Вінницький національний технічний університет

Факультет інформаційних технологій та комп'ютерної інженерії

Кафедра програмного забезпечення

**Бакалаврська кваліфікаційна робота**

на тему: «Розробка вебзастосунку для розпізнавання зображень за допомогою нейронної мережі»

Виконав: студент 4 курсу
групи 4ПІ-21б
спеціальності
121 – Інженерія програмного забезпечення
(шифр і назва напряму підготовки, спеціальності)
Дерев'янчук А. Ю.
(прізвище та ініціали)
Керівник: к.т.н., доц. каф ПЗ Черноволик Т.О.
(прізвище та ініціали)
Рецензент к.т.н.,доц, каф.ОТ Тарновський М.Г.
(прізвище та ініціали)

**Допущено до захисту**
Завідувач кафедри ПЗ
д.т.н., проф. Романюк О.Н.
«09» 06 2025 р.

ВНТУ – 2025

Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра програмного забезпечення
Рівень вищої освіти перший бакалаврський
Галузь знань 12 – Інформаційні технології
Спеціальність 121 – Інженерія програмного забезпечення
Освітньо-професійна програма – Інженерія програмного забезпечення

ЗАТВЕРДЖУЮ
Завідувач кафедри ПЗ
Романюк О.Н._____
«21» березня 2025 р.

**З А В Д А Н Н Я**
**НА БАКАЛАВРСЬКУ КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ**

Дерев'янчуку Андрію Юрійовичу

1. Тема роботи – «Розробка вебзастосунку для розпізнавання зображень за допомогою нейронної мережі».

Керівник роботи: Черноволик Галина Олександрівна, к.т.н., доцент, затверджені наказом вищого навчального закладу від 20 березня 2025 р. № 97.

2. Строк подання студентом роботи 2 червня 2025 р.

3. Вихідні дані до роботи: мова програмування Python, бібліотеки flask, flask-cors, opencv-python-headless, numpy, matplotlib, tensorflow, середовище розробки PyCharm.

4. Зміст розрахунково-пояснювальної записки: вступ; аналіз розвитку вебзастосунку для розпізнавання зображень за допомогою нейронної мережі; розробка структури та алгоритмів програмного продукту; розробка програмних модулів та реалізації вебзастосунку для розпізнавання зображень за допомогою нейронної мережі ; висновки; список використаних джерел; додатки.

5. Перелік графічного матеріалу: титульний слайд; актуальність теми; мета, об'єкт та предмет дослідження; задачі дослідження, наукова новизна, практична цінність одержаних результатів; порівняльний аналіз аналогів.

6. Консультанти розділів роботи

| Розділ | Прізвище, ініціали та посада Консультанта | Підпис, дата | |
|---|---|---|---|
| | | завдання видав | завдання прийняв |
| 1-4 | Черноволик Г. О., к.т.н., доцент | 24.03.25 | 01.06.25 |

7. Дата видачі завдання 24 березня 2025 р.

КАЛЕНДАРНИЙ ПЛАН

| № з/п | Назва етапів бакалаврської кваліфікаційної роботи | Строк виконання етапів роботи | Примітка |
|---|---|---|---|
| 1 | Аналіз розвитку вебзастосунку для розпізнавання зображень за допомогою нейроної мережі | 25.03.25 – 03.04.25 | вик |
| 2 | Розробка структури та алгоритмів програмного продукту | 04.04.25 – 15.04.25 | вик |
| 3 | Варіантний аналіз та обгрунтування вибору засобів програмної реалізації | 16.04.25 – 17.04.25 | вик |
| 4 | Розробка програмних модулів та реалізації вебзастосунку для розпізнавання зображень за допомогою нейроної мережі | 18.04.25 – 27.04.25 | вик |
| 5 | Тестування вебзастосунку | 28.04.25 – 11.05.25 | вик |
| 6 | Оформлення матеріалів до захисту БКР | 12.05.25 – 30.05.25 | вик |

Студент _____ Дерев'янчук А.Ю.
(підпис) (прізвище та ініціали)

Керівник бакалаврської кваліфікаційної роботи _____ Черноволик Г. О.
(підпис) (прізвище та ініціали)

# АНОТАЦІЯ

УДК 004.932.2:004.93

Дерев’янчук А. Ю. Вебзастосунок для розпізнавання зображень за допомогою нейронної мережі: бакалаврська кваліфікаційна робота зі спеціальності 121 – інженерія програмного забезпечення, освітня програма – інженерія програмного забезпечення. Вінниця: ВНТУ, 2025. 91 с.

На укр. мові. Бібліогр. : 30 назв ; рис. : 32 ; табл. 2.

Вебінструмент, що базується на вебтехнологіях, забезпечує: розпізнавання об'єктів на зображеннях за заданими категоріями та визначення їх розташування. Передбачено можливість взаємодії з нейронною мережею через вебінтерфейс. Для розробки застосовано мову Python, модуль Flask для розробки серверної частини, бібліотеку React та фреймворк Bootstrap для побудови вебдодатків, а також загальноприйняті стандарти машинного навчання для нейронних мереж. Застосування цієї системи сприятиме зменшенню потреби у людських ресурсах під час вивчення об'єктів на зображенях та підвищенню ефективності роботи підприємства або окремих користувачів вебдодатку. Отримані в бакалаврській кваліфікаційній роботі результати можна використати для розпізнавання зображень за допомогою нейронної мережі.

Ключові слова: Нейронна мережа, розпізнавання зображень, вебдодаток, python, flask, tensorflow, http, react, bootstrap.

# ABSTRACT

UDC 004.932.2:004.93

Derevianchuk A. Y. Web Application for Image Recognition Using a Neural Network: bachelor's qualification work in specialty 121 - software engineering, educational program - software engineering. Vinnytsia: VNTU, 2025. 91 p.

In Ukrainian. Bibliography: 30 titles; fig.: 32; tab. 2.

The web tool, based on web technologies, provides: object recognition in images according to specified categories and determination of their location. Interaction with a neural network via a web interface is provided. Python language, Flask module for backend development, React library and Bootstrap framework for building web applications, as well as generally accepted machine learning standards for neural networks were used for development. The application of this system will help reduce the need for human resources when studying objects in images and increase the efficiency of enterprises or individual web application users. The results obtained in the bachelor's qualification work can be used for image recognition using a neural network.

Keywords: Neural network, image recognition, web application, python, flask, tensorflow, http, react, bootstrap.

ЗМІСТ

# ВСТУП

**Обґрунтування вибору теми дослідження**. У сучасному світі стрімкий поступ хмарних технологій та штучного інтелекту є одним з ключових трендів у галузі інформаційних технологій. Хмарні технології надають можливість зберігати та обробляти величезні обсяги даних, що забезпечує швидку та ефективну роботу сучасних додатків і сервісів. Провідні компанії, що спеціалізуються на хмарних технологіях, пропонують широкий спектр послуг, серед яких зберігання даних, обчислювальні ресурси, бази даних та інструменти аналітики.

Водночас, у сфері штучного інтелекту також відбуваються значні зміни. Сучасні моделі штучного інтелекту здатні виконувати складні завдання, такі як розпізнавання образів, обробка мови, машинний переклад та навіть створення творчих робіт. Ці досягнення стали можливими завдяки поєднанню великих обсягів даних, потужних обчислювальних ресурсів та передових алгоритмів машинного навчання.

Інтеграція ШІ та хмарних технологій відкриває нові перспективи для розробників та підприємств. Хмарні платформи надають інструменти для розгортання та масштабування моделей штучного інтелекту. Це дозволяє швидко впроваджувати інноваційні рішення, спрощуючи процеси розробки та навчання моделей.

На сьогоднішній день штучний інтелект активно застосовується у різних галузях. У медицині штучний інтелект допомагає у виявленні захворювань. Фінансовий сектор використовує алгоритми штучного інтелекту для виявлення шахрайських операцій та оптимізації інвестиційних стратегій. У маркетингу штучний інтелект аналізує поведінку споживачів, допомагаючи персоналізувати рекламні кампанії.

Ще одним актуальним напрямом розвитку штучного інтелекту є розробка вебзастосунків на основі нейронних моделей. Такі додатки можуть забезпечувати швидкий доступ до роботи з нейромережами, генерувати відповіді на запити,

враховуючи вподобання користувача, та полегшувати користування інтернетом загалом, надаючи безліч корисних функцій. Оскільки нейронні мережі можуть навчатися індивідуально для різних потреб, автоматизуючи рутинну або необхідну роботу, ця тема є досить широкою та актуальною для детальнішого вивчення та розробок.

Програмне забезпечення, що розпізнає зображення за допомогою нейромереж, хоч і демонструє високу продуктивність, має низку типових недоліків.

Нерідко точність розпізнавання не досягає абсолютного значення. Системи можуть припускатися помилок, якщо зображення або відео низької якості, з незвичайних ракурсів, або ж об'єкти, які необхідно ідентифікувати, відсутні у навчальному наборі даних моделі. Це здатне зумовлювати хибні визначення або повну неможливість ідентифікації об'єкта.

Також викликає занепокоєння питання приватності та захищеності даних. Оскільки для аналізу зображення, як правило, передаються на віддалені сервери, виникають питання щодо їхнього зберігання, використання та доступу до них, що особливо актуально у випадку чутливої інформації.

Додатково, таке програмне забезпечення зазвичай має обмежену гнучкість. Воно використовує вже натреновані моделі, що означає складність у адаптації до розпізнавання надзвичайно специфічних або нових об'єктів, які не були враховані під час початкового навчання. Можливість самостійного навчання для кінцевого користувача у подібних готових рішеннях, переважно, відсутня.

Ураховуючи всі ці недоліки існуючих рішень, актуальною є розробка власного вебзастосунку для розпізнавання зображень за допомогою нейронної мережі.

**Зв'язок роботи з науковими програмами, планами, темами.** Робота виконувалася згідно плану виконання наукових досліджень на кафедрі програмного забезпечення.

**Мета та завдання дослідження.** Метою є збільшення точності та оперативності розпізнавання об’єктів на зображеннях в режимі реального часу за

рахунок розробки вебзастосунку для розпізнавання зображень за допомогою нейронної мережі.

**Основними завданнями дослідження є:**

– розробити архітектуру вебзастосунку для розпізнавання зображень;

– розробити структуру додатку;

– розробити дизайн та графічні елементи;

– розробити метод інтерактивного адаптивного навчання нейронної мережі в умовах вебзастосунку;

– розробити алгоритми роботи системи;

– розробити функціонал системи;

– провести тестування розробленої системи.

**Об’єкт дослідження** – метод інтерактивного адаптивного навчання, призначений для керування та спостереження за функціонуванням комп'ютерної нейромережевої системи розпізнавання зображень.

**Предмет дослідження** – засоби управління і моніторингу роботи інтерактивного адаптивного навчання нейронної мережі в умовах вебзастосунку.

**Методи дослідження.** У процесі дослідження та реалізації вебзастосунку для розпізнавання зображень було використано низку формальних і прикладних методів. Основу теоретичної частини склали методи дискретної математики**,** теорії графів і теорії алгоритмів, які дозволили моделювати логіку обробки вхідних даних, оптимізувати структури збереження інформації та забезпечити ефективність роботи алгоритмів на етапі попередньої обробки та сегментації зображень. Зокрема, графові структури використовувались для аналізу зв’язків між піксельними областями, а алгоритмічні методи дозволили мінімізувати обчислювальні витрати.

**Новизна отриманих результатів.** Розроблено метод інтерактивного адаптивного навчання нейронної мережі, який інтегровано у вебзастосунок для розпізнавання зображень. Була реалізована модульна архітектура, що поєднує інтерфейс вебзастосунку з серверною частиною, яка відповідає за обробку

зображень нейронною мережею.

Застосовано механізми кешування, формування логів, а також збереження результатів взаємодії користувача для подальшого аналізу ефективності моделі. Отримані результати свідчать про підвищення точності моделі з кожним етапом донавчання, що особливо помітно у випадках розпізнавання нестандартних чи погано структурованих зображень.

Розроблений метод дозволяє створити адаптивний вебзастосунок, здатний ефективно функціонувати в умовах змінного середовища та постійного оновлення вхідних даних. Така система може використовуватись у сферах, де точність класифікації зображень має критичне значення: медичні зображення, моніторинг безпеки, аграрні технології, контроль якості у виробництві тощо.

**Практична цінність отриманих результатів**. Практична цінність одержаних результатів полягає в тому, що розроблено вебзастосунок, алгоритми та методи які дозволяють розпізнавати зображення за допомогою нейронної мережі.

**Особистий внесок здобувача.** Усі наукові результати, викладені у бакалаврській кваліфікаційній роботі, отримані автором особисто. У друкованих працях, опублікованих у співавторстві, автору належать такі результати: використання модифікованної нейронної мережі для розпізнавання зображень [1].

**Апробація матеріалів бакалаврської кваліфікаційної роботи.** Основні положення БКР доповідалися та обговорювалися на Міжнародній конференції «Молодь в науці: дослідження, проблеми, перспективи Вінницького національного технічного університету (МН ВНТУ–2025).

# 1 АНАЛІЗ РОЗВИТКУ ВЕБЗАСТОСУНКУ ДЛЯ РОЗПІЗНАВАННЯ ЗОБРАЖЕНЬ ЗА ДОПОМОГОЮ НЕЙРОНОЇ МЕРЕЖІ

## 1.1 Аналіз стану вебзастосунків для розпізнавання зображень

Застосування підходу, відомого як «глибоке навчання», стало поштовхом для створення найпродуктивніших систем штучного інтелекту за останнє десятиліття. Яскравими прикладами таких систем є програми розпізнавання мовлення для смартфонів або найновіший автоматичний перекладач від Google. По суті, «нейронні мережі» є передовою концепцією в штучному інтелекті, що становить суть глибокого навчання. Варто зазначити, що Уоррен Маккалок і Уолтер Піттс, два науковці з Чиказького університету, вперше представили концепцію нейронних мереж ще у 1944 році. У 1952 році вони продовжили свою роботу в Массачусетському технологічному інституті, де вони створили те, що іноді вважають першою кафедрою когнітивних наук [3]. Осередком досліджень як у нейронауці, так і в обчислювальних науках є нейромережі. Модулі глибинного навчання потребують великої кількості інформації, оскільки на ній ґрунтується процес навчання. Обсяг набутих знань безпосередньо залежить від обсягу даних. Метод машинного навчання, що має назву нейронні мережі, дозволяє комп'ютерам опановувати виконання різних завдань через аналіз навчальних прикладів. Ці приклади зазвичай створюються вручну. Скажімо, для вивчення потрібних зображень, таких як люди, будівлі, авто, тварини тощо, використовується система розпізнавання об'єктів. На цих зображеннях система визначатиме візуальні патерни, які послідовно співвідносяться з конкретними мітками. Після навчання на зображеннях, нейронну мережу можна застосовувати в різноманітних областях, як-от розпізнавання облич, підрахунок транспортних засобів або відстеження об'єктів з камери дрона. Нейронна мережа, змодельована за зразком людського мозку, включає мільйони щільно сполучених простих обчислювальних елементів. Переважна більшість сучасних нейронних мереж складаються з шарів елементів і є «прямими», тобто інформація може поширюватися через них лише в одному визначеному напрямку (див. рис. 1.1).

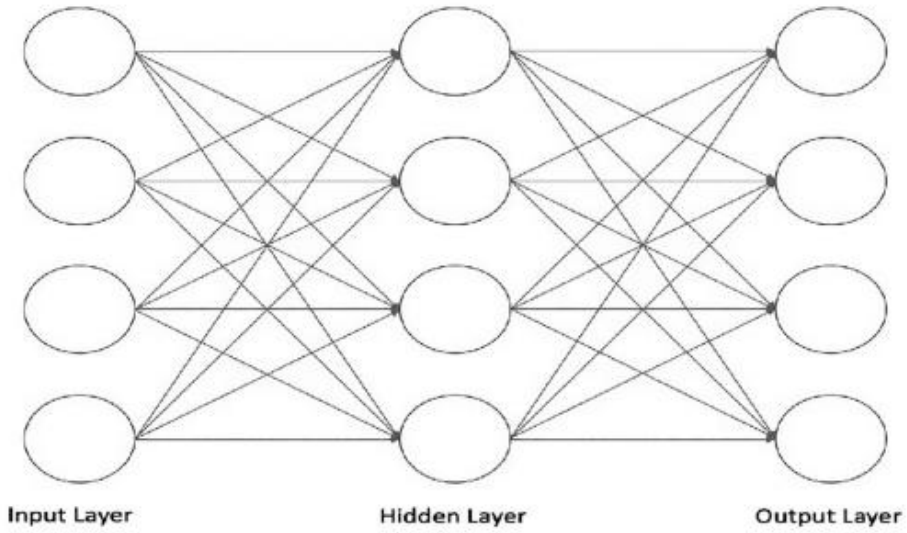


Рисунок 1.1 – Шари нейронної мережі

Під час тренування нейронної мережі, всі її ваги спочатку отримують значення: або випадкові числа, або результати спеціальних методів, як-от ініціалізація Глорота чи Хейма. Від правильного встановлення ваг напряму залежить швидкість та стабільність збіжності моделі. Далі мережа здійснює передачу даних: від вхідних даних до вихідних, де кожен шар, спираючись на ці дані та встановлені ваги, обчислює свої вихідні значення. Отримання прогнозованих результатів моделі – фінальний етап цього процесу, відомого як пряме поширення. Як тільки передбачені результати отримано, настає час обчислення функції втрат, яку іноді називають просто «втратою». Ця функція визначає величину різниці між фактичними та передбаченими значеннями [2].

Завдяки потужності обчислень і здатності оперувати великими масивами інформації, штучний інтелект знайшов застосування в різних сферах. Він дозволяє покращити якість прогнозів та аналізу, а також відкриває нові горизонти для взаємодії та адаптації до потреб користувачів. Це дає можливість автоматизувати завдання, що раніше потребували значних людських ресурсів.

Найбільші стрімінгові сервіси, такі як Netflix, застосовують системи персоналізованих рекомендацій. Це лише один з багатьох прикладів того, як штучний інтелект інтегровано в різні сфери діяльності по всьому світу. Алгоритми машинного навчання Netflix аналізують перегляди, уподобання та

взаємодію користувачів з платформою. На основі цих даних вони пропонують фільми та серіали, які, ймовірно, зацікавлять конкретних глядачів. Збільшення тривалості часу, який користувачі проводять на платформі, сприяє поліпшенню їхнього досвіду та підвищує задоволеність від використання сервісу [12].

Автомобілі Tesla, а також інші автоматизовані транспортні засоби застосовують штучний інтелект, щоб гарантувати безпечний та раціональний рух без участі водія [13].

Крім того, штучний інтелект (ШІ) застосовується в медичній діагностиці: після аналізу отриманих медичних зображень навчена нейромережа здатна виявити хворобу та допомогти медикам у встановленні діагнозу та вирішенні щодо лікування.

Технології розпізнавання предметів за допомогою нейромереж вже міцно увійшли у функціонал сучасних мобільних ґаджетів. Скажімо, під час розблокування смартфона реалізується нейронна мережа для ідентифікації обличчя в режимі реального часу. Це значно підвищує рівень захисту пристрою, водночас забезпечуючи швидкий доступ до його функцій [3].

Фотографія – це візуальне відображення, захоплене світлочутливим сенсором або плівкою за допомогою оптичного інструменту, наприклад, фотоапарата. Вона створює нерухоме зображення, щоб відтворити сцену реальності, яке зберігається як файл на відповідному носії. Для обробки цього зображення нейронною мережею, необхідно спочатку його підготувати. Попередня обробка фотографії може передбачати зміну розміру, нормалізацію значень пікселів та інші процедури для стандартизації вхідних даних. Підготовлені фотографії проходять через різні шари нейромережі. Ці шари включають згорткові та повністю з'єднані шари. Кожен шар виконує різноманітні операції над зображенням, щоб сприяти визначенню особливостей об’єктів. Нейронна мережа аналізує зображення, щоб виявити такі характеристики, як форма, текстура та кольори. Нейронна мережа здатна розпізнавати різні риси на різних рівнях абстракції за допомогою згорткових шарів. Нейронна мережа групує об’єкти на зображенні відповідно до конкретних категорій або класів

після виявлення рис. Це може бути досягнуто за допомогою повністю з'єднаних шарів, які можна використовувати для визначення ймовірності приналежності об'єкта до певного класу. На завершальному етапі нейромережа виводить результати розпізнавання об'єктів, які можна візуалізувати у вигляді текстових міток на зображенні або списку класів, а також ймовірності їх належності до певного класу.

Також нейромережа задіяна для виявлення об'єктів на відео. Відео – це відтворення картинок, що змінюються в реальному часі. Кожну з цих картинок звемо кадром. Коли ці кадри з'являються дуже швидко, то виникає ілюзія руху. Відео можна зафіксувати камерою, або скласти з окремих зображень-кадрів, з'єднаних у послідовний потік. Кожний кадр відео передає інформацію про те, що відбувається у сцені, в конкретний момент часу. Ступінь плавності руху у відео залежить від кількості кадрів, що показуються за секунду. Швидкість зміни кадрів у відео зазвичай фіксована, наприклад, 24, 30 чи 60 кадрів за секунду, але може різнитися в залежності від налаштувань запису або програвання.

Коли нейронна мережа починає обробку, відео розбивається на окремі кадри. Далі нейромережа опрацьовує кожний кадр. На кожному з кадрів нейромережа розпізнає об’єкти шляхом аналізу пікселів та пошуку характерних рис об'єктів. Результати виявлення об'єктів можна показати в реальному часі або записати на пристрій для подальшого аналізу. Це може бути показ відео з виділенням знайдених об’єктів або створення звіту про знайдені об'єкти та їхні параметри [26].

### 1.2 Порівняльний аналіз аналогів

Стрімкий розвиток цифрового простору стимулює інтеграцію штучного інтелекту в програмне забезпечення та додатки, створені для вирішення широкого спектру задач. Нейронні мережі дають змогу організаціям адаптуватися до мінливих трендів та запитів клієнтів, тим самим покращуючи продуктивність. Машинне навчання розкриває безліч перспектив для бізнесу. Нейронні мережі прискорюють бізнес-процеси та підвищують їхню

ефективність в умовах конкурентної боротьби на ринку.

Однією з ключових сфер застосування штучного інтелекту є комп'ютерний зір. Алгоритми глибинного навчання та машинного навчання показують вражаючі результати, зокрема в області розпізнавання об'єктів. Люди з легкістю розпізнають людей, предмети, сцени та візуальні деталі, переглядаючи фотографії чи відео. Мета комп'ютерного зору полягає у тому, щоб навчити комп'ютер здатності, притаманній людині: розуміти зображення на рівні, близькому до людського. Розпізнавання об'єктів здійснюється різними способами. Машинне навчання та глибинне навчання стали основними підходами до розпізнавання предметів в останні роки. Хоча обидві методики використовуються для розпізнавання об'єктів на зображеннях, їх підходи відрізняються.

Глибоке навчання нині набуло широкої популярності у сфері розпізнавання об'єктів. Згортові нейронні мережі (CNN) та інші глибинні моделі використовують автоматичний процес вивчення характеристик об'єкта для його ідентифікації. Скажімо, CNN здатна відрізнити котів від собак, аналізуючи тисячі зображень під час навчання та визначаючи ключові відмінності між цими тваринами.

Вибір між машинним та глибинним навчанням залежить від задачі та особливостей програми, яку потрібно реалізувати. Машинне навчання може виявитися ефективним у багатьох ситуаціях, зокрема, коли відомі важливі ознаки зображень, що використовуються для розрізнення об'єктів різних класів. Ключовими факторами при виборі між машинним та глибинним навчанням є доступність обчислювальних ресурсів (потужний комп'ютер) та велика кількість зображень для навчання. Якщо організація або людина не можуть забезпечити ці вимоги, машинне навчання стає оптимальним вибором. Глибоке навчання, як правило, ефективніше з великими наборами даних, а використання потужного графічного процесора (GPU) сприяє прискоренню процесу тренування моделі [21].

Google Lens – це надзвичайно корисний сервіс від Google. Він дозволяє

визначати об'єкти, розпізнавати текст, рослини та тварин. Крім цього, він вміє перекладати тексти у реальному часі, використовуючи камеру вашого ґаджета або зображення, що вже є у вашому пристрої(див. рисунок 1.2).

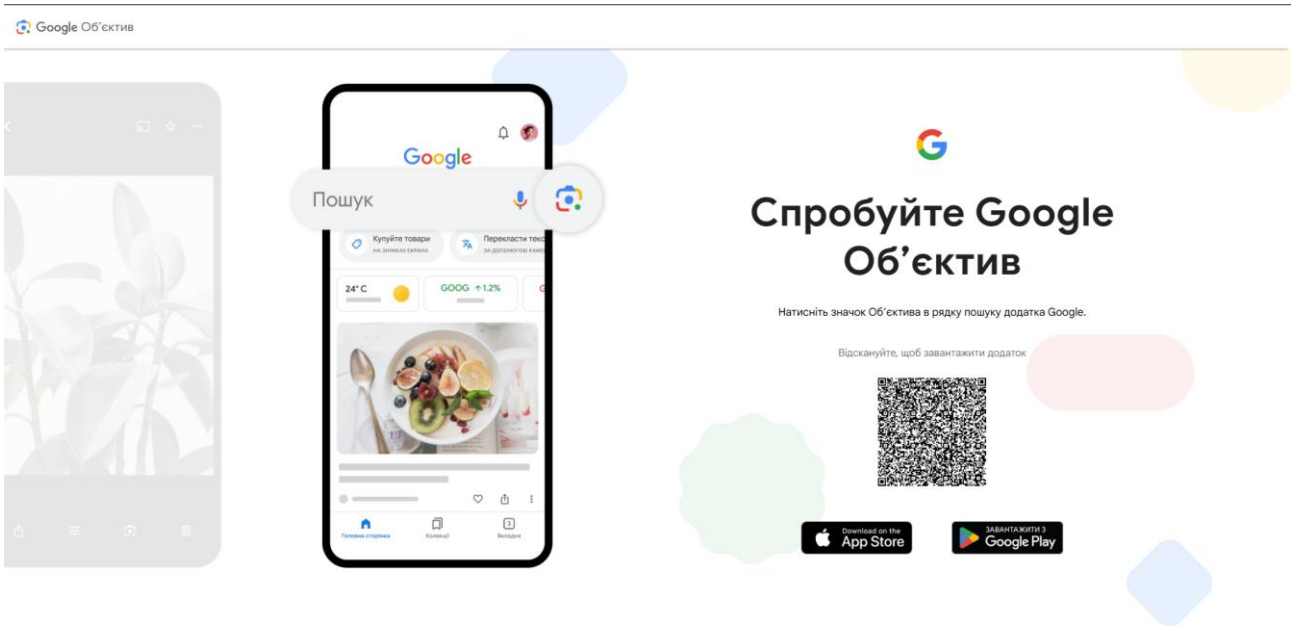


Рисунок 1.2 – Інтерфейс вебзастосунку Google Lens

Amazon Rekognition – це сервіс, розроблений Amazon Web Services (AWS), який спеціалізується на аналізі фотографій та відео. З його допомогою можна виявляти різні об'єкти, людей, текст(див. рисунок 1.3).

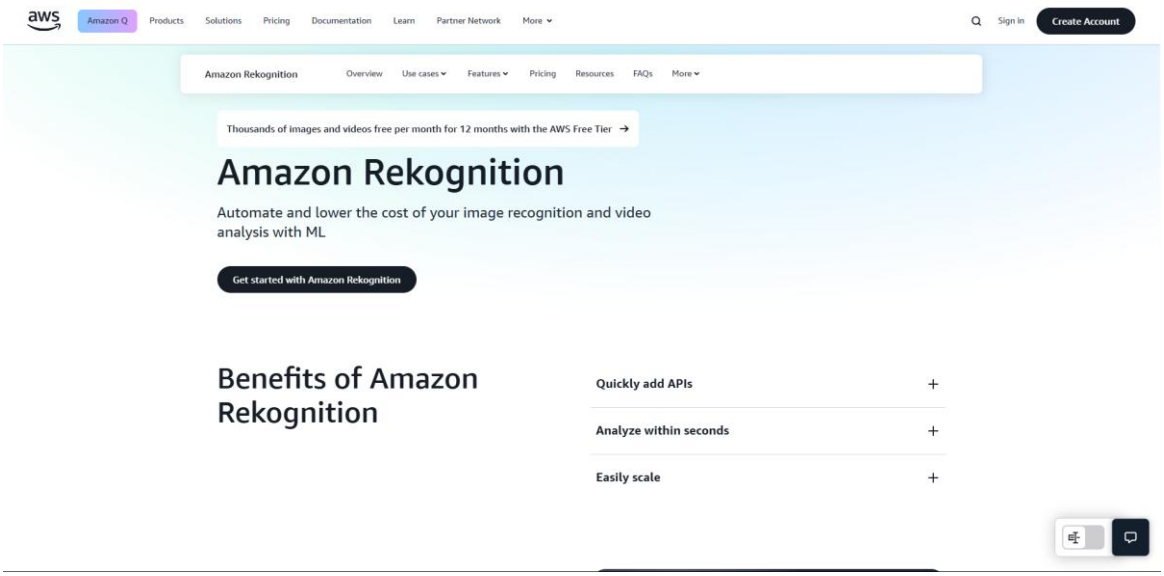


Рисунок 1.3 – Інтерфейс вебзастосунку Amazon Rekognition

Clarifai – це платформа штучного інтелекту, що надає можливості розпізнавання зображень та відео на високому рівні для багатьох сфер (дивитись рисунок 1.4).

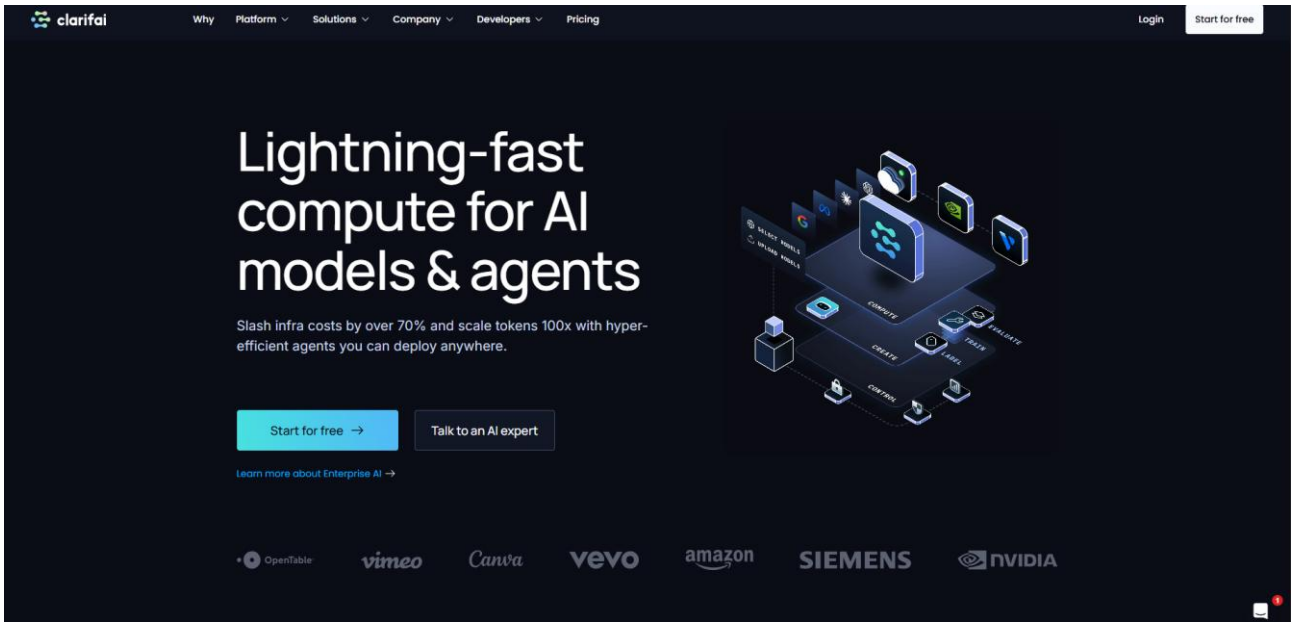


Рисунок 1.4 – Інтерфейс вебзастосунку Clarifai

Результати порівняння аналогів зведено в табл. 1.1.

Таблиця 1.1 – Порівняльна характеристика аналогів

| Функціонал | Google Lens | Amazon Rekognition | Clarifai | Власний застосунок |
|---|---|---|---|---|
| Завантаження зображень та відео | - | + | + | + |
| Розпізнавання зображень | + | + | + | + |
| Розпізнавання об’єктів на відео | - | + | + | + |
| Самонавчання на основі даних | - | - | + | + |
| Інтерфейс користувача | + | - | - | + |
| Модель використання | + | - | - | + |
| Загальна оцінка | 50% | 50% | 66% | 100% |

Для розпізнавання об'єктів із застосуванням глибинного навчання використовуються два основні методи: навчання моделей «з нуля» та застосування заздалегідь навченої моделі.

Навчання глибинної мережі з самого початку передбачає збір значного обсягу даних із мітками та розробку архітектури мережі, здатної вивчати відповідні ознаки, для подальшого створення моделі. Досягнуті результати часто є вражаючими, але цей підхід потребує значної кількості даних для навчання та складного налаштування шарів і вагових коефіцієнтів згорткової нейронної мережі [16].

Якщо використовувати попередньо навчену модель, це може значно полегшити процес навчання. Більшість програм глибокого навчання застосовують підхід передачі навчання, процес, що передбачає точне налаштування вже перевіреної моделі. Така модель може мати вже готові знання, тому можна подати нові дані з невідомими класами, аби створити потрібну нейронну мережу. Також існують базові моделі, котрі пропонують лише попередньо сконструйовану архітектуру мережі, не проходячи жодних попередніх тренувань. Хоча ці моделі вимагають додаткових налаштувань і більше інформації для тренування, кінцевий результат може виявитися кращим. У будь-якому разі, використання попередньо навченої моделі спрощує навчання і забезпечує позитивний результат.

Щоб виконати функцію розпізнавання об’єктів за допомогою машинного навчання, необхідно почати з підготовленої колекції зображень, після чого обрати необхідні характеристики для кожного зображення. Розділивши ці характеристики на окремі категорії, модель машинного навчання використовує цю інформацію для аналізу та класифікації нових об’єктів. Для розробки точної моделі класифікації та локалізації об’єктів, швидше за все, знадобиться застосування різних алгоритмів машинного навчання та методів вилучення характеристик, які пропонують безліч різних комбінацій[17].

### 1.3 Аналіз інтерактивного адаптивного навчання нейронної мережі в умовах вебзастосунку

За останнє десятиліття, нейронні мережі для розпізнавання об'єктів зросли якісно. Еволюція архітектур суттєво поглибилася і покращилася, включаючи в себе традиційні підходи, такі як згорткові нейронні мережі, машини опорних векторів, а також методи, що базуються на регіонах: Fast R-CNN, детектори одного кадру та YOLO. Сучасні моделі здатні оперативно й з високою точністю визначати об’єкти на відео та зображеннях, а також відносити їх до певних категорій.

З 1990-х років, коли були представлені перші згорткові нейронні мережі, розвиток нейронних мереж для розпізнавання об’єктів став надзвичайно цікавою та динамічною сферою. За цей період було досягнуто значних успіхів, що дозволили нейромережам реалізовувати завдання класифікації та виявлення об’єктів з вражаючою точністю. Це відкрило безліч перспективних можливостей у багатьох сферах, зокрема в медичній діагностиці, системах безпеки та спостереження, суттєво підвищуючи продуктивність і точність процесів. Автомобілі, що застосовують нейромережі для розпізнавання об’єктів на дорозі, є яскравим прикладом реального застосування цих технологій. Робототехніка також використовує нейромережі для створення більш інтелектуальних та самостійних систем.

Зважаючи на все це, етичні аспекти та ймовірні наслідки неправомірного застосування технологій розпізнавання об'єктів вимагають особливої уваги. Щоб ці технології служили на благо суспільства та були безпечними, критично важливо їх ретельно регулювати та застосовувати відповідально. Технологічні гіганти та науково-дослідні організації продовжують вкладати кошти в розробку передових моделей та методів, прагнучи зробити нейромережі ще досконалішими та надійнішими.

Враховуючи динамічний розвиток цієї сфери, цілком логічно передбачати, що нейромережі для розпізнавання об'єктів дедалі більше інтегруватимуться в наше щоденне життя, породжуючи як нові перспективи, так і певні виклики.

Задля гарантування безпечного та продуктивного застосування цієї технології, а також для максимально ефективного використання її потенціалу, вкрай важливо постійно тримати руку на пульсі [27].

Попри значний поступ, актуальні нейронні моделі також демонструють недоліки. Наприклад, вони можуть вимагати значних обчислювальних потужностей та тривалого часу на процес навчання. Недостатнє або надмірне навчання деяких моделей здатне призвести до погіршення показників на нових вхідних даних. Крім того, можливо, що наявні моделі не завжди можуть розпізнати конкретні типи об'єктів, особливо за умов недостатнього освітлення чи обмеженої видимості.

Скажімо, атаки з викривленням зображення використовують для генерування спеціально модифікованих зображень, що знижує точність роботи нейромережі. Зловмисники можуть застосовувати цей метод для ускладнення процесу розпізнавання об'єктів та спотворення результатів класифікації.

Інший різновид атак – це перехресні мережі, де зловмисники можуть маніпулювати зовнішніми даними або взаємодіяти з кількома нейронними мережами для впливу на результати. Це може бути вкрай загрозливо у випадках, коли важлива точність і безпека класифікації, наприклад, у медичній діагностиці чи системах безпеки.

Ще один вид небезпеки – атаки на персональні дані моделі. Зловмисник може розголошувати секретну інформацію або зламати систему, якщо він матиме доступ до чутливих даних, як-от ваги моделі або навчальні дані [28].

Одним із пунктів бакалаврської роботи є створення методу, здатної точно розпізнавати клас і розташування об'єктів на фото або відео. Перевагою методу інтерактивного адаптивного навчання нейронної мережі в умовах вебзастосунку є те, що його завжди можна покращувати, навчаючи новим класам. Розроблений метод дозволяє використовувати нейронну мережу як у комерційних, так і в особистих цілях, без зайвих зусиль та без використання обчислювальних ресурсів власного пристрою.

У результаті розробки даного методу процес зберігається у вихідному

файлі і відправляється сервером до вебсторінки (див. рисунок 1.4).

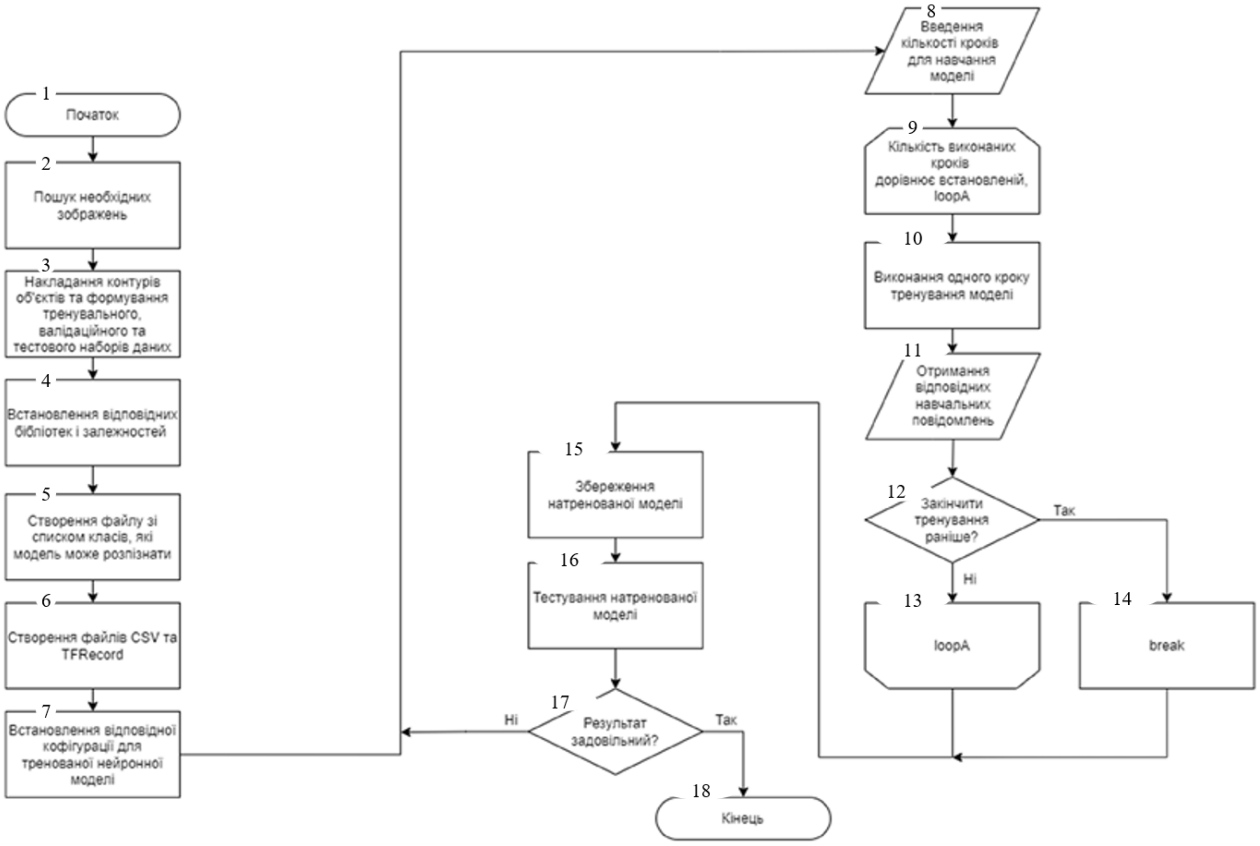


Рисунок 1.4 – Алгоритм навчання нейронної мережі

Усі ці операції відбуваються миттєво і не вимагають від користувача жодних додаткових маніпуляцій, окрім завантаження файлу. Це забезпечує зручний досвід та швидке оброблення даних, не витрачаючи власні ресурси.

### 1.4 Постановка задач дослідження

Провівши аналіз методу інтерактивного адаптивного навчання нейронної мережі в умовах вебзастосунку, порівняння аналогів було поставлено такі задачі дослідження:

- розробити архітектуру вебзастосунку для розпізнавання зображень;
- розробити структуру додатку;
- розробити дизайн та графічні елементи;
- розробити метод розпізнавання зображень нейронною мережею;
- розробити алгоритми роботи системи;

- розробити функціонал системи;
- провести тестування розробленої системи.

### 1.5 Висновки

У першому розділі було проведено аналіз проблематики та можливих реалізацій у сфері нейромереж та розробки на їх основі вебзастосунків.

Було розглянуто існуючі технології, які дозволяються побачити весь спектр доступних технологій та різних підходів до розробки вебдодатків, що базуються на навчених нейронних моделях. Детальний аналіз різних підходів дозволив детальніше ознайомитися з потребами в даній області та краще спланувати мету дослідження.

На основі цього було сформовано завдання бакалаврської роботи, яке охоплює розробку вебдодатку, що базується на основі модифікованої нейронної моделі розпізнавання об’єктів. Мета проєкту передбачає побудову комфортного вебзастосунку для будь-якого пристрою, що включає в себе роботу з натренованою нейронною моделлю розпізнавання медіа-об’єктів на зображеннях та відеофайлах без використання великої кількості ресурсів.

Також було розглянуто різні варіанти та середовища розробки відповідних вебзастосунків на базі нейромереж. Після певних обміркувань було обрано найкомфортніші для застосування у даному проєкті.

Таким чином, перший розділ став теоретичним фундаментом для реалізації майбутнього проєкту у даній сфері.

# 2 РОЗРОБКА СТРУКТУРИ ТА АЛГОРИТМІВ ПРОГРАМНОГО ЗАСТОСУНКУ

## 2.1 Аналіз вхідних даних системи

Програмне забезпечення – це сукупність інструкцій, даних або програм, які слугують для управління комп'ютером та виконання поставлених задач. Воно включає системне програмне забезпечення, яке керує апаратними компонентами комп'ютера і забезпечує базову функціональність, зокрема операційну систему, та прикладне програмне забезпечення, що виконує конкретні завдання для користувача, такі як текстові редактори, веббраузери чи графічні редактори.

При розробці веборієнтованого інструменту розпізнавання медіа-об’єктів на основі модифікованої нейронної мережі виникла необхідність вибору відповідного програмного забезпечення. У даній роботі програмне забезпечення можна розділити на дві основні частини: ПЗ для тренування та тестування нейронних моделей та ПЗ для розробки вебдодатків.

Щодо програмного забезпечення для роботи з нейромережами, то тренування та тестування здійснювалися з використанням бібліотеки TensorFlow. Це відкрита бібліотека машинного навчання, розроблена Google. Вона призначена не лише для створення та навчання моделей глибокого навчання, але й для розв'язання інших задач машинного навчання та числових обчислень. TensorFlow пропонує гнучкість і масштабованість, що дозволяє розробникам створювати моделі для різноманітних платформ, від мобільних пристроїв до розподілених систем [6].

Сам процес навчання та перевірки здійснювався у Google Colab. Цей хмарний сервіс представляє собою інтерактивне середовище для написання та запуску коду на мові Python. Google Colab користується обчислювальними потужностями хмари, що дозволяє реалізовувати важкі обчислювальні задачі [11].

Для навчання нейронної моделі було обрано модель SSD MobileNet V2 FPNlite. Модель SSD MobileNet V2 FPNLite являє собою модифікацію моделі

розпізнавання об'єктів, яка інтегрує ефективність MobileNet V2 з можливостями FPN. Дана модель забезпечує точне і швидке розпізнавання об'єктів та є особливо корисною для роботи в умовах обмежених обчислювальних ресурсів, наприклад, на мобільних пристроях або у веббраузерах.

MobileNet v2 – це результативна архітектура для глибокого навчання, що використовує розподілену згортку, яка значно скорочує кількість параметрів і обчислень, при цьому зберігаючи високу продуктивність [14].

SSD – це методика розпізнавання об’єктів, котра реалізує детекцію за один прохід.

FPN – це архітектура нейронної мережі, яка використовує багаторівневу піраміду ознак, що дозволяє ефективніше виявляти об'єкти різних розмірів та покращувати точність детекції на усіх рівнях масштабування. Спрощена версія FPN оптимізована для зменшення обчислювальних затрат, зберігаючи при цьому високу точність розпізнавання.

Модель SSD Mobile Net V2 FPN lite є придатною для інтеграції з вебсайтами завдяки її властивостям, які забезпечують оптимальне співвідношення між швидкістю та точністю, а також можливостям оптимізації для роботи в умовах обмежених ресурсів.

Розглядаючи програмні засоби для розробки вебзастосунків, у цій роботі було застосовано React та Bootstrap для розробки і дизайну інтерфейсу користувача, а також Flask для реалізації серверної частини [20].

React – це бібліотека JavaScript, призначена для побудови інтерфейсів взаємодії з користувачем. Вона надає розробникам можливість творити масштабовані та динамічні вебзастосунки, базуючись на компонентному підході, що забезпечує ефективне повторне використання програмного коду. Ключовими особливостями React є застосування віртуального DOM для прискорення оновлення інтерфейсу, односпрямоване потокопередавання даних, що полегшує моніторинг змін, та JSX, розширений синтаксис JavaScript для кодування інтерфейсів у вигляді HTML-подібного коду. Крім того, React інтегрує передові інструменти, зокрема React Hooks, що надають можливість використовувати стан

та інші можливості у функціональних компонентах. Завдяки своїй адаптивності та функціональності, React займає позицію одного з найпопулярніших інструментів для веброзробки [9].

Bootstrap – найвідоміший інструмент для створення вебінтерфейсів, побудований на HTML, CSS і JavaScript. Ця платформа надає широкий вибір вже готових компонентів, які дають змогу швидко і без зайвих зусиль розробляти сучасні та зручні вебсайти. Bootstrap також містить корисні інструменти CSS, серед яких сітка та типографіка, що сприяють розробці приємних для ока і функціональних дизайнів для вебдодатків. Він дає можливість розробникам зосереджуватися на функціях своїх проєктів, прискорюючи створення та розгортання проєктів через наявність великого арсеналу готових рішень [15].

Flask – це мікрофреймворк для створення вебзастосунків мовою Python. Він надає легкий та адаптивний шлях до конструювання вебсайтів, даючи змогу швидко збирати вебдодатки різного ступеня складності. Flask пропонує базовий набір інструментів для розгортання вебсервера, але також може бути удосконалений через різні розширення для роботи з базами даних, авторизацією, формами та іншими можливостями. Завдяки простому для розуміння синтаксису та відмінно задокументованій структурі, він є надзвичайно популярним для швидкої розробки вебдодатків на Python [7].

Отже, ключовою складовою процесу створення цього вебдодатку є визначний редактор PyCharm. Версія PyCharm professional вражає своєю зрозумілістю та легкістю в роботі, завдяки інтуїтивному інтерфейсу. Редактор надає можливості для розширення функціональності через численні плагіни та розширення, адаптовані до різних мов програмування, фреймворків та інструментів. З цієї причини PyCharm виявився надзвичайно корисним інструментом для розробки вебзастосунків, забезпечуючи зручний інтерфейс, масштабованість та інтегровані інструменти для роботи з широким спектром мов та вебтехнологій [18].

Апаратне забезпечення для керування нейронними мережами охоплює спеціалізовані обчислювальні пристрої та елементи, що гарантують високу

швидкість обробки та ефективність при здійсненні комплексних математичних операцій, властивих навчанню нейронних мереж та їхньому тестуванню.

Оскільки в цьому дослідженні для розв'язання поставленої задачі використовувався хмарний сервіс для розробки та тренування моделей Google Colab, то було залучено його ключові складові апаратного забезпечення.

У цьому контексті серцем системи є процесор AMD Ryzen 7 9700X CPU @ 3.80GHz. Цей процесор від AMD розроблений для обробки значних обчислювальних навантажень, що робить його чудовим вибором для навчання моделей машинного навчання. AMD Ryzen відомий своєю надійністю та безперебійною роботою, що є ключовим аспектом для тривалих обчислювальних операцій, необхідних для навчання нейронних мереж. Процесори цього типу демонструють низький ризик виникнення помилок та високу стабільність, що гарантує безперервний процес навчання моделі, мінімізуючи втрати даних і часу. Ці процесори оснащені новітніми інструкціями, зокрема AVX-512, які оптимізують обчислювальні завдання та підвищують швидкість обробки числових даних. Гнучкість використання процесора для попередньої обробки даних у поєднанні з GPU або TPU для навчання моделей забезпечує ефективний розподіл обчислювальних ресурсів. Процесори AMD Ryzen використовуються для виконання скриптів, обробки даних, підготовки навчальних наборів та інших операцій, які не потребують спеціалізованого апаратного прискорення. Висока продуктивність, стабільність та підтримка значних обсягів оперативної пам'яті роблять його ідеальним для роботи з великими моделями машинного навчання, забезпечуючи ефективне та безперебійне виконання обчислювальних задач [29].

Розглядаючи графічні процесори, Google Colab відкриває доступ до найсучасніших GPU, серед яких NVIDIA A100, T4 та інші. Для тренування та налагодження необхідної нейронної моделі було застосовано GPU NVIDIA Tesla T4. Побудований на архітектурі Turing, графічний процесор T4 оптимізовано для високопродуктивних обчислень з наголосом на енергоефективності. Він надає високу продуктивність з відносно низьким споживанням енергії, що робить його ефективним вибором для довгострокових обчислювальних задач. GPU T4

містить тензорні ядра, які спроектовані для прискорення операцій глибинного навчання, особливо множення матриць та кумулятивних обчислень. Тензорні ядра значно прискорюють процес навчання моделей нейронних мереж, особливо для задач, що базуються на матричних операціях, таких як згорткові нейронні мережі. Ядра CUDA, вбудовані в графічний процесор, здатні ефективно обробляти великі обсяги даних паралельно, що сприяє прискоренню навчання та тестування моделей. У Google Colab GPU T4 надається користувачам як компонент хмарної інфраструктури. Це дозволяє дослідникам та розробникам використовувати потужні обчислювальні ресурси без потреби в власному дорогому обладнанні. Графічний процесор T4 включає прискорене навчання моделям нейронних мереж, ефективну обробку великих обсягів даних, високу продуктивність, а також сумісність з численними популярними фреймворками, що сприяє експериментам з різними архітектурами нейронних мереж. Вищесказане робить графічний процесор NVIDIA Tesla T4 відмінним вибором для навчання та тестування нейронних мереж у хмарному середовищі Google Colab [30].

Ще одним необхідним елементом апаратного забезпечення під час роботи з нейронними модулями є оперативна пам'ять. Google Colab відкриває доступ до великого обсягу оперативної пам'яті. А саме, він пропонує 12,7 ГБ доступної оперативної пам'яті. Це критично важливо при роботі з великими наборами даних. Оперативна пам'ять використовується для зберігання даних, які потребують швидкого доступу під час обчислень, наприклад, параметри моделі та проміжні результати. За наявності достатньої кількості оперативної пам'яті можна виконувати обчислення з великими обсягами даних без проблем із продуктивністю. Як хмарний сервіс, Google Colab забезпечує дослідників та розробників доступом до значної кількості оперативної пам'яті без потреби у власному обладнанні. Достатній обсяг оперативної пам'яті збільшує продуктивність, скорочує час очікування та забезпечує стабільність, що робить Google Colab привабливим інструментом для експериментів і розробок в області машинного навчання та штучного інтелекту (див. рисунок 2.1).

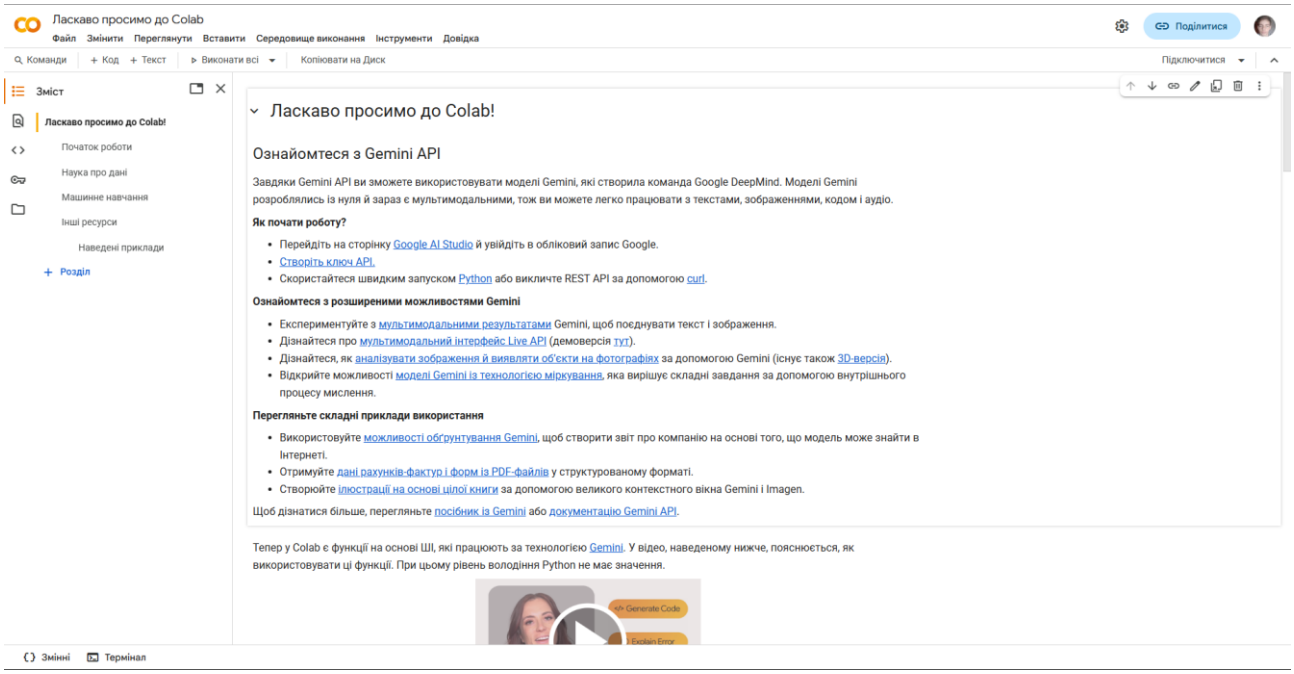


Рисунок 2.1 – Головний екран вебсайту Google Colab

Важливою складовою роботи з нейромережами є наявність достатньої кількості місця на диску для зберігання даних. Google Colab пропонує 107,7 ГБ вільного простору. Дискова система, використана Google Colab, демонструє високу продуктивність та надійність під час вирішення обчислювальних задач, що потребують швидкого доступу до великих обсягів інформації. SSD-накопичувачі можуть суттєво пришвидшити обробку критично важливих даних у проєктах, пов'язаних з машинним навчанням та аналізом великих даних. Завдяки інтеграції з Google Drive користувачі можуть без проблем зберігати, отримувати доступ та ділитися своїми даними, що робить Google Colab зручним інструментом для дослідників, розробників та викладачів. Дані, що зберігаються на диску Google Colab, захищені шифруванням. Це гарантує захист даних від несанкціонованого доступу та захищає конфіденційність користувача. Твердотільні накопичувачі менш схильні до механічних пошкоджень, ніж жорсткі диски, через відсутність рухомих частин. Це підвищує надійність та термін служби диска, а також зменшує ризик втрати даних [4].

## 2.2 Розробка інтерфейсу програмного додатку

Графічний інтерфейс користувача (GUI) надає можливість взаємодії з електронікою через графічні складові: вікна, значки, кнопки та списки вибору. На противагу текстовому інтерфейсу, GUI звільняє користувачів від потреби вводити команди у вигляді тексту. Головна відмінність – взаємодія з графічними об'єктами відбувається за допомогою миші, клавіатури або сенсорного екрану, замість використання текстового терміналу [19].

Ось основні складові графічного інтерфейсу користувача:

1. Вікна – прямокутні ділянки на екрані, в котрих може бути розміщено різноманітний контент: тексти, зображення, а також інші графічні складники.
2. Іконки – маленькі графічні зображення, які представляють собою програму, файл або інший об'єкт. Вони сприяють швидкому доступ до цих об'єктів.
3. Меню – список доступних опцій або команд, з котрих користувач може вибрати потрібну. Меню бувають випадаючими, контекстними або головними, кожен з яких має свої власні функції та спосіб застосування.
4. Кнопки – графічні елементи, на які можна натискати, щоб запустити певну дію, наприклад, відкрити програму або зберегти файл.
5. Панель інструментів – панель, що містить кнопки та інші елементи керування, які дозволяють оперативно виконувати найчастіше використовувані команди.
6. Форми та поля введення – це поля, призначені для введення користувачами різних даних, як наприклад текст чи необхідні файли.

В цілому розроблений вебдодаток виглядає наступним чином (див. рис.2.2).

Графічний інтерфейс є набагато інтуїтивнішим, особливо для тих, хто не має досвіду роботи з текстовими командами. Графічні елементи легко впізнаються та зрозумілі.

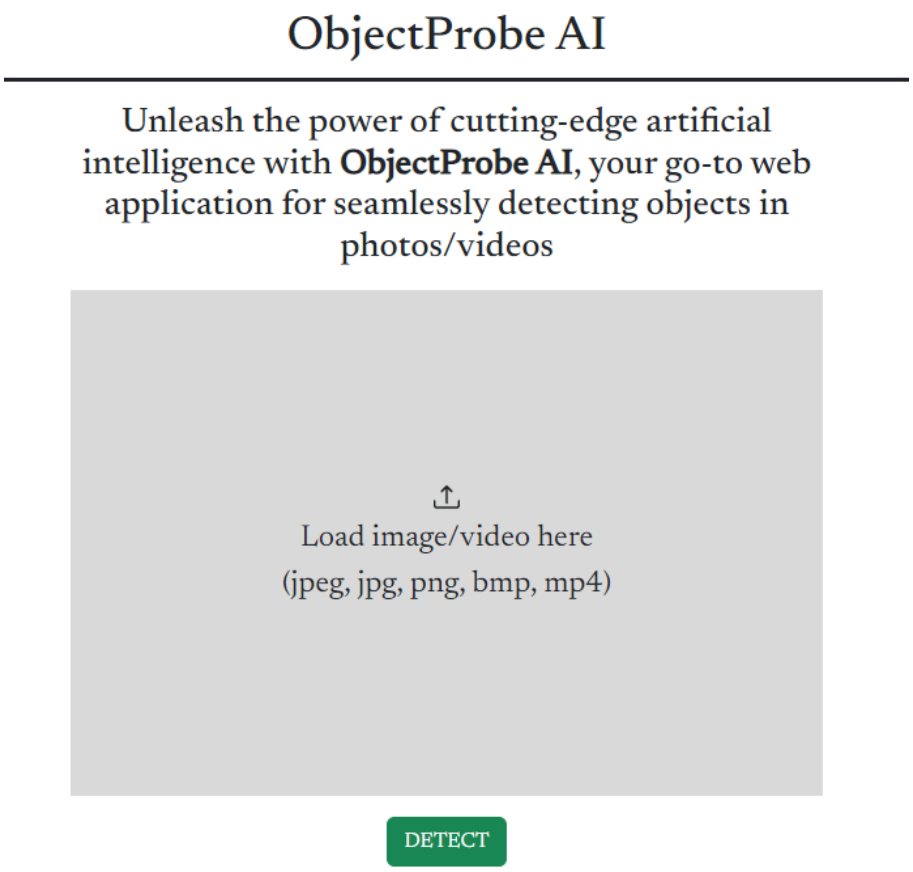


Рисунок 2.2 – Зовнішній вигляд головної сторінки розробленого вебдодатка

За допомогою миші або іншого пристрою введення користувачі можуть швидко виконувати завдання, не потребуючи запам'ятовування складних команд. Графічний інтерфейс користувача – це графічний інтерфейс із миттєвим візуальним зворотним зв'язком, який допомагає користувачам зрозуміти їхні дії та отримані результати, а також дає змогу ширшому колу людей, у тому числі й тим, хто не володіє технічними знаннями, зручно використовувати розроблений додаток.

Розглядаючи складові застосунку ізольовано, можна визначити такі компоненти графічного інтерфейсу, а саме заголовок, опис додатку та робоча зона.

При відкриті розробленого вебдодатку користувача зустрічає відповідний заголовок (див. рисунок 2.3).


Рисунок 2.3 – Зовнішній вигляд заголовку

Заголовок графічного інтерфейсу користувача відіграє ключову роль у впізнаваності та орієнтації користувача. Зазвичай, його розміщують у верхній частині вікна або сторінки. Цей елемент є критичним для зручності використання, суттєво полегшує навігацію та загальне сприйняття інтерфейсу користувачем.

Назва сайту відіграє не менш важливу роль у залученні відвідувачів та формуванні першого враження про вебресурс. Вона може суттєво впливати на ставлення до розробленого додатку. Легка для запам'ятовування та приваблива назва допоможе виокремитися серед інших ресурсів та підвищити комфорт користування платформою. Назва може відображати основні цілі та цінності додатка, сприяючи встановленню зв'язку з цільовою аудиторією.

Крім того, назва може бути розроблена з урахуванням специфіки цільової аудиторії, використовуючи мову, зрозумілу та близьку для неї. Важливо, щоб назва включала релевантні ключові слова та терміни, які відображають суть контенту та послуг, що пропонуються, сприяючи покращенню видимості сайту у пошукових системах.

У цій роботі заголовок виконує роль репрезентанта. Він дає назву розробленій технології, що сприяє розумінню користувачем мети створеного додатку ще до початку взаємодії з сайтом.

Одразу після заголовку користувач зустрічає поле з короткою інформацією про розроблений додаток (див. рисунок 2.4).

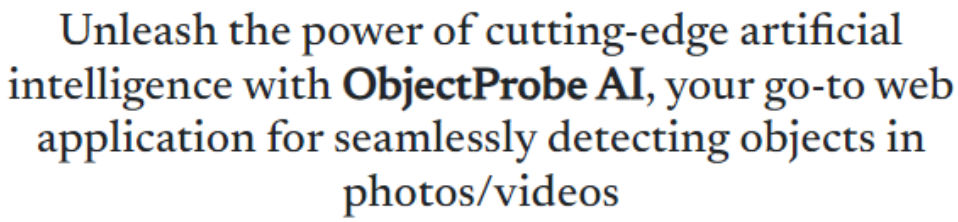


Рисунок 2.4 – Поле з короткою інформацією про розроблений додаток

Короткий виклад даних на вебсайті здатний відчутно поліпшити взаємодію з відвідувачами. Зручна структура дозволяє сконцентруватися на ключових моментах товару чи послуги, полегшуючи користувачам розуміння матеріалу. Відвідувачі швидше засвоюють головне, не витрачаючи час на перегляд об'ємних сторінок, та економлять час на пошук потрібної інформації. Стисла та конкретна інформація сприяє кращому запам'ятовуванню, робить платформу більш доступною та привабливою для повторного звернення. Проста структура допомагає уникнути втрати уваги через надлишок інформації або надмірно складний виклад.

У цьому розділі розробленого додатка подано стислу інформацію про створений вебдодаток. Це потрібно для швидкого ознайомлення користувача з можливостями, які пропонує ця розробка.

Ознайомившись з функціоналом та цілями розробленої вебсторінки, користувач безпосередньо переходить до взаємодії з вебдодатком (див. рис. 2.5).

Розробка зрозумілої та інтуїтивної частини для взаємодії з вебдодатком – найважливіше завдання при його створенні. Простий та інтуїтивно зрозумілий дизайн дозволяє користувачу легко орієнтуватися та знаходити потрібну інформацію, не докладаючи зайвих зусиль. В результаті, користувач припускається менше помилок і покращує загальне враження від використання.

Прості розділи дають можливість користувачам миттєво виконувати дії та поставлені завдання, адже їм не доводиться марнувати час на осмислення громіздких інструкцій чи пошук необхідних функцій. Спрощений дизайн значно розширює доступність вебдодатку, роблячи його придатним для широкого кола

користувачів, зокрема й тих, хто не має технічних знань.

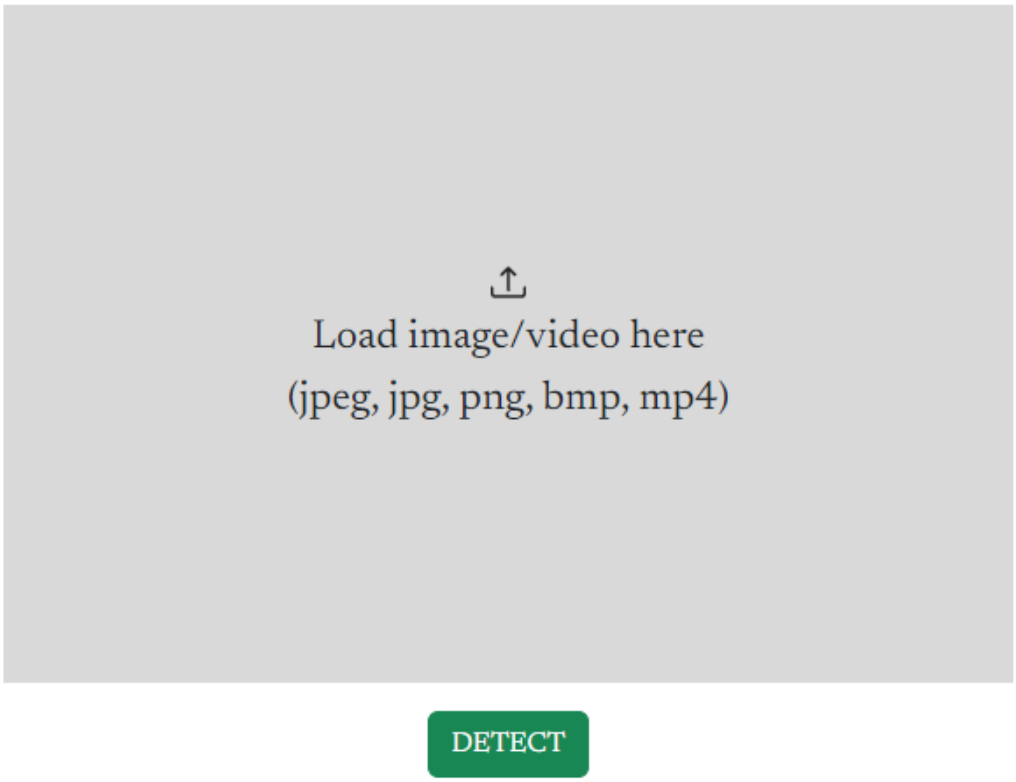


Рисунок 2.5 – Розділ взаємодії з вебдодатком

Користувачі мають змогу сформувати позитивне враження про вебдодаток, насолоджуючись легкістю використання зрозумілого та ефективного інтерфейсу. Простота взаємодії та чіткість навігації можуть сприяти збільшенню конверсії, оскільки користувачі досягають поставлених цілей, виконуючи необхідні або бажані дії значно швидше.

У полі взаємодії з додатком користувачеві надається доступ до завантаження зображення або відеофайлу (див. рисунок 2.6).

При натисканні на вкладку завантаження файлів з'явиться вікно, що дозволяє обрати файл потрібного формату для завантаження на сайт.

Після вибору файлу користувач матиме змогу переглянути його на своєму екрані до відправлення на сервер. Це дозволить уникнути помилок при завантаженні фото чи відео.

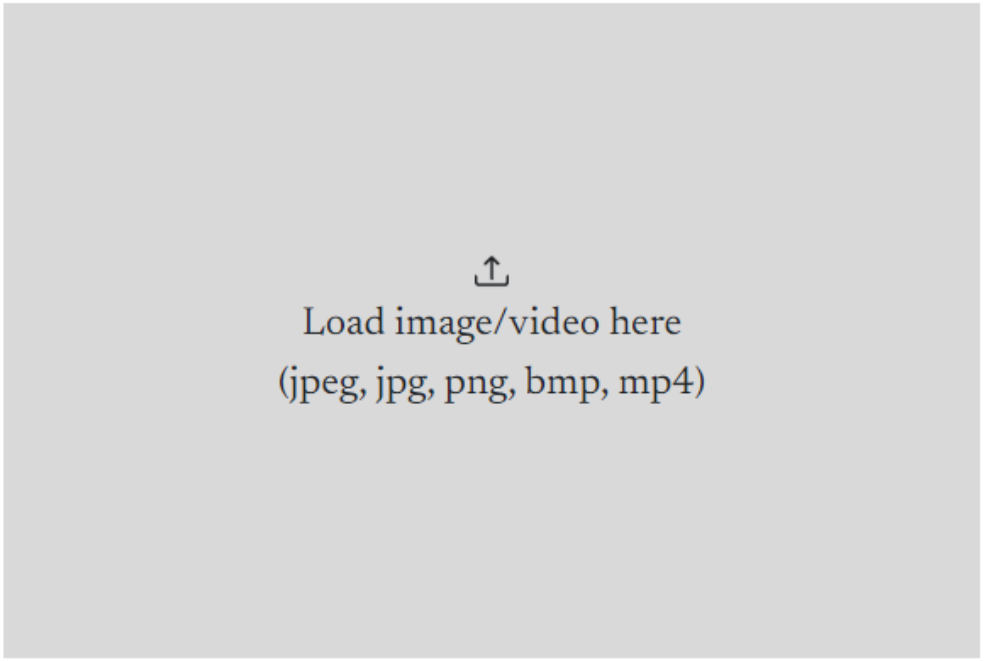


Рисунок 2.6 – Розділ для завантаження зображення або відеофайлу

Отже, користувачі отримують зручний та інтуїтивно зрозумілий інтерфейс для роботи з вебдодатком. Це сприятиме позитивному досвіду взаємодії з вебзастосунком, утриманню наявних користувачів та залученню нових.

### 2.3 Метод інтерактивного адаптивного навчання нейронної мережі

Під час розробки методу інтерактивного адаптивного навчання нейронної мережі, призначеного для взаємодії з нейронною мережею, критично важливо ретельно продумати кожен елемент системи. У процесі використання вебзастосунку користувач може оцінювати та коригувати результати розпізнавання, а система миттєво враховує ці зміни для підвищення точності та адаптації моделі до конкретного контексту.

Для цього було створено UML діаграму діяльності, яка візуалізує основні сценарії взаємодії користувача з програмою (див. рисунок 2.7).

На діаграмі активності зображено взаємодію користувача з вебзастосунком, що стосується завантаження файлу на ресурс. Розпочинається процес з визначення формату файлу, тобто користувач має змогу завантажити на сайт зображення або відеоматеріал. Після того, як файл потрапляє на сервер, програма

визначає, чи це фотографія, чи відео, та відображає оброблену інформацію на екрані користувача.


Рисунок 2.7 – Блок-схема інтерактивного адаптивного навчання нейронної мережі

Далі користувач приймає рішення, чи потрібно йому завантажити кінцевий результат з сайту, після чого він натискає на оброблену картинку або відео. Завершується процес отриманням результату роботи нейронної мережі.

### 2.4 Розробка алгоритмів роботи системи

Розроблений вебзастосунок працює наступним чином. При вході за відповідним посиланням, користувач отримує доступ до роботи з розробленим вебзастосунком.

Далі користувач вибирає відповідний файл, який потрібно опрацювати розробленою нейромережею. Це може бути забраження форматів JPG, JPEG, PNG, BMP (див. рисунок 2.8) або відеофайл формату MP4 (див. рисунок 2.9).

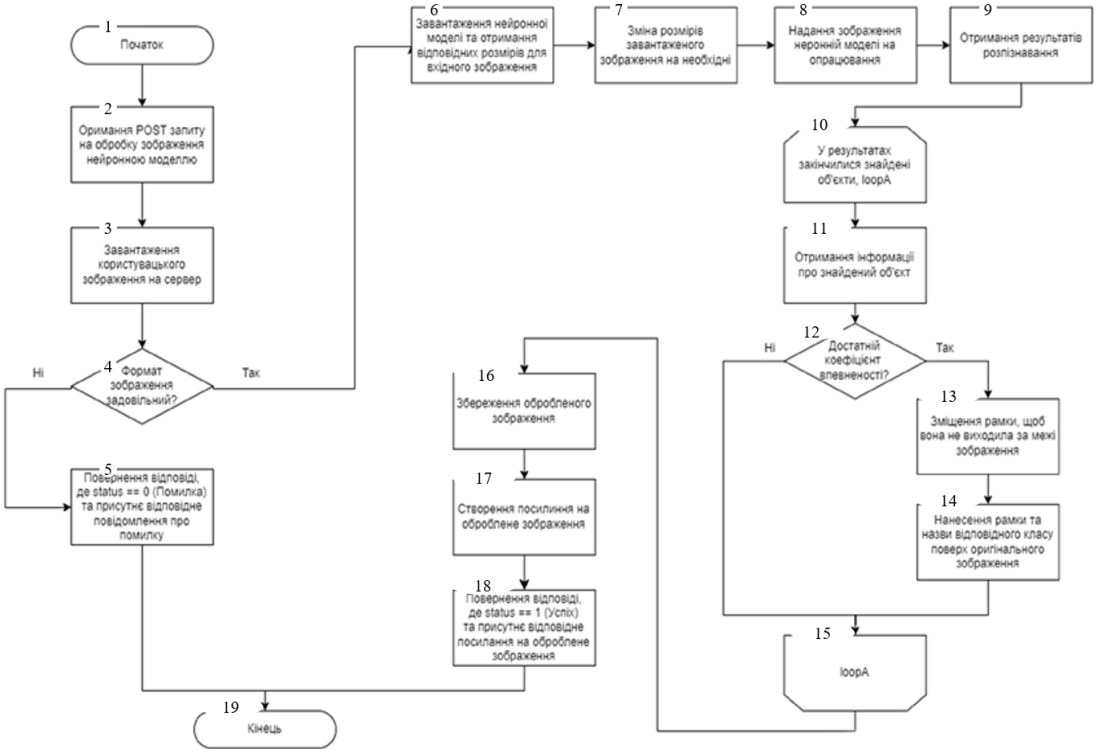


Рисунок 2.8 – Алгоритм обробки зображень

Після того, як завантажений необхідний файл, і натиснуто кнопку “DETECT”, файл відправляється на сервер, де починається його обробка нейронною мережею.

Отримавши файл, сервер спочатку перевіряє його формат на відповідність. Здійснивши потрібні перевірки, відбувається завантаження навченої нейромережі, і далі починається обробка вхідного файлу для його підготовки до опрацювання нейронною моделлю.

Після отримання оброблених даних від нейромережі, починається процес постобробки файлу. Він включає в себе накладання відповідних контурів і написів на подане зображення або відео.

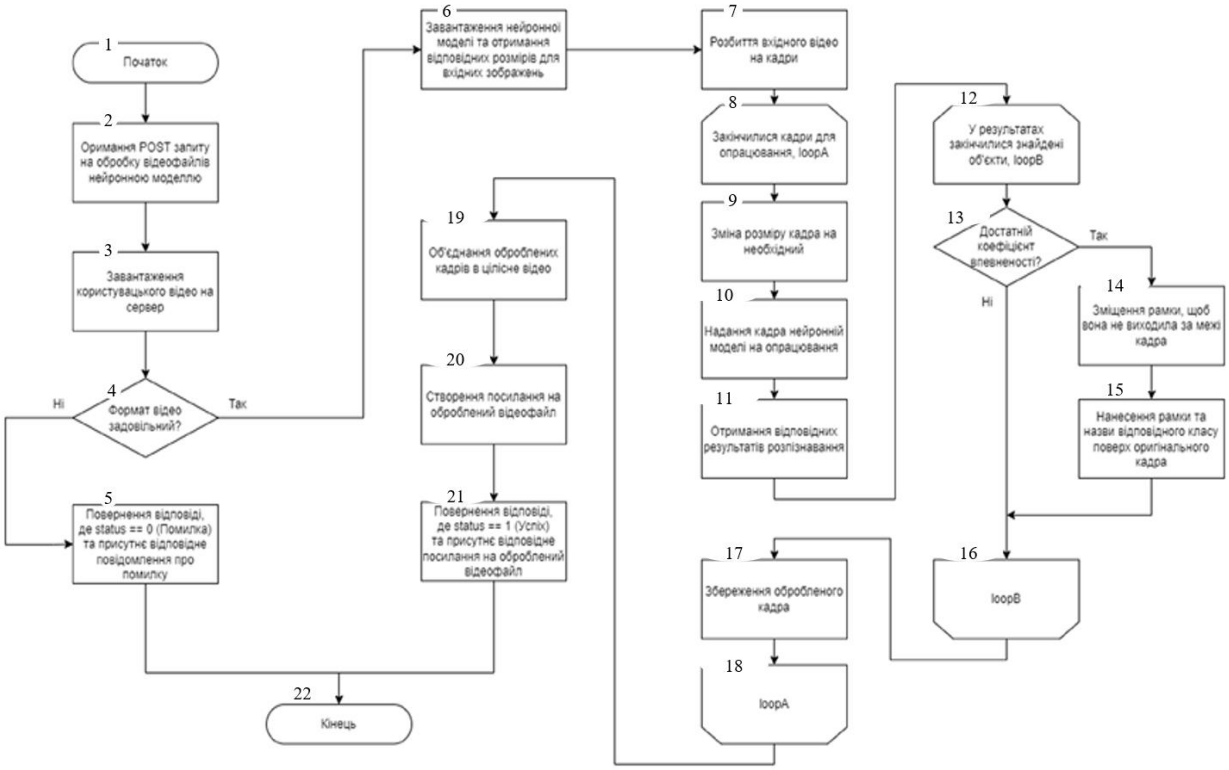


Рисунок 2.9 – Алгоритм обробки відео

Після завершення обробки користувач має змогу завантажити готовий файл на свій локальний пристрій. Якщо це зображення, потрібно натиснути на нього для завантаження. Якщо це відеофайл, необхідно натиснути кнопку “DOWNLOAD” під відео.

Якщо користувачеві не сподобалась виконана робота нейронною мережею, він може спробувати ще раз завантажити картинку чи відео, тим самим вдосконалить обізнання нейронної мережі та можливо отримає бажаний ним результат.

## 2.5 Висновки

У другому розділі продемонстровано використання різноманітних складових, які використовуються для розробки та запуску вебдодатку, заснованого на модифікованій нейронній моделі.

Наведено приклади програмного та апаратного забезпечення, застосовного для навчання нейронної моделі та подальшого розгортання веб застосунку. Проаналізовано позитивні аспекти використання конкретного програмного та апаратного забезпечення, а також визначено мінімальні вимоги для розгортання цього вебдодатку.

Окремо було оглянуто та описано файли, які використовуються для формування та підтримки роботи вебдодатку. Це дозволяє прослідкувати логіку функціонування розробленого додатку та заглибитись у деталі його роботи.

В цьому розділі також приділено увагу графічному інтерфейсу користувача. Визначено вплив графічного інтерфейсу на взаємодію користувачів з додатком, а також способи залучення та утримання цільової аудиторії. Описано основні переваги використаного графічного рішення та додано відповідні ілюстрації для наочності.

# 3 РОЗРОБКА ПРОГРАМНИХ МОДУЛІВ ТА РЕАЛІЗАЦІЇ ВЕБЗАСТОСУНКУ ДЛЯ РОЗПІЗНАВАННЯ ЗОБРАЖЕНЬ ЗА ДОПОМОГОЮ НЕЙРОНОЇ МЕРЕЖІ

## 3.1 Варіантний аналіз і обґрунтування вибору засобів для реалізації програмного забезпечення

Аби навчити глибинну мережу з «нуля», спершу слід зібрати великий масив розмічених даних, далі спроектувати архітектуру мережі, здатну засвоїти потрібні функції, та зрештою створити модель. Результати можуть вражати, проте цей підхід потребує величезної кількості тренувальних даних та тонкого налаштування шарів і ваг CNN.

Значно спростити процес навчання можна, використовуючи вже готові, попередньо навчені моделі. Більшість застосувань глибинного навчання використовують метод перенесення знань – процес, що передбачає доведення до потрібного стану вже перевіреної моделі. Така модель може містити необхідні знання, тому можна подати нові дані з невідомими класами та отримати бажану нейромережу. Також, доступні базові моделі, які пропонують лише архітектуру мережі без попереднього навчання. Вони вимагають додаткового налаштування, а відповідно, й більше даних для навчання, проте це може призвести до кращого результату. У будь-якому разі, попередня підготовка моделі сприяє спрощенню навчання та дозволяє досягти кращих показників.

Щоб запустити функцію розпізнавання об'єктів, залучаючи машинне навчання, слід почати з підібраної колекції зображень, попередньо підготовлених, та далі визначити ключові ознаки для кожного зображення. Розподіливши ці ознаки за окремими категоріями, модель машинного навчання використовує ці дані для аналізу та класифікації нових об'єктів. Створення точної моделі, яка здатна класифікувати та визначати розташування об'єктів, зазвичай потребує використання різних алгоритмів машинного навчання та методів виокремлення ознак, що пропонують широке розмаїття комбінацій. Машинне навчання в процесі розпізнавання об'єктів обирає оптимальне поєднання ознак

та класифікаторів для більш адаптивного навчання. У підсумку, це дає змогу досягти точних результатів, використовуючи мінімальну кількість необхідної інформації.

Якщо постає питання вибору між машинним та глибинним навчанням, оптимальний метод для розпізнавання об’єктів визначається специфікою програми та поставленої задачі. Машинне навчання може бути корисним інструментом у багатьох ситуаціях, особливо якщо є розуміння, які саме ознаки або особливості зображень використовуються для розрізнення класів об'єктів. Ключовими аспектами при виборі між машинним та глибинним навчанням є наявність потужного обчислювального ресурсу та значної кількості зображень для навчання. У випадку, коли особа або організація не має можливості забезпечити ці вимоги, машинне навчання виглядає як найкращий вибір для вирішення поставленого завдання. Технології глибинного навчання, як правило, демонструють кращі результати при роботі з більшими обсягами даних зображень, а потужний графічний процесор (GPU) сприяє зменшенню часу, необхідного для тренування моделі.

Необхідно зосередитись також на виборі середовища для розробки нейронних мереж.

Для створення, тренування та застосування нейронних мереж є у наявності відомі інтерфейси, як-от: TensorFlow, PyTorch, Keras і MXNet. Вони надають великий перелік функцій, оптимізацій та рівнів абстрагування, завдяки чому розробники можуть обрати найоптимальніший фреймворк для своїх задач[5], [8].

Навчання нейронних мереж – процес, що потребує чималих обчислювальних потужностей. Такі сервіси, як Google Colab та Amazon SageMaker, поряд із власними рішеннями з використанням GPU чи TPU, дають змогу тренувати нейронки як на локальних машинах, так і в хмарі [7].

Для показу функціонування нейронних мереж та дослідження їхньої продуктивності задіюють такі помічники, як TensorBoard, Visdom, Matplotlib та інші. Ці засоби добре знані та сприяють розробникам у кращому налаштуванні нейронних мереж задля досягнення бажаних результатів.

### 3.2 Аналіз апаратного забезпечення для програмного додатку

Апаратне забезпечення – це комп'ютер або електронна система, що виконує різноманітні завдання, потрібні для опрацювання даних, зберігання інформації, вводу та виводу даних, а також зв'язку між різними складовими системи. Апаратне забезпечення для обробки нейронних мереж надає необхідну продуктивність для вивчення складних моделей машинного навчання, зокрема глибоких нейронних мереж. Скажімо, графічні процесори та TPU істотно зменшують час навчання моделей, одночасно обробляючи великі обсяги інформації. Сучасні процесори та оперативна пам'ять забезпечують швидке опрацювання та збереження тимчасових даних, а швидкісні SSD-накопичувачі дають можливість швидко зберігати великі обсяги даних та отримувати до них доступ. Мережевий інтерфейс сприяє ефективній діяльності кластерної системи, яка застосовується для навчання моделей на великих обсягах даних.

У даній роботі для навчання та перевірки нейромережі було застосовано Google Colab, а для розміщення вебзастосунку використано локальний комп'ютер. Отже, апаратні вимоги можуть змінюватися залежно від конкретних поставлених задач.

Для підтримки вебзастосунків, побудованих на основі навчених нейронних моделей, обладнання гарантує ефективне опрацювання запитів користувачів. Сервер, оснащений достатньо потужним процесором та відповідним об'ємом оперативної пам'яті, забезпечує базову обчислювальну потужність, у той час як графічний процесор прискорює обробку даних моделями нейронних мереж.

Якщо розглядати розгортання вебдодатку з мінімальним залученням ресурсів, варто зупинитися виключно на центральному процесорі, уникаючи зайвих витрат на графічні. В даному екземплярі розгортання відбувалося на базі AMD Ryzen 7 9700X. Цей процесор оптимально підходить для реалізації базових операцій з даними та забезпечення взаємодії з вебзастосунками, що не потребують значних обчислювальних можливостей. AMD Ryzen 7 9700X вирізняється енергоефективністю, споживаючи незначну кількість енергії, що критично важливо з економічної та екологічної точок зору. Завдяки наявності

кеш-пам'яті, процесор здатний оперативно обробляти дані та виконувати задачі. Відтак, AMD Ryzen 7 9700X виступає вдалим рішенням для розгортання вебзастосунків на основі нейронних мереж, які не вимагають значних обчислювальних ресурсів (див. рисунок 3.1).

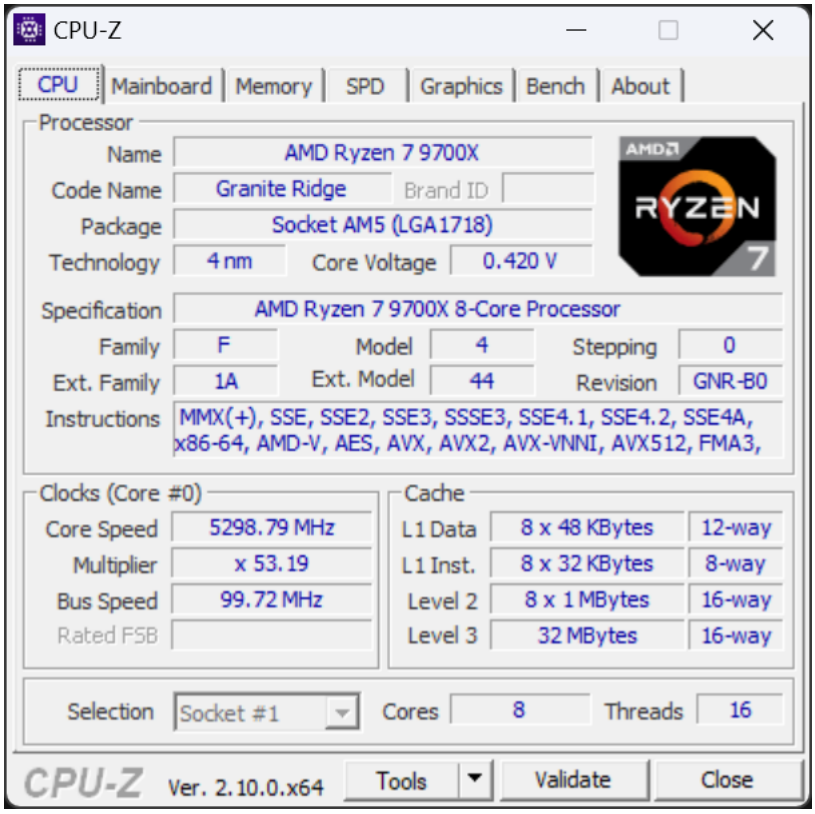


Рисунок 3.1 – Роботоспроможність процесора AMD Ryzen 7 9700X

У випадку, якщо вебзастосунок може залучати графічний процесор, це сприятиме пришвидшенню операцій, пов'язаних з нейронними мережами, що критично важливо в сучасних вебдодатках. В даній роботі було використано графічний процесор NVIDIA GeForce RTX 4070 SUPER. Висока обчислювальна потужність RTX 4070 SUPER зумовлює його оптимальність для обробки складних алгоритмів нейронних мереж. Цей графічний процесор підтримує технологію NVIDIA CUDA. Це дає змогу задіювати графічний процесор для обчислень, істотно прискорюючи функціонування нейронної мережі. Графічний процесор RTX 4070 SUPER оснащено декількома ядрами, що дозволяють здійснювати паралельні обчислення, забезпечуючи більш швидке та ефективне

виконання операцій. Отже, цей графічний процесор є доволі вдалим рішенням для забезпечення високої продуктивності та підтримки технологій, необхідних для роботи з нейронними мережами (див. рисунок 3.2).

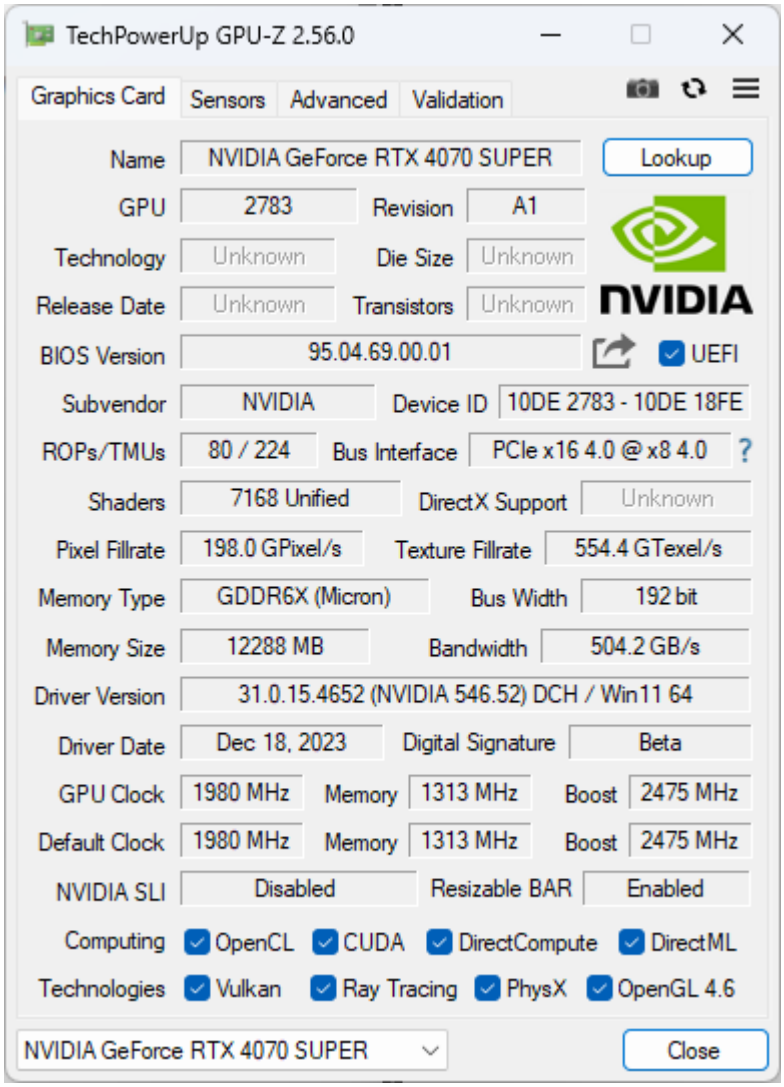


Рисунок 3.2 – Роботоспроможність графічного процесора NVIDIA GeForce RTX 4070 SUPER

Швидкісні твердотільні накопичувачі гарантують оперативний доступ до даних і моделей, а надійне мережеве устаткування відповідає за швидку та безперебійну передачу даних. Системи резервного копіювання, відновлення, охолодження і безперебійного живлення гарантують стабільність і безпеку всього апаратного забезпечення. Враховуючи, що розроблений додаток

сконструйовано з метою економії ресурсів, він цілком може бути розгорнутий на обладнанні з відносно скромними характеристиками.

Необхідною складовою частиною для цього вебдодатку є також належний обсяг оперативної пам'яті. Зважаючи на те, що додаток створено з акцентом на економію ресурсів, його функціонування має бути бездоганним при обсязі пам'яті в діапазоні 4-5 ГБ. Для розміщення цього вебзастосунку на локальному пристрої було використано дві планки пам'яті Kingston Fury Beast EXPO DDR5, кожна об'ємом 16 ГБ. Загальний обсяг ОЗП становить 32 ГБ, що дозволяє вебдодаткам опрацьовувати великі масиви даних та без проблем запускати великі навчені нейронні мережі. Надлишковий обсяг оперативної пам'яті сприяє прискоренню обробки даних, забезпечуючи більше простору для кешу та тимчасових даних, тим самим підвищуючи загальну продуктивність вебдодатків. Наявність достатньої ОЗП гарантує ефективну обробку обсягів даних, необхідних для функціонування нейронної мережі. Завдяки значному обсягу оперативної пам'яті, паралельні операції, зокрема обробка даних нейронної мережі, можуть виконуватися без затримок або зниження продуктивності.

Отже, застосування 2-ох планок по 16 ГБ оперативної пам'яті є ефективним рішенням для забезпечення високої продуктивності та ефективності вебдодатків, які базуються на навчених нейронних мережах.

Ще одним компонентом обладнання, що необхідний для запуску вебзастосунку, є система зберігання даних. Оскільки цій програмі не потрібно багато пам'яті, достатньо диску малого розміру для безперебійної роботи. На пристрої, де було розгорнуто вебдодаток, встановлено M.2 SSD Kingston на 1 ТБ. Твердотільні накопичувачі значно швидші при зчитуванні та записі даних, ніж звичайні жорсткі диски.

SSD-накопичувачі прискорюють запуск вебзастосунків і менш схильні до збоїв через фізичні пошкодження або зношення в порівнянні з жорсткими дисками, адже у сучасних SSD-дисках немає механічних частин. SSD-диски споживають менше електроенергії, дозволяючи заощаджувати ресурси та зменшуючи тепловиділення.

Відтак, твердотільний накопичувач Kingston може виявитися вигідним варіантом для запуску вебдодатків, заснованих на навчених нейронних мережах, через свою швидкість, надійність та економію енергії.

### 3.3 Аналіз програмних файлів застосунку

Тека з файлами вихідного вебдодатку, що містить всі використані файли та папки виглядає наступним чином (див. рисунок 3.3).

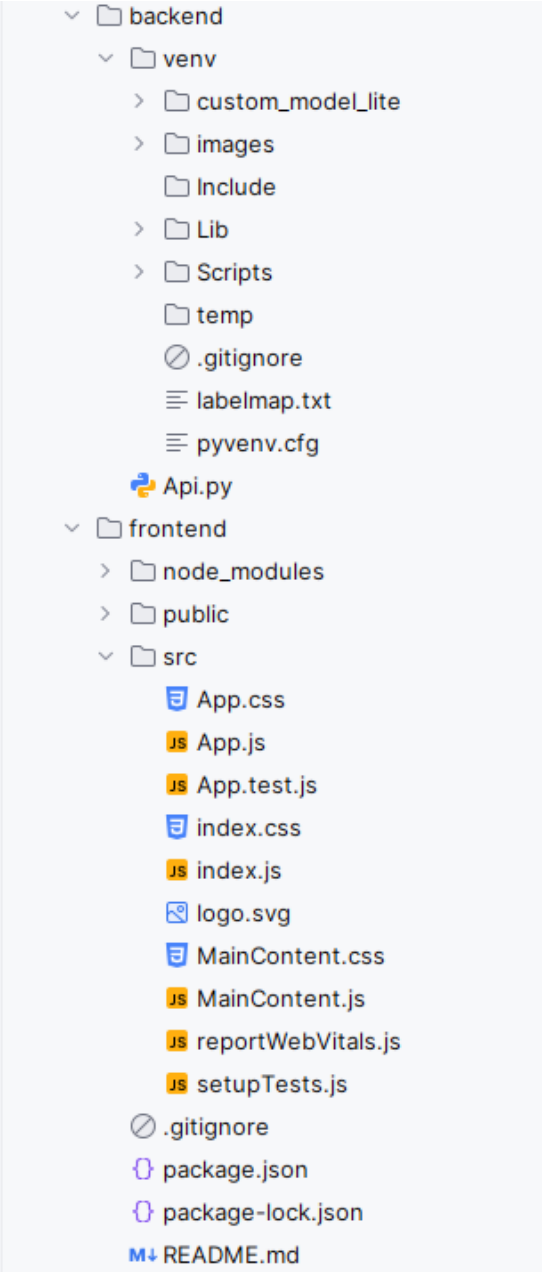


Рисунок 3.3 – Тека з необхідними файлами та папками для коректного функціонування розробленого вебдодатка

Загалом, файли розробленого вебдодатку можна згрупувати за трьома основними категоріями: файли, що відповідають за налаштування робочого середовища та підключення необхідних бібліотек; файли, які формують серверну частину; і, нарешті, файли, що забезпечують візуальне оформлення графічного інтерфейсу вебзастосунку.

Серверна частина додатку була розроблення з використанням Flask і зберігається в теці venv (див. рисунок 3.4).

Основні файли серверної частини розробленого вебзастосунку на основі модифікованої нейромережі в теці venv є наступними:

1) custom_model_lite – це каталог, який містить в собі модель нейронної мережі. В середині цього каталогу розміщуються файли, що репрезентують навчену нейромережу. Саме завдяки цій папці, розроблений вебдодаток отримує можливість використовувати натреновану нейронну модель;

2) images – це директорія, де розміщуються завантажені користувачами фотографії та відео, перш ніж їх обробляє розроблена нейронна мережа для розпізнавання зображення;

3) scripts – це каталог, що його згенерувала Flask-програма. Він містить скрипти, необхідні для керування застосунком та його оточенням. Ці файли використовуються для активації та деактивації віртуального середовища Python, що було налаштоване для конкретного проєкту Flask. Такі скрипти суттєво полегшують керування додатками Flask та роблять розробку, запуск і адміністрування більш комфортними;

4) temp – це каталог, де зберігаються оброблені нейронною мережею картинки та відеоматеріали. Тут містяться ці зображення й відеозаписи з відповідними мітками, що ідентифікують розпізнані медіа-елементи та їхню класифікацію. Це вихідні файли створеної вебпрограми, котрі надсилаються користувачеві;

5) labelmap.txt – це текстовий файл, який використовується разом з навченою нейронною моделлю, щоб відображати класи, з якими модель вміє

працювати. Кожен рядок у цьому файлі описує окремий клас об'єктів. Ці мітки є ключем для розпізнавання об'єктів, які здатна розпізнавати модель. Завдяки цьому користувач отримує докладніший вивід після того, як нейронна мережа обробить потрібний файл;

6) api.py – файл, що втілює базову логіку бекенду везастосунку. Цей файл забезпечує з’єднання з найбільш навченим нейронним модулем для розпізнавання об’єктів. В api.py реалізовано механізм обробки HTTP-запитів, що надходять від користувачів. Він генерує відповіді на HTTP-запити. Цей файл є ключовим компонентом вебдодатка, оскільки контролює обробку запитів та комунікацію з клієнтом. Крім того, він проводить валідацію вхідних даних, перевіряючи їх на наявність помилок та неточностей.

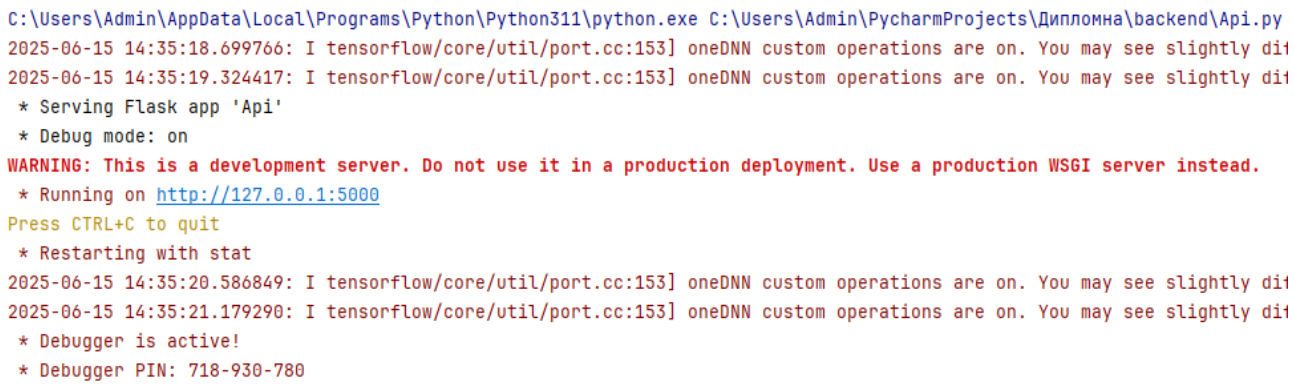


Рисунок 3.4 – Робота системи backend

Основні файли налаштувань середовища виконання і підключених бібліотек є наступними:

1) node_modules – це каталог у JavaScript проєкті, де зберігаються всі залежності (бібліотеки, модулі), інстальовані за допомогою Node.js, менеджера пакетів. Коли новий пакет додається в проєкт за допомогою npm install чи yarn add, він здебільшого завантажується з мережі та зберігається в каталозі node_modules.

Кожен пакет зазвичай міститься у власному підкаталозі, і в кожному підкаталозі можуть бути власні залежності. Це сприяє керуванню проєктом, оскільки залежності та їхні версії розділені й ізольовані одна від одної.

Під час запуску проєкту Node.js автоматично шукатиме всі залежності в каталозі node_modules та використовуватиме їх у проєкті. Це робить можливим використання сторонніх бібліотек та модулів у проєкті без потреби додавати код у репозиторій проєкту [22];

2) package.json – цей файл є вмістилищем метаінформації про проєкт. У ньому зберігаються відомості, а саме назва, версія, роз'яснення, ім'я автора, скрипти та необхідні компоненти. Він також містить перелік усіх залежностей проєкту, які легко можна інсталювати за допомогою спеціального менеджера пакетів. Файл package.json виконує функцію налаштовувача скриптів, що потрібні для автоматичного виконання різних операцій у рамках проєкту, а саме збирання коду, перевірка та запуск сервера;

3) package-lock.json – автоматично згенерований артефакт після виконання npm install чи yarn install. Він містить докладну інформацію про кожну встановлену бібліотеку, включаючи залежності та конкретні версії. Завдяки package-lock.json всі розробники, які працюють над додатком, гарантовано використовують ідентичні версії пакетів, що сприяє стабільності та передбачуваності робочого середовища;

4) backend\venv\pyvenv.cfg – це конфігураційний файл, призначений для віртуального оточення Python, створеного за допомогою утиліти venv. У цьому файлі зберігаються налаштування, які стосуються конкретно віртуального середовища, зокрема вказується шлях до версії Python, задіяної при його створенні, а також шлях до стандартної бібліотеки Python. Функціональність цього файлу полягає в підтримці управління та налаштування віртуального середовища Python, а також в наданні розуміння відповідних параметрів, застосованих у конкретному оточенні;

5) backend\venv\Lib – це каталог, який містить ключові стандартні бібліотеки Python, зокрема: os, sys та math. Ці бібліотеки, які поставляються

разом із Python, доступні для використання в будь-якому програмному середовищі Python. У разі встановлення зовнішньої бібліотеки за допомогою такого інструменту управління пакунками Python, як pip, її буде збережено в цій папці. Зазвичай каталог Lib є критичним компонентом середовища розробки Python, де зосереджені бібліотеки та модулі, необхідні програмі для виконання різноманітних завдань;

6) .gitignore – це текстовий файл, у якому перелічено файли та каталоги, які Git має ігнорувати під час команди git add. Цей файл є корисним, якщо потрібно працювати з системою контролю версій Git. Створюється автоматично за допомогою Pycharm;

7) README.md – цей файл вміщує загальні відомості про проект, зокрема назву проекту, його короткий опис, інструкції для встановлення й запуску, а також загальний огляд функціональності.

Оскільки React був використаний для розробки вебдодатку, структура файлів графічного інтерфейсу підпорядковується специфічним правилам. React спирається на розроблені компоненти для побудови вебсайту, формуючи дерево компонентів, що візуалізує структуру інтерфейсу. Кожен компонент React є базовим елементом, здатним відображати певні частини інтерфейсу, та може включати в себе HTML, CSS і JavaScript для функціональності та візуального налаштування (див. рисунок 3.5).

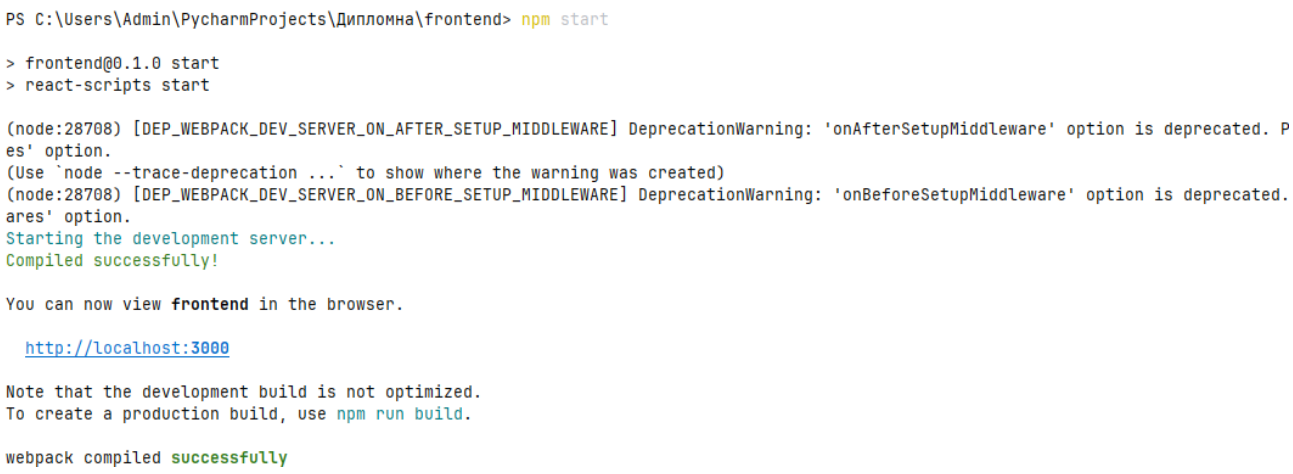


Рисунок 3.5 – Робота системи frontend

Основні файли графічної частини розробленого вебдодатку, який функціонує на основі модифікованої нейронної мережі, знаходяться в папці src\ і включають:

1) Header.js – файл, що містить компонент, що відповідає за відображення заголовку розробленого вебдодатка;

2) Header.css – файл стилів для шапки сайту, тобто для компоненту Header.js. Цей файл містить додаткове оформлення, наприклад, налаштування шрифтів та параметри меж відображення елементів шапки;

3) MainContent.js – це файл, у якому розташовано компонент, що керує показом головного вмісту сайту. У цьому компоненті описується візуальний вигляд основної області вебдодатку. Окрім того, він визначає взаємодію з відвідувачем, тобто оновлення компонента відповідно до його дій. Саме в цьому файлі реалізується процес отримання даних від користувача, таких як файли, та створення запитів до сервера для обробки завантаженої інформації;

4) MainContent.css – це файл стилів, призначений для файлу MainContent.js. В ньому зібрані додаткові стильові правила, які стосуються налаштування шрифтів, колірного оформлення, а також зовнішнього вигляду конкретних елементів. Завдяки цьому файлу реалізуються особливості дизайну інтерфейсу, які виходять за рамки можливостей фреймворку Bootstrap [23];

5) index.js – файл, який визначає початкову точку, де програма React з'єднується з кореневим елементом HTML в DOM. Це дозволяє React відтворювати компоненти головної програми на вебсторінці;

6) index.css – це файл у React-проєкті, який слугує для глобального стилю, що впливає на всю програму. В цьому файлі зазвичай знаходяться стилі, призначені для використання в усіх компонентах вашого застосунку. Будь-який стиль, прописаний тут, застосовується до всіх елементів, за умови, що він не конфліктує з більш специфічним стилем, визначеним у компонентах з вищим пріоритетом;

7) reportWebVitals.js – це файл у React-проектах, який застосовується для вимірювання та звітування про показники вебзастосунків. Ці дані дозволяють

оцінити, як функціонує програма у реальних умовах та виявити потенційні «вузькі місця» продуктивності. Це сприяє оптимізації додатку та загальному поліпшенню взаємодії з користувачем.

8) Додатково до файлів, розташованих в директорії src\, структуру і візуальний вигляд додатку визначають файли, що містяться у папці public\;

public – це сховище статичних файлів, що лишаються недоторканими Webpack і стають доступними як є після фінальної збірки проекту. У ній зберігаються такі файли, як HTML-шаблон або іконки, котрі використовуються на вашому вебсайті. Файли у цій папці дозволяють вам налаштувати базову структуру HTML, задати параметри PWA та контролювати індексацію пошуковими системами [24].

### 3.4 Розробка модулів вебзастосунку

Алгоритм вебзастосунку, побудованого на основі нейронної моделі розпізнавання об'єктів на вхідних зображеннях, дає змогу завантажувати знімки, отримувати зображення та зберігати їх на сервері, попередньо обробляти зображення, завантажувати навчені нейронні моделі, ідентифікувати об'єкти на зображеннях, формувати результати розпізнавання після обробки і генерувати відповіді.

Користувач відкриває вебзастосунок і завантажує зображення за допомогою форми на вебсторінці. Зображення відправляється на сервер через POST запит.

Сервер отримує зображення та тимчасово розміщує його у файловій системі або пам'яті. Далі сервер здійснює базову перевірку зображення, тобто перевіряє тип файлу, який було надіслано.

Після огляду зображення, сервер завантажує з локального сховища наперед навчену нейронну модель для розпізнавання об'єктів. Після успішного завантаження моделі, сервер витягує необхідну інформацію про вхідні дані цієї моделі (див. рисунок 3.1).

```
interpreter = Interpreter(model_path=modelpath)
interpreter.allocate_tensors()

input_details = interpreter.get_input_details()
output_details = interpreter.get_output_details()
height = input_details[0]['shape'][1]
width = input_details[0]['shape'][2]
float_input = (input_details[0]['dtype'] == np.float32)
```

Рисунок 3.1 – Завантаження натренованої моделі з локального сховища

Після відповідних перевірок і завантаження моделі, зображення завантажується в пам'ять. Далі відбувається зміна розміру зображення і перетворення у формат, який повинна отримати натренована нейронна модель на вхід (див. рисунок 3.2).

```
image = cv2.imread(image_path)
if image is None:
    continue

image_rgb = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
imH, imW, _ = image.shape
image_resized = cv2.resize(image_rgb, dsize:(width, height))
input_data = np.expand_dims(image_resized, axis=0)
```

Рисунок 3.2 – Опрацювання зображення для коректної подачі на вхід моделі

Наступним кроком є подача зображення на вхід нейронної мережі. Нейронна мережа обробляє його, ідентифікує об'єкти на ньому, класифікує та повертає клас, координати та ймовірності виявлених об'єктів (див. рисунок 3.3).

Результати розпізнавання зазнають обробки для представлення у зручному для користувача вигляді. Відтак, всі об'єкти, чия вірогідність менше 0,6, будуть відфільтровані.

```
interpreter.set_tensor(input_details[0]['index'], input_data)
interpreter.invoke()

try:
    boxes = interpreter.get_tensor(output_details[0]['index'])[0]
    classes = interpreter.get_tensor(output_details[1]['index'])[0]
    scores = interpreter.get_tensor(output_details[2]['index'])[0]
except Exception as e:
    print("Error getting outputs:", e)
    continue
```

Рисунок 3.3 – Обробка зображення нейронною моделлю та отримання вихідних даних

Надалі на початкове зображення наноситься відповідне обрамлення та позначки, котрі ідентифікують знайдені об'єкти. Зважаючи на те, що нейронна модель видала вихідні дані для зображення, зменшеного у розмірах, необхідно розрахувати координати для первинного розміру та застосувати їх безпосередньо на зображення (див. рисунок 3.4).

```
ymin = int(max(1, box[0] * imH))
xmin = int(max(1, box[1] * imW))
ymax = int(min(imH, box[2] * imH))
xmax = int(min(imW, box[3] * imW))

cv2.rectangle(image, (xmin, ymin), (xmax, ymax), (10, 255, 0), 2)
object_name = labels[int(classes[i])]
label = f'{object_name}: {int(score * 100)}%'
labelSize, baseLine = cv2.getTextSize(label, cv2.FONT_HERSHEY_SIMPLEX, fontScale: 0.7, thickness: 2)
label_ymin = max(ymin, labelSize[1] + 10)
cv2.rectangle(image, (xmin, label_ymin - labelSize[1] - 10),
              (xmin + labelSize[0], label_ymin + baseLine - 10), (255, 255, 255), cv2.FILLED)
cv2.putText(image, label, org: (xmin, label_ymin - 7),
            cv2.FONT_HERSHEY_SIMPLEX, fontScale: 0.7, color: (0, 0, 0), thickness: 2)
```

Рисунок 3.4 – Накладання відповідної рамки і міток на оригінальне зображення

Після того, як оброблене зображення з'явиться на сервері, його необхідно відправити користувачеві. Для цього створюється URL-адреса, яка вказує на картинку. Отже, у відповіді на запит користувача надсилається посилання на

згенероване зображення і відповідний код, який засвідчує успішне завершення обробки, тобто без жодних проблем.

Далі, вже на боці клієнта, буде видно вихідне зображення разом з виділеними об'єктами та мітками (див. рисунок 3.5).

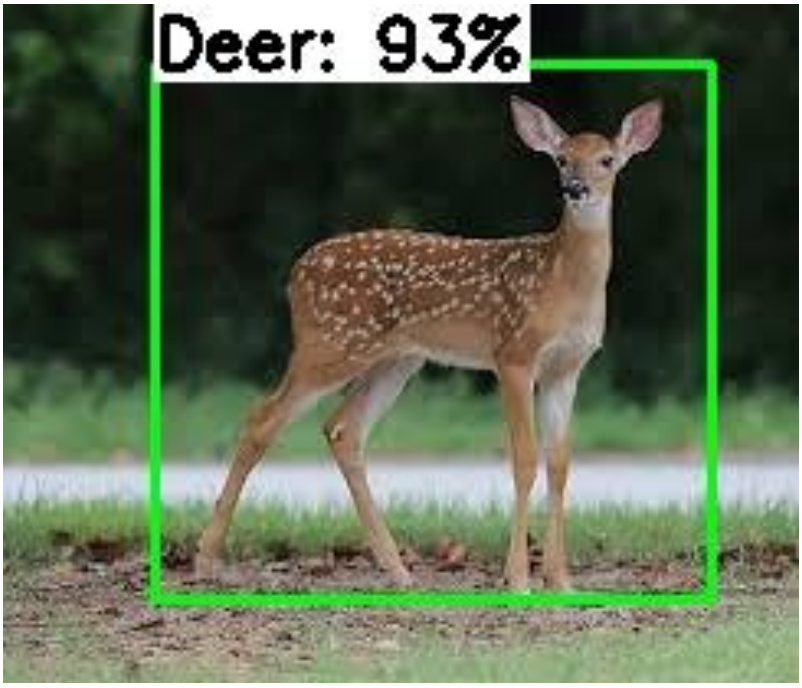


Рисунок 3.5 – Приклад обробленого зображення нейронною мережею

Отже, створений вебдодаток дає змогу автоматично та оперативно виявляти об'єкти на зображеннях. Таке рішення здатне розширити автоматизацію в різних сферах: від охоронних систем до медичної діагностики, додатково покращуючи ефективність і точність виконання поставлених задач.

Алгоритм роботи розробленого вебдодатку з відеофайлом, що надається, поділяється на декілька ключових етапів.

Користувач завантажує відеофайл у вебдодаток через зручний інтерфейс. Потім відеофайл передається на сервер, де він обробляється кадр за кадром. Сервер використовує бібліотеку OpenCV для розділення відео на окремі кадри (див. рисунок 3.6).

```
def extractImages(filename):  1 usage
    global heightVid, widthVid, framesCount
    parent_dir = PATH_TO_IMAGES
    pathToVideo = os.path.join(parent_dir, filename)
    directory = filename.rsplit('.', 1)[0]
    pathOut = os.path.join(parent_dir, directory)

    if not os.path.exists(pathOut):
        os.mkdir(pathOut)

    count = 0
    vidcap = cv2.VideoCapture(pathToVideo)
    heightVid = vidcap.get(cv2.CAP_PROP_FRAME_HEIGHT)
    widthVid = vidcap.get(cv2.CAP_PROP_FRAME_WIDTH)

    while True:
        success, image = vidcap.read()
        if not success:
            break
        cv2.imwrite(os.path.join(pathOut, f"frame{count}.jpg"), image)
        count += 1
```

Рисунок 3.6 – Розбивання відеофайлу на кадри

Кожен окремий кадр трансформується в RGB-формат, де його розмір адаптується до потреб моделі. Після цього формується тензор, що відповідає вимогам для подальшої обробки нейронною мережею.

Далі підготовлений кадр надсилається на вхід нейронної моделі. Остання виконує розпізнавання та класифікацію об'єктів, що містяться на зображенні. На виході отримуємо дані про координати та категорію кожного виявленого об'єкта. Інформація про ці об'єкти інтегрується у кожен відтворений кадр, зокрема, навколо виявленого об'єкта малюється рамка та вказується назва відповідної категорії.

Після обробки кожного кадру результат зберігається і збирається у відеопотік (див. рисунок 3.7).

```
fourcc = cv2.VideoWriter_fourcc(*'mp4v')
downloadName = filename.rsplit( sep: '.', maxsplit: 1)[0] + '_detected.mp4'
sizeVid = (int(widthVid), int(heightVid))
downloadLink = os.path.join(temppath, downloadName)

videoDownload = cv2.VideoWriter(downloadLink, fourcc, 15, sizeVid)
for j in range(framesCount):
    frame_path = os.path.join(resVid, f"frame{j}.jpg")
    if os.path.exists(frame_path):
        tempFrame = cv2.imread(frame_path)
        videoDownload.write(tempFrame)

videoDownload.release()
```

Рисунок 3.7 – Формування цілісного відео з окремих кадрів

По закінченню обробки кожного кадру відео складається у фінальний відеофайл. У ньому зберігається вся отримана інформація про розпізнані елементи. Цей файл розміщується на сервері, де стає доступним для завантаження або перегляду користувачем через вебдодаток.

Отже, вебдодаток, що базується на модернізованій нейронній моделі розпізнавання об'єктів, забезпечує повний цикл обробки відео. Він охоплює процес від завантаження відео користувачем до отримання результатів щодо розпізнаних об'єктів. Це гарантує швидкий та ефективний аналіз відеоконтенту для різноманітних потреб, таких як моніторинг, виявлення нетиповостей та автоматизована інвентаризація об'єктів. Використання нейронних мереж сприяє високій точності та швидкості обробки. Це є ключовим аспектом у багатьох сучасних промислових секторах.

### 3.5 Розробка модулів нейронної мережі

Алгоритм навчання нейронної мережі стартує з підготовки відповідного масиву даних для навчання. Для забезпечення коректного функціонування нейронної моделі необхідно сформувати три окремі набори даних з позначеннями позицій та класів об’єктів: навчальний, валідаційний та тестовий.

Кожен набір має своє чітке призначення під час тренування моделі. Також потрібно сформувати перелік, що міститиме всі можливі класи, які модель зможе розпізнавати.

Після того, як розробник сформував тренувальні набори, можна переходити до налаштування моделі та середовища розробки. Слід обрати тип та архітектуру нейронної мережі. Вибір цих параметрів визначається цільовим призначенням нейромережі та технічними характеристиками середовища виконання. Окрім того, потрібно інсталювати всі необхідні бібліотеки та залежності для забезпечення комфортного тренування та тестування моделі.

Після завершення налаштувань і підготовки всіх необхідних даних для тренування, слід перетворити всю інформацію, зокрема перелік класів та датасети, у формати файлів, придатних для навчання нейронної мережі.

Коли підготовку та налаштування моделі завершено, можна розпочинати тренування. Спочатку необхідно визначити кількість кроків навчання моделі. Кожен крок передбачає повний перебір усього тренувального набору даних. Залежно від підходу до тренування, результати навчання можуть відображатися після кожного кроку або через визначену їх кількість (див. рисунок 3.8).

```
INFO:tensorflow:Step 100 per-step time 0.855s
I0325 01:07:34.598938 134465021710336 model_lib_v2.py:705] Step 100 per-step time 0.855s
INFO:tensorflow:{'Loss/classification_loss': 0.710647,
 'Loss/localization_loss': 0.43443936,
 'Loss/regularization_loss': 0.15347166,
 'Loss/total_loss': 1.298558,
 'learning_rate': 0.0319994}
I0325 01:07:34.599390 134465021710336 model_lib_v2.py:708] {'Loss/classification_loss': 0.710647,
 'Loss/localization_loss': 0.43443936,
 'Loss/regularization_loss': 0.15347166,
 'Loss/total_loss': 1.298558,
 'learning_rate': 0.0319994}
```

Рисунок 3.8 – Приклад інформаційних повідомлень під час навчання

Після завершення заданої кількісті кроків або дочасного зупинення навчання, модель конвертується в загальний формат і зберігається на локальному або хмарному сховку.

### 3.6 Висновки

У цьому розділі проаналізовано функціонування ключових операцій та засад обробки інформації в розробленому вебдодатку.

Тут зосереджено увагу на даних, з якими працює вебдодаток. Представлено докладний опис та пояснення процесу обробки даних різних типів.

Проаналізовано ключові етапи обробки модифікованою нейронною мережею для розпізнавання медіа-об'єктів на користувацьких зображеннях та відео. Подано структуру коду, що забезпечує обробку різних аспектів для досягнення необхідного результату.

Завдяки ретельно представленим алгоритмам, легко зрозуміти загальну логіку вебдодатку. Також можна відстежити взаємодію між сервером та клієнтською частиною розробленого інструменту.

Крім того, детально розглянуто загальний алгоритм навчання нейронної моделі. Описано процес формування необхідних файлів та виконання налаштувань моделі. Результатом є отримана нейронна модель розпізнавання медіа-об'єктів, що використовується для аналізу зображень та відео.

# 4 ТЕСТУВАННЯ ВЕБЗАСТОСУНКУ

## 4.1 Тестування системи

Процес тестування вебзастосунку, що використовує навчену нейронну модель для розпізнавання медіаоб'єктів, було здійснено на етапі навчання, через створення тренувального та валідаційного наборів даних. Тренувальний набір даних застосовувався для навчання нейронної моделі, а валідаційний – для оцінки її ефективності в процесі навчання.

Перший етап тестування реалізовано під час навчання нейронної моделі. Завдяки використанню Google Colab для процесу навчання, було забезпечено зручний доступ до TensorBoard, що дозволило відслідковувати якість навчання в реальному часі.

TensorBoard – це засіб візуалізації та моніторингу для TensorFlow, який супроводжує аналіз і налаштування процесу навчання нейронної мережі. Виступаючи як випробувальний майданчик для розроблених моделей, TensorBoard демонструє низку корисних функцій. Цей інструмент забезпечує можливість відстежувати ключові показники роботи моделі, зокрема точність, величину втрат та швидкість навчання, що особливо важливо в процесі тренування. Додатково, TensorBoard надає функціонал візуалізації архітектури моделі, включаючи графіки обчислень, що полегшує розробнику розуміння структури та взаємодії між окремими шарами і процесами.

Також, TensorBoard відкриває можливість відстежувати еволюцію параметрів моделі в часі та використовувати вбудовані інструменти для аналізу й візуалізації даних. Наприклад, доступна побудова графіків для порівняння різних моделей або застосування метричних діаграм для моніторингу збіжності навчання. Загалом, TensorBoard постає як потужний інструмент для аналізу та оптимізації процесу навчання нейронних мереж, сприяючи покращенню результатів тестування розроблених моделей [10].

Loss/classification_loss – це метрика, яка використовується для вимірювання того, наскільки точно модель класифікує об'єкти на зображенні. У

контексті нейронних мереж, які розпізнають об'єкти, цей показник показує, наскільки ефективно модель визначає категорію або клас, до якого належить кожен об'єкт на зображенні (див. рисунок 4.1).

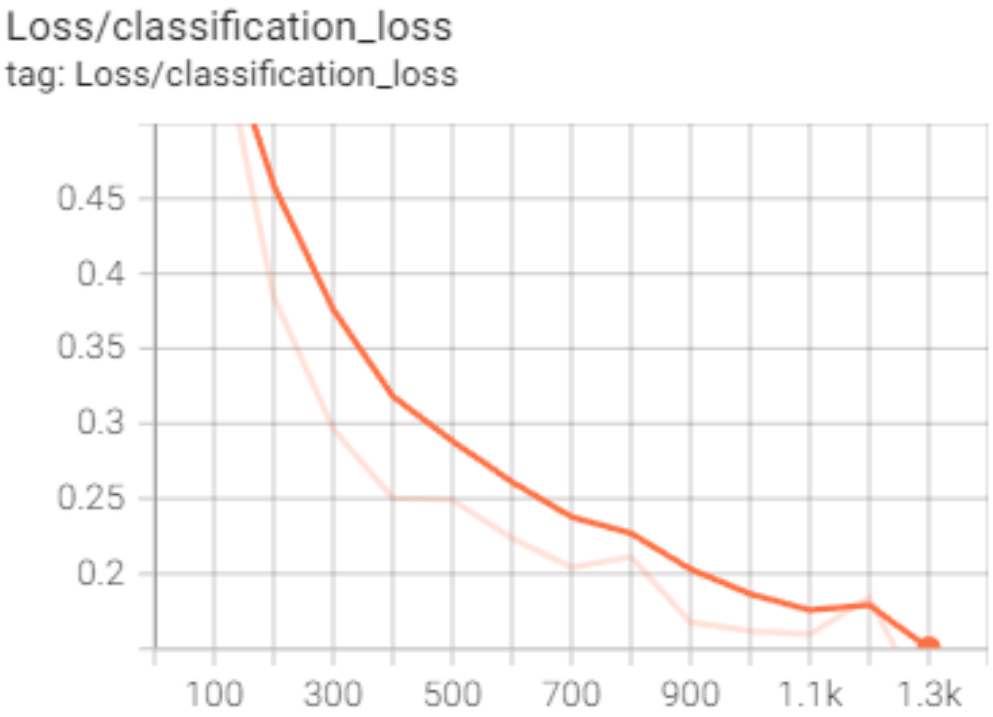


Рисунок 4.1 – Точність класифікації об'єктів розробленої нейронної моделі

Розмір цієї втрати має надзвичайне значення для розуміння успішності навчання моделі. Скажімо, низький показник втрат під час класифікації свідчить про чудову здатність моделі розпізнавати об'єкти на зображенні, в той час як високий – сигналізує про труднощі з класифікацією медійних об’єктів, і, можливо, необхідність оптимізації для поліпшення результатів.

Наступний показник втрат демонструє точність передбачення розташування об’єктів. Іншими словами, він вказує на те, наскільки точно навчена нейронна модель може окреслити межі кожного об’єкта на зображенні (див. рисунок 4.2).


Рисунок 4.2 – Точність прогнозування положення об'єктів розробленої нейронної моделі

Loss/localization_loss – це показник, що використовується для оцінювання чіткості визначення розташування об'єктів на малюнку, або якості їхнього знаходження за допомогою моделі об'єктів на картинці. Стосовно нейронних мереж для виявлення об'єктів, цей індикатор вказує, з якою точністю модель визначає координати областей для кожного об'єкта. Значення втрат вимірює розбіжність між передбаченими координатами об'єкта та його справжніми координатами.

Для локалізації традиційно застосовуються різноманітні функції втрат, як-от середньоквадратична помилка або втрати L1. Величина цієї втрати є визначальною для оцінювання точності локалізації об’єктів, отриманої за моделлю. Незначне значення втрат локалізації свідчить про ефективне визначення моделлю положення об'єктів на зображенні, а значне – може вказувати на потребу в покращенні методу локалізації або налаштування параметрів навчання моделі.

Наступним показником буде показник регуляризації (див. рисунок 4.3).

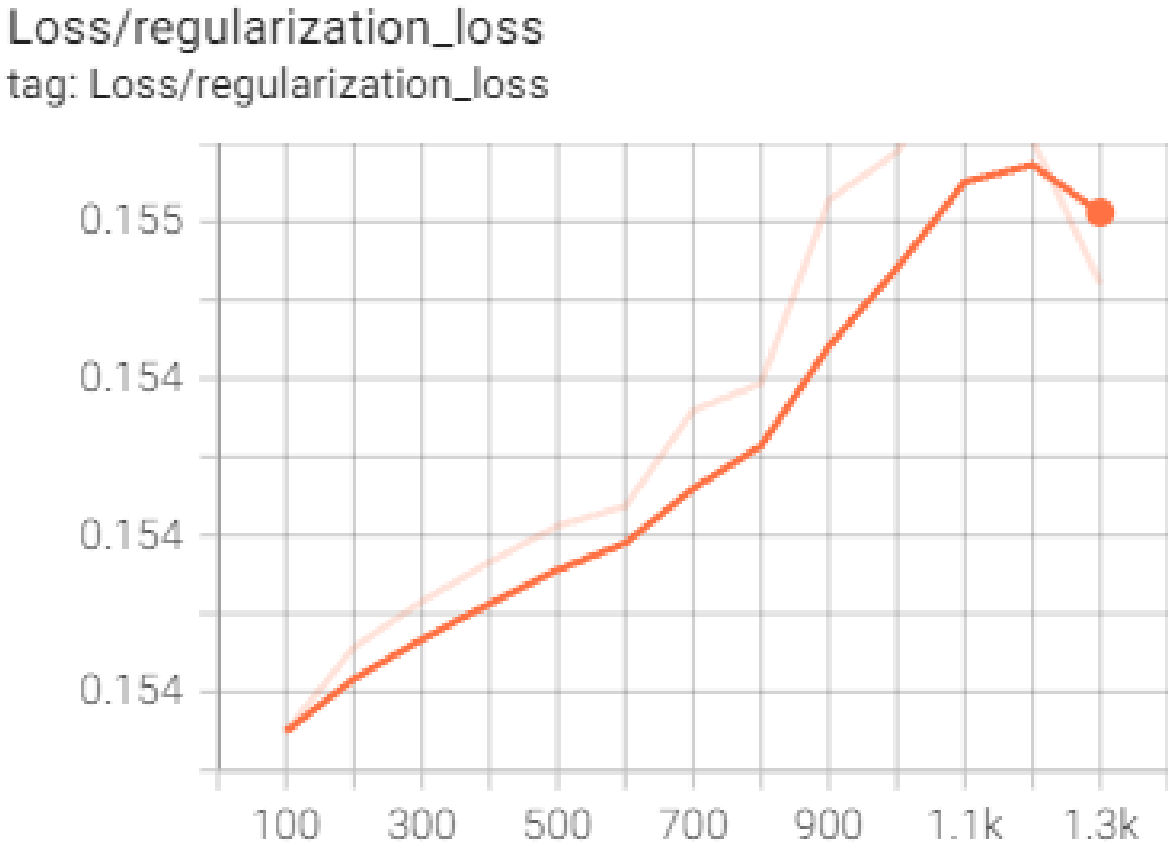


Рисунок 4.3 – Контроль перенавчання нейронної моделі

Втрата/регуляризація_втрати – це складова функцій втрат у нейронних мережах, яка слугує для контролю перенавчання навченої нейронної моделі. Цей компонент відіграє ключову роль у недопущенні перенавчання та сприяє покращенню здатності моделі до узагальнення на нові дані.

Існує безліч видів регуляризації, але загальна мета полягає в тому, щоб обмежити складність моделі, уникнути надмірних значень ваг параметрів та забезпечити більш стабільну роботу моделі.

Останній показник втрат вказує на загальні втрати при роботі з зображеннями (див. рисунок 4.4).

Загальні втрати виступають визначальним індикатором, що застосовується для оцінювання поступу моделі під час процесу навчання. Зменшення загальних втрат впродовж навчання моделі свідчить про наближення моделі до оптимального рішення та про належну класифікацію й локалізацію об'єктів на вхідному зображенні.

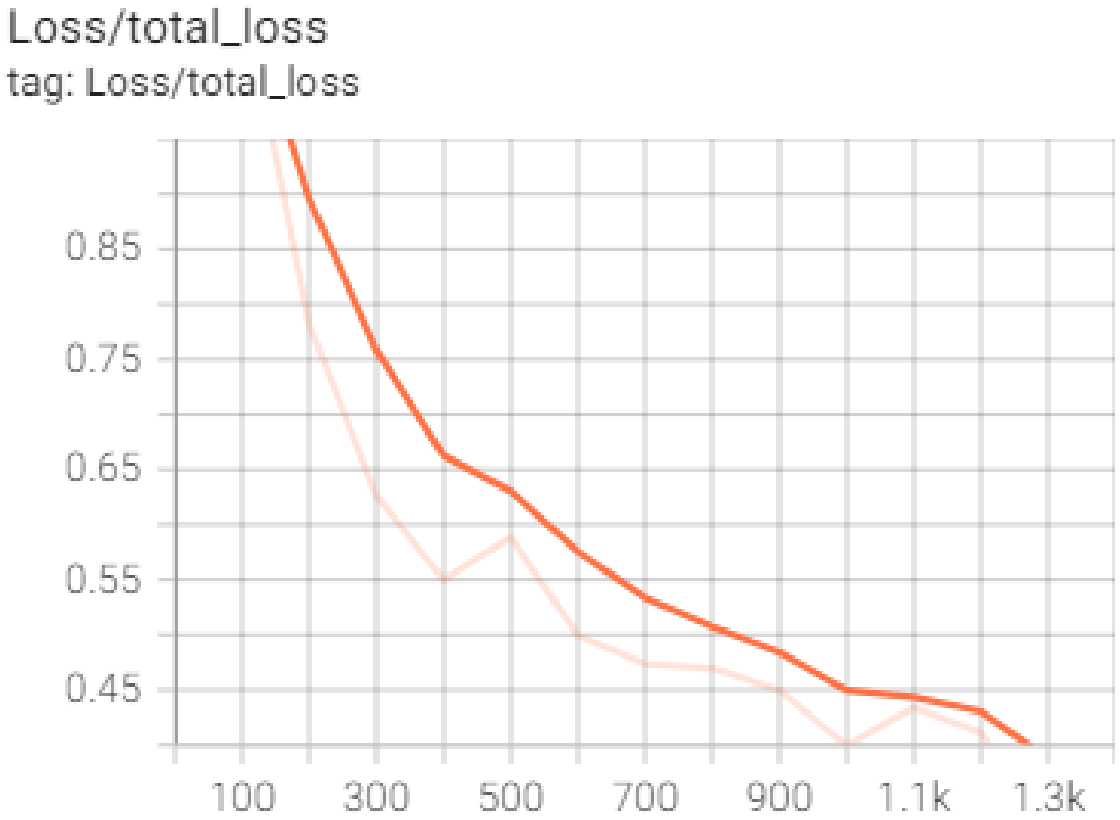


Рисунок 4.4 – Загальні втрати розробленої нейронної моделі

Loss/total_loss – це сукупний підсумок всіх складових втрат, що беруться до уваги під час тренування нейромережі. Це показник, який вміщує різноманітні чинники, зокрема втрати класифікації, локалізації та регуляризації, що використовуються для оцінки успішності моделі в процесі навчання.

Головна мета тренування нейронної мережі – мінімізація загальних втрат, щоб модель могла точно та продуктивно визначати об'єкти на зображенні.

Окрім того, під час тестування створеної моделі, можливо, будуть помітні відмінності від прогнозованих результатів.

На рисунку зображено приклад невірного розпізнавання координат об’єктів, тому що в тренувальному наборі даних було недостатньо інформації, тому модель не змогла прослідкувати схожість (див. рисунок 4.5).

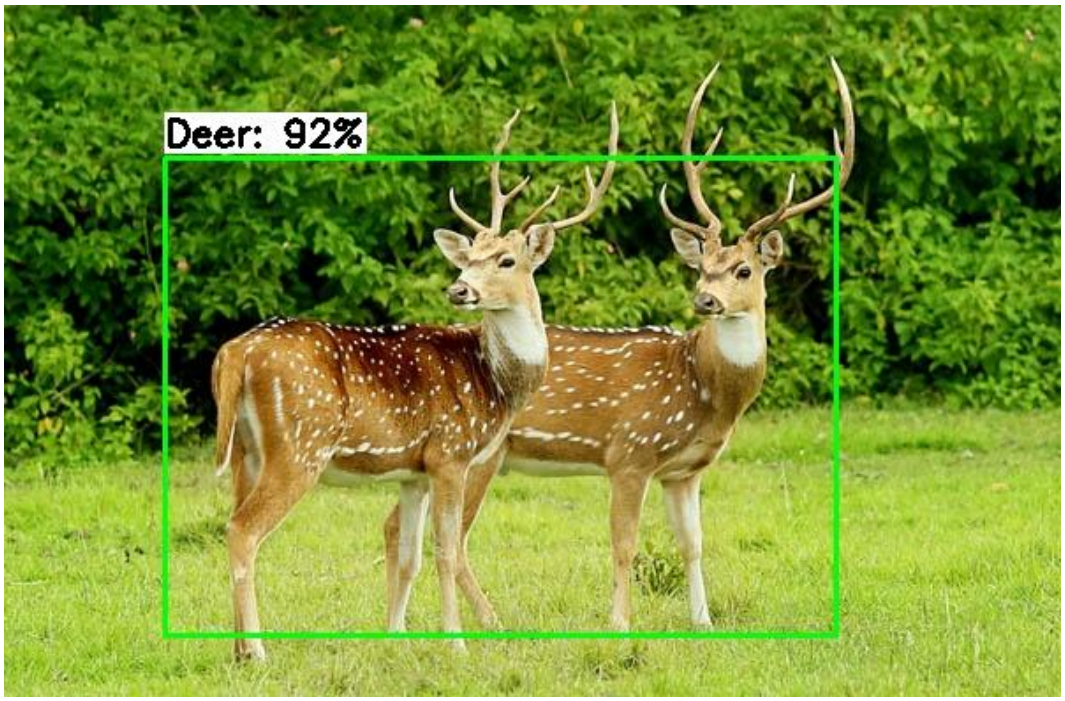


Рисунок 4.5 – Приклад невірного знаходження положення об’єктів розробленою нейронною мережею

Під час обробки зображень нейронна мережа може помилитися. Причиною цього є недостатній обсяг даних, використаних для навчання. Цей недолік може призвести до того, що розроблена модель даватиме неправильні відповіді, неправильно класифікуючи об'єкти або визначаючи їхнє положення, якщо вхідне зображення значно відрізняється від тренувального набору. Щоб покращити результати розпізнавання об'єктів у такому випадку, необхідно оновити наявний набір даних та збільшити обсяг даних для тренування. Водночас, тренування може тривати тривалий час, навіть при використанні графічного процесора.

### 4.2 Розробка інструкції користувача

Інструкція користувача передбачає визначення технічних вимог для запуску програмного продукту. Деталі щодо мінімальної та рекомендованої конфігурації розміщено в таблиці 4.1.

Вебзастосунок дозволяє користувачам завантажувати зображення або відео, після чого нейронна мережа автоматично аналізує вміст і визначає, що

зображено на медіа-об’єкті (люди, тварини, предмети), сцени (наприклад, парк чи офіс), дії (рух, жести), текст (OCR) або емоції на обличчях.

Таблиця 4.1 – Вимоги до апаратного забезпечення

| Вимоги до конфігурування | Рекомендовано | Мінімально |
|---|---|---|
| Процесор | AMD Ryzen 7 9700X | Intel Core i7 |
| Оперативна пам'ять | 32ГБ RAM | 16 ГБ RAM |
| Вільний простір на диску | 15 ГБ | 4 ГБ |
| Відеокарта | NVIDIA GeForce RTX 4070 SUPER або аналогічна | NVIDIA GeForce GTX 1660 Ti |
| Операційна система | Windows 10 або macOS Big Sur | Windows 7 або macOS Mojave |
| Монітор | Роздільна здатність не менше 1920x1080 пікселів, діагональ 21" або більше | Роздільна здатність не менше 1280x720 пікселів, діагональ 15" або більше |

### 4.3 Висновки

У цьому розділі головний акцент було зроблено на тестуванні нейронної мережі, яке було здійснено.

Тестування під час процесу навчання дає можливість стежити за ходом тренування нейромережі у реальному часі. Окрім цього, воно дозволяє зберігати та аналізувати отримані дані стосовно втрат нейромережі.

Розглянуто різні види втрат, які можуть виникати у нейронних мережах. Це важливо для виявлення проблем та подальшої оптимізації нейромережі, щоб досягти кращих показників при розпізнаванні медіа-об'єктів.

# ВИСНОВКИ

У результаті бакалаврської кваліфікаційної роботи було розроблено вебзастосунок для розпізнавання зображень за допомогою нейронної мережі. Нейронна мережа, створена для вебзастосунку, є досить оперативною при обробці зображень, що дозволяє їй функціонувати навіть на малопотужних пристроях. Це стає можливим завдяки використанню сучасної архітектури нейромережі в її основі. Для розробки було використано середовище розробки PyCharm. Бакалаврську кваліфікаційну роботу реалізовано згідно методичних вказівок [25].

Вебзастосунок створено для покращення вражень користувачів, забезпечуючи швидку та легку взаємодію з розробленою нейронною моделлю. Це відкриває широкі можливості як для бізнесу, так і для індивідуального використання. Інтерфейс вебзастосунку зручний та інтуїтивно зрозумілий, що значно скорочує час на адаптацію для нових користувачів, дозволяючи їм максимально ефективно використовувати всі його функції. Таким чином, цей вебінструмент є потужним засобом для автоматизації аналізу візуальних даних, пропонуючи швидке та точне розпізнавання зображень.

Під час варіантного аналізу мов програмування було обрано мову програмування Python для розробки системи.

У першому розділі було закладено теоретичний фундамент проєкту, де проведено аналіз проблем та існуючих технологій у сфері розробки вебзастосунків на основі нейронних мереж для розпізнавання зображень. Цей детальний огляд дозволив чітко сформулювати мету бакалаврської роботи: створення комфортного вебзастосунку, що ефективно працює на будь-якому пристрої, та використовує модифіковану нейронну модель для розпізнавання медіа-об'єктів на зображеннях та відео з мінімальним споживанням ресурсів, а також обґрунтувати вибір оптимальних середовищ розробки.

У другому розділі детально розглянуто складові, необхідні для розробки та запуску вебзастосунку на основі модифікованої нейронної моделі. Тут було

продемонстровано програмне та апаратне забезпечення для навчання моделі та розгортання додатку, включно з аналізом його переваг і мінімальних вимог. Окремо описано файли, що забезпечують логіку функціонування вебзастосунку, а також приділено значну увагу графічному інтерфейсу користувача, його впливу на взаємодію та способам залучення аудиторії, з відповідними ілюстраціями для наочності.

У третьому розділі докладно проаналізовано ключові операції та принципи обробки інформації в розробленому вебзастосунку. Основну увагу приділено даним різних типів та їх обробці модифікованою нейронною мережею для розпізнавання медіа-об'єктів на зображеннях і відео. Представлено структуру коду, що забезпечує цей процес, а також детально описано взаємодію між сервером та клієнтською частиною. Окремо розглянуто алгоритм навчання нейронної моделі, включно з формуванням необхідних файлів та налаштуванням, що дозволило отримати готову модель для аналізу зображень та відео.

У четвертому розділі основну увагу приділено тестуванню нейронної мережі, що здійснювалося як під час навчання, так і після нього. Моніторинг процесу тренування в реальному часі дозволив відстежувати та аналізувати втрати нейромережі, а також виявляти різні їх види, що є важливим для подальшої оптимізації та досягнення кращих показників розпізнавання медіа-об'єктів. Зокрема, було виявлено неточності у роботі моделі, які проявляються, коли вхідні дані суттєво відрізняються від тренувального набору, що призводить до помилок у класифікації або локалізації об'єктів.

# СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Дерев'янчук А. Ю., Черноволик Г. О. Використання модифікованної нейронної мережі для розпізнавання зображень/ Міжнародній конференції «Молодь в науці: дослідження, проблеми, перспективи Вінницького національного технічного університету (МН ВНТУ–2025): збірник доповідей [Електронний ресурс] — Режим доступу до ресурсу: https://conferences.vntu.edu.ua/index.php/mn/mn2025/paper/view/25646

2. I. Goodfellow, Y. Bengio, A. Courville "Deep Learning", 2016. 773 с. [Електронний ресурс] – Режим доступу до ресурсу: https://www.deeplearningbook.org/

3. Хайрова Н. Ф. Сучасні технології Web-програмування : навч. посібник / Н. Ф. Хайрова, С. В. Петрасова; Нац. техн. ун-т "Харків. політехн. ін-т". – Харків : Панов А. М., 2020. – 112 с.

4. S Poonkuntran, Rajesh Kumar Dhanraj, Balaburugan Balusamy Object Detection with Deep Learning Models, 2022. 276 с. [Електронний ресурс] – Режим доступу до ресурсу: https://books.google.com.ua/books?redir_esc=y&hl=ru&id=jIGIEAAAQBAJ&q

5. TensorFlo v2.16.1 API Documentation [Електронний ресурс] – Режим доступу до ресурсу: https://www.tensorflow.org

6. Welcom To Colab – Colab – Google [Електронний ресурс] – Режим доступу до ресурсу: https://colab.research.google.com/

7. Welcome to Flask — Flask Documentation [Електронний ресурс] – Режим доступу до ресурсу: https://flask.palletsprojects.com/en/2.0.x/

8. Keras: the Python deep learning API [Електронний ресурс] – Режим доступу до ресурсу: https://keras.io/

9. TensorBoard: TensorFlow's visualization toolkit [Електронний ресурс] – Режим доступу до ресурсу https://www.tensorflow.org/tensorboard

10. React. The library for web and native user interfaces. Quick Start [Електронний ресурс] – Режим доступу до ресурсу: https://react.dev/learn

11. Hands-On Transfer Learning with Python [Електронний ресурс] – Режим доступу до ресурсу: https://books.google.com.ua/books?id=aPFsDwAAQBAJ&lpg=PP1&ots=kQr2emWu0&dq=why%20Python%20is%20the%20best%20language%20for%20neural%20network%20development&lr&hl=uk&pg=PP1#v=onepage&q=why%20Python%20is%20the%20best%20language%20for%20neural%20network

12. Neural Networks in Netflix [Електронний ресурс] – Режим доступу до ресурсу: https://www.linkedin.com/pulse/neural-networks-netflix-akanksha-bhatt/

13. Transforming Transportation: Tesla's Vision Neural Networks and Full Self-Driving Capability [Електронний ресурс] – Режим доступу до ресурсу:https://medium.com/@ghaffar_qureshi40/transforming-transportation-teslas-vision-neural-networks-and-full-self-driving-capability-895a4c22cb65

14. MobileNet SSD v2: URL:https://roboflow.com/model/mobilenet-ssd-v2 (дата звернення 21.04.2025).

15. Build fast, responsive sites with Bootstrap [Електронний ресурс] – Режим доступу до ресурсу:https://getbootstrap.com/docs/5.3/getting-started/introduction/

16. S. J. Russell, P. Norvig "Artificial Intelligence: A Modern Approach" Third Edition, 2010. 1152 с. [Електронний ресурс] – Режим доступу до ресурсу: https://people.engr.tamu.edu/guni/csce421/files

17. Нейронні мережі (Neural network, NN) [Електронний ресурс] – Режим доступу до ресурсу:https://evergreens.com.ua/ua/development-services/neural-network.html

18. Ivan Vasilev , Daniel Slater , Gianmario Spacagna , Peter Roelants Python Deep Learning: Exploring deep learning techniques and neural network, 2019. – 379с.

19. Python 3.12.3 Documentation [Електронний ресурс] – Режим доступу до ресурсу: https://docs.python.org/3/

20. Introduction to Node.js [Електронний ресурс] – Режим доступу до ресурсу: https://nodejs.org/en/learn/getting-started/introduction-to-nodejs

21. Machine Learning, ML [Електронний ресурс] – Режим доступу до ресурсу: https://www.it.ua/knowledge-base/technology-innovation/machine-learning

22. Node.js Tutorial [Електронний ресурс] – Режим доступу до ресурсу: https://www.w3schools.com/nodejs/

23. What is Bootstrap: A Beginner's Guide [Електронний ресурс] – Режим доступу до ресурсу: https://careerfoundry.com/en/blog/web-development/what-is-bootstrap-a-beginners-guide/

24. Progressive web apps [Електронний ресурс] – Режим доступу до ресурсу: https://developer.mozilla.org/en-US/docs/Web/Progressive_web_apps

25. Методичні вказівки до виконання бакалаврських кваліфікаційних робіт для студентів спеціальності 121 "Інженерія програмного забезпечення" / Уклад. О.Н. Романюк, Г.О. Черноволик – Вінниця: ВНТУ, 2022. – 52с.

26. What are Neural Networks? [Електронний ресурс] – Режим доступу до ресурсу: https://www.datacamp.com/blog/what-are-neural-networks

27. Embedded Vision Services [Електронний ресурс] – Режим доступу до ресурсу: https://www.phytec.eu/en/leistungen/embedded-vision

28. Attacks on Artificial Intelligence (IV): Privacy Attacks [Електронний ресурс] – Режим доступу до ресурсу: https://telefonicatech.com/en/blog/attacks-ai-privacy-attacks

29. AMD Ryzen 7 9700X processor review: finding hidden reserves [Електронний ресурс] – Режим доступу до ресурсу: https://mezha.media/en/reviews/amd-ryzen-7-9700x-processor-review-finding-hidden-reserves

30. GPU as a Service [Електронний ресурс] – Режим доступу до ресурсу: https://www.beyond.pl/en/services/gpu-as-a-service-gpuaas

# ДОДАТКИ

69

**Додаток А**

**Технічне завдання**

Міністерство освіти і науки України
Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії

ЗАТВЕРДЖУЮ
Завідувач кафедри ПЗ
Романюк О.Н.
«25» березня 2025 р.

**Технічне завдання**
**на бакалаврську кваліфікаційну роботу «Розробка вебзастосунку для розпізнавання зображень за допомогою нейронної мережі»**
**за спеціальністю**
**121 – Інженерія програмного забезпечення**

Керівник бакалаврської кваліфікаційної роботи:
к.т.н., доц. каф. ПЗ Черноволик Г. О.
«24» 03 2025 року.
Виконав:
студент гр. 4ПІ-21б Дерев'янчук А. Ю.
«24» 03 2025 року.

м. Вінниця – 2025 рік

**1. Найменування та галузь застосування**

Бакалаврська кваліфікаційна робота: «Розробка вебзастосунку для розпізнавання зображень за допомогою нейронної мережі».

Галузь застосування – вебтехнології.

**2. Підстава для розробки.**

Завдання на роботу, яке затверджене на засіданні кафедри програмного забезпечення – протокол №18 від 10 лютого 2025 р.

**3. Мета та призначення розробки.**

Метою даного проекту є створення веборієнтованого засобу на основі модифікованої нейронної мережі для класифікації та знаходження положень відповідних об’єктів, при обробці зображень або відео.

**4. Вихідні дані для проведення НДР**

Джерелом інформації є технічна та науково-технічна література, технічна документація, публікації у періодичних виданнях та електронні статті у мережі Інтернет.

**5. Технічні вимоги**

Вхідні дані — зображення або відео.

Вихідні дані — інформація про положення об’єктів на зображеннях або відео та можливість зберігання оброблених медіафайлів на власний пристрій.

**6. Конструктивні вимоги.**

Інтерфейс програми повинен відповідати естетичним та ергономічним вимогам, повинна бути зручною в обслуговуванні та керуванні.

Графічна та текстова документація повинна відповідати діючим стандартам України.

**7. Перелік технічної документації, що пред’являється по закінченню робіт:**

–пояснювальна записка до бакалаврської кваліфікаційної роботи;

–технічне завдання;

–лістинги програми.

### 8. Вимоги до рівня уніфікації та стандартизації

При розробці програмних засобів слід дотримуватися уніфікації і ДСТУ.

### 9. Стадії та етапи розробки

| № з/п | Назва етапів бакалаврської кваліфікаційної роботи | Строк виконання етапів роботи |
|---|---|---|
| 1 | Аналіз розвитку вебзастосунку для розпізнавання зображень за допомогою нейроної мережі | 25.03.25 – 03.04.25 |
| 2 | Розробка структури та алгоритмів програмного продукту | 04.04.25 – 15.04.25 |
| 3 | Варіантний аналіз та обгрунтування вибору засобів програмної реалізації | 16.04.25 – 17.04.25 |
| 4 | Розробка програмних модулів та реалізації вебзастосунку для розпізнавання зображень за допомогою нейроної мережі | 18.04.25 – 27.04.25 |
| 5 | Тестування вебзастосунку | 28.04.25 – 11.05.25 |
| 6 | Оформлення матеріалів до захисту БКР | 12.05.25 – 30.05.25 |

### 10. Порядок контролю та прийняття.

Усі етапи бакалаврської кваліфікаційної роботи контролюються науковим керівником згідно плану по виконанню роботи. Прийняття бакалаврської кваліфікаційної роботи здійснюється ДЕК, затвердженою зав. кафедрою згідно з графіком захисту.

72

**Додаток Б**

**ПРОТОКОЛ ПЕРЕВІРКИ КВАЛІФІКАЦІЙНОЇ РОБОТИ**

Назва роботи: «Розробка вебзастосунку для розпізнавання зображень за допомогою нейронної мережі»

Тип роботи: бакалаврська кваліфікаційна робота
(бакалаврська кваліфікаційна робота / магістерська кваліфікаційна робота)

Підрозділ: кафедра програмного забезпечення, факультет ІТКІ, 4ПІ-21Б
(кафедра, факультет, навчальна група)

Коефіцієнт подібності текстових запозичень, виявлених у роботі системою StrikePlagiarism 6,8%

Висновок щодо перевірки кваліфікаційної роботи (відмітити потрібне)

■ Запозичення, виявлені у роботі, є законними і не містять ознак плагіату, фабрикації, фальсифікації. Роботу прийняти до захисту

☐ У роботі не виявлено ознак плагіату, фабрикації, фальсифікації, але надмірна кількість текстових запозичень та/або наявність типових розрахунків не дозволяють прийняти рішення про оригінальність та самостійність її виконання. Роботу направити на доопрацювання.

☐ У роботі виявлено ознаки плагіату та/або текстових маніпуляцій як спроб укриття плагіату, фабрикації, фальсифікації, що суперечить вимогам законодавства та нормам академічної доброчесності. Робота до захисту не приймається.

Експертна комісія:

_____ _____
(прізвище, ініціали, посада) (підпис)

_____ _____
(прізвище, ініціали, посада) (підпис)

Особа, відповідальна за перевірку _____ Черноволик Г. О.

З висновком експертної комісії ознайомлений(-на)

Керівник _____ к.т.н., доц. каф. ПЗ Черноволик Г. О.
(підпис) (прізвище, ініціали, посада)

Здобувач _____ Дерев'янчук А. Ю.
(підпис) (прізвище, ініціали)

# Додаток В

## Лістинг програми

## api.py

```
from flask import Flask, jsonify, send_from_directory, request
from flask_cors import CORS
from werkzeug.utils import secure_filename

import os
import cv2
import numpy as np
import glob
from tensorflow.lite.python.interpreter import Interpreter

PATH_TO_IMAGES = './venv/images/test'
PATH_TO_MODEL = './venv/custom_model_lite/detect.tflite'
PATH_TO_LABELS = './venv/labelmap.txt'
PATH_TO_TEMP = './venv/temp'
heightVid = 0
widthVid = 0
framesCount = 0

app = Flask(__name__)
app.config['UPLOAD_FOLDER'] = PATH_TO_IMAGES
app.config['DOWNLOAD_FOLDER'] = PATH_TO_TEMP
ALLOWED_EXTENSIONS = {'png', 'jpg', 'jpeg', 'bmp', 'mp4'}

def extractImages(filename):
    global heightVid, widthVid, framesCount
    parent_dir = PATH_TO_IMAGES
    pathToVideo = os.path.join(parent_dir, filename)
    directory = filename.rsplit('.', 1)[0]
    pathOut = os.path.join(parent_dir, directory)

    if not os.path.exists(pathOut):
        os.mkdir(pathOut)

    count = 0
    vidcap = cv2.VideoCapture(pathToVideo)
    heightVid = vidcap.get(cv2.CAP_PROP_FRAME_HEIGHT)
```

```
            widthVid = vidcap.get(cv2.CAP_PROP_FRAME_WIDTH)

            while True:
                success, image = vidcap.read()
                if not success:
                    break
                cv2.imwrite(os.path.join(pathOut, f"frame{count}.jpg"), image)
                count += 1

            framesCount = count
            return pathOut

        def allowed_file(filename):
            return '.' in filename and filename.rsplit('.', 1)[1].lower() in
ALLOWED_EXTENSIONS

        @app.route("/uploads/<path:name>")
        def download_file(name):
            return send_from_directory(app.config['DOWNLOAD_FOLDER'], name,
as_attachment=True)

        @app.route('/api/upload', methods=['POST'])
        def predict(modelpath=PATH_TO_MODEL, imgpath=PATH_TO_IMAGES,
lblpath=PATH_TO_LABELS, temppath=PATH_TO_TEMP):
            d = {}
            isVideo = False

            if 'file' not in request.files:
                return jsonify({'error': "Couldn't upload file", 'status': 0})

            file = request.files['file']
            if file.filename == '':
                return jsonify({'error': "Couldn't upload file", 'status': 0})

            if not file or not allowed_file(file.filename):
                return jsonify({'error': "Incorrect file type", 'status': 0})

            filename = secure_filename(file.filename)
            file.save(os.path.join(app.config['UPLOAD_FOLDER'], filename))

            if filename.lower().endswith('.mp4'):
                PATH_TO_CREATED_FRAMES = extractImages(filename)
```

```
            isVideo = True
            min_conf = 0.75
            images       =       glob.glob(os.path.join(PATH_TO_CREATED_FRAMES,
'*.[jp][pn]g')) + glob.glob(
                os.path.join(PATH_TO_CREATED_FRAMES, '*.bmp'))
        else:
            min_conf = 0.6
            images = [os.path.join(imgpath, filename)]

        # Load label map
        with open(lblpath, 'r') as f:
            labels = [line.strip() for line in f.readlines()]

        # Load model
        interpreter = Interpreter(model_path=modelpath)
        interpreter.allocate_tensors()

        # Get model details
        input_details = interpreter.get_input_details()
        output_details = interpreter.get_output_details()
        height = input_details[0]['shape'][1]
        width = input_details[0]['shape'][2]
        float_input = (input_details[0]['dtype'] == np.float32)

        for image_path in images:
            image = cv2.imread(image_path)
            if image is None:
                continue

            image_rgb = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
            imH, imW, _ = image.shape
            image_resized = cv2.resize(image_rgb, (width, height))
            input_data = np.expand_dims(image_resized, axis=0)

            if float_input:
                input_data = (np.float32(input_data) - 127.5) / 127.5

            interpreter.set_tensor(input_details[0]['index'], input_data)
            interpreter.invoke()

            # Get outputs safely
            try:
```

```
                boxes = interpreter.get_tensor(output_details[0]['index'])[0]
                classes                                                    =
interpreter.get_tensor(output_details[1]['index'])[0]
                scores = interpreter.get_tensor(output_details[2]['index'])[0]
            except Exception as e:
                print("Error getting outputs:", e)
                continue

            if not isinstance(scores, (list, np.ndarray)) or len(scores) == 0:
                continue

            for i in range(len(scores)):
                if i >= len(boxes) or i >= len(classes):
                    continue

                score = scores[i]
                if isinstance(score, np.ndarray):
                    score = score.item()

                if score > min_conf and score <= 1.0:
                    box = boxes[i]
                    if isinstance(box, np.ndarray) and box.shape[0] == 4:
                        ymin = int(max(1, box[0] * imH))
                        xmin = int(max(1, box[1] * imW))
                        ymax = int(min(imH, box[2] * imH))
                        xmax = int(min(imW, box[3] * imW))

                        cv2.rectangle(image, (xmin, ymin), (xmax, ymax), (10,
255, 0), 2)
                        object_name = labels[int(classes[i])]
                        label = f'{object_name}: {int(score * 100)}%'
                        labelSize,     baseLine     =     cv2.getTextSize(label,
cv2.FONT_HERSHEY_SIMPLEX, 0.7, 2)
                        label_ymin = max(ymin, labelSize[1] + 10)
                        cv2.rectangle(image, (xmin, label_ymin - labelSize[1] -
10),
                                      (xmin   +   labelSize[0],   label_ymin   +
baseLine - 10), (255, 255, 255), cv2.FILLED)
                        cv2.putText(image, label, (xmin, label_ymin - 7),
                                    cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0, 0, 0),
2)
```

```
            if isVideo:
                PATH_TO_RESULT = filename.rsplit('.', 1)[0]
                resVid = os.path.join(temppath, PATH_TO_RESULT)
                if not os.path.exists(resVid):
                    os.mkdir(resVid)
                detectName = os.path.split(image_path)[-1]
                cv2.imwrite(os.path.join(resVid, detectName), image)
            else:
                detectName = filename.rsplit('.', 1)[0] + '_detected.' + filename.rsplit('.', 1)[1]
                cv2.imwrite(os.path.join(temppath, detectName), image)
                return jsonify({'status': 1, 'image': '/uploads/' + detectName})

        if isVideo and widthVid and heightVid and framesCount:
            fourcc = cv2.VideoWriter_fourcc(*'mp4v')
            downloadName = filename.rsplit('.', 1)[0] + '_detected.mp4'
            sizeVid = (int(widthVid), int(heightVid))
            downloadLink = os.path.join(temppath, downloadName)

            videoDownload = cv2.VideoWriter(downloadLink, fourcc, 15, sizeVid)
            for j in range(framesCount):
                frame_path = os.path.join(resVid, f"frame{j}.jpg")
                if os.path.exists(frame_path):
                    tempFrame = cv2.imread(frame_path)
                    videoDownload.write(tempFrame)

            videoDownload.release()
            return jsonify({'status': 1, 'video': '/uploads/' + downloadName})

        return jsonify({'error': "Processing failed", 'status': 0})

    cors = CORS(app, resources={'/*': {'origins': 'http://localhost:3000'}})

    if __name__ == "__main__":
        app.run(debug=True)
```

# MainContent.js

```
    import './MainContent.css'
```

```
import Button from 'react-bootstrap/Button';
import React, { useState } from 'react';
import ReactPlayer from 'react-player';

function MainContent() {
  const [file, setFile] = useState(0);
  const [download, setDownload] = useState('');
  const [error, setError] = useState('')
  const [fileLink, setFileLink] = useState('');
  const [fileFormat, setFileFormat] = useState('');
  const [downloadIsVid, setDownloadIsVid] = useState(false);

  function Refresh(e) {
    console.log("START REFRESH");
    e.preventDefault();
    setFile(0);
    setError('');
    setDownload('');
    setFileLink('');
    setFileFormat('');
    setDownloadIsVid(false);
  }

  function handleChangeFile(e) {
    console.log("START INPUT");
    e.preventDefault();
    const files = e.target.files;
    console.log(files[0]);
    if(files[0] === undefined) return;
    const                          fileSize                          =
parseFloat(((files[0].size/1024)/1024).toFixed(4));
    console.log(fileSize);
    console.log(files[0].type);
    if(files && files[0]){
      if(fileSize < 10){
        setFile(files[0]);
        setFileLink(URL.createObjectURL(files[0]));
        setFileFormat(files[0].type);
      } else{
        setError("File too big (More than 10 MB)")
      }
    } else {
```

```
            setError("Incorect file input")
          }
        };

        function handleSubmit(e) {
          e.preventDefault();
          if(file !== 0) {
            setError('');
            const data = new FormData();
            data.append('file', file);
            console.log(file);
            fetch('http://localhost:5000/api/upload',{
              method: 'POST',
              headers: {
                'Access-Control-Allow-Origin': '*'
              },
              body: data,
            }).then(async (resp) => {
              const json = await resp.json()
              console.log(json)
              if(json.status === 1 && json.image){
                setDownload('http://localhost:5000' + json.image)
              } else if(json.status === 1 && json.video){
                setDownload('http://localhost:5000' + json.video)
                setDownloadIsVid(true)
              } else {
                setError(json.error)
              }
            })
          } else {
            setError("Choose your image");
          }
        };

        return (
          <div className="MainContent text-center mt-3">
            <h2 className='fs-3'>Unleash the power of cutting-edge artificial
intelligence
            with <b>ObjectProbe AI</b>, your go-to web application for
seamlessly
            detecting objects in photos/videos</h2>
            <div className={error === '' ?
```

```
            'd-none'
            :
            'mt-3 text-danger'
            }>
            {error}
            </div>
            {download === '' ?
              <form onSubmit={handleSubmit}>
              <label htmlFor='getFile' className={file === 0 ? 'Load mt-3 d-
block' : 'mt-2 d-block'}>
              {file === 0 ?
              <div>
              <svg xmlns="http://www.w3.org/2000/svg" width="20" height="20"
fill="currentColor" className="bi bi-upload m-1" viewBox="0 0 16 16"
stroke='black' strokeWidth='0.2'>
                <path d="M.5 9.9a.5.5 0 0 1 .5.5v2.5a1 1 0 0 0 1 1h12a1 1 0 0
0 1-1v-2.5a.5.5 0 0 1 1 0v2.5a2 2 0 0 1-2 2H2a2 2 0 0 1-2-2v-2.5a.5.5 0 0 1
.5-.5"/>
                <path d="M7.646 1.146a.5.5 0 0 1 .708 0l3 3a.5.5 0 0 1-
.708.708L8.5 2.707V11.5a.5.5 0 0 1-1 0V2.707L5.354 4.854a.5.5 0 1 1-.708-
.708z"/>
              </svg>
              <div>Load image/video here<br/>
              (jpeg, jpg, png, bmp, mp4)</div>
              </div>
              :
              <div className='h6 text-secondary mt-3'>
                <img className={fileFormat === 'video/mp4' ? 'd-none' : 'd-
inline mb-2'} src={fileLink} alt='preview' width='40%' height='40%'/>
                <div className={fileFormat === 'video/mp4' ? 'mb-2 d-flex
justify-content-center' : 'd-none'}>
                <ReactPlayer url={fileLink} controls={true} muted/>
                </div>
                <br/>-To change the file, click HERE-
              </div>
              }
            </label>
            <input type='file' name='file' id='getFile' className='d-none'
multiple={false}
            onChange={handleChangeFile}
            accept='.jpg,.png,.jpeg,.bmp,.mp4'/>
            <Button type='submit' variant='success' className='mt-
```

```
3'>DETECT</Button>
            </form>
            :
            <div>
              {downloadIsVid ?
              <div className='mt-3 text-success h5'>
              Click on the button to download created Video<br/>
              <p className='text-secondary h6'>(If there are no labels on the
video, then no object was detected)</p>
              <div className='mb-2 d-flex justify-content-center'>
                <ReactPlayer url={download} controls={true} muted/>
              </div>
              <br/>
              <Button className='mt-3' onClick={Refresh}>
                Load another file
              </Button>
              <Button className='mt-3 ms-3 bg-success' href={download}>
                Download
              </Button>
              </div>
              :
              <div className='mt-3 text-success h5'>
              Click on the photo to download<br/>
              <p className='text-secondary h6'>(If there are no labels on the
foto, then no object was detected)</p>
              <a href={download} download>
              <img  className='d-inline  m-1'  src={download}  alt='download'
width='40%' height='40%'/>
              </a>
              <br/>
              <Button className='mt-3' onClick={Refresh}>
                Load another file
              </Button>
              </div>
              }
            </div>
            }
          </div>
        );
      }
      export default MainContent;
```

# Додаток Г

## Ілюстративна частина

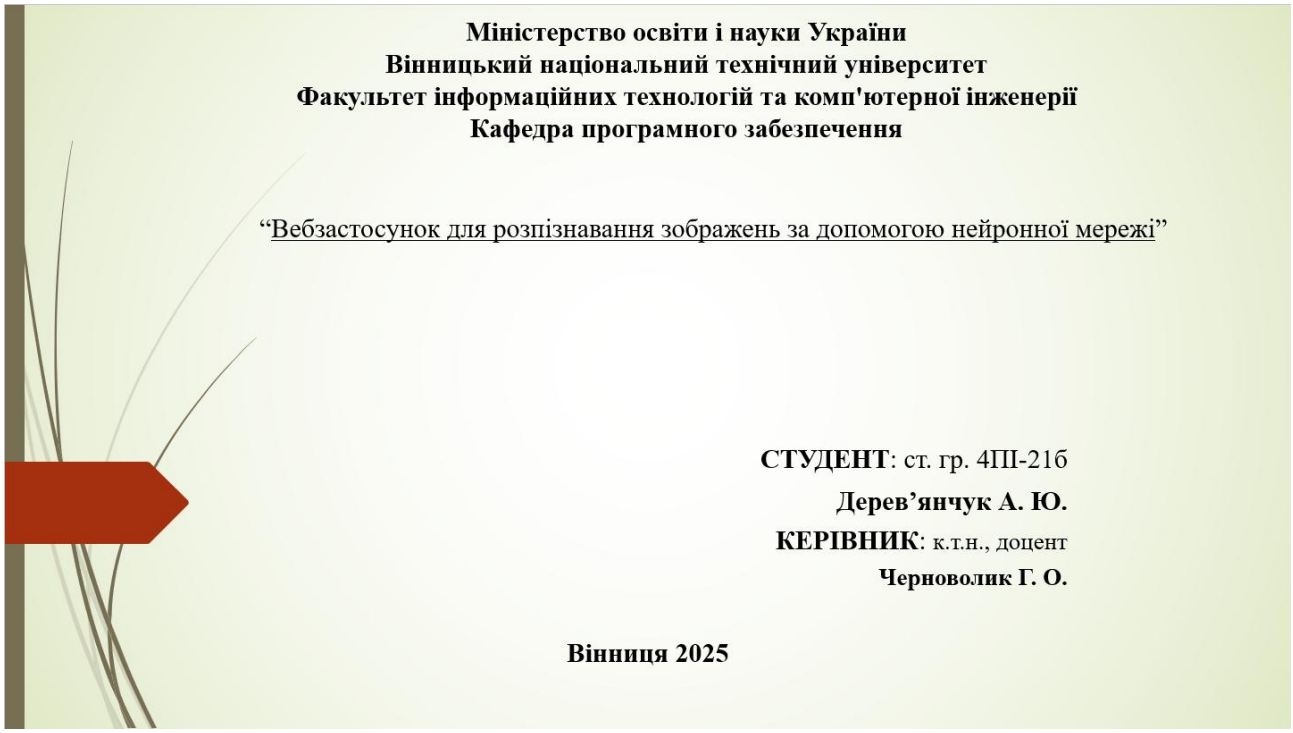


Рисунок Г.1 – слайд 1

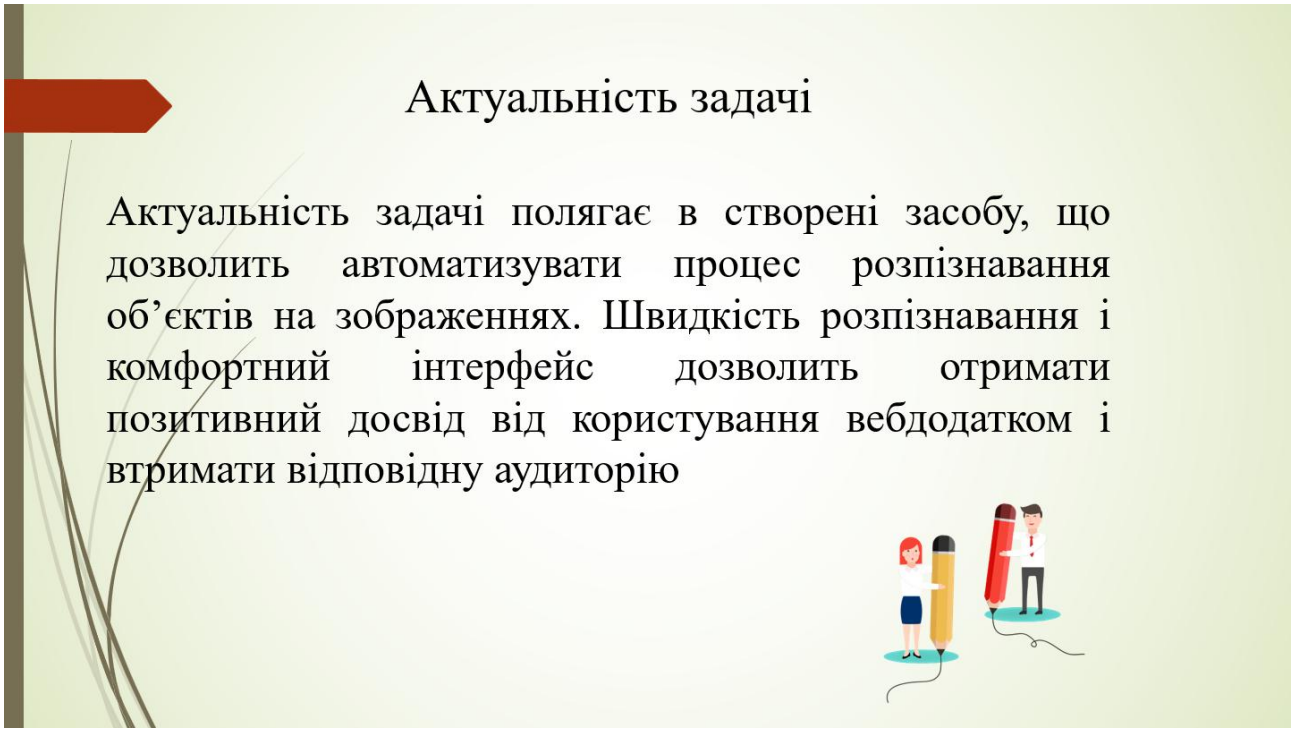


Рисунок Г.2 – слайд 2

- Мета та завдання дослідження. Метою є збільшення точності та оперативності розпізнавання об’єктів на зображеннях в режимі реального часу за рахунок розробки вебзастосунку для розпізнавання зображень за допомогою нейронної мережі.
- Об’єкт дослідження – метод інтерактивного адаптивного навчання, призначений для керування та спостереження за функціонуванням комп'ютерної нейромережевої системи розпізнавання зображень.
- Предмет дослідження – засоби управління і моніторингу роботи комп’ютерної нейромережевої системи розпізнавання зображень.

Рисунок Г.3 – слайд 3

## Задачі

1) розробити архітектуру вебзастосунку для розпізнавання зображень;
2) розробити структуру додатку;
3) розробити дизайн та графічні елементи;
4) розробити метод інтерактивного адаптивного навчання нейронної мережі в умовах вебзастосунку;
5) розробити алгоритми роботи системи;
6) розробити функціонал системи
7) провести тестування розробленої системи.

Рисунок Г.4 – слайд 4

## Новизна отриманих результатів

- У межах дослідження було розроблено метод інтерактивного адаптивного навчання нейронної мережі, який інтегровано у вебзастосунок для розпізнавання зображень. Була реалізована модульна архітектура, що поєднує інтерфейс вебзастосунку з серверною частиною, яка відповідає за обробку зображень нейронною мережею.
- Застосовано механізми кешування, формування логів, а також збереження результатів взаємодії користувача для подальшого аналізу ефективності моделі. Отримані результати свідчать про підвищення точності моделі з кожним етапом донавчання, що особливо помітно у випадках розпізнавання нестандартних чи погано структурованих зображень.
- Розроблений метод дозволяє створити адаптивний вебзастосунок, здатний ефективно функціонувати в умовах змінного середовища та постійного оновлення вхідних даних. Така система може використовуватись у сферах, де точність класифікації зображень має критичне значення: медичні зображення, моніторинг безпеки, аграрні технології, контроль якості у виробництві тощо.

Рисунок Г.5 – слайд 5

## Аналоги

- Google Lens – це надзвичайно корисний сервіс від Google. Він дозволяє визначати об'єкти, розпізнавати текст, рослини та тварин. Крім цього, він вміє перекладати тексти у реальному часі, використовуючи камеру вашого ґаджета або зображення, що вже є у вашому пристрої.
- Amazon Rekognition – це сервіс, розроблений Amazon Web Services (AWS), який спеціалізується на аналізі фотографій та відео. З його допомогою можна виявляти різні об'єкти, людей, текст. Він також здатен розпізнавати обличчя та визначати емоції. Цей інструмент призначено переважно для корпоративних користувачів та розробників.
- Clarifai – це платформа штучного інтелекту, що надає можливості розпізнавання зображень та відео на високому рівні для багатьох сфер.

Рисунок Г.6 – слайд 6

## Порівняльний аналіз аналогів

| Функціонал | Google Lens | Amazon Rekognition | Clarifai | Власний застосунок |
|---|---|---|---|---|
| Завантаження зображень та відео | - | + | + | + |
| Розпізнавання зображень | + | + | + | + |
| Розпізнавання об'єктів на відео | - | + | + | + |
| Самонавчання на основі даних | - | - | + | + |
| Інтерфейс користувача | + | - | - | + |
| Модель використання | + | - | - | + |
| Загальна оцінка | 50% | 50% | 66% | 100% |

Рисунок Г.7 – слайд 7

## Алгоритм навчання нейронної моделі

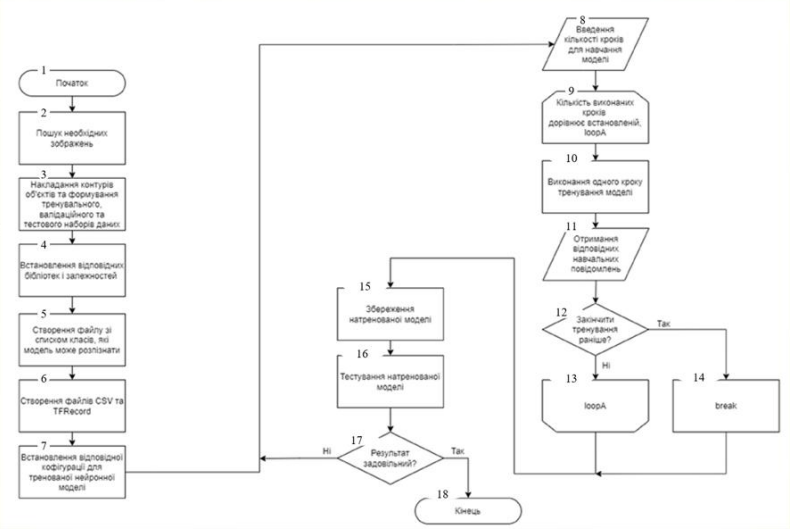


Рисунок Г.8 – слайд 8

## Процес тренування нейронної моделі

Процес тренування нейронної моделі включає подачу вхідних даних, обчислення виходу, порівняння виходу з реальними значеннями, обчислення втрат, і корекцію ваг для мінімізації помилки. Важливим етапом є вибір відповідної архітектури моделі, яка визначає, як шари нейронної мережі організовані і взаємодіють між собою. У даній роботі була вибрана архітектура MobileNet V2.

Рисунок Г.9 – слайд 9

## Середовище тренування нейромережі

Для тренування нейронної моделі використовувався хмарний сервіс для машинного навчання Google Colab, який дозволяє запускати та розробляти Python-код у браузері з використанням апаратних прискорювачів, GPU та TPU.

Рисунок Г.10 – слайд 10

## Взаємодія вебдодатка з нейронною моделлю

Взаємодія з розробленою нейронною моделлю відбувається на серверній частині вебзастосунку. Серверна частина створена на Flask, вона обробляє запити від клієнта, приймає файли, і передає їх нейронній моделі для розпізнавання об'єктів. У свою чергу нейромережа повертає ідентифіковані об'єкти та їхні координати, що потім накладаються на відповідний файл і доставляються користувачеві.

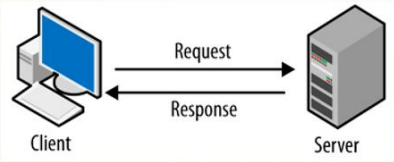


Рисунок Г.11 – слайд 11

## Розроблений метод

Метод інтерактивного адаптивного навчання нейронної мережі передбачає безперервне удосконалення моделі розпізнавання зображень безпосередньо під час її експлуатації у вебзастосунку. Користувачі отримують можливість взаємодіяти з результатами класифікації, коригуючи помилки або підтверджуючи правильні відповіді, що автоматично враховується в процесі донавчання моделі. Метод поєднує принципи зворотного зв’язку, локального донавчання та контекстної адаптації, формуючи гнучку, самонавчальну систему штучного інтелекту.

Рисунок Г.12 – слайд 12

Блок-схема інтерактивного адаптивного навчання нейронної мережі

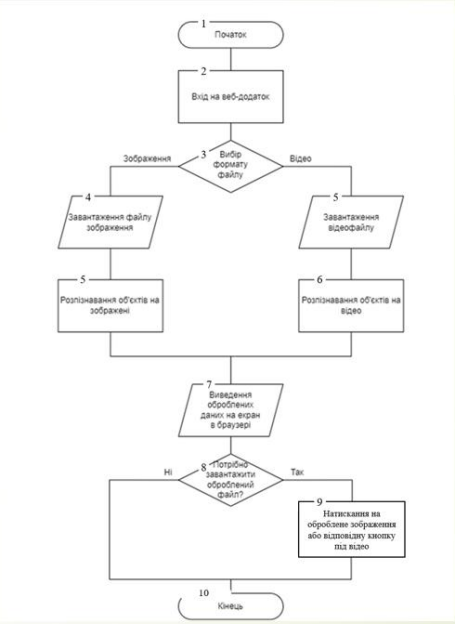


Рисунок Г.13 – слайд 13

## Обробка зображень нейронною моделлю

При отриманні запиту від клієнта, на сервері зображення спочатку обробляється для підготовки до подальшого розпізнавання. Далі воно надсилається до нейронної мережі, де відбувається аналіз зображення для розпізнавання об'єктів на ньому. Після аналізу мережа повертає результати, які використовуються для подальшої обробки та відображення на клієнтському пристрої.

Рисунок Г.14 – слайд 14


Рисунок Г.15 – слайд 15


Рисунок Г.16 – слайд 16

## Програмне забезпечення для роботи з нейронною моделлю

Для навчання та комфортного тестування нейронної моделі було використано інструменти, що надає бібліотека TensorFlow. Дана бібліотека підтримує широкий спектр алгоритмів і може працювати на різних платформах, що робить її ідеальним засобом для роботи з нейромережами.

Рисунок Г.17 – слайд 17

## Графіки втрат нейронної моделі

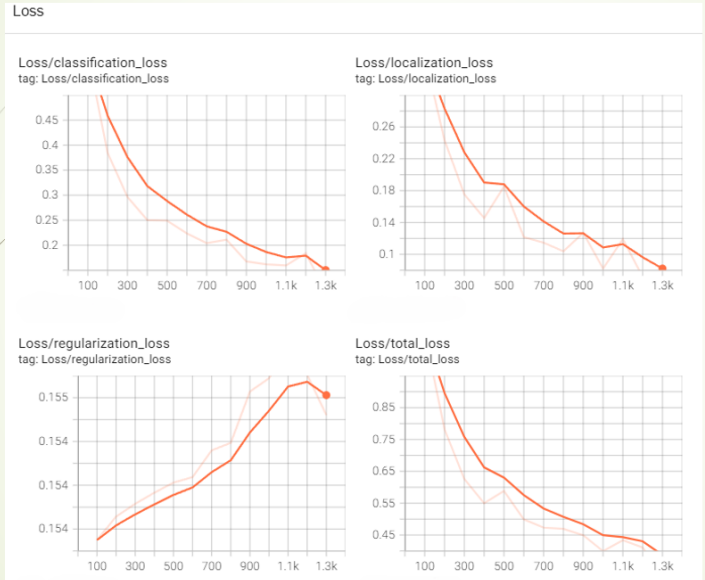


Рисунок Г.18 – слайд 18

# Висновки

У результаті бакалаврської кваліфікаційної роботи було розроблено вебзастосунок для розпізнавання зображень за допомогою нейронної мережі. Нейронна мережа, створена для вебзастосунку, є досить оперативною при обробці зображень, що дозволяє їй функціонувати навіть на малопотужних пристроях. Це стає можливим завдяки використанню сучасної архітектури нейромережі в її основі. Для розробки було використано середовище розробки PyCharm.

Вебзастосунок створено для покращення вражень користувачів, забезпечуючи швидку та легку взаємодію з розробленою нейронною моделлю. Це відкриває широкі можливості як для бізнесу, так і для індивідуального використання. Інтерфейс вебзастосунку зручний та інтуїтивно зрозумілий, що значно скорочує час на адаптацію для нових користувачів, дозволяючи їм максимально ефективно використовувати всі його функції. Таким чином, було розроблено архітектуру вебзастосунку для розпізнавання зображень, розроблено структуру додатку, розроблено дизайн та графічні елементи, розроблено метод інтерактивного адаптивного навчання нейронної мережі в умовах вебзастосунку, розроблено алгоритми роботи системи, розроблено функціонал системи та протестовано розроблену систему.

Рисунок Г.19 – слайд 19

# Публікації й апробації

- **Апробація матеріалів бакалаврської кваліфікаційної роботи.** Основні положення БКР доповідалися та обговорювалися на Міжнародній конференції «Молодь в науці: дослідження, проблеми, перспективи Вінницького національного технічного університету (МН ВНТУ–2025).


Рисунок Г.20 – слайд 20