КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ТАРАСА ШЕВЧЕНКА

Факультет комп'ютерних наук та кібернетики

Кафедра інтелектуальних програмних систем

Кваліфікаційна робота

на здобуття ступеня бакалавра

за спеціальністю 121 Інженерія програмного забезпечення

на тему:

ЗАСТОСУВАННЯ МЕТОДІВ МАШИННОГО НАВЧАННЯ ДО ЗАДАЧ АВТОМАТИЧНОЇ КЛАСИФІКАЦІЇ ВЗУТТЯ У МАГАЗИНІ

Виконав студент 4-го курсу

Данило ТЮТЮШКІН _____

(підпис)

Науковий керівник:

доцент, кандидат фіз.-мат. наук

Ярослав ЛІНДЕР _____

(підпис)

Засвідчую, що в цій роботі немає запозичень з праць інших авторів без відповідних посилань.

Студент _____

(підпис)

Роботу розглянуто і допущено до захисту на засіданні кафедри інтелектуальних програмних систем «____» _____ 2022 р.

протокол № ___

Завідувач кафедри Олександр ПРОВОТАР _____

(підпис)

Київ – 2022

ЗМІСТ

ВСТУП

З розвитком технологій та зі зростанням популярності нейронних мереж та машинного навчання, з’явилось безліч можливостей замінювати ручну рутинну роботу найманного працівника більш точною та автоматичною системою розпізнавання тих чи інших об’єктів. Найчастіше, технології машинного навчання та розпізнавання (класифікації) зображень використовують на підприємствах задля автоматизації та пришвидшення роботи. Саме збільшення продуктивності праці та водночас зменшення впливу людської помилки підштовхує до розробки систем, які полегшать життя працівникам та позбавлять їх рутинної роботи.

Актуальність роботи полягає у адаптивності системи до роботи із задачами розпізнавання та класифікації об’єктів на підприємствах.

Мета й завдання роботи. Метою роботи є розробка багаторівневої системи розпізнавання та класифікації об’єктів на основі їх зображень. Для досягнення цієї мети поставлено такі завдання:

- визначити об’єм даних (фотозображень) для обробки;
- підготувати (обробити) вхідний набір зображень для навчання моделі;
- оцінити якість навчання та можливі похибки моделі;
- спроектувати та розробити інтерфейс користувача для зручного використання класифікації зображень.

Інструментами розробки є мова програмування Python, фреймворк глибокого навчання PyTorch [1], а саме допоміжну бібліотеку Detectron2 [2] (для класифікації зображень) та Tensorflow [3] в парі з моделлю PSPNet [4] (для семантичної сегментації), та допоміжні технології для створення користувацького інтерфейсу (мова JavaScript: ReactJS, NodeJs фреймворки)

РОЗДІЛ 1
ОСОБЛИВОСТІ ТЕХНОЛОГІЙ КЛАСИФІКАЦІЇ ЗОБРАЖЕНЬ

Для розробки системи розпізнавання було обрано наступні технології:

- Python в якості мови програмування;
- PyTorch (Detectron2) в якості фреймворку для класифікації зображень.

1.1. Мова програмування

Мовою програмування було обрано **Python**. Мова Python - інтерпретована, об’єктно-орієнтована мова програмування високого рівня з динамічною семантикою. Динамічність та структурованість цієї високорівневої мови роблять Python привабливим для розробки додатків, особливо таких, що складаються з декількох частин.

Легкий та зрозумілий синтаксис Python-а шанується читабельністю і, отже, допомагає швидкій та якісній розробці додатку. Python - це мова із відкритим кодом, безкоштовна для використання, та може безперешкодно запускатись на будь-якій сучасній оперативній системі, що в поєднанні із високими можливостями вбудованими в цю мову роблять її одною із фаворитів серед усіх існуючих.

Оскільки в мові Python відсутній крок компіляції, розробка програм стає ще більш швидкою та зручною. Програмісти можуть без затримок вносити зміни, запускати та тестувати свій код. Відлагодження програм Python також неймовірно просте: помилка або неправильний ввід ніколи не спричиняють помилку сегментації [5]. Проте в мові все одно реалізована система винятків, так як логічні помилки всередині коду можливі в будь-якому випадку.

У світі існує багато різних реалізацій мови Python. Проте найпопулярніша та найзручніша реалізація пов'язана з іншою відомою мовою, а саме C. Це також популярна мова програмування, що із своїх значущих переваг має швидкісіть роботи програми, написаної на ній. А так як велика кількість бібліотек в Python написані саме мовою C - він стає додатково також продуктивним та ефективним при специфічних задачах.

Якраз можна додати, що Python - це найбільш вживана мова розробки, коли йдеться про **машинне навчання та обробку даних**. Для нього створено безліч фреймворків, які покращують та полегшують життя програмістів, а отже підвищують продуктивність в цілому.

Додаток, який розробляється для реалізації системи розпізнавання має обробляти великий набір вхідних даних (тисячі зображень) та запускати на них навчання моделі для класифікації зображень, тобто використовуються усі позитивні сторони мови Python.

1.2. Платформа класифікації зображень

PyTorch - це система глибокого навчання з відкритим кодом, створена для гнучкості та модульності для досліджень, зі стабільністю та підтримкою, необхідною для розгортання виробництва. PyTorch пропонує пакет Python для високорівневих функцій, таких як обчислення тензорів (наприклад, NumPy), з потужним прискоренням графічного процесора та TorchScript для легкого переходу між нетерплячим та графічним режимом. З останнім випуском PyTorch, фреймворк забезпечує виконання на основі графіків, розподілене навчання, мобільне розгортання та квантування.

PyTorch використовує техніку, яка називається автодиференціацією в зворотному режимі, що дозволяє розробникам довільно змінювати поведінку **нейронної мережі** з нульовою затримкою або накладними витратами, прискорюючи ітерації досліджень, що отримало назву “Динамічні нейронні мережі” [6].

За допомогою TorchScript PyTorch забезпечує простоту у використанні та гнучкість у нетерплячому режимі, одночасно переходячи до графічного режиму для швидкості, оптимізації та функціональності в середовищах виконання C ++.

Розглянемо особливості нейронних мереж, адже фреймворк PyTorch активно використовує їх для створення моделей розпізнавання.

Нейронна мережа є обчислювальною навчальною системою, яка використовує мережу функцій для розуміння та перекладу введених даних однієї форми у бажаний результат, як правило, в іншій формі. Концепція штучної нейронної мережі була породжена людською біологією та тим, як нейрони людського мозку функціонують разом, щоб зрозуміти вклади людських почуттів. Нейронні мережі - це лише один із багатьох інструментів та підходів, що використовуються в алгоритмах **машинного навчання**. Сама нейронна мережа може використовуватися як частина в багатьох різних алгоритмах машинного навчання для обробки складних даних у просторі, який можуть зрозуміти комп'ютери. Нейронні мережі сьогодні застосовуються до багатьох реальних проблем, включаючи розпізнавання мови та зображень, фільтрування спаму, медичну діагностику тощо [7] (рис. 1).

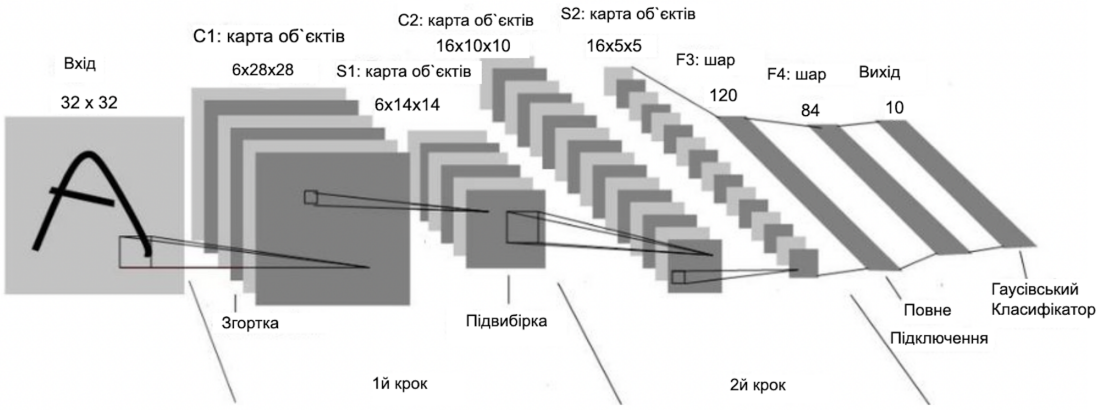

Рис. 1. Приклад класифікації фотозображення нейронною мережею

Для створення моделей розпізнавання запускається процес **навчання нейронної мережі** на вхідних даних, який складається з наступних кроків:

- визначення нейронної мережі, яка має деякі параметри, що піддаються вивченню (ваги);
- ітерація над набором вхідних даних;
- процес проведення даних крізь нейронну мережу;
- обчислення похибок (loss) - наскільки отриманий результат віддалений від правильного;
- пропагування градієнтів назад до параметрів нейронної мережі;
- оновлення ваг (weight) мережі за формулою:
  *weight = weight - learning_rate * gradient* ;
- повторення попередніх кроків (ітерацій) для змінених вагів нейронної мережі [8].

Задля простоти роботи із потужним фреймворком PyTorch було вирішено використовувати зручну його реалізацію у платформі Detectron2.

Detectron2 - це платформа наступного покоління для розпізнавання та сегментації об'єктів. Detectron2 був побудований Facebook AI Research (FAIR) для підтримки швидкого впровадження та оцінки нових досліджень комп'ютерного зору. Він включає реалізації для таких алгоритмів виявлення об’єктів:

- маска R-CNN;
- швидкий R-CNN;
- TensorMask;
- RPN та інші.

На рисунку 2 зображений приклад роботи Маски (Mask) R-CNN та використання сегментації екземплярів (завдання виявлення та розмежування кожного окремого об’єкта, який цікавить, що з’являється на зображенні)

Рис. 2. Приклад сегментації екземплярів за допомогою Detectron2

Mask R-CNN - це концептуально проста, гнучка та загальна структура для сегментації екземплярів об’єктів. Її підхід ефективно виявляє об’єкти на зображенні, одночасно створюючи високоякісну маску сегментації для кожного екземпляра. Метод, який називається Mask R-CNN, розширює швидший R-CNN, додаючи гілку для прогнозування маски об’єкта паралельно існуючій гілці для розпізнавання обмежуючого поля. Mask R-CNN простий у навчанні і додає лише невеликі накладні витрати на швидший R-CNN, що працює зі швидкістю 5 кадрів в секунду [9].

Більше того, Mask R-CNN легко узагальнити до інших завдань, наприклад, дозволяючи оцінювати людські пози в тих же рамках. Mask R-CNN демонструє найкращі результати у всіх трьох доріжках набору завдань COCO (широкомасштабний датасет для сегментації даних), включаючи сегментацію екземплярів, виявлення об’єкта обмежувальної рамки та виявлення ключових точок людини (або об’єкту). Насправді Mask R-CNN перевершує всі існуючі одномодельні записи у кожному завданні, включаючи переможців виклику COCO 2016. Простий та ефективний підхід слугує надійною базовою лінією та допомагає полегшити майбутні дослідження у визнанні рівня екземплярів.

Існує два етапи маски RCNN. По-перше, він формує пропозиції щодо регіонів, де може бути об’єкт, на основі вхідного зображення. По-друге, він передбачає клас об'єкта, уточнює обмежувальну рамку та генерує маску на рівні пікселів об'єкта на основі пропозиції першого етапу. Обидва етапи пов'язані з структурою магістралі.

Магістраль - це глибока нейронна мережа стилю FPN. Він складається із шляху знизу вгору, шляху зверху вниз та бічних з'єднань. Шляхом знизу вгору може бути будь-який ConvNet, зазвичай ResNet або VGG, який витягує функції з

необроблених зображень. Шлях зверху вниз генерує карту пірамід, яка за розмірами схожа на шлях знизу вгору. Бічні зв’язки - це згортання та додавання операцій між двома відповідними рівнями двох шляхів. FPN перевершує інші одиночні ConvNet головним чином з тієї причини, що він підтримує сильні семантичні функції в різних масштабах роздільної здатності.

На рисунку 3 продемонстровано приклад роботи Mask R-CNN.

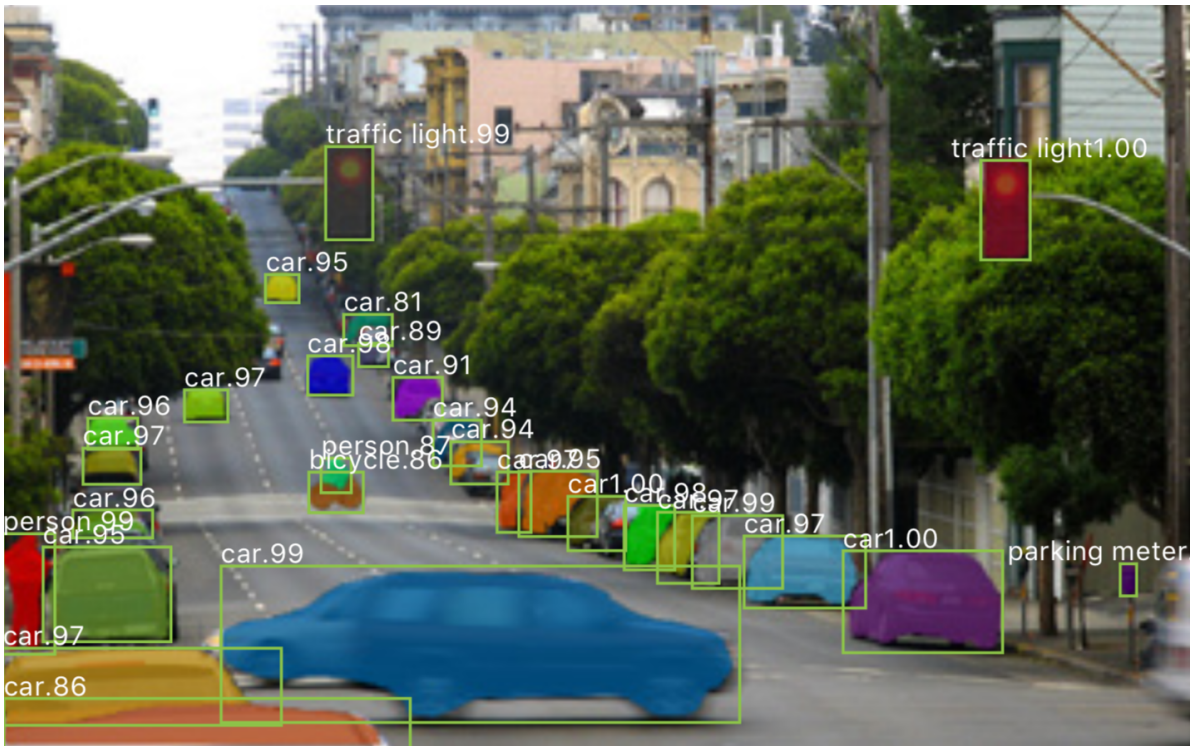

Рис. 3. Приклад роботи Mask R-CNN

Ідея **першого етапу** полягає в тому, що легка нейронна мережа під назвою RPN сканує весь шлях FPN зверху-вниз (далі - “карта об'єктів”) і пропонує регіони, які можуть містити об'єкти. Незважаючи на те, що

сканування карти об’єктів є ефективним способом, нам потрібен метод прив’язки об’єктів до його вихідного розташування зображення. Для розуміння процесу введемо термін якорів. **Якорі** - це набір зон із заздалегідь визначеними місцями та масштабами щодо зображень. Класи правдивості (Ground-truth classes) (на цьому етапі класифікуються лише двійкові файли об'єктів або фону) та обмежувальні рамки присвоюються окремим якорям відповідно до деякого значення IoU (Intersection over Union - перетин двух зон: очікуваної та отриманої при розпізнаванні) . Оскільки якорі з різними масштабами прив'язуються до різних рівнів карти об'єктів, RPN використовує ці якорі, щоб з'ясувати, де з карти об'єктів "повинен" потрапити об'єкт і який розмір його обмежувального поля. Тут ми можемо погодитись, що згортання, зменшення та збільшення вибірки дозволять зберегти об’єкти, що залишаються в тих самих відносних місцях, що й об’єкти на вихідному зображенні, і не буде їх псувати.

На **другому етапі** інша нейронна мережа приймає запропоновані регіони на першому етапі і призначає їх до кількох конкретних областей рівня карти об'єктів, сканує ці області та генерує класи об'єктів (багатокатегоріальні класифіковані), обмежуючі поля (bounding-boxes) та маски. Процедура схожа на RPN. Відмінності полягають у тому, що без допомоги якорів другий етап використовував фокус, який називається ROIAlign (операція з вилучення невеликої карти об’єктів з кожного IP у завданнях на основі виявлення та сегментації), щоб знайти відповідні ділянки карти об'єктів, і існує гілка, що генерує маски для кожного об'єкта на рівні пікселів. Робота маски R-CNN є завершеною.

Структура Mask R-CNN має наступний вигляд (рис. 4):

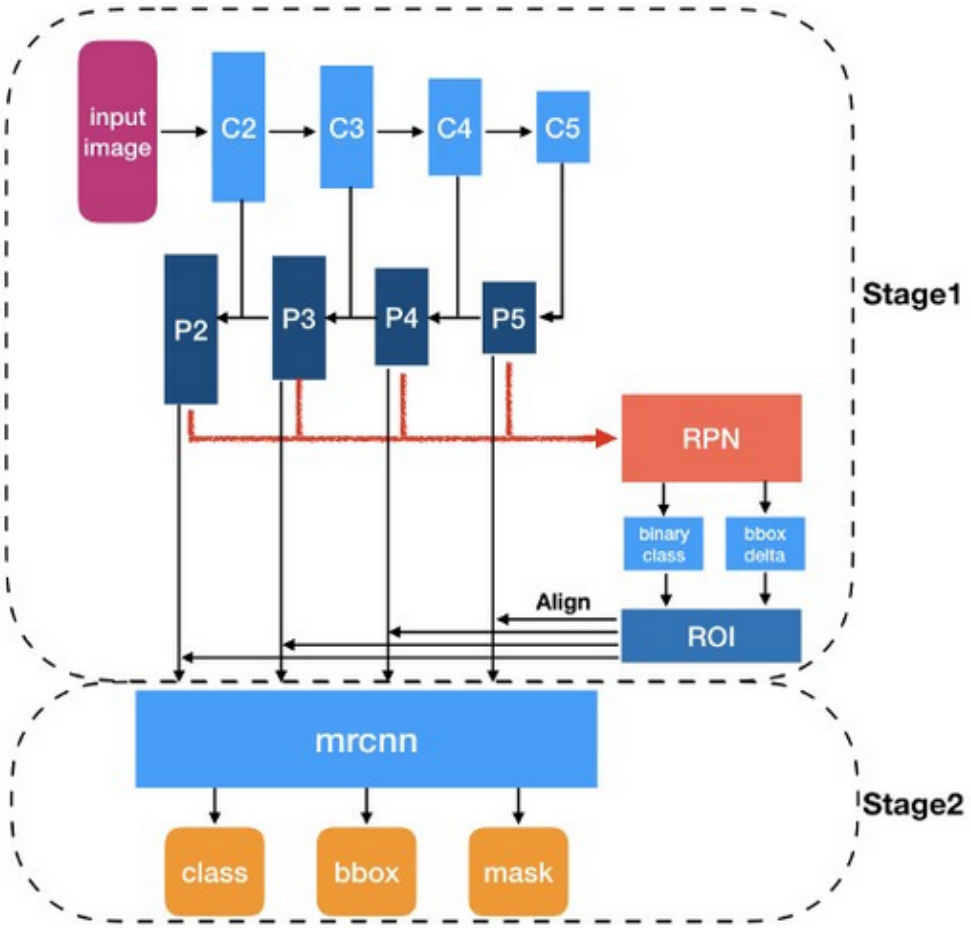

Рис. 4. Структура Mask R-CNN

РОЗДІЛ 2
МОЖЛИВОСТІ ПРОГРАМНОГО ПРОДУКТУ

2.1. Опис системи та постановка проблеми

Обрана назва системи – Modular Recognition – українською перекладається як “Модульне розпізнавання”. Суть системи полягає в зручній та швидкій класифікації об’єктів дослідження. Як вже згадувалось раніше, система неабияка гнучка, проте для демонстрації треба трохи конкретизувати задачу.

Проблема: отже, уявімо склад товарів, на який щодня надходить безліч одиниць чоловічого та жіночого взуття. Зрозуміло, що ці товари надходять від різних постачальників та від різних фірм взуття. Товари необхідно сортувати по фірмах на складі, та вміти швидко зрозуміти, якої фірми це взуття (один постачальник може привозити багато різних фірм взуття, що додає складнощів). А також - вміти автоматично зчитати інформацію про **розмір, колір** та **модель** взуття зі знайденої етикетки.

2.2. Запропонований розв’язок задачі

Створена система, що дозволяє, користуючись зручним інтерфейсом, посилати кожну пару взуття, що приходить на склад, на класифікацію за допомогою завчасно навченої моделі розпізнавання об’єктів, знаходження об’єкту на фотозображенні та відповідно класифікації до певної фірми взуття. (Кожна фірма взуття має свою оригінальну наліпку (без штрих-коду, бо якщо штрих-код присутній, то всю інформацію можна легко дістати з нього) за виглядом якої є можливість віднести пару взуття до тієї чи іншої фірми).

Розглянемо детальні кроки авторизованого користувача системи при класифікації однієї пари взуття. (для наочності до кожного кроку прикріплений скріншот вигляду системи):

1. На рисунку 5 показано можливість користувачем переглядати класи, на яких була навчена модель розпізнавання. Тут можна дізнатись назву певного класу взуття, його бренд (фірму) та автора, що створив відповідний клас.

   Відповідно класифікованими можуть бути лише класи, які внесені в систему, та на яких була навчена модель.

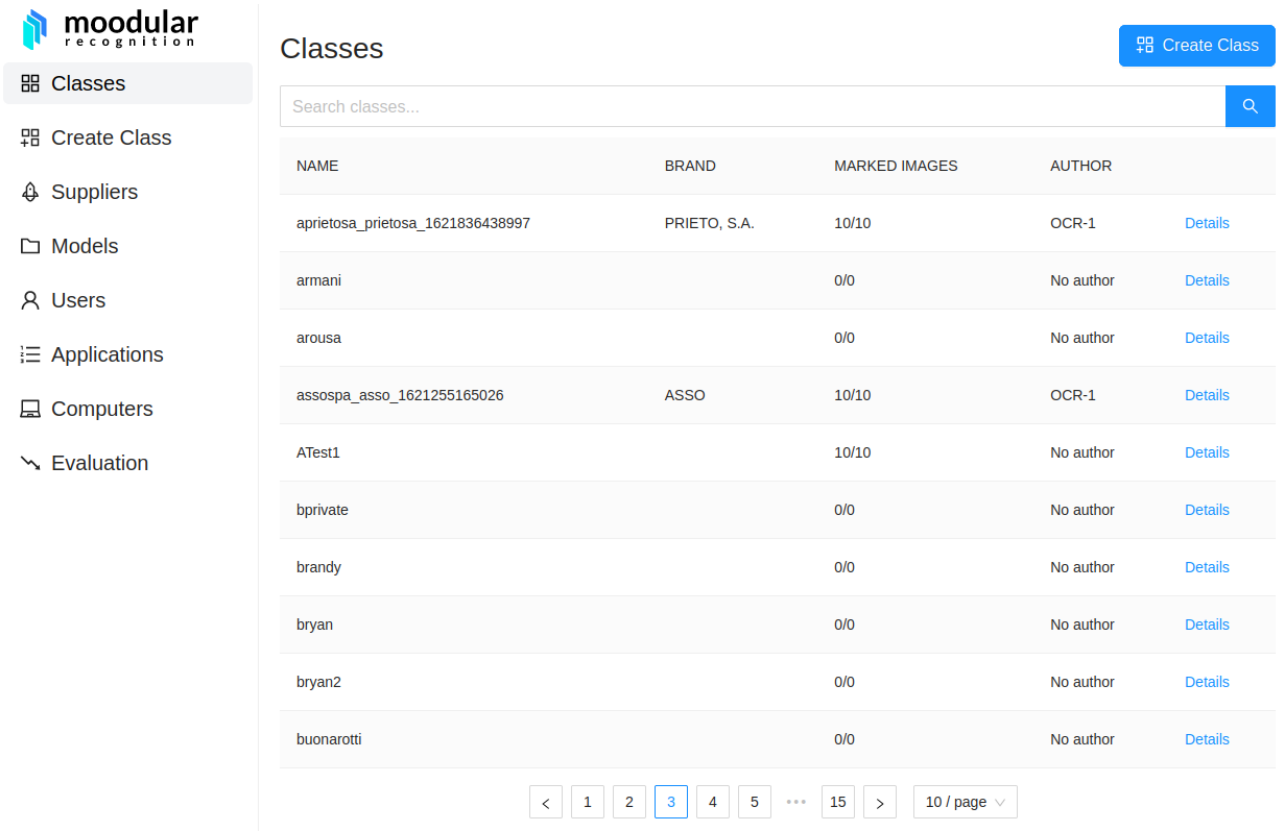

Рис. 5. Сторінка переліку класів навченої моделі

2. Після того, як користувач упевнився в наявності усіх потрібних класів взуття - він може переходити до процесу класифікації зображень, використовуючи зручний інтерфейс для внесення зображень, чи, навіть підключення камери комп’ютера до системи та прямої передачі даних з камери (рис. 6).

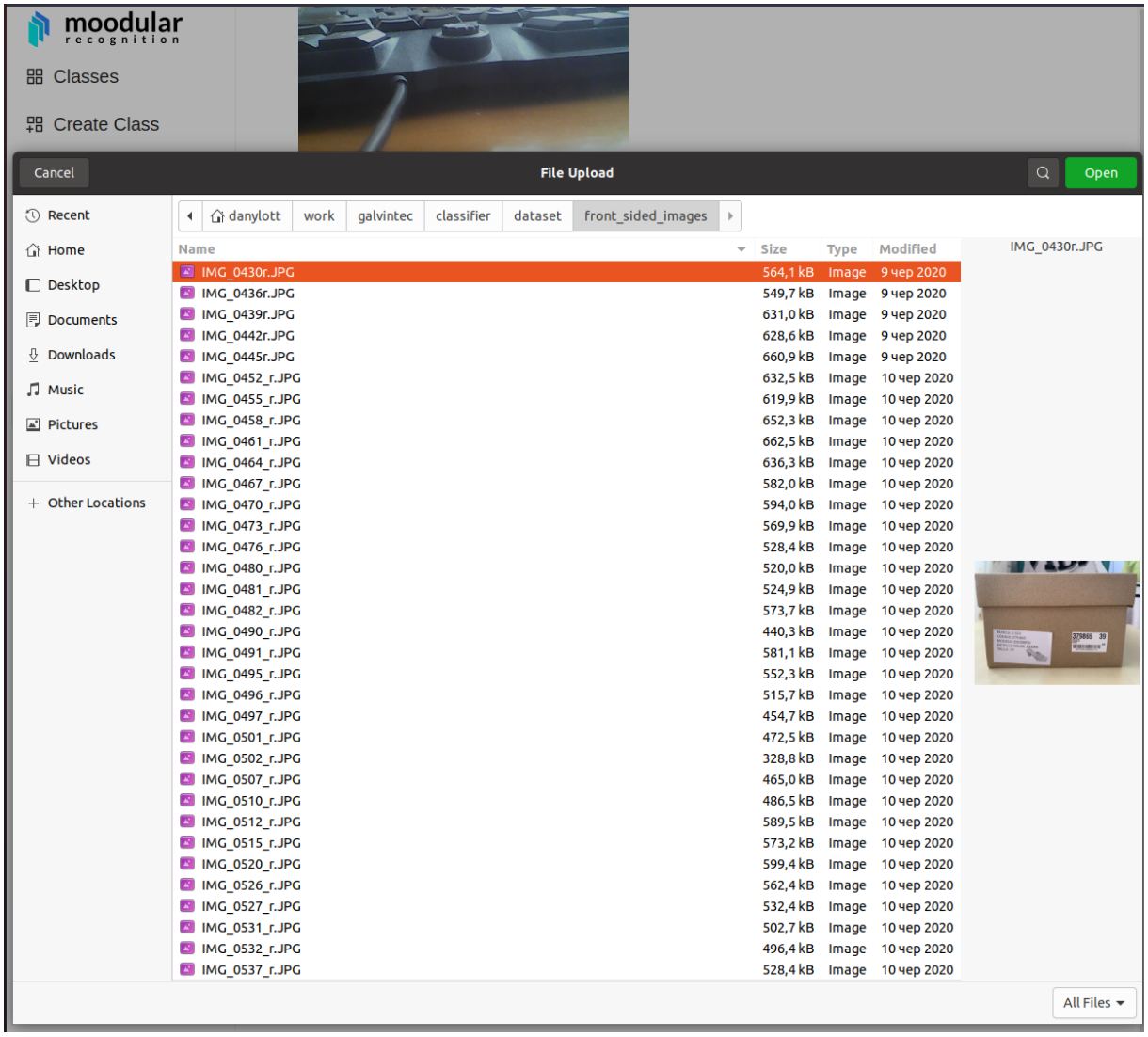

Рис. 6. Вибір зображення (вручну) для розпізнавання

На рисунку 7 зображений результат класифікації відповідно обраного зображення. Також додана можливість розпізнавання та зчитування певних потрібних значень з картинки - таких як розмір, колір та модель взуття за допомогою спеціальних розміток, що зберігаються для кожного класу в базі даних (зелені рамки на рисунку 7), та технології розпізнавання символів з картинки - OCR [10], а саме її реалізації за допомогою Tesseract [11] .

Також цікаво спостерігати за точністю “маски” (залиті чорним кольором краї фотозображення), що показує якість моделі розпізнавання.

Рис. 7. Результати класифікації зображень

3. На рисунку 8 показана можливість підключити камеру з приватного комп’ютеру та проводити класифікацію у реальному часі, натискаючи кнопку “Take Snapshot”, коли коробка стоїть перед камерою.

Рис. 8. Підключення до камери та результати класифікації

2.3. Повна роботи системи

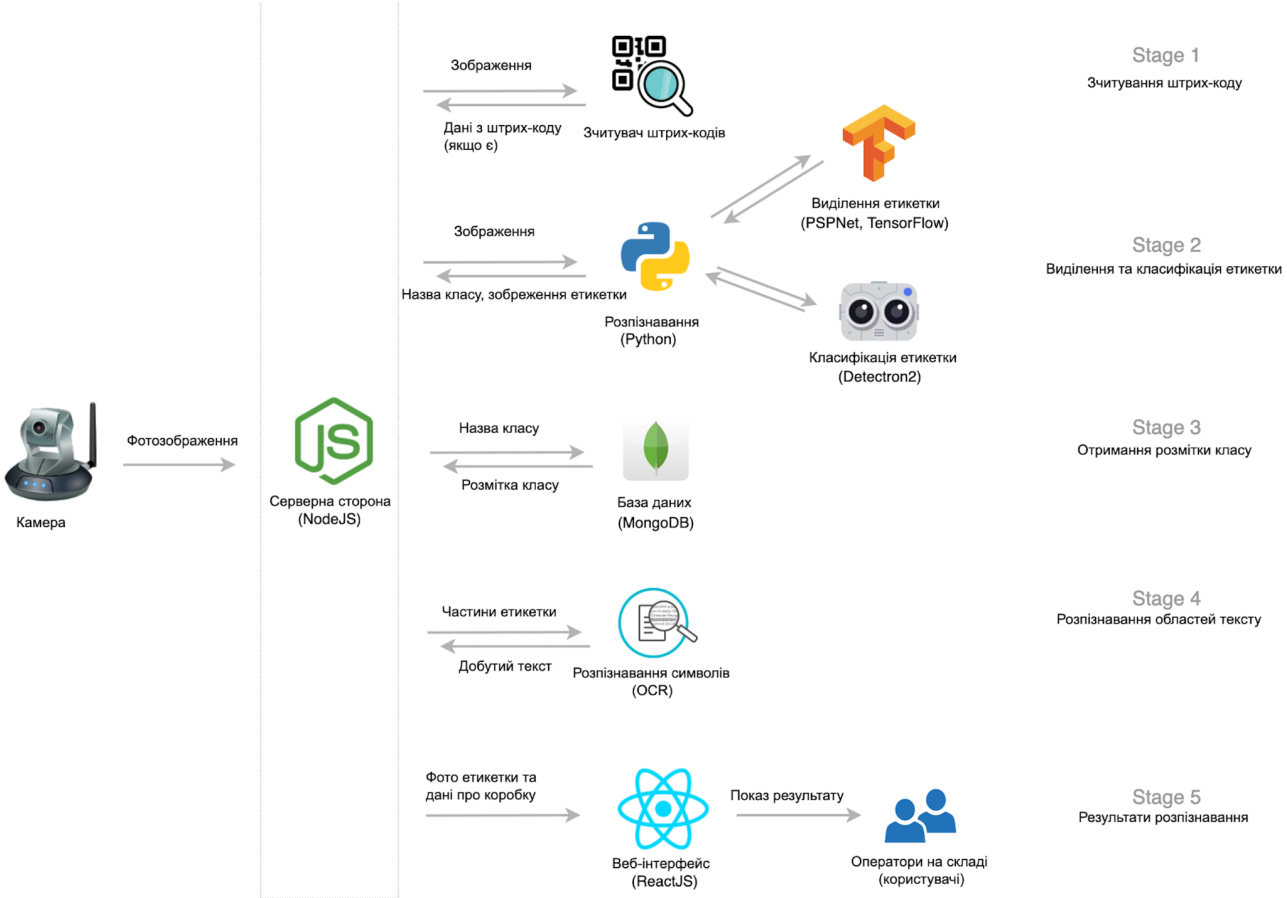

Рис. 9. Повна схема роботи системи

На рисунку 9 зображена повна схема роботи системи:

- отримання вхідних зображень взуття;
- навчання моделі на відповідні класи взуття для подальшого розпізнавання;
- класифікація етикетки на коробках;
- добування даних про колір, розмір та модель взуття;
- збереження інформації до бази даних складу чи магазину.

РОЗДІЛ 3
ОСОБЛИВОСТІ РОЗРОБКИ

3.1.Розробка моделі розпізнавання

Для розробки моделі розпізнавання було вирішено обрати платформу **Google Colab.** Colaboratory, або скорочено “Colab” - це продукт Google Research. Colab дозволяє будь-кому писати та виконувати довільний код python через браузер, і особливо добре підходить для машинного навчання, аналізу даних та навчання. Більш технічно, Colab - це розміщена послуга ноутбуків Jupyter, яка не потребує налаштування, одночасно надаючи безкоштовний доступ до обчислювальних ресурсів, включаючи графічні процесори. Саме на цій платформі була навчена модель класифікації коробок із взуттям.

[09/07 12:30:51 d2.data.build]: Distribution of instances among all 88 categories:

category	#instances	category	#instances	category	#instances
mare	3	arousa	5	armani	3
brandy	3	bryan	3	clarks	8
core	8	courtset	3	dacota	3
dilceida	3	emmshu	3	fluchos	3
fredperry	3	geox	8	gg	3
harmoni	3	heritage	5	krackcore	7
krackcore2	3	krackkids	3	levis_kids	3
lol	3	love	3	newbalance	3
nike	4	nike_sb	3	pepejeans	3
positivity	3	sacut	3	sandrafontan	3
sotavento	8	stonefly	3	unisa	3
vans	3	velilla	7	violeta	3
walk	4	krackcore6	6	angel_infan..	3
bprivate	3	bryan2	3	buonarotti	3
coolway	3	cossimo2	3	dulceida2	3
isteria	3	jenker	3	krack_by_ied	3
krackcore7	3	krack_harmony	3	krack_harmo..	3
krackcore3	3	krackcore4	6	krackcore5	3
krackcore8	3	krackcore9	3	krackkids2	3
krackkids3	3	krackkids4	5	milatrend	3
nice	6	roberto_tor..	4	roberto_tor..	3
roberto_tor..	3	rocio_camacho	3	sandra_font..	3
sandra_font..	3	sandrafontan2	3	viguera	3
wonders	3	sandra_font..	3	nice2	3
harmony2	7	krack_herit..	9	krack_herit..	3
sandra_font..	3	krackcore10	5	dulceida3	3
krack_herit..	3	krack398	3	mou	3
mou2	3	krackcore11	3	krackcore12	4
krackcore13	3	krackcore14	3	roberto_tor..	3
rocio_camac..	3				
total	323				

Рис. 10. перелік усіх класів (88 класів) навченої моделі

3.2. Перевірка якості навченої моделі

Після навчання моделі, було перевірено кожен клас на якість розпізнавання та враховано похибку розпізнавання контурів кожної етикетки (рис. 11).

Накладаючи маску розпізнаної етикетки на зображення, дуже легко оцінити якість розпізнавання моделі. А відсоткове значення вказує, наскільки модель впевнена у тому чи іншому розпізнаванні.

Рис. 11. Перевірка якості навченої моделі

Оцінка моделі (% від ідеального результату):

- Точність побудови маски етикетки: 91%;
- Точність класифікації: 98%;
- Точність знаходження прямокутника з етикеткою: 93%.

Загальний час навчання моделі: 4 год.

Кількість вхідних зображень: 323.

3.3. Реалізація інтерфейсу взаємодії з моделлю

Для реалізації інтерфейсу користувача була використана мова **Javascript**, а саме **React.js (front-end)** для інтерфейсу, а **Node.js (back-end)** для роботи з даними.

React - це найпопулярніша та гнучка бібліотека JavaScript для швидкої та якісної розробки інтерфейсу додатків. Ця бібліотека популярна тим, що використовує так званий “компонентний” підхід, тобто невеликі незалежні частини коду, кожен з яких відповідає за ту чи іншу сторінку користувацького інтерфейсу.

Node.js - це платформа, побудована на середовищі виконання Chrome що дозволяє працювати із мовою JavaScript не тільки на стороні браузеру, а ще й із серверного боку. Node.js використовують для розробки логіки додатку, та також підключення до моделей бази даних, тобто щоб мати можливість передати необхідну інформацію від сховища даних до інтерфейсу користувача.

Для можливості добування інформації про колір, розмір та модель взуття для кожного класу передбачена можливість виділити необхідні регіони з інформацією (рис. 12)

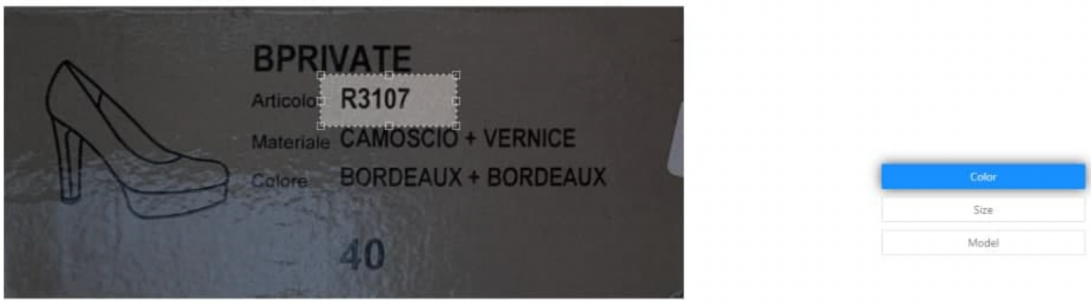

Рис. 12. Виділення регіонів з інформацією про етикетку

Для забезпечення можливості подальшого навчання на нових коробках (донавчання старої моделі, або навчання нових) реалізований функціонал нанесення маски (мануальне виділення етикетки) на кожне зображення (рис. 13).

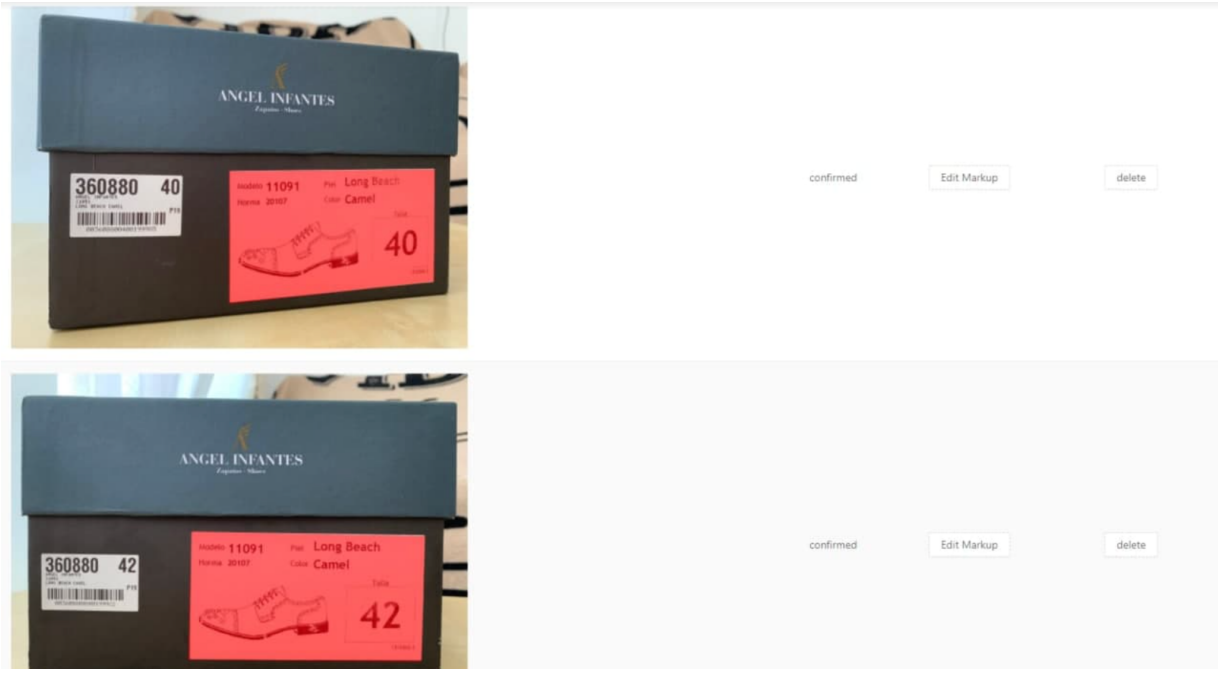

Рис. 13. Нанесені розміки етикеток на коробки

3.4. База даних

Для зберігання інформації про класи, фотозображення, що до них належать, користувачів системи та результати розпізнавання використовувалась база даних **MongoDB.**

MongoDB - це документо-орієнтована NoSQL база даних, яка використовується для великого обсягу зберігання даних. Всупереч стандартному (SQL) підходу - NoSQL база даних MongoDB використовує замість таблиць і рядків - колекції (набори документів) та документи (пари ключ-значення в яких зберігаються усі необхідні дані).

ВИСНОВКИ

Було виконано такі завдання:

- визначено об’єм даних (фотозображень) для обробки;
- підготовлено вхідний набір зображень для навчання моделі;
- оцінено якість навчання та можливі похибки моделі;
- спроектовано та розроблено інтерфейс користувача для зручного використання класифікації зображень;
- розпізнавання зон із текстом.

Поставлена мета є досягнутою. Надалі можливо додати:

- групування моделей та підтримка багатьох моделей одночасно;
- автоматичне відслідковування появи нової коробки перед камерою.

Даний продукт може бути використаний будь-якою компанією для оптимізації процесу прийому товару.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Imambi, Sagar & Kolla, Bhanu & Kanagachidambaresan, G.. (2021). PyTorch. 10.1007/978-3-030-57077-4_10;
2. Vung, Pham & Pham, Minh Chau & Dang, Tommy. (2020). Road Damage Detection and Classification with Detectron2 and Faster R-CNN. 5592-5601. 10.1109/BigData50022.2020.9378027;
3. Schell, Uli. (2022). Keras/Tensorflow. 10.3139/9783446472440.010.
4. Yang, Qiong & Yu, Lifeng. (2021). Recognition of Taxi Violations Based on Semantic Segmentation of PSPNet and Improved YOLOv3. Scientific Programming. 2021. 1-13. 10.1155/2021/4520190.
5. Wolf, Steffen. (2020). Machine Learning for Instance Segmentation;
6. Chaudhary, Anmol & Chouhan, Kuldeep & Gajrani, Jyoti & Sharma, Bhavna. (2020). Deep Learning With PyTorch. 10.4018/978-1-7998-3095-5.ch003;
7. Purves, Dale. (2021). Neural Networks. 10.1007/978-3-030-71064-4_5;
8. Bosc, Tom. (2016). Learning to Learn Neural Networks;
9. Zhang, Wenchao & Fu, Chong & Xie, Haoyu & Zhu, Mai & Tie, Ming & Chen, Junxin. (2021). Global context aware RCNN for object detection. Neural Computing and Applications. 1-13. 10.1007/s00521-021-05867-1;
10. Pathak, Kavita & Saraf, Shreya & Wagh, Sayali & Vishwanath, Dr. (2022). OCR Studymate. International Journal for Research in Applied Science and Engineering Technology. 10. 2241-2246. 10.22214/ijraset.2022.41103.
11. Rodríguez Leal, Carlos. (2020). Tesseract. 10.13140/RG.2.2.22886.86088.