. Також система випереджає

аналог за такими параметрами як точністю розпізнавання об’єктів у відеопотоці та підвищеною швидкодією

Одиничний параметричний індекс розраховуємо за формулою [20]:

$$q_i = \frac{P_i}{P_{базі}}. \quad (4.1)$$

де q_i – одиничний параметричний індекс, розрахований за i-м параметром;

P_i – значення i-го параметра виробу;

$P_{базі}$ – аналогічний параметр базового виробу-аналога, з яким проводиться порівняння.

Загальні технічні та економічні характеристики розробки представлено в таблиці 4.4.

Таблиця 4.4 – Основні техніко-економічні показники аналога та розробки, що проектується

Показник	Варіанти		Відносний показник якості	Коефіцієнт вагомості параметра
	Базовий (товар-конкурент)	Новий (інноваційне рішення)		
1	2	3	4	5
Процесор (тактова частота)	3,5	1,5	2,33	25%
Оперативна пам'ять	16	4	4	25%
Об'єм диску	100	40	2,5	20%
Мережевий канал	1	2	2	20%
Швидкість мережевого каналу	100	50	2	10%

Нормативні параметри оцінюємо показником, який отримує одне з двох значень: 1 – пристрій відповідає нормам і стандартам; 0 – не відповідає.

Груповий показник конкурентоспроможності за нормативними параметрами розраховуємо як добуток частинних показників за кожним параметром за формулою [20]:

$$I_{НП} = \prod_{i=1}^{n} q_i \,, \tag{4.2}$$

де $I_{нп}$ – загальний показник конкурентоспроможності за нормативними параметрами;

q_i – одиничний (частинний) показник за i-м нормативним параметром;

n – кількість нормативних параметрів, які підлягають оцінюванню.

За нормативними параметрами розроблюваний пристрій відповідає вимогам ДСТУ, тому $I_{нп} = 1$.

Значення групового параметричного індексу за технічними параметрами визначаємо з урахуванням вагомості (частки) кожного параметра [20]:

$$I_{ТП} = \sum_{i=1}^{n} q_i \cdot \alpha_i \,, \tag{4.3}$$

де $I_{тп}$ – груповий параметричний індекс за технічними показниками (порівняно з виробом-аналогом);

q_i – одиничний параметричний показник i-го параметра;

α_i – вагомість i-го параметричного показника, $\sum_{i=1}^{n} \alpha_i = 1$;

n – кількість технічних параметрів, за якими оцінюється конкурентоспроможність.

Проведемо аналіз параметрів згідно даних таблиці 4.4.

$$I_{mn} = 2{,}33 \cdot 0{,}25 + 4 \cdot 0{,}25 + 2{,}5 \cdot 0{,}2 + 2 \cdot 0{,}2 + 2 \cdot 0{,}1 = 2{,}68.$$

Груповий параметричний індекс за економічними параметрами розраховуємо за формулою [20]:

$$I_{ЕП} = \sum_{i=1}^{m} q_i \cdot \beta_i \,, \tag{4.4}$$

де $I_{ЕП}$ – груповий параметричний індекс за економічними показниками;

q_i – економічний параметр i-го виду;

β_i – частка i-го економічного параметра, $\sum_{i=1}^{m} \beta_i = 1$;

m – кількість економічних параметрів, за якими здійснюється оцінювання.

Проведемо аналіз параметрів згідно даних таблиці .

$$I_{ЕП}=0{,}65 \cdot 0{,}5 + 0{,}74 \cdot 0{,}5 = 0{,}70.$$

На основі групових параметричних індексів за нормативними, технічними та економічними показниками розрахуємо інтегральний показник конкурентоспроможності за формулою [20]:

$$K_{ІНТ} = I_{НП} \cdot \frac{I_{ТП}}{I_{ЕП}} \,, \tag{4.5}$$

$$K_{ІНТ} = 1 \cdot 2{,}68 / 0{,}70 = 3{,}8.$$

Інтегральний показник конкурентоспроможності $К_{ІНТ} > 1$, отже розробка переважає відомі аналоги за своїми техніко-економічними показниками.

4.3 Розрахунок витрат на проведення науково-дослідної роботи

Витрати, пов’язані з проведенням науково-дослідної роботи на тему «Інформаційна технологія розпізнавання об'єктів у відеопотоці», під час планування, обліку і калькулювання собівартості науково-дослідної роботи групуємо за відповідними статтями.

До статті «Витрати на оплату праці» належать витрати на виплату основної та додаткової заробітної плати керівникам відділів, лабораторій, секторів і груп, науковим, інженерно-технічним працівникам, конструкторам, технологам, креслярам, копіювальникам, лаборантам, робітникам, студентам, аспірантам та іншим працівникам, безпосередньо зайнятим виконанням конкретної теми, обчисленої за посадовими окладами, відрядними розцінками, тарифними ставками згідно з чинними в організаціях системами оплати праці.

Основна заробітна плата дослідників

Витрати на основну заробітну плату дослідників ($З_о$) розраховуємо у відповідності до посадових окладів працівників, за формулою [20]:

$$З_о = \sum_{i=1}^{k} \frac{M_{ni} \cdot t_i}{T_p}, \quad (4.6)$$

де k – кількість посад дослідників залучених до процесу досліджень;

M_{ni} – місячний посадовий оклад конкретного дослідника, грн;

t_i – число днів роботи конкретного дослідника, дн.;

T_p – середнє число робочих днів в місяці, T_p=21 дні.

$$З_о = 17120 \cdot 15 / 21 = 11673 \; грн.$$

Витрати на основну заробітну плату робітників ($З_р$) за відповідними найменуваннями робіт НДР на тему «Інформаційна технологія розпізнавання об'єктів у відеопотоці» розраховуємо за формулою:

$$З_р = \sum_{i=1}^{n} C_i \cdot t_i, \quad (4.7)$$

де C_i – погодинна тарифна ставка робітника відповідного розряду, за виконану відповідну роботу, грн/год;

t_i – час роботи робітника при виконанні визначеної роботи, год.

Проведені розрахунки зведемо до табл. 4.5.

Таблиця 4.5 – Витрати на заробітну плату дослідників

Найменування посади	Місячний посадовий оклад, грн	Оплата за робочий день, грн	Число днів роботи	Витрати на заробітну плату, грн
Керівник проекту	17120	778,2	15	11673
Інженер-дослідник (програміст)	16850	765,9	15	11489
Консультант (менеджер служби доставки)	17000	772,7	4	3091
Фахівець 1-ї категорії	8500	386,4	15	5795
Всього				32048

Погодинну тарифну ставку робітника відповідного розряду C_i можна визначити за формулою:

$$C_i = \frac{M_M \cdot K_i \cdot K_c}{T_p \cdot t_{зм}}, \quad (4.8)$$

де M_M – розмір прожиткового мінімуму працездатної особи, або мінімальної місячної заробітної плати (в залежності від діючого законодавства), приймемо $Мм$=6500 грн;

K_i – коефіцієнт міжкваліфікаційного співвідношення для встановлення тарифної ставки робітнику відповідного розряду (табл. Б.2, додаток Б) [20];

K_c – мінімальний коефіцієнт співвідношень місячних тарифних ставок робітників першого розряду з нормальними умовами праці виробничих об’єднань і підприємств до законодавчо встановленого розміру мінімальної заробітної плати.

T_p – середнє число робочих днів в місяці, приблизно T_p = 21 дн;

$t_{зм}$ – тривалість зміни, год.

Проведені розрахунки зведемо до табл. 4.6.

$C_1 = 6500{,}00 \cdot 1{,}1 \cdot 1{,}65 / (21 \cdot 8) = 72{,}4$ грн.

$З_{р1} = 72{,}4 \cdot 4{,}32 = 312{,}7$ грн.

Таблиця 4.6 – Величина витрат на основну заробітну плату робітників

Найменування робіт	Тривалість роботи, год	Розряд роботи	Погодинна тарифна ставка, грн	Величина оплати на робітника грн
Встановлення обладнання для проведення досліджень	4,32	2	72,4	312,7
Інсталяція програмного забезпечення розробки програмного забезпечення	6,5	3	88,8	577,4
Встановлення цифрових обчислювальних систем та мережевого доступу	3,15	4	98,7	310,9
Відлагодження програмних модулів аналітичного дослідження	5,2	5	111,9	581,7
Всього				1782,8

Додаткова заробітна плата дослідників та робітників

Додаткову заробітну плату розраховуємо як 10 … 12% від суми основної заробітної плати дослідників та робітників за формулою:

$$З_{дод} = (З_о + З_р) \cdot \frac{Н_{дод}}{100\%}, \qquad (4.9)$$

де $Н_{дод}$ – норма нарахування додаткової заробітної плати. Приймемо 11%.

$З_{дод}$ = (32048+ 1782,8) · 11 / 100% = 3721,35 грн.

Нарахування на заробітну плату дослідників та робітників розраховуємо як 22% від суми основної та додаткової заробітної плати дослідників і робітників за формулою:

$$З_н = (З_о + З_р + З_{дод}) \cdot \frac{Н_{зп}}{100\%} \quad (4.10)$$

де $Н_{зп}$ – норма нарахування на заробітну плату. Приймаємо 22%.

$Зн$ = (32048+ 1782,8+3721,35) · 22 / 100% = 8261,4 грн.

До статті «Сировина та матеріали» належать витрати на сировину, основні та допоміжні матеріали, інструменти, пристрої та інші засоби і предмети праці, які придбані у сторонніх підприємств, установ і організацій та витрачені на проведення досліджень за темою «Інформаційна технологія розпізнавання об'єктів у відеопотоці».

Витрати на матеріали ($М$), у вартісному вираженні розраховуються окремо по кожному виду матеріалів за формулою:

$$М = \sum_{j=1}^{n} Н_j \cdot Ц_j \cdot К_j - \sum_{j=1}^{n} В_j \cdot Ц_{вj}, \quad (4.11)$$

де $Н_j$ – норма витрат матеріалу j-го найменування, кг;

n – кількість видів матеріалів;

$Ц_j$ – вартість матеріалу j-го найменування, грн/кг;

$К_j$ – коефіцієнт транспортних витрат, ($К_j$ = *1,1 ... 1,15*);

$В_j$ – маса відходів j-го найменування, кг;

$Ц_{вj}$ – вартість відходів j-го найменування, грн/кг.

Проведені розрахунки зведемо до табл. 4.7.

Таблиця 4.7 – Витрати на матеріали

Найменування матеріалу, марка, тип, сорт	Ціна за 1 кг, грн	Норма витрат, кг	Вартість витраченого матеріалу, грн
Папір канцелярський офісний (А4) BUROMAX Premium	174,9	1	174,9
Папір для заміток (А5) BUROMAX Premium Light	118	1	118
Начиння канцелярське BUROMAX Premium	187	1	187
Всього			479,9
З врахуванням коефіцієнта транспортування			527,89

Витрати на комплектуючі *($К_в$)*, які використовують при проведенні НДР на тему «Інформаційна система підтримки підприємств малого бізнесу у сфері послуг».

Витрати на комплектуючі К, що були використані під час виконання даного етапу роботи, розраховуються за формулою:

$$К = \sum_{1}^{n} Н_i \cdot Ц_i \cdot К_i \text{ грн.}, \quad (4.12)$$

де $Н_i$ – кількість комплектуючих *i*-го виду, шт.;

$Ц_i$ – ціна комплектуючих *i*-го виду, грн.;

$К_i$ – коефіцієнт транспортних витрат, $К_i$ = (1,1…1,15);

n – кількість видів комплектуючих.

Зроблені розрахунки бажано звести до табл. 4.8.

Таблиця 4.8 – Витрати на комлектуючі

Найменування комплектувальних	Кількість	Ціна за штуку, грн.	Сума, грн.
Роутер XT-MB-0312	1	2850	2850
Кабель USB для передачі даних	1	356	356
Всього з врахування коефіцієнт транспортних витрат			3526,60

До статті «Спецустаткування для наукових (експериментальних) робіт» належать витрати на виготовлення та придбання спецустаткування необхідного для проведення досліджень, також витрати на їх проектування, виготовлення, транспортування, монтаж та встановлення.

Балансову вартість спецустаткування розраховуємо за формулою:

$$B_{спец} = \sum_{i=1}^{k} Ц_i \cdot C_{пр.i} \cdot K_i \ , \qquad (4.13)$$

де $Ц_i$ – ціна придбання одиниці спецустаткування даного виду, марки, грн;

$C_{пр.i}$ –кількість одиниць устаткування відповідного найменування, які придбані для проведення досліджень, шт.;

K_i – коефіцієнт, що враховує доставку, монтаж, налагодження устаткування тощо, ($K_i = 1,10…1,12$);

k – кількість найменувань устаткування.

$B_{спец} = 9650 \cdot 1 \cdot 1,11 = 10615$ *грн.*

Отримані результати зведемо до табл. 4.9.

Таблиця 4.9 – Витрати на придбання спецустаткування по кожному виду

Найменування устаткування	Кількість, шт	Ціна за одиницю, грн	Вартість, грн
Мережеве обладнання передачі цифрових даних	1	9650	10615
Всього			10615

До статті «Програмне забезпечення для наукових (експериментальних) робіт» належать витрати на розробку та придбання спеціальних програмних засобів і програмного забезпечення, (програм, алгоритмів, баз даних) необхідних для проведення досліджень, також витрати на їх проектування, формування та встановлення.

Балансову вартість програмного забезпечення розраховуємо за формулою:

$$B_{прг} = \sum_{i=1}^{k} Ц_{iпрг} \cdot C_{прг.i} \cdot K_i \ , \qquad (4.14)$$

де $Ц_{iпрг}$ – ціна придбання одиниці програмного засобу даного виду, грн;

$C_{прг.i}$ – кількість одиниць програмного забезпечення відповідного найменування, які придбані для проведення досліджень, шт.;

K_i – коефіцієнт, що враховує інсталяцію, налагодження програмного засобу тощо, ($K_i = 1,10…1,12$);

k – кількість найменувань програмних засобів.

$B_{прг} = 5890 \cdot 1 \cdot 1,11 = 6479$ *грн.*

Отримані результати зведемо до табл. 4.10:

Таблиця 4.10 – Витрати на придбання програмних засобів по кожному виду

Найменування програмного засобу	Кількість, шт	Ціна за одиницю, грн	Вартість, грн
Середовище програмування Visual Studio Code	1	5890	6479
Платформа Docker	1	8560	9416
Всього			15895

В спрощеному вигляді амортизаційні відрахування по кожному виду обладнання, приміщень та програмному забезпеченню тощо, розраховуємо з використанням прямолінійного методу амортизації за формулою:

$$A_{обл} = \frac{Ц_б}{T_в} \cdot \frac{t_{вик}}{12} \ , \qquad (4.15)$$

де $Ц_б$ – балансова вартість обладнання, програмних засобів, приміщень тощо, які використовувались для проведення досліджень, грн;

$t_{вик}$ – термін використання обладнання, програмних засобів, приміщень під час досліджень, місяців;

$T_в$ – строк корисного використання обладнання, програмних засобів, приміщень тощо, років.

$A_{обл} = (27599 \cdot 3) / (2 \cdot 12) = 3449,88$ *грн.*

Проведені розрахунки зведемо до табл. 4.11.

Таблиця 4.11 – Амортизаційні відрахування по кожному виду обладнання

Найменування обладнання	Балансова вартість, грн	Строк корисного використання, років	Термін використання обладнання, місяців	Амортизаційні відрахування, грн
Персональний комп'ютер Core I3-2700	27599	2	3	3449,88
Робоче місце інженера-розробника (дослідника програмного забезпечення)	6560	20	2	54,67
Графічні пристрої виводу інформації	7500	2	2	625,00
Офісна оргтехніка	8329	2	1	347,04
Всього				4476,58

Витрати на силову електроенергію (B_e) розраховуємо за формулою:

$$B_e = \sum_{i=1}^{n} \frac{W_{yi} \cdot t_i \cdot Ц_e \cdot K_{вni}}{\eta_i}, \qquad (4.16)$$

де W_{yi} – встановлена потужність обладнання на визначеному етапі розробки, кВт;

t_i – тривалість роботи обладнання на етапі дослідження, год;

$Ц_e$ – вартість 1 кВт-години електроенергії, грн; (вартість електроенергії визначається за даними енергопостачальної компанії), приймемо $Ц_e$ = 7,5 грн;

$K_{впі}$ – коефіцієнт, що враховує використання потужності, $K_{впі} < 1$;

η_i – коефіцієнт корисної дії обладнання, $\eta_i < 1$.

$B_e = 0,25 \cdot 310,0 \cdot 7,5 \cdot 0,5 / 0,8 = 363,28$ грн.

До статті «Службові відрядження» дослідної роботи на тему «Інформаційна технологія розпізнавання об'єктів у відеопотоці» належать витрати на відрядження штатних працівників, працівників організацій, які працюють за договорами цивільно-правового характеру, аспірантів, зайнятих розробленням досліджень, відрядження, пов’язані з проведенням випробувань машин та приладів, а також витрати на відрядження на наукові з’їзди, конференції, наради, пов’язані з виконанням конкретних досліджень.

Витрати за статтею «Службові відрядження» розраховуємо як 20…25% від суми основної заробітної плати дослідників та робітників за формулою:

$$B_{св} = (З_о + З_р) \cdot \frac{H_{св}}{100\%}, \qquad (4.17)$$

де $H_{св}$ – норма нарахування за статтею «Службові відрядження», приймемо $Н_{св} = 20\%$.

$B_{св} = (32048 + 1782,8) \cdot 20 / 100\% = 6766,1$ грн.

До статті «Інші витрати» належать витрати, які не знайшли відображення у зазначених статтях витрат і можуть бути віднесені безпосередньо на собівартість досліджень за прямими ознаками.

Витрати за статтею «Інші витрати» розраховуємо як 50…100% від суми основної заробітної плати дослідників та робітників за формулою:

$$I_в = (З_о + З_р) \cdot \frac{H_{ів}}{100\%}, \qquad (4.18)$$

де $Н_{ів}$ – норма нарахування за статтею «Інші витрати», приймемо $Н_{ів}$ = 50%.

$І_в$ = (32048+ 1782,8) · 50 / 100% = 16915,24 грн.

До статті «Накладні (загальновиробничі) витрати» належать: витрати, пов'язані з управлінням організацією; витрати на винахідництво та раціоналізацію; витрати на підготовку (перепідготовку) та навчання кадрів; витрати, пов'язані з набором робочої сили; витрати на оплату послуг банків; витрати, пов'язані з освоєнням виробництва продукції; витрати на науково-технічну інформацію та рекламу та ін.

Витрати за статтею «Накладні (загальновиробничі) витрати» розраховуємо як 100…150% від суми основної заробітної плати дослідників та робітників за формулою:

$$В_{нзв} = (З_о + З_р) \cdot \frac{Н_{нзв}}{100\%}, \quad (4.19)$$

де $Н_{нзв}$ – норма нарахування за статтею «Накладні (загальновиробничі) витрати», приймемо $Н_{нзв}$ = 100%.

$В_{нзв}$ = (32048+ 1782,8) · 100 / 100% = 33830,48 грн.

Витрати на проведення науково-дослідної роботи на тему «Інформаційна технологія розпізнавання об'єктів у відеопотоці». розраховуємо як суму всіх попередніх статей витрат за формулою:

$$В_{заг} = З_о + З_р + З_{дод} + З_н + М + К_в + В_{спец} + В_{прг} + А_{обл} + В_е + В_{св} + В_{сп} + І_в + В_{нзв}. \quad (4.20)$$

$В_{заг}$=2048+1782,8+3721,35+8261,4+527,89+3526,60+10615+15895+4476,58+ +363,28 +6766,1 +16915,24 +33830,48=139790,9 грн.

Загальні витрати *ЗВ* на завершення науково-дослідної (науково-технічної) роботи та оформлення її результатів розраховується за формулою:

$$ЗВ = \frac{В_{заг}}{\eta}, \quad (4.21)$$

де η - коефіцієнт, який характеризує етап (стадію) виконання науково-дослідної роботи, приймемо η=0,5.

$$ЗВ = 139790,9 / 0,5 = 279581,79 \text{ грн.}$$

4.4 Розрахунок економічної ефективності науково-технічної розробки при її можливій комерціалізації потенційним інвестором

В ринкових умовах узагальнюючим позитивним результатом, що його може отримати потенційний інвестор від можливого впровадження результатів тієї чи іншої науково-технічної розробки, є збільшення у потенційного інвестора величини чистого прибутку.

Результати дослідження проведені за темою «Інформаційна технологія розпізнавання об'єктів у відеопотоці» передбачають комерціалізацію протягом 3-х років реалізації на ринку.

В цьому випадку основу майбутнього економічного ефекту будуть формувати:

ΔN – збільшення кількості споживачів яким надається відповідна інформаційна послуга у періоди часу, що аналізуються;

N – кількість споживачів яким надавалась відповідна інформаційна послуга у році до впровадження результатів нової науково-технічної розробки, приймемо 1 особа

$Ц_б$ – вартість послуги у році до впровадження інформаційної системи, приймемо 3000,00 грн;

$\pm\Delta Ц_о$ – зміна вартості послуги від впровадження результатів, приймемо зростання на 500,00 грн.

Можливе збільшення чистого прибутку у потенційного інвестора $\Delta П_i$ для кожного із 3-х років, протягом яких очікується отримання позитивних результатів від можливого впровадження та комерціалізації науково-технічної розробки, розраховуємо за формулою [20]:

$$\Delta П_i = (\pm \Delta Ц_o \cdot N + Ц_o \cdot \Delta N)_i \cdot \lambda \cdot \rho \cdot (1 - \frac{\vartheta}{100}), \quad (4.22)$$

де λ – коефіцієнт, який враховує сплату потенційним інвестором податку на додану вартість. У 2021 році ставка податку на додану вартість складає 20%, а коефіцієнт $\lambda = 0,8333$;

ρ – коефіцієнт, який враховує рентабельність інноваційного продукту). Приймемо ρ =40%;

ϑ – ставка податку на прибуток, який має сплачувати потенційний інвестор, у 2023 році $\vartheta = 18\%$;

Збільшення чистого прибутку 1-го року:

$\Delta П_1 = (1 \cdot 500 + 3000 \cdot 600) \cdot 0,83 \cdot 0,4 \cdot (1 - 0,18/100\%) = 358821,06$ грн.

Збільшення чистого прибутку 2-го року:

$\Delta П_2 = (1 \cdot 500 + 3000 \cdot (600 + 400)) \cdot 0,83 \cdot 0,4 \cdot (1 - 0,18/100\%) = 598392,75$ грн.

Збільшення чистого прибутку 3-го року:

$\Delta П_3 = (1 \cdot 500 + 3000 \cdot (600 + 400 + 300)) \cdot 0,83 \cdot 0,4 \cdot (1 - 0,18/100\%) = 777769,58$ грн.

Приведена вартість збільшення всіх чистих прибутків *ПП*, що їх може отримати потенційний інвестор від можливого впровадження та комерціалізації науково-технічної розробки:

$$ПП = \sum_{i=1}^{T} \frac{\Delta П_i}{(1+\tau)^t}, \quad (4.23)$$

де $\Delta П_i$ – збільшення чистого прибутку у кожному з років, протягом яких виявляються результати впровадження науково-технічної розробки, грн;

T – період часу, протягом якого очікується отримання позитивних результатів від впровадження та комерціалізації науково-технічної розробки, роки;

τ – ставка дисконтування, за яку можна взяти щорічний прогнозований рівень інфляції в країні, τ =18%;

t – період часу (в роках) від моменту початку впровадження науково-технічної розробки до моменту отримання потенційним інвестором додаткових чистих прибутків у цьому році.

$$ПП = 358821,06 /(1+0,18)^1+598392,75 /(1+0,18)^2+777769,58 /(1+0,18)^3=$$
$$= 1166754,45 \text{ грн.}$$

Величина початкових інвестицій PV, які потенційний інвестор має вкласти для впровадження і комерціалізації науково-технічної розробки:

$$PV = k_{інв} \cdot ЗВ, \quad (4.24)$$

де $k_{інв}$ – коефіцієнт, що враховує витрати інвестора на впровадження науково-технічної розробки та її комерціалізацію, приймаємо $k_{інв}$=2;

$ЗВ$ – загальні витрати на проведення науково-технічної розробки та оформлення її результатів, приймаємо 279581,79 грн.

$PV = k_{інв} \cdot ЗВ = 2 \cdot 279581,79 = 559163,59$ грн.

Абсолютний економічний ефект $E_{абс}$ для потенційного інвестора від можливого впровадження та комерціалізації науково-технічної розробки становитиме:

$$E_{абс} = ПП - PV \quad (4.25)$$

де $ПП$ – приведена вартість зростання всіх чистих прибутків від можливого впровадження та комерціалізації науково-технічної розробки, 1166754,45грн;

PV – теперішня вартість початкових інвестицій, 559163,59 грн.

$E_{абс} = ПП - PV$ = 1166754,45- 559163,59 = 607590,86 грн.

Внутрішня економічна дохідність інвестицій $E_в$, які можуть бути вкладені потенційним інвестором у впровадження та комерціалізацію науково-технічної розробки:

$$E_в = \sqrt[T_ж]{1 + \frac{E_{абс}}{PV}} - 1, \quad (4.26)$$

де $E_{абс}$ – абсолютний економічний ефект вкладених інвестицій, грн;

PV – теперішня вартість початкових інвестицій, грн;

$T_ж$ – життєвий цикл науково-технічної розробки, тобто час від початку її розробки до закінчення отримування позитивних результатів від її впровадження, 3 роки.

$$E_в = \sqrt[T_ж]{1 + \frac{E_{абс}}{PV}} - 1 = (1+607590{,}86 / 559163{,}59)^{1/3} - 1 = 0{,}47.$$

Мінімальна внутрішня економічна дохідність вкладених інвестицій $\tau_{мін}$:

$$\tau_{мін} = d + f, \quad (4.27)$$

де d – середньозважена ставка за депозитними операціями в комерційних банках; в 2023 році в Україні d =0,1;

f – показник, що характеризує ризикованість вкладення інвестицій, приймемо 0,25.

$\tau_{мін}$ = 0,1+0,25 = 0,35 < 0,47 свідчить про те, що внутрішня економічна дохідність інвестицій $E_в$, які можуть бути вкладені потенційним інвестором у впровадження та комерціалізацію науково-технічної розробки вища мінімальної внутрішньої дохідності. Тобто інвестувати в науково-дослідну роботу за темою «Інформаційна технологія онтологічного моделювання бази знань з організації бібліотеки» доцільно.

Період окупності інвестицій $T_{ок}$ які можуть бути вкладені потенційним інвестором у впровадження та комерціалізацію науково-технічної розробки:

$$T_{ок} = \frac{1}{E_в}, \quad (4.28)$$

де $E_в$ – внутрішня економічна дохідність вкладених інвестицій.

$T_{ок}$ = 1 / 0,47 = 2,1 р.

$T_{ок}$ < 3-х років, що свідчить про комерційну привабливість науково-технічної розробки і може спонукати потенційного інвестора профінансувати впровадження даної розробки та виведення її на ринок.

4.5 Висновки до розділу 4

Згідно проведених досліджень рівень комерційного потенціалу розробки за темою «Інформаційна технологія розпізнавання об'єктів у відеопотоці» становить 42 бали, що, свідчить про комерційну важливість проведення даних досліджень оскільки рівень комерційного потенціалу розробки високий.

При оцінюванні рівня конкурентоспроможності, згідно узагальненого коефіцієнту конкурентоспроможності розробки, науково-технічна розробка переважає існуючі аналоги приблизно в 3,8 рази.

Також термін окупності становить 2,1 роки, що менше 3-х років, що свідчить про комерційну привабливість науково-технічної розробки і може спонукати потенційного інвестора профінансувати впровадження даної розробки та виведення її на ринок.

Отже можна зробити висновок про доцільність проведення науково-дослідної роботи за темою «Інформаційна технологія розпізнавання об'єктів у відеопотоці».

ВИСНОВКИ

У ході виконання магістерської кваліфікаційної роботи розроблено інформаційну технологію розпізнавання об’єктів у відеопотоці.

В роботі було виконано дослідження найкращих підходів вирішення задачі автоматизованого розпізнавання об’єктів у відеопотоці. Досліджено архітектури CNN, RNN, LSTM, їх переваги і недоліки. Було описано відкриті набори даних та вибраний датасет Duke MTMC для навчання нейромережі для розпізнавання людей у відеопотоці. В якості системи-аналогу вибрано веб-ресурс PyImageSearch People Counter. Визначено архітектуру нейромережі, а саме автоенкодер на основі технології VGG. Було розроблено алгоритм реідентифікації об’єктів у відеопотоці. В процесі порівняння можливих мов для програмної реалізації інформаційної технології розпізнавання об’єктів у відеопотоці було обрано мову програмування Python. Для розробки нейронної мережі вибрано пакети Pandas, NumPy та PyTorch. В сукупності вони дозволяють розробляти рішення великої складності. Було розроблено та протестовано програмне забезпечення розпізнавання об’єктів у відеопотоці, на основі попередніх досліджень, яке цілком справляється з поставленою задачею. Для порівняння результатів роботи програми з програмою-аналогом було опрацьовано 80 відео. При автоматизованому розпізнавання та підрахунку людей розроблений додаток показав кращий результат відносно програми аналогу принаймні на 2,67%.

Згідно проведених економічних досліджень рівень комерційного потенціалу розробки становить 42 бали, що, свідчить про комерційну важливість проведення даних досліджень оскільки рівень комерційного потенціалу розробки високий. При оцінюванні рівня конкурентоспроможності, згідно узагальненого коефіцієнту конкурентоспроможності розробки, науково-технічна розробка переважає існуючі аналоги приблизно в 3,8 рази. Також термін окупності становить 2,1 роки, що менше 3-х років, а це свідчить про комерційну привабливість науково-технічної розробки і може спонукати потенційного інвестора профінансувати впровадження даної розробки та виведення її на ринок.

Отже всі завданні виконано у повному обсязі, мету робото досягнуто.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Я.К. Левчук, В.С. Озеранський. Технологія розпізнавання об’єктів у відеопотоці. Матеріали Всеукраїнської науково-практичної інтернет-конференції «Молодь в науці: дослідження, проблеми, перспективи (МН-2023)» − [Електронний ресурс]. – https://conferences.vntu.edu.ua/index.php/all-fksa/all-fksa-2023/paper/view/19942
2. Autoencoders – [Електронний ресурс] – http://ufldl.stanford.edu/tutorial/unsupervised/Autoencoders/.
3. Auto-Encoder: What Is It? And What Is It Used For? – [Електронний ресурс] – https://towardsdatascience.com/auto-encoder-what-is-it-and-what-is-it-used-for-part-1-3e5c6f017726
4. Introduction to autoencoders – [Електронний ресурс] – https://www.jeremyjordan.me/autoencoders/
5. Згорткова нейронна мережа – просте пояснення CNN та її застосування – [Електронний ресурс] – https://evergreens.com.ua/ua/articles/cnn.html
6. Bradski G. Learning OpenCV - Computer Vision with the OpenCV Library / G. Bradski.,2018 – 580 с.
7. Машинне навчання з використанням мікрокомп’ютерів – [Електронний ресурс] – Режим доступу: http://man.gov.ua/files/49/Machine_Nav4ann_Mogilniy.pdf
8. Python documentation – [Електронний ресурс] – https://www.python.org/doc/
9. Image Classification on ImageNet – [Електронний ресурс] – Режим доступу: https://paperswithcode.com/sota/image-classification-on-imagenet
10. Eric Rothstein Morris, Ralf C. Staudemeyer, Understanding LSTM – a tutorial into Long Short-Term Memory Recurrent Neural Networks, 2019 – 42c.
11. P. Viola and M.J. Jones, «Robust real-time face detection», International Journal of Computer Vision, vol. 57, no. 2, 2004., pp.137–154
12. Yilmaz A. Object Tracking: A Survey / A. Yilmaz, O. Javed, M. Shah // ACM Computing Surveys. – 2006. – Vol. 38, № 4. – P. 13-45.
13. Messer K. et al. Performance characterization of face recognition algorithms and their sensitivity to severe illumination changes – ICB. – 2006.

14. Face Recognition with OpenCV // OpenCV 2.4.13.4 documentation. URL: https://docs.opencv.org/2.4/modules/contrib/doc/facerec/facerec_tutorial
15. Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun, Deep Residual Learning for Image Recognition 2015 – 12c.
16. Benyamin Ghojogh, The Theory Behind Overfitting, Cross Validation, Regularization, Bagging, and Boosting: Tutorial, 2019 – 23c.
17. Dor Bank, Noam Koenigstein, Raja Giryes, Autoencoders, 2020– 19c.
18. Weiyang Liu, Yandong Wen, Zhiding Yu, Ming Li, Bhiksha Raj, and Le Song. SphereFace: Deep Hypersphere Embedding for Face Recognition. In Conference on Computer Vision and Pattern Recognition (CVPR), 2017 – 13c.
19. Подоба Віталій. У яких випадках мова програмування Python є правильним вибором? – [Електронний ресурс] – Режим доступу: http://www.vitaliypodoba.com/2015/06/python-application/
20. Методичні вказівки до виконання економічної частини магістерських кваліфікаційних робіт / Уклад. : В. О. Козловський, О. Й. Лесько, В. В. Кавецький. – Вінниця : ВНТУ, 2021. – 42 с.

ДОДАТКИ

Додаток А (обов'язковий)

Протокол перевірки кваліфікаційної роботи на наявність текстових запозичень

ПРОТОКОЛ ПЕРЕВІРКИ КВАЛІФІКАЦІЙНОЇ РОБОТИ
НА НАЯВНІСТЬ ТЕКСТОВИХ ЗАПОЗИЧЕНЬ

Назва роботи: Інформаційна технологія розпізнавання об'єктів у відеопотоці

Тип роботи: магістерська кваліфікаційна робота
(БДР, МКР)

Підрозділ кафедра комп'ютерних наук, ФІІТА
(кафедра, факультет)

Показники звіту подібності Unicheck

Оригінальність 85,2% Схожість 14,8%

Аналіз звіту подібності (відмітити потрібне):

✓ Запозичення, виявлені у роботі, оформлені коректно і не містять ознак плагіату.

□ Виявлені у роботі запозичення не мають ознак плагіату, але їх надмірна кількість викликає сумніви щодо цінності роботи і відсутності самостійності її виконання автором. Роботу направити на розгляд експертної комісії кафедри.

□ Виявлені у роботі запозичення є недобросовісними і мають ознаки плагіату та/або в ній містяться навмисні спотворення тексту, що вказують на спроби приховування недобросовісних запозичень.

Ознайомлені з повним звітом подібності, який був згенерований системою Unicheck щодо роботи.

Автор роботи _____ Левчук Я.К.

Керівник роботи _____ Озеранський В.С.

Опис прийнятого рішення

Магістерську кваліфікаційну роботу допущено до захисту

Особа, відповідальна за перевірку _____ Озеранський В.С.

Додаток Б (обов'язковий)

Лістинг програми

```
import numpy as np
import cv2

from typing import List
from videostream import VideoStram
from my_detector import DetectorV5
from deep_sort.deep.feature_extractor import Extractor
from deep_sort import build_tracker
from utils.parser import get_config

class Program:
    def __init__(self) -> None:
        # self.vs = VideoStram('./demo/test.mp4')
        self.vs = VideoStram('./2.mov')
        self.yolo = DetectorV5(location="cpu")
        self.extractor = Extractor(
            "./deep_sort/deep/checkpoint/ckpt.t7",
            use_cuda=False
            )
        cfg = get_config()
        cfg.merge_from_file("./configs/deep_sort.yaml")
        self.deepsort = build_tracker(cfg, use_cuda=False)

    def _get_detections(self, image: np.ndarray) -> np.ndarray:
        return self.yolo.predict(image)



    def _make_crops(self, image: np.ndarray, boxes: np.ndarray) -> List[np.ndarray]:
        return [image[box[1]:box[3], box[0]:box[2]] for box in boxes]

    def _xyxy_to_tlwh(self, bbox_xyxy):
        x1,y1,x2,y2 = bbox_xyxy
        t = x1
        l = y1
        w = int(x2-x1)
        h = int(y2-y1)
        return t,l,w,h

    def run(self) -> None:
```

```
        for i in range(3000):
            frame = self.vs.get_frame()
            if i > 0:
                boxes, scores = self.yolo.predict(frame)
                crops = self._make_crops(frame, boxes)
                if crops:
                    features = self.extractor(crops)
                    bboxes = [self._xyxy_to_tlwh(box) for box in boxes]
                    bboxes = np.array(bboxes)
                    outputs = self.deepsort.update(
                        bboxes,
                        scores,
                        frame,
                        features,
                    )
                    for track in outputs:
                        objectID, centroid = track.track_id, track.get_centroid()
                        x1,y1,x2,y2 = [int(i) for i in track.to_tlbr()]
                        cv2.rectangle(frame, (x1 ,y1),(x2, y2),(0,255,0), 2)
                        text = "ID {}".format(objectID)
                        cv2.putText(frame, text, (int(centroid[0]) - 10, int(centroid[1]) - 10),
                            cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0, 255, 0), 2)
                        cv2.circle(frame, (int(centroid[0]), int(centroid[1])), 4, (0, 255, 0), -1)
                self.vs.write(frame)
                # cv2.imshow('name', frame)
                # else:
                    # self.vs.write(frame)
                print(i)

Program().run()

from yolov5 import YOLOv5
from enum import Enum
from typing import List, Tuple
from torch import Tensor

import code
import cv2
import numpy as np
from yolov5.models.common import BottleneckCSP

class ModelLocation(Enum):
    cuda = 'cuda'
    cpu = 'cpu'
```

```
class DetectorV5:
    def __init__(self, location: str="cuda", path: str="./detector/YOLOv5/yolov5s.pt") -> None:
        location: ModelLocation = self._validate_model_location(location)
        self.model = YOLOv5(path, location.name)

    def predict(self, image: np.ndarray) -> np.ndarray:
        result = self.model.predict(image)
        detections = self._get_boxes_and_scores_from_yolo_xyxy(result.xyxy[0])
        return detections

    def _get_boxes_and_scores_from_yolo_xyxy(self, result: Tensor) -> Tuple[np.ndarray, np.ndarray]:
        boxes: List[np.ndarray] = list()
        scores: List[float] = list()
        for detection in result:
            if not detection[5].eq(0):
                continue
            box: Tensor = detection[:4].cpu()
            box: np.ndarray = np.array(box).astype('int32')
            scores.append(detection[4].cpu())
            boxes.append(box)
        return np.array(boxes), np.array(scores)

    def _validate_model_location(self, location: str) -> ModelLocation:
        try:
            location = ModelLocation(location)
        except ValueError:
            raise ValueError('Model location must be "cuda" or "cpu"')
        return location

import numpy as np
import torch
import atexit

from .deep.feature_extractor import Extractor
from .sort.nn_matching import NearestNeighborDistanceMetric
from .sort.preprocessing import non_max_suppression
from .sort.detection import Detection
from .sort.tracker import Tracker

__all__ = ['DeepSort']

class DeepSort(object):
```

```
        def __init__(self, model_path, max_dist=0.2, min_confidence=0.3, nms_max_overlap=1.0, max_iou_distance=0.7, max_age=70, n_init=3, nn_budget=100, use_cuda=True):
            self.min_confidence = min_confidence
            self.nms_max_overlap = nms_max_overlap
            # self.extractor = Extractor(model_path, use_cuda=use_cuda)
            max_cosine_distance = max_dist
            nn_budget = 100
            metric = NearestNeighborDistanceMetric("cosine", max_cosine_distance, nn_budget)
            self.tracker = Tracker(metric, max_iou_distance=max_iou_distance, max_age=max_age, n_init=n_init)

        def update(self, bbox_xywh, confidences, ori_img, features):
            self.height, self.width = ori_img.shape[:2]
            # generate detections
            # features = self._get_features(bbox_xywh, ori_img)
            # bbox_tlwh = self._xywh_to_tlwh(bbox_xywh)
            # print(bbox_xywh)
            # print(bbox_tlwh)
            bbox_tlwh = bbox_xywh
            detections = [Detection(bbox_tlwh[i], conf, features[i]) for i,conf in enumerate(confidences) if conf>self.min_confidence]
            # run on non-maximum supression
            boxes = np.array([d.tlwh for d in detections])
            scores = np.array([d.confidence for d in detections])
            indices = non_max_suppression(boxes, self.nms_max_overlap, scores)
            detections = [detections[i] for i in indices]
            # update tracker
            self.tracker.predict()
            self.tracker.update(detections)
            # output bbox identities
            outputs = []
            # for track in self.tracker.tracks:
            #     if not track.is_confirmed() or track.time_since_update > 1:
            #         continue
            #     box = track.to_tlwh()
            #     x1,y1,x2,y2 = self._tlwh_to_xyxy(box)
            #     track_id = track.track_id
            #     outputs.append(np.array([x1,y1,x2,y2,track_id], dtype=np.int))
            # if len(outputs) > 0:
            #     outputs = np.stack(outputs,axis=0)
            # return outputs
            return self.tracker.tracks
        @staticmethod
        def _xywh_to_tlwh(bbox_xywh):
```

```
    if isinstance(bbox_xywh, np.ndarray):
        bbox_tlwh = bbox_xywh.copy()
    elif isinstance(bbox_xywh, torch.Tensor):
        bbox_tlwh = bbox_xywh.clone()
    bbox_tlwh[:,0] = bbox_xywh[:,0] - bbox_xywh[:,2]/2.
    bbox_tlwh[:,1] = bbox_xywh[:,1] - bbox_xywh[:,3]/2.
    return bbox_tlwh

def _xywh_to_xyxy(self, bbox_xywh):
    x,y,w,h = bbox_xywh
    x1 = max(int(x-w/2),0)
    x2 = min(int(x+w/2),self.width-1)
    y1 = max(int(y-h/2),0)
    y2 = min(int(y+h/2),self.height-1)
    return x1,y1,x2,y2

def _tlwh_to_xyxy(self, bbox_tlwh)
    x,y,w,h = bbox_tlwh
    x1 = max(int(x),0)
    x2 = min(int(x+w),self.width-1)
    y1 = max(int(y),0)
    y2 = min(int(y+h),self.height-1)
    return x1,y1,x2,y2

def _xyxy_to_tlwh(self, bbox_xyxy):
    x1,y1,x2,y2 = bbox_xyxy

    t = x1
    l = y1
    w = int(x2-x1)
    h = int(y2-y1)
    return t,l,w,h

def _get_features(self, bbox_xywh, ori_img):
    im_crops = []
    for box in bbox_xywh:
        x1,y1,x2,y2 = self._xywh_to_xyxy(box)
        im = ori_img[y1:y2,x1:x2]
        im_crops.append(im)
    if im_crops:
        features = self.extractor(im_crops)
    else:
        features = np.array([])
    return features
```

```
class TrackState:
    """
    Enumeration type for the single target track state. Newly created tracks are
    classified as `tentative` until enough evidence has been collected. Then,
    the track state is changed to `confirmed`. Tracks that are no longer alive
    are classified as `deleted` to mark them for removal from the set of active
    tracks.
    """
    Tentative = 1
    Confirmed = 2
    Deleted = 3
class Track:
    """
    A single target track with state space `(x, y, a, h)` and associated
    velocities, where `(x, y)` is the center of the bounding box, `a` is the
    aspect ratio and `h` is the height.
    Parameters
    ----------
    mean : ndarray
        Mean vector of the initial state distribution.
    covariance : ndarray
        Covariance matrix of the initial state distribution.
    track_id : int
        A unique track identifier.
    n_init : int
        Number of consecutive detections before the track is confirmed. The
        track state is set to `Deleted` if a miss occurs within the first
        `n_init` frames.
    max_age : int
        The maximum number of consecutive misses before the track state is
        set to `Deleted`.
    feature : Optional[ndarray]
        Feature vector of the detection this track originates from. If not None,
        this feature is added to the `features` cache.
    Attributes
    ----------
    mean : ndarray
        Mean vector of the initial state distribution.
    covariance : ndarray
        Covariance matrix of the initial state distribution.
    track_id : int
        A unique track identifier.
    hits : int
```

```
    Total number of measurement updates.
age : int
    Total number of frames since first occurance.
time_since_update : int
    Total number of frames since last measurement update.
state : TrackState
    The current track state.
features : List[ndarray]
    A cache of features. On each measurement update, the associated feature
    vector is added to this list.
"""

def __init__(self, mean, covariance, track_id, n_init, max_age,
             feature=None):
    self.mean = mean
    self.covariance = covariance
    self.track_id = track_id
    self.hits = 1
    self.age = 1
    self.time_since_update = 0
    self.state = TrackState.Tentative
    self.features = []
    if feature is not None:
        self.features.append(feature)
    self._n_init = n_init
    self._max_age = max_age
def get_centroid(self):
    ret = self.to_tlwh()
    ret[2:] = ret[:2] + ret[2:]
    return (ret[0] + ret[2]) / 2, (ret[1] + ret[3]) / 2

def to_tlwh(self):
    """Get current position in bounding box format `(top left x, top left y,
    width, height)`.
    Returns
    -------
    ndarray
        The bounding box.
    """
    ret = self.mean[:4].copy()
    ret[2] *= ret[3]
    ret[:2] -= ret[2:] / 2
    return ret
```

```
def to_tlbr(self):
    """Get current position in bounding box format `(min x, min y, max x,
    max y)`.
    Returns
    -------
    ndarray
        The bounding box.
    """
    ret = self.to_tlwh()
    ret[2:] = ret[:2] + ret[2:]
    return ret
def predict(self, kf):
    """Propagate the state distribution to the current time step using a
    Kalman filter prediction step.
    Parameters
    ----------
    kf : kalman_filter.KalmanFilter
        The Kalman filter.
    """
    self.mean, self.covariance = kf.predict(self.mean, self.covariance)
    self.age += 1
    self.time_since_update += 1

def update(self, kf, detection):
    """Perform Kalman filter measurement update step and update the feature
    cache.
    Parameters
    ----------
    kf : kalman_filter.KalmanFilter
        The Kalman filter.
    detection : Detection
        The associated detection.
    """
    self.mean, self.covariance = kf.update(
        self.mean, self.covariance, detection.to_xyah())
    self.features.append(detection.feature)

    self.hits += 1
    self.time_since_update = 0
    if self.state == TrackState.Tentative and self.hits >= self._n_init:
        self.state = TrackState.Confirmed

def mark_missed(self):
    """Mark this track as missed (no association at the current time step).
```

```
        """
        if self.state == TrackState.Tentative:
            self.state = TrackState.Deleted
        elif self.time_since_update > self._max_age:
            self.state = TrackState.Deleted

    def is_tentative(self):
        """Returns True if this track is tentative (unconfirmed).
        """
        return self.state == TrackState.Tentative

    def is_confirmed(self):
        """Returns True if this track is confirmed."""
        return self.state == TrackState.Confirmed

    def is_deleted(self):
        """Returns True if this track is dead and should be deleted."""
        return self.state == TrackState.Deleted

# vim: expandtab:ts=4:sw=4
from __future__ import absolute_import
import numpy as np
from . import kalman_filter
from . import linear_assignment
from . import iou_matching
from .track import Track

class Tracker:
    """
    This is the multi-target tracker.
    Parameters
    ----------
    metric : nn_matching.NearestNeighborDistanceMetric
        A distance metric for measurement-to-track association.
    max_age : int
        Maximum number of missed misses before a track is deleted.
    n_init : int
        Number of consecutive detections before the track is confirmed. The
        track state is set to `Deleted` if a miss occurs within the first
        `n_init` frames.

    Attributes
    ----------
    metric : nn_matching.NearestNeighborDistanceMetric
```

```
    The distance metric used for measurement to track association.
max_age : int
    Maximum number of missed misses before a track is deleted.
n_init : int
    Number of frames that a track remains in initialization phase.
kf : kalman_filter.KalmanFilter
    A Kalman filter to filter target trajectories in image space.
tracks : List[Track]
    The list of active tracks at the current time step.
"""
def __init__(self, metric, max_iou_distance=0.7, max_age=70, n_init=3):
    self.metric = metric
    self.max_iou_distance = max_iou_distance
    self.max_age = max_age
    self.n_init = n_init
    self.kf = kalman_filter.KalmanFilter()
    self.tracks = []
    self._next_id = 1

def predict(self):
    """Propagate track state distributions one time step forward.
    This function should be called once every time step, before `update`.
    """
    for track in self.tracks:
        track.predict(self.kf)

def update(self, detections):
    """Perform measurement update and track management.

    Parameters
    ----------
    detections : List[deep_sort.detection.Detection]
        A list of detections at the current time step.

    """
    # Run matching cascade.
    matches, unmatched_tracks, unmatched_detections = \
        self._match(detections)

    # Update track set.
    for track_idx, detection_idx in matches:
        self.tracks[track_idx].update(
            self.kf, detections[detection_idx])
    for track_idx in unmatched_tracks:
```

```
            self.tracks[track_idx].mark_missed()
        for detection_idx in unmatched_detections:
            self._initiate_track(detections[detection_idx])
        self.tracks = [t for t in self.tracks if not t.is_deleted()]

        # Update distance metric.
        active_targets = [t.track_id for t in self.tracks if t.is_confirmed()]
        features, targets = [], []
        for track in self.tracks:
            if not track.is_confirmed():
                continue
            features += track.features
            targets += [track.track_id for _ in track.features]
            track.features = []
        self.metric.partial_fit(
            np.asarray(features), np.asarray(targets), active_targets)

    def _match(self, detections):

        def gated_metric(tracks, dets, track_indices, detection_indices):
            features = np.array([dets[i].feature for i in detection_indices])
            targets = np.array([tracks[i].track_id for i in track_indices])
            cost_matrix = self.metric.distance(features, targets)
            cost_matrix = linear_assignment.gate_cost_matrix(
                self.kf, cost_matrix, tracks, dets, track_indices,
                detection_indices)

            return cost_matrix

        # Split track set into confirmed and unconfirmed tracks.
        confirmed_tracks = [
            i for i, t in enumerate(self.tracks) if t.is_confirmed()]
        unconfirmed_tracks = [
            i for i, t in enumerate(self.tracks) if not t.is_confirmed()]

        # Associate confirmed tracks using appearance features.
        matches_a, unmatched_tracks_a, unmatched_detections = \
            linear_assignment.matching_cascade(
                gated_metric, self.metric.matching_threshold, self.max_age,
                self.tracks, detections, confirmed_tracks)

        # Associate remaining tracks together with unconfirmed tracks using IOU.
        iou_track_candidates = unconfirmed_tracks + [
            k for k in unmatched_tracks_a if
```

```
            self.tracks[k].time_since_update == 1]
        unmatched_tracks_a = [
            k for k in unmatched_tracks_a if
            self.tracks[k].time_since_update != 1]
        matches_b, unmatched_tracks_b, unmatched_detections = \
            linear_assignment.min_cost_matching(
                iou_matching.iou_cost, self.max_iou_distance, self.tracks,
                detections, iou_track_candidates, unmatched_detections)

        matches = matches_a + matches_b
        unmatched_tracks = list(set(unmatched_tracks_a + unmatched_tracks_b))
        return matches, unmatched_tracks, unmatched_detections

    def _initiate_track(self, detection):
        mean, covariance = self.kf.initiate(detection.to_xyah())
        self.tracks.append(Track(
            mean, covariance, self._next_id, self.n_init, self.max_age,
            detection.feature))
        self._next_id += 1
```

Додаток В (обов'язковий)

ІЛЮСТРАТИВНА ЧАСТИНА

ІНФОРМАЦІЙНА ТЕХНОЛОГІЯ РОЗПІЗНАВАННЯ ОБ'ЄКТІВ У ВІДЕОПОТОЦІ

Виконала: студентка 2 курсу, групи 2КН-22м
спеціальності 122 – Комп'ютерні науки
(шифр і назва напряму підготовки, спеціальності)
Левчук Я.К.
(прізвище та ініціали)
Керівник: к.т.н., доцент каф. КН
Озеранський В.С.
(прізвище та ініціали)
« 07 » 12 2023 р.

Вінниця ВНТУ – 2023 рік

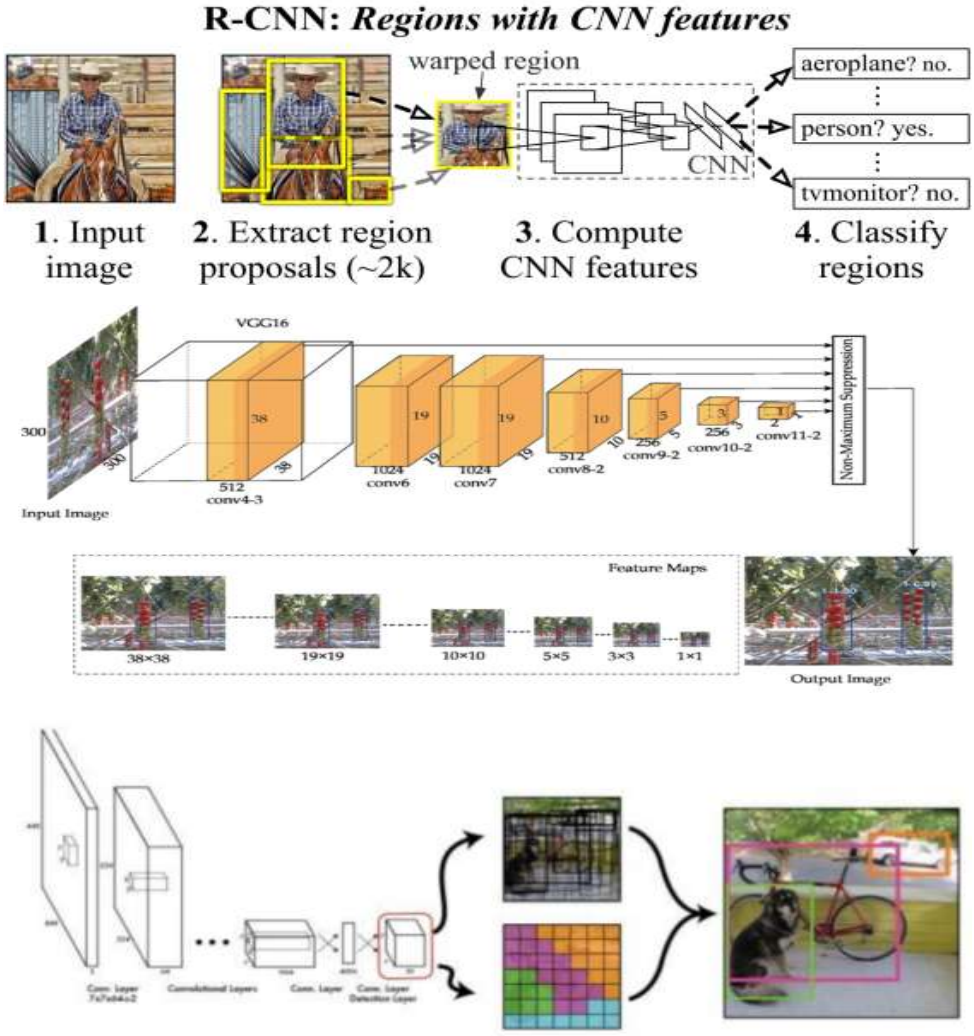


Рисунок В.1 – Існуючі технології розпізнавання об’єктів

Рисунок В.2 – Програма аналог

Рисунок В.3 – Наявні набори даних

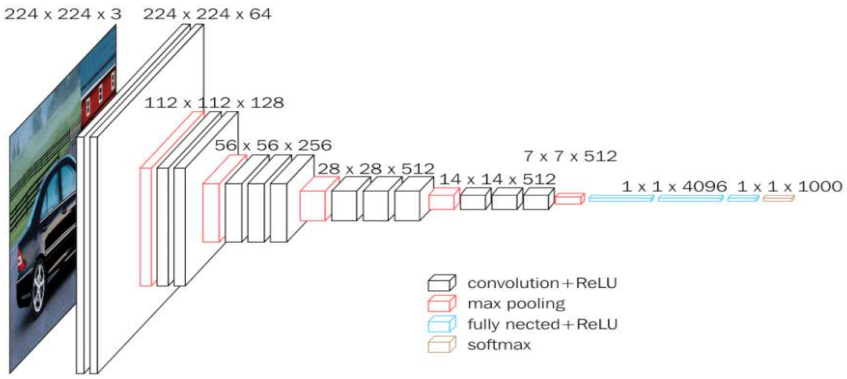


Рисунок В.4 – Базова архітектура нейронної мережі VGG

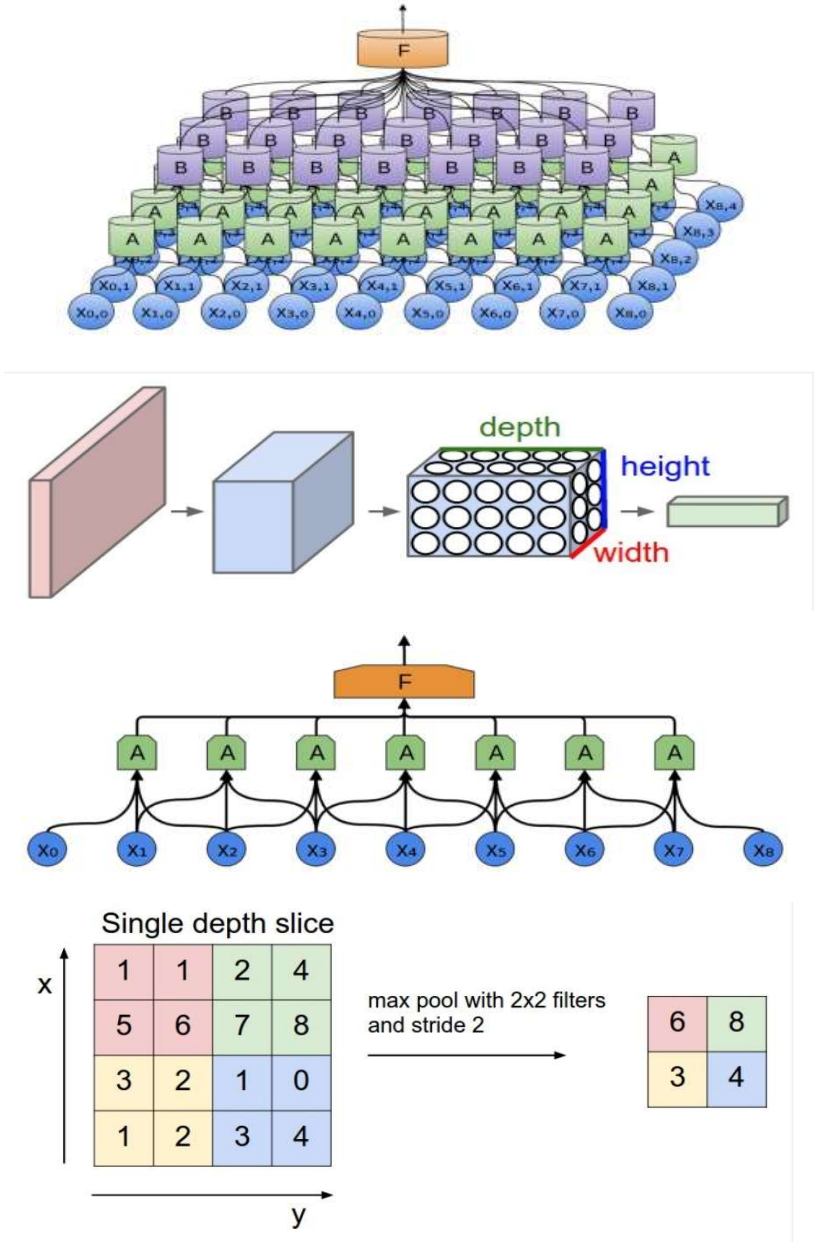
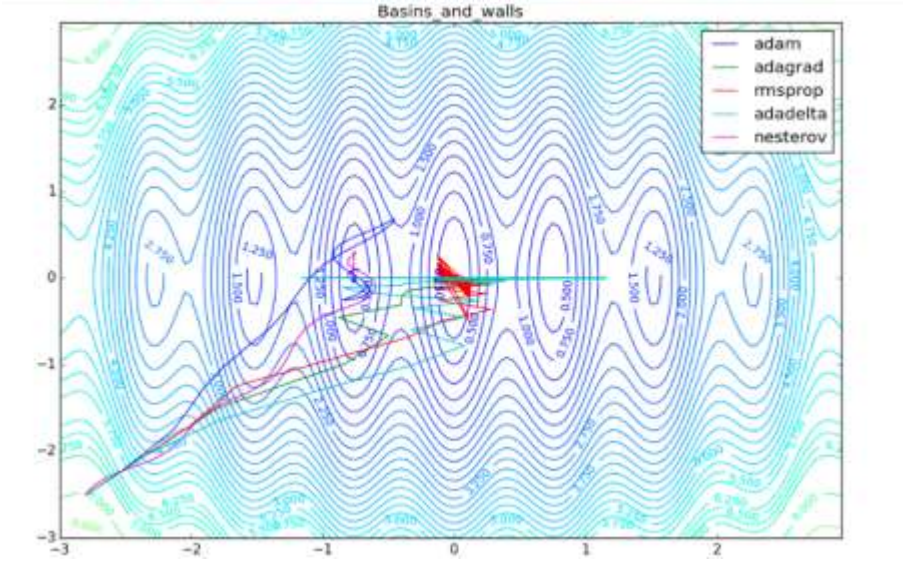


Рисунок В.5 – Використання згорткової нейронної мережі для розпізнавання об’єктів



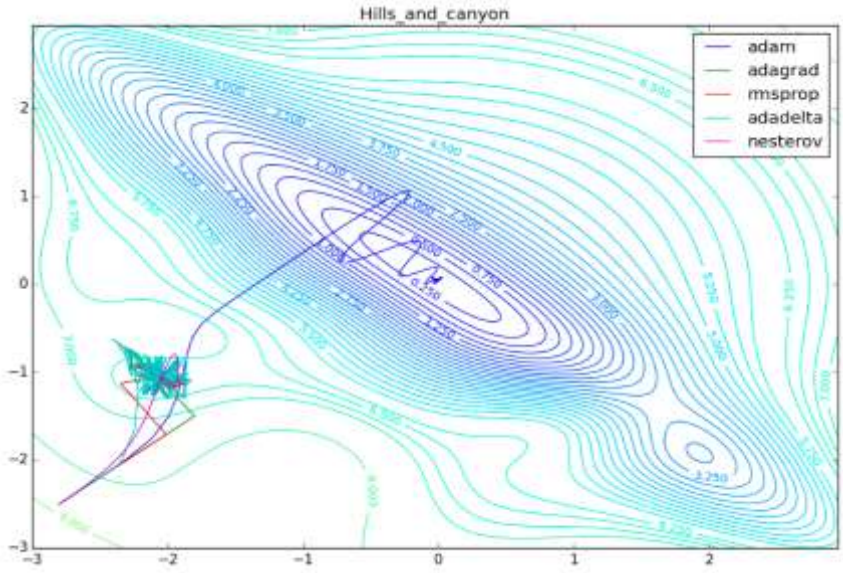


Рисунок В.6 – Порівняння методів оптимізації навчання нейронної мережі

Рисунок В.7 – Алгоритм розпізнавання об’єктів у відеопотоці

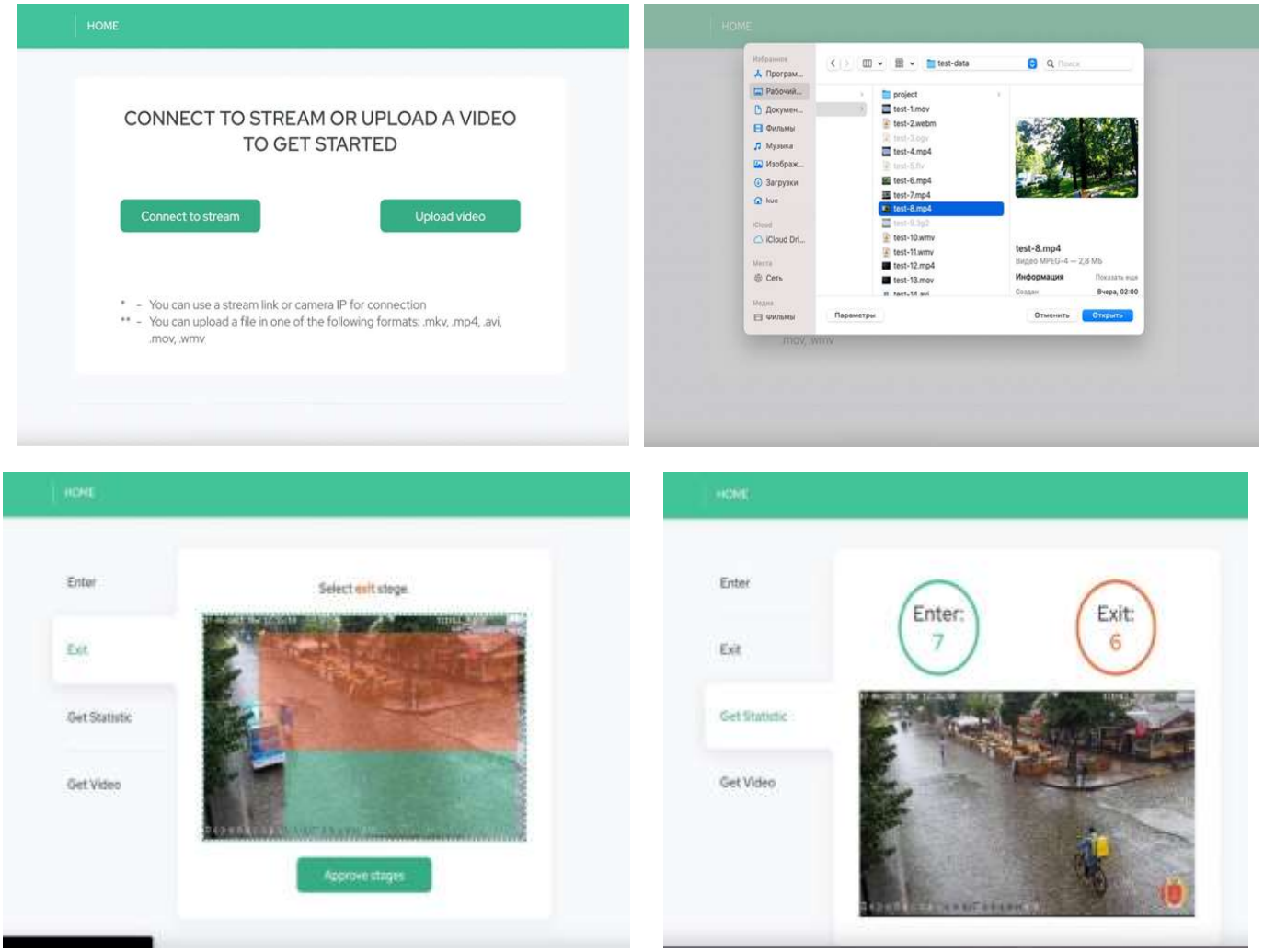


Рисунок В.8 – Графічний інтерфейс програмного забезпечення розпізнавання об’єктів у відеопотоці

Додаток Г (довідниковий)

Інструкція користувача

Після запуску програми відкривається вікно початкової активності (рис. Г.1).

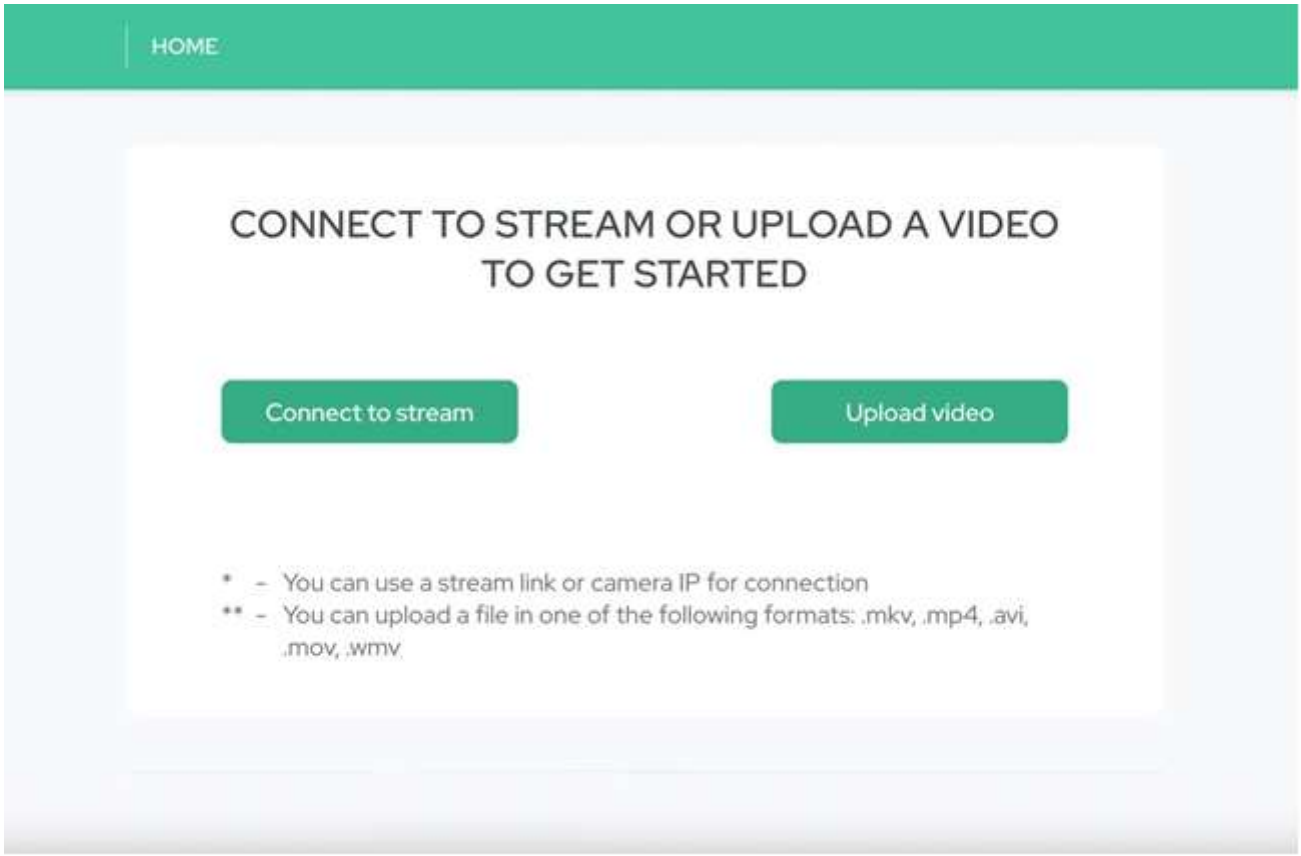


Рисунок Г.1 – Вікно початкової активності

Користувачу необхідно обрати спосіб для введення в програмне забезпечення початкових даних. Наявні можливості як підключення до камери , так і завантаження файлу.

Для наведеного далі тестування був обраний шлях завантаження підготовлених відео. Після натискання на кнопку «Upload video» відкривається стандартне вікно вибору файлу для завантаження. Одномоментно дозволяється обрати лише один файл у форматах mkv, mp4, avi, mov, wmv (рис. Г.2).

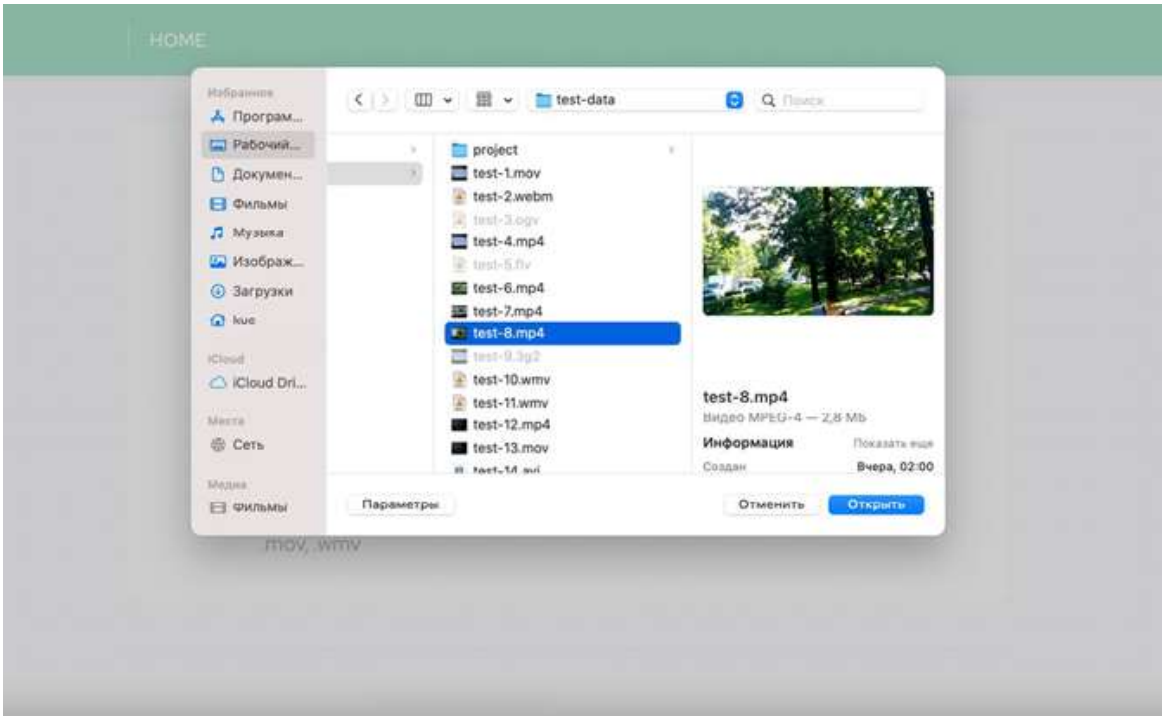


Рисунок Г.2 – Вікно вибору відео файлу

Оскільки відеофайли можуть бути великих розмірів, впроваджено лоадер для повідомлення користувача про процес завантаження відео (рис. Г.3).

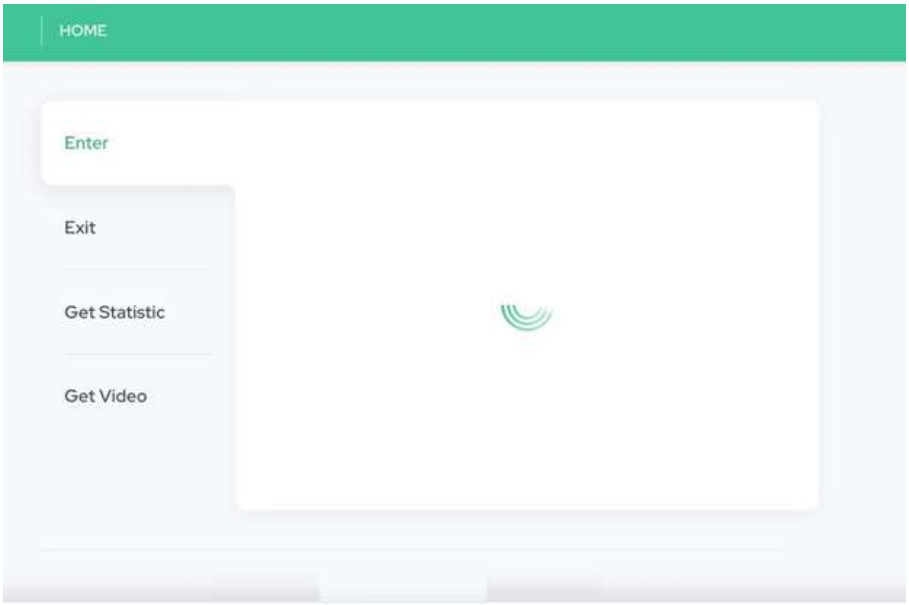


Рисунок Г.3 – Вікно процесу завантаження відео

Після успішного завантаженню файлу, програмний модуль повертає перший кадр для нанесення розмітки полігонів для опису зон переходів. Вікно нанесення розмітки полігону для опису зони входу зображено на рис Г.4.

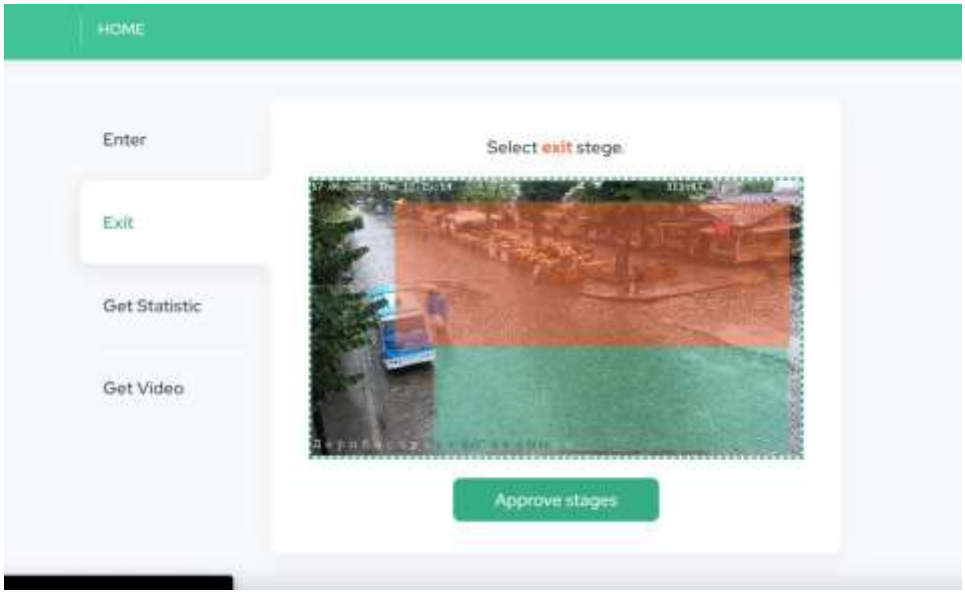


Рисунок Г.4 – Вікно нанесення розмітки полігону для опису зони виходу

Програмне забезпечення фіксує пересування об’єктів, в якості яких виступають люди, відносно цих полігонів та оновлює статистичні дані, що далі надсилаються користувачеві та виводяться на екран (рис Г.5).

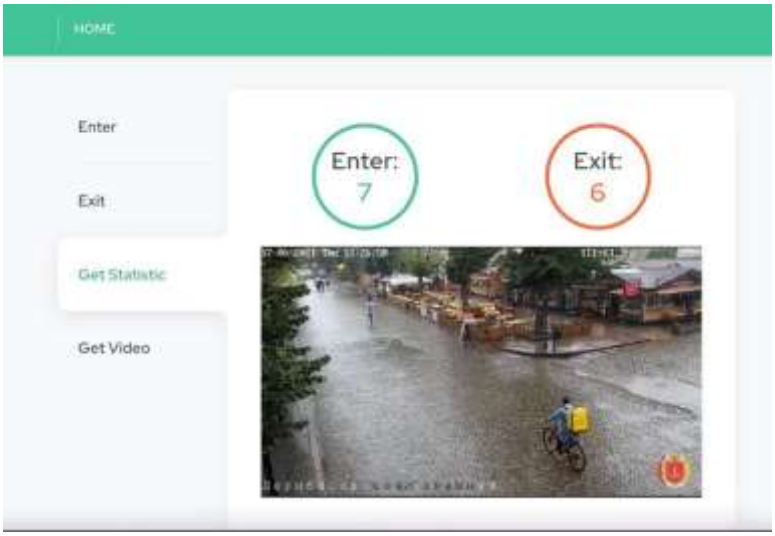


Рисунок Г.5 – Вікно виведення результатів

У програмного застосунка є можливість завантажити відео, до якого було застосовано алгоритм автоматизованого розпізнавання об’єктів та їх підрахунку у відеопотоці. При натисненні на кнопку «Get video» розпочинається процес завантаження опрацьованого файлу, який в подальшому відкривається користувачем (рис. Г.6).

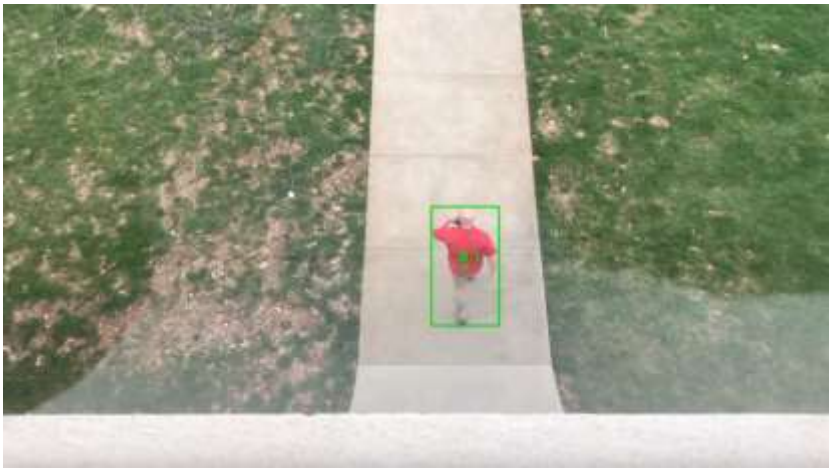


Рисунок Г.6 – Приклад опрацьованого відео

Додаток Д (довідниковий)

Довідка про впровадження

МАЛЕ НАУКОВО-ВИРОБНИЧЕ ПІДПРИЄМСТВО "ТОВ "ІТІ"
Україна, 21021, м.Вінниця, вул. Келецька, 56

№ ____ від "__" _____ 20__ р.

ДОВІДКА

Дана Левчук Ярославі Костянтинівні в тому, що результати, одержані нею в процесі виконання магістерської кваліфікаційної роботи, а саме алгоритми та програмні засоби розпізнавання та підрахунку об'єктів у відеопотоці, планується використати в розробках ТОВ «ІТІ».

Зам. директора ТОВ «ІТІ» Бодяк В.М.