МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
СУМСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ

Кафедра прикладної математики та моделювання складних систем

Допущено до захисту
Завідувач кафедри ПМ та МСС
_____ Коплик І.В.
(підпис)
«___» _____2022 р.

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня «магістр»
спеціальність 113 «Прикладна математика»
освітньо-професійна програма «Наука про дані та моделювання складних систем»

тема роботи: «**Моделювання розпізнавання номерного знаку автомобіля за відео з веб-камери з використанням технологій комп'ютерного зору**»

Виконавець
Шапошніков Владислав Олександрович
_____
(підпис)

Науковий керівник
док. фіз.-мат. наук, професор
Лисенко Олександр Володимирович _____
(підпис)

Суми – 2022

РЕФЕРАТ

Кваліфікаційна робота: 41 с., 24 рис,12 джерел.

Мета роботи: Створення системи для моделювання розпізнавання номерного знаку автомобіля за відео з веб-камери з використанням технологій комп’ютерного зору.

Об’єкт дослідження: процес розпізнавання номерного знаку автомобіля за відео з веб-камери з використанням технологій комп’ютерного зору.

Предмет дослідження: характеристики розпізнавання номерного знаку автомобіля за відео з веб-камери з використанням технологій комп’ютерного зору.

У роботі проведено огляд технологій комп’ютерного зору, принципів та методів, на основі яких розробляють системи для розпізнавання номерного знаку автомобіля. Було створено систему розпізнавання номерного знаку автомобіля за відео з веб-камери. Для перевірки роботи систему було використано зображення з датасету Autoria.UA та власні відеозаписи з ip камер автомийки Hose. Фактично використовувався набір фотографій та кадрів з відеозаписів різної роздільної здатності та якості. Створено дві версії системи розпізнавання номерних знаків автомобіля. Версія 1.0 використовує контурний аналіз для локалізації номерних знаків та easyOCR для розпізнавання тексту. Виявилась досить повільною у роботі та не завжди розпізнавала номера. Версія 2.0 використовує моделі комп’ютерного зору YOLOv5. Швидкість та якість цієї системи значно вища.

Ключові слова: КОМПЬ’ЮТЕРНИЙ ЗІР,НЕЙРОМЕРЕЖІ, КОНТУРНИЙ АНАЛІЗ, YOLOV5

ЗМІСТ

ВСТУП

Завдання розпізнавання автомобільних номерних знаків користується попитом у різних системах контролю транспортних засобів. Такі системи використовують технології комп'ютерного зору. Використовувати системи розпізнавання номерного знаку автомобіля можна для різних цілей:

- на дорогах для виявлення порушників правил дорожнього руху (ПДР);
- на автомобільній мийці;
- відстеження транспорту у розшуку;
- обмеження доступу на території під охороною;
- організація платного доступу для автомобілів, наприклад, можна налаштувати автоматичний процес оплати;
- та багато інших цілей.

Проте для різних задач потрібна різна швидкість роботи системи. Наприклад, на дорогах для виявлення порушників правил дорожнього руху (ПДР) вимоги до швидкості роботи таких систем невисокі: час від порушення ПДР до сповіщення про штраф може бути доба і більше, у випадку роботи аналогічної системи, наприклад, на автомобільній мийці, такий час зменшується залежно від задач від хвилини до десятків хвилин. Також треба виділити і інші проблеми які спостерігаються у наявних системах:

- об'єкти можуть хаотично переміщатися або виїжджати з поля зору камери, тому необхідно з'єднати їх з об'єктами, які раніше були на відео;
- може спостерігатися розмиття зображення якщо об'єкт рухався дуже швидко;
- об'єкти можуть виглядати різними з різних точок, тому необхідно послідовно ідентифікувати один і той самий об'єкт з усіх точок;

Тому розробка таких систем з високою швидкістю роботи та з виправленням існуючих проблем є задачею важливою та актуальною.

Мета роботи: Створення системи для моделювання розпізнавання номерного знаку автомобіля за відео з веб-камери з використанням технологій комп’ютерного зору.

Об’єкт дослідження: процес розпізнавання номерного знаку автомобіля за відео з веб-камери з використанням технологій комп’ютерного зору.

Предмет дослідження: характеристики розпізнавання номерного знаку автомобіля за відео з веб-камери з використанням технологій комп’ютерного зору.

РОЗДІЛ 1
ТЕХНОЛОГІЇ КОМП'ЮТЕРНОГО ЗОРУ, ПРИНЦИПИ ТА МЕТОДИ (ОГЛЯД ЛІТЕРАТУРИ)

1.1. Технології комп'ютерного зору

1.1.1. Поняття комп'ютерного зору

Комп'ютерне бачення або зір – технологія для створення систем, які можуть відстежувати та визначати об'єкти(вони отримують інформацію у вигляді зображень).

Сьогодні комп'ютерний зір є дуже актуальною науковою областю, яка вирішує задачі для здобуття високого рівня розуміння цифрових зображень [1]. Ці задачі включають методи збору, обробки, аналізу та розуміння цифрових зображень. Існує 4 типи таких задач[2]:

- Classification - класифікує зображення за типом об'єкта, який воно містить;
- Semantic segmentation – на зображенні яке містить об'єкт одного класу позначає кожен піксель;
- Object detection – виявлення об'єктів потрібного класу та створення обмежувального поля навколо кожного з них;
- Instance segmentation – позначає пікселі на об'єктах кожного класу.

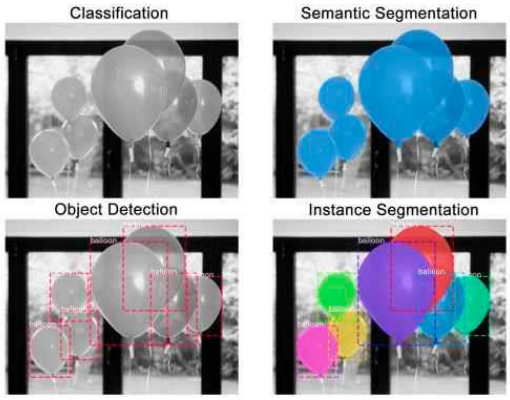

Рисунок 1- ілюстрація типів комп'ютерного зору

Окрім використання для розпізнавання номерних існує дуже багато варіантів використання комп'ютерного зору. Важливе місце комп'ютерний зір займає в медицині. Його використовують для обробки відеоданих щоб визначити діагноз пацієнта. Дані отримую за допомогою мікроскопу, рентгену, УЗД, та ін. Інформацією яка може бути отримана з таких відеоданих є виявлення пухлин або інших злоякісних змін.

Також комп'ютерний зір використовують в промисловості. Тут інформація допомогає підтримувати виробничий процес. Наприклад, коли треба провести контроль якості деталей.

Дуже часто комп'ютерний зір використовують у військовій галузі. Найчастіше це виявлення ворога або керування ракетами.

Одним із нових галузей застосування комп'ютерного зору є автономні транспортні засоби. Системи які базуються на комп'ютерному баченні допомогають водію у керуванні. Також існують повністю автономні транспортні засоби.[3]

1.1.2. Як працює комп’ютерний зір

Місія комп'ютерного зору – навчити обчислювальну машину бачити та розуміти оточення за допомогою цифрових фотографій та відеозаписів. Для досягнення цієї мети використовуються три компоненти:

- отримання зображень;
- обробка інформації;
- аналіз даних.

Для отримання зображення використовують веб-камери, цифрові та дзеркальні фотоапарати, а також професійні 3D-камери та лазерні далекоміри.

Далі проводиться низькорівнева обробка зображень для визначення країв, точок і сегментів зображення, які являють собою прості геометричні фігури.

Популярними методами низькорівневого аналізу є:

- виділення кордонів або edge detection;
- сегментація;
- класифікація та виявлення об'єктів.

Виділення кордонів передбачає різноманітність математичних методів, за допомогою яких ідентифікуються точки у зображеннях. Алгоритм аналізує малюнок і переводить його в набір вигнутих відрізків та ліній. Цей метод використовується для виділення найважливіших частин зображення, що дозволяє зменшити кількість даних, що обробляються.

Рисунок 2- приклад зображення обробленого за допомогою методу виділення кордонів

Сегментація зазвичай використовується для визначення розташування об'єктів та меж на зображеннях. У процесі обробки алгоритм надає мітку кожному пікселю, щоб у подальшому їх можна було поєднати за певними характеристиками.

В результаті виходить набір сегментів, що охоплюють усі частини зображення або витягнуті з нього контури.

Рисунок 3- приклад сегментації зображення

Класифікація надає нам інформацію про зміст зображень. Класифікація зображень є основою іншого, складнішого алгоритму в комп'ютерному зорі — виявлення об'єктів. Це дозволяє, наприклад, відрізнити одному зображенні людину від машини.

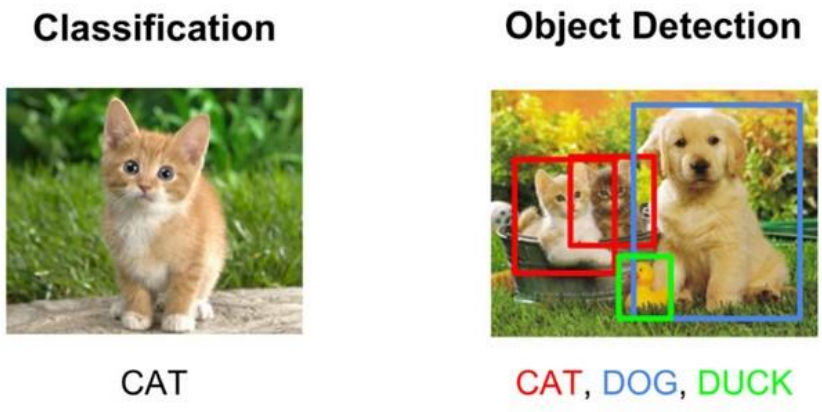

Рисунок 4- приклад алгоритму класифікації

Аналіз та розуміння зображень – це останній крок у комп'ютерному зорі, що дозволяє машинам приймати власні рішення. На цьому етапі використовують високорівневі дані, отримані з попереднього кроку. Прикладом високорівневого аналізу може бути відображення тривимірної сцени, розпізнавання чи відстеження об'єктів.[4]

1.2. Принципи та методи пошуку

1.2.1. Контурний аналіз – пошук прямокутного контуру

Недоліком цього методу є те, що він працює тільки при наявності добре читабельного контуру з високою роздільною здатністю і рівною границею а також при відсутності перешкод між камерою і номерним знаком.

Суть контурного аналізу: фільтрується зображення для знаходження кордонів, потім проводиться виділення всіх знайдених контурів та їх аналіз. Через залежність від якості контуру даний метод є непрактичний, оскільки контур номерного знаку може бути нечітким.

1.2.2. Аналіз частини кордонів

Більш стабільним методом буде аналіз лише частини рамки номерного знаку. Спочатку виділяються контури, після чого є всі вертикальні прямі. «Для будь-яких двох прямих, розташованих недалеко один від одного, з невеликим зрушенням по осі Y, з правильним відношенням відстані між ними до їх тривалості, розглядається гіпотеза того, що номер розташовується між ними» [5].

1.2.3. Метод Віоли-Джонса

Метод Віоли-Джонса- алгоритм виявлення об'єктів на зображенні який специфікується на розпізнаванні обличчя але його можна також використовувати і для інших об'єктів [6].

Метод можна розбити на дві частини: алгоритм навчання та алгоритм розпізнавання.

Принципи, на яких базується метод:

- інтегральне уявлення зображень, що значно прискорює обчислення необхідних об'єктів;
- використовуються ознаки Хаара, для пошуку кутових точок;
- побудова класифікатора з урахуванням алгоритму бустингу (від англ.boost – покращення, посилення) для визначення найбільш підходящих ознак для об'єкта, що шукається, на даній частині зображення;
- всі ознаки надходять на вхід класифікатора, який дає результат «правильно» чи «брехня»;
- для швидкого відкидання незначних вікон, без об'єкта, використовуються каскади ознак.

Недоліками цього методу є довгий час навчання а також використання великої кількості даних для навчання класифікатора. Також якщо кут нахилу більше 30 градусів точність розпізнавання сильно падає. Окрім цього, через занадто сильне освітлення або затемнення алгоритм може не дати результату.

Проте цей метод має також і багато переваг:

- виявлення більш як одного об'єкта;
- використання простих класифікаторів;
- висока швидкість пошуку;

- можливість використання у відеопотоці.

1.2.4. K-nearest

Дуже простий але результативний метод.

Принцип методу:

a) Зображення символів сортується по класам та записується в базу;

b) Між ними вводиться міра відстані;

c) В процесі розпізнавання по черзі розраховується дистанція між обраним символом і всіма символами в базі. Серед к найближчих сусідів можуть зустрічатися представники різних класів, тоді символ отримує такий самий клас як у більшості символів поблизу;

Проблемою цього методу є необхідність швидкого розрахунку дистанції між даними(переведення їх до бінарного типу і використання XOR). Бінаризація непередбачуваний процес і у випадку якщо номерний знак погано забруднений або потертий можуть виникнути проблеми. Проте цю проблему можна вирішити при наявності великої бази символів, зафіксованих у різних умовах.

Головною перевагою цього методу є його простота, тому налагодження алгоритму на отримання потрібного результату проводиться без затруднень.[7]

1.2.5. Кореляційний метод

Принципом роботи цього методу є формування для кожного класу об'єктів в декартовій площині ознак деякої «еталонної області». Будь який розпізнаний об'єкт буде відноситись до класу що відповідає найближчій «еталонної області»(область формується за допомогою допустимих перетворень еталонних векторів класу).

Для розпізнавання застосовуються операції розрахунку коваріації вхідного сигналу з гіпотетичним.

Такі методи ще називають «зіставлення шаблонів», бо суть цих методів у тому, щоб обрати деякий зразок і потім порівнювати вхідні зображення з цим зразком. У разі невизначеності щодо параметрів, застосовуються адаптивні підходи або перебирають усі можливі варіанти.

Переваги методу:

- Передбачуваний і добре вивчений результат, якщо шум хоч трохи відповідає обраній моделі
- Можливість розпізнати погано читаємий шрифт

Недоліки:

- Великі витрати

1.2.6. Адаптивне розпізнавання

Можливі 2 підходи:

1) Шрифтовий підхід

Програма аналізує характеристики шрифту та заносить їх у свою базу еталонних характеристик[8]. Після цього програма зможе розпізнати конкретний шрифт. Таким чином, для роботи програми у різних областях треба провести навчання для потрібних шрифтів[9].

Недоліки:

- Алгоритму необхідно мати дані про потрібний шрифт;
- Необхідні налаштування на конкретний шрифт;

Переваги:

- Якщо є детальна інформація про символи що використовуються можливо побудувати досить точний алгоритм розпізнавання;

2) Безшрифтовий підхід

Алгоритми аналізують різні ознаки букв будь-якого шрифту і розміру.

Недоліки:

- Значно нижча точність в порівнянні з шрифтовим підходом;
- Низький коефіцієнт надійності розпізнавання;

Переваги:

- Робота алгоритму на відміну від шрифтового методу не залежить від характеристик та типу шрифту;
- Простота навчання, можливість автоматичного навчання;

1.2.7. Mask R-CNN

Щоб зрозуміти Mask R-CNN, давайте спочатку обговоримо архітектуру Faster R-CNN, яка працює в два етапи:

1. Перший етап складається з двох мереж, магістральної (ResNet, VGG, Inception тощо) та регіональної мережі. Ці мережі запускаються один раз для зображення, щоб надати набір пропозицій регіону. Пропозиції регіонів – це регіони на карті об’єктів, які містять об’єкт.
2. На другому етапі мережа передбачає обмежувальні прямокутники та клас об'єкта для кожної запропонованої області, отриманої на етапі 1. Кожна запропонована область може мати різний розмір, тоді як повністю зв'язані рівні в мережах завжди потребують вектора фіксованого розміру для прогнозування. Розмір цих запропонованих регіонів фіксується за допомогою або RoI pool (який дуже схожий на MaxPooling), або методу RoIAlign.

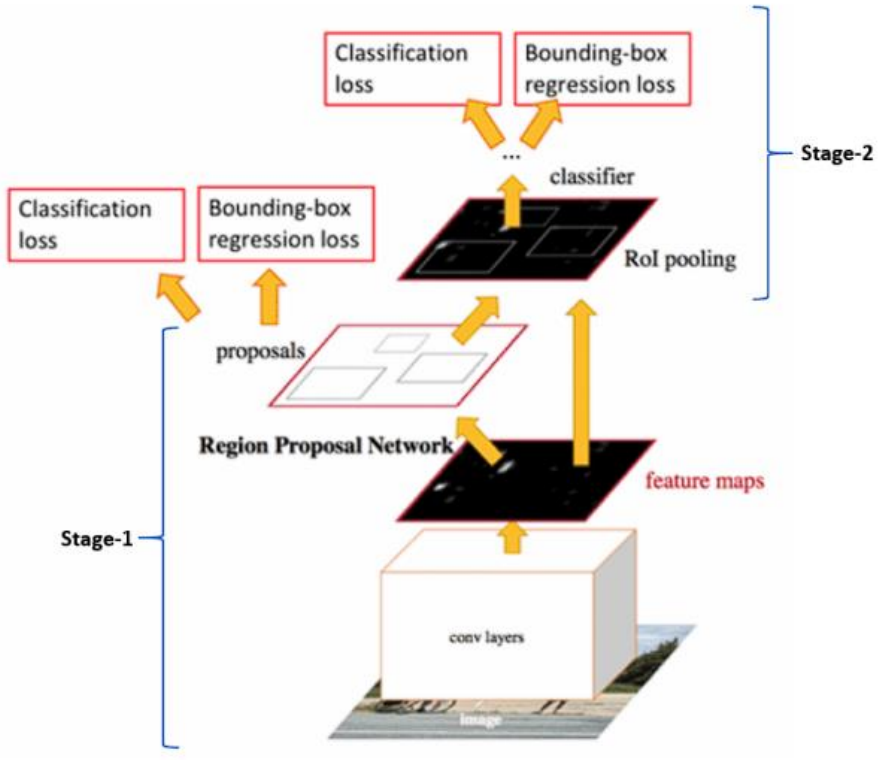

Рисунок 5-архітектура faster R-CNN

Mask R-CNN є розширенням Faster R-CNN з додатковою гілкою для прогнозування масок сегментації (RoI).

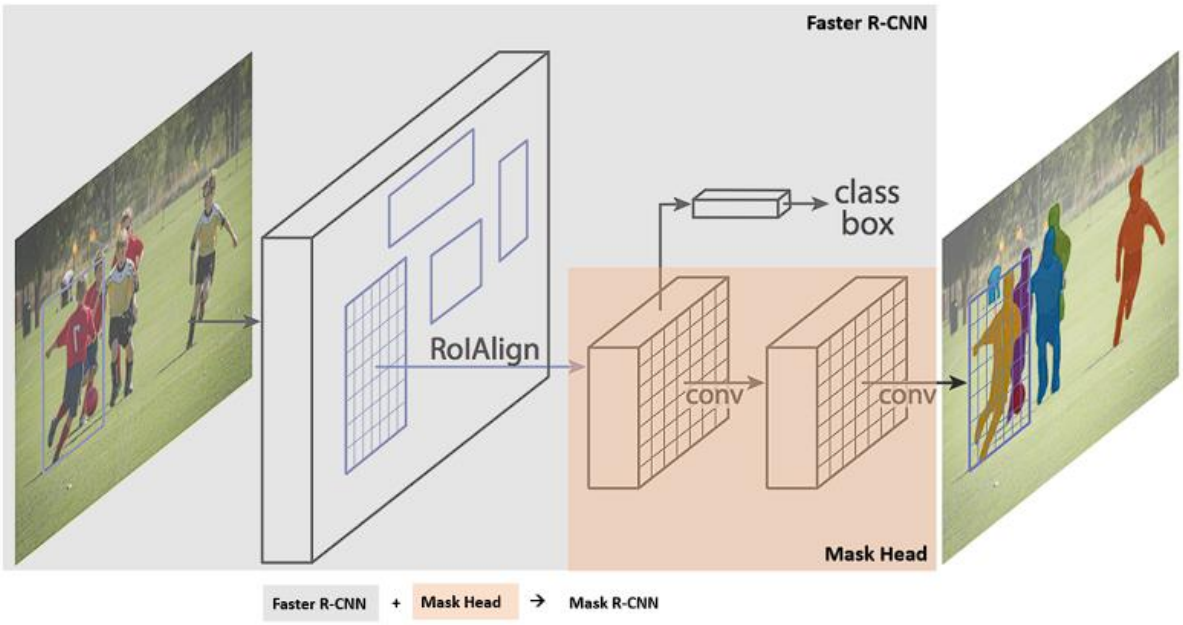

Рисунок 6- структура mask R-CNN

Виділення маски відбувається в class-agnostic стилі: маски визначаються окремо для кожного класу, а потім завдяки незалежного класифікатора обирається маска класу, який переміг. Стверджується, що такий підхід ефективніший, ніж той, хто спирається на апріорне знання класу.

1.2.8. Нейромережі

Різні типи нейронних мереж використовуються для різних цілей, наприклад, для передбачення послідовності слів ми використовуємо рекурентні нейронні мережі, точніше LSTM, так само для класифікації зображень ми використовуємо згорточні нейронні мережі. Перш ніж перейти до різних типів нейронних мереж, спочатку розглянемо структуру звичайної нейромережі. У звичайній нейронній мережі є три типи шарів:

1. Вхідні шари: це шар, на якому ми надаємо вхідні дані для нашої моделі. Кількість нейронів у цьому шарі дорівнює загальній кількості ознак у наших даних (кількості пікселів у випадку зображення).

2. Прихований шар: вхідні дані з вхідного шару потім подаються до прихованого шару. Може бути багато прихованих шарів залежно від нашої моделі та розміру даних. Кожен прихований шар може мати різну кількість нейронів, яка, як правило, перевищує кількість функцій. Вихідні дані кожного шару обчислюються шляхом множення матриці вихідних даних попереднього шару на вагові коефіцієнти цього шару, які можна дізнатися, а потім шляхом додавання зміщень, які можна дізнатися, з наступною функцією активації, яка робить мережу нелінійною.

3. Вихідний рівень: вихідні дані з прихованого шару потім подаються в логістичну функцію, наприклад sigmoid або softmax, яка перетворює вихідні дані кожного класу в оцінку ймовірності кожного класу.

Потім дані вводяться в модель і виходять результати з кожного шару. Цей крок називається прямою передачею, потім ми обчислюємо похибку за

допомогою функції похибок, деякими поширеними функціями похибок є крос-ентропія, похибка квадратичних втрат тощо. Після цього ми зворотне поширення в модель шляхом обчислення похідних. Цей крок називається зворотним поширенням, який в основному використовується для мінімізації втрат. Але я хочу розглянути детальніше варіанти нейромереж які я використовував в роботі, а саме: згорткові та рекурентні нейромережі.

Ось базовий код Python для нейронної мережі з випадковими входами та двома прихованими шарами.

```
activation = lambda x: 1.0/(1.0 + np.exp(-x)) # sigmoid function
input = np.random.randn(3, 1)
hidden_1 = activation(np.dot(W1, input) + b1)
hidden_2 = activation(np.dot(W2, hidden_1) + b2)
output = np.dot(W3, hidden_2) + b3
```

Згорткові нейромережі

Згорткові нейромережі або ковнети — це нейронні мережі, які мають спільні параметри. Уявіть, що у вас є образ. Його можна представити у вигляді кубоїда, який має свою довжину, ширину (розмір зображення) і висоту (оскільки зображення зазвичай мають червоні, зелені та сині канали).

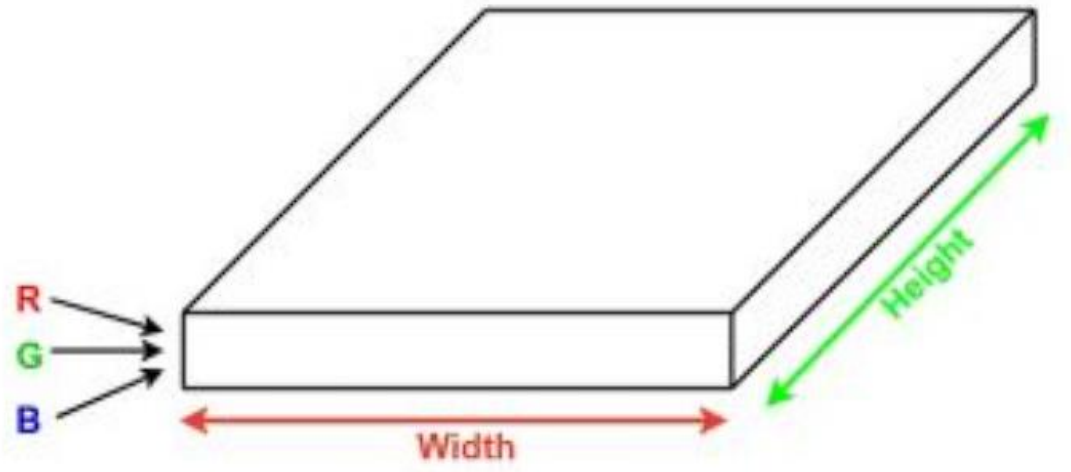

Рисунок 7- згорткова нейромережа

Тепер уявіть, що ви берете невелику ділянку цього зображення і запускаєте на ньому невелику нейронну мережу, скажімо, з k виходами та представляєте їх вертикально. Тепер проведіть цією нейронною мережею по всьому зображенню, в результаті ми отримаємо інше зображення з іншою шириною, висотою та глибиною. Замість лише каналів R, G та B тепер у нас більше каналів, але менша ширина та висота. Ця операція називається згорткою. Якщо розмір патча такий самий, як і зображення, це буде звичайна нейронна мережа. Через цей маленький фрагмент у нас менше ваги.

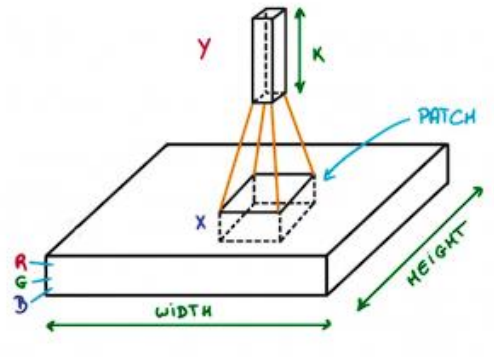

Рисунок 8-згорткова нейромережа

Тепер давайте трохи поговоримо про математику, яка бере участь у всьому процесі згортки.

- Шари згортки складаються з набору фільтрів, які можна вивчати (патч на зображенні вище). Кожен фільтр має малу ширину та висоту та таку саму глибину, як і вхідний об’єм (3, якщо вхідний шар є вхідним зображенням).
- Наприклад, якщо нам потрібно виконати згортку на зображенні розміром 34x34x3. Можливий розмір фільтрів може бути axax3, де «a» може бути 3, 5, 7 тощо, але малий порівняно з розміром зображення.
- Під час прямого проходу ми крок за кроком ковзаємо кожним фільтром по всьому вхідному об’єму, де кожен крок називається кроком (який може мати значення 2, 3 або навіть 4 для багатовимірних зображень) і

обчислюємо скалярний добуток між вагами фільтрів і патча від вхідного обсягу.

- Коли ми пересуваємо наші фільтри, ми отримаємо 2-D вихід для кожного фільтра, і ми складемо їх разом, і в результаті ми отримаємо вихідний об’єм, глибина якого дорівнює кількості фільтрів. Мережа дізнається всі фільтри.

Рекурентні нейронні мережі

Рекурентна нейронна мережа (RNN) — це тип нейронної мережі , де результати попереднього кроку подаються як вхідні дані для поточного кроку . У традиційних нейронних мережах усі входи та виходи незалежні один від одного, але у випадках, наприклад, коли потрібно передбачити наступне слово речення, потрібні попередні слова, і, отже, виникає потреба запам’ятати попередні слова. Так виник RNN, який вирішив цю проблему за допомогою прихованого шару. Головною і найважливішою особливістю RNN є прихований стан , який запам’ятовує деяку інформацію про послідовність.

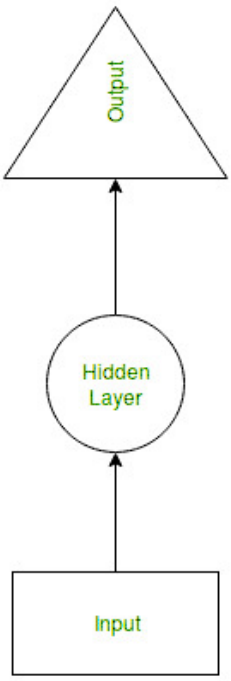

Рисунок 9-рекурентна нейромережа

RNN мають «пам’ять» , яка запам’ятовує всю інформацію про те, що було обчислено. Він використовує однакові параметри для кожного входу, оскільки виконує однакове завдання на всіх входах або прихованих шарах для створення виходу. Це зменшує складність параметрів, на відміну від інших нейронних мереж.

Для більш чіткого поняття принципу роботи наведу приклад:

Приклад: припустимо, що існує більш глибока мережа з одним вхідним рівнем, трьома прихованими рівнями та одним вихідним рівнем. Тоді, як і в інших нейронних мережах, кожен прихований рівень матиме власний набір ваг і зміщень, скажімо, для прихованого шару 1 ваги та зміщення (w1, b1), (w2, b2) для другого прихованого шару та (w3, b3) для третього прихованого шару. Це означає, що кожен із цих рівнів незалежний від іншого, тобто вони не запам’ятовують попередні виходи.

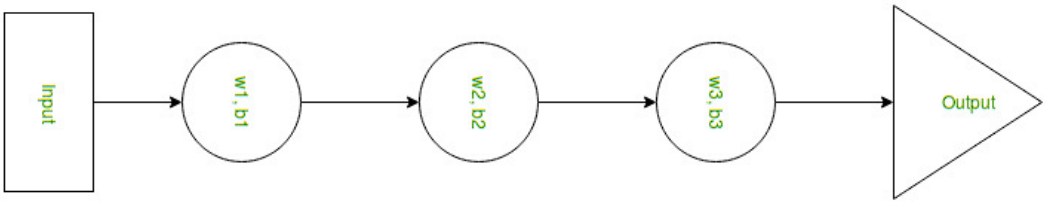

Рисунок 10-приклад рекурентної нейромережі

Тепер RNN зробить наступне:

- RNN перетворює незалежні активації в залежні активації, надаючи однакові ваги та зміщення для всіх рівнів, таким чином зменшуючи складність збільшення параметрів і запам’ятовування кожного попереднього виходу, надаючи кожен вихід як вхідний сигнал для наступного прихованого рівня.
- Таким чином, ці три шари можна об’єднати разом, щоб ваги та зміщення всіх прихованих шарів були однаковими в одному повторюваному шарі.

Отже, можемо зробити висновок що нейромережі мають наступні переваги та недоліки

Переваги:

- При правильному налаштуванні та навчанні може працювати ефективніше ніж інші методи;
- Стійкість до спотворень символів(при великому навчальному масиві)[12];

Недоліки:

- Складна реалізація;
- В багатошаровій мережі неможливий аналіз аномальної поведінки;

1.2.9. OpenCV

OpenCV — це величезна бібліотека з відкритим вихідним кодом для комп’ютерного зору, машинного навчання та обробки зображень, вона відіграє важливу роль у роботі в режимі реального часу, що дуже важливо для нашої системи розпізнавання.

Щоб ідентифікувати шаблон зображення та його різноманітні характеристики, ми використовуємо векторний простір і виконуємо математичні операції над цими функціями.

Функціональність OpenCV

- Введення/виведення зображення/відео,обробка, відображення (ядро, imgproc, highgui)
- Виявлення об’єктів/функцій (objdetect, features2d, nonfree)
- Монокулярний або стереокомп’ютерний зір на основі геометрії (calib3d, зшивання, videostab)
- Комп'ютерна фотографія (фото, відео, суперзйомка)
- Машинне навчання та кластеризація (ml, flann)

- Прискорення CUDA (gpu)

Обробка зображень

Обробка зображення — це метод виконання деяких операцій із зображенням, щоб отримати покращене зображення та/або витягти з нього деяку корисну інформацію.

Якщо говорити про основне визначення обробки зображень, то «Обробка зображень — це аналіз і обробка оцифрованого зображення, особливо з метою покращення його якості».

Цифрове зображення:

Зображення можна визначити як двовимірну функцію f(x, y), де x і y є просторовими (площинними) координатами, а амплітуда будь-якої пари координат (x, y) називається інтенсивність або рівень сірого зображення в цій точці. Іншими словами, зображення — це не що інше, як двовимірна матриця (тривимірна у випадку кольорових зображень), яка визначається математичною функцією f(x, y) у будь-якій точці, яка дає значення пікселя в цій точці зображення. зображення, значення пікселя описує, наскільки яскравим є цей піксель і якого кольору він має бути.

Обробка зображень — це в основному обробка сигналів, у якій входом є зображення, а виходом є зображення або характеристики відповідно до вимог, пов’язаних із цим зображенням.

Обробка зображення в основному включає наступні три етапи:

1. Імпорт зображення
2. Аналіз і обробка зображення
3. Вихід, результатом якого може бути змінене зображення або звіт, який базується на аналізі зображення

РОЗДІЛ 2
МОДЕЛЬ СИСТЕМИ РОЗПІЗНАВАННЯ НОМЕРНИХ ЗНАКІВ АВТОМОБІЛІВ З ВІДЕОПОТОКУ

Знаходження об'єктів на зображенні або у відео-потоці це завдання з галузі комп'ютерного зору, яке вирішується різними підходами, у моїй роботі розпізнавання виконується за допомогою різних типів нейронних мереж.

Розглянемо інструменти які використовуються у магістерській роботі.

Для завдань сегментації я використав архітектуру згорткових мереж під назвою Mask R-CNN.

Другий інструмент-бібліотека для розпізнавання тексту, в нашому випадку використовується tesseract від гугл.

Для нормалізації області з номерним знаком використовується opencv.

Для того щоб отримати точність близько 97 відсотків ми реалізували свою розпізнавалку тексту на базі згорткових та рекурентних шарів.

Архітектура цієї мережі виглядає приблизно так:

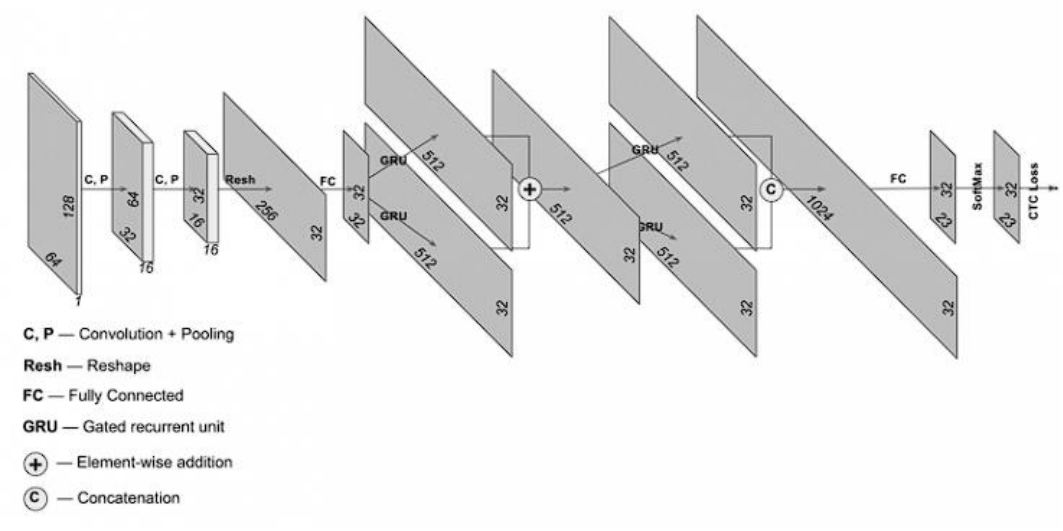

Рисунок 11 – архитектура OCR(розпізнавалки тексту)

Далі ознайомимось детальніше з інструментами та алгоритмами проекту

2.1. Варіант 1: контурний аналіз

Для першої версії нашої системи я використав контурний аналіз.

Контур об'єкта – це замкнута межа навколо рухомого об'єкта, яку можна представити у вигляді упорядкованої послідовності координатних пар.

$$E = [e_1, e_2, \dots e_n | e = P], \quad (1)$$

$$P = [x, y], \quad (1)$$

Де:

E- контур об'єкта

$e_1, e_2, \dots e_n$- пікселі контуру

n- загальна кількість точок контуру

P- координатна пара

x,y – координати точок на кордоні контуру

Для того, щоб подання контуру було одномірним використаємо масив центромасних відстаней.

Масив центромасних відстаней - одномірна дискретна впорядкована послідовність, члени якої дорівнють відстані від центру мас контуру до точки на його межі.

$$R = [r_1, r_2, \dots r_n], \quad (2)$$

Де:

R - Масив центромасних відстаней;

r - Відстань від центру мас до точки на межі контуру;

n – загальна кількість точок, що становлять контур.

Відстань від центру мас до точки на межі контуру я обчислюю за формулою:

$$r = \sqrt{(cx - x)^2 + (cy - y)^2}, \quad (3)$$

Де:

r - Відстань від центру мас до точки на межі контуру;

x,y – координати точок на кордоні контуру

cx та cy – координати центру мас

2.2. Варіант 2: нейронні мережі

Нейромережа це деякий набір нейронів поєднаних між собою. Принцип роботи нейромережі базується на перетворенні вхідного вектору даних в вихідний.[10]. Для того щоб навчити нейромережу в нейрони записують певний еталон, потім подається вхідний сигнал і порівнюють його з еталоном. Відмінність між еталоном і вхідним сигналом додається до еталону. Чим більше вхідних впливів тим краще навчиться мережа.[11].

Незважаючи на велику різноманітність варіантів нейронних мереж, усі вони мають спільні риси. Так, всі вони, так само, як і мозок людини, складаються з великої кількості пов'язаних між собою однотипних елементів – нейронів.

Стан нейрона я визначаю за формулою:

$$S = \sum_{i=1}^{n} x_i \omega_i , \quad (4)$$

Де:

n – число входів нейрона

x_i - значення i-го входу нейрона

ω_i - вага i-го синапсу.

Потім визначається значення аксону нейрона за формулою:

$$Y=f(S)Y=f(S), \quad (5)$$

Де:

f – деяка функція, що називається активаційною.Як активаційну функцію я використовую так званий сигмоїд, який має наступний вигляд:

$$f(x) = \frac{1}{1+e^{-ax}}, \quad (6)$$

Основна перевага цієї функції в тому, що вона диференційована по всій осі абсцис і має дуже просту похідну:

$$f'(x)=\alpha f(x)(1-f(x))f'(x)=\alpha f(x)(1-f(x)), \quad (7)$$

2.2.1. Нейронна мережа зворотного розповсюдження

Для навчання нейромережі я знаходжу певну функціональну залежність Y=F(X) де X – вхідний, а Y – вихідний вектори. Під час навчання ставиться завдання мінімізації цільової функції помилки нейромережі , яка знаходиться за методом найменших квадратів:

$$E(w) = \frac{1}{2}\sum_{j-1}^{p}(y_i - d_i)^2, \quad (8)$$

Де:

y_i – значення j-го виходу нейромережі,

d_i – цільове значення j-го виходу,

p - Число нейронів у вихідному шарі.

Навчання нейромережі виконується за методом градієнтного спуску

$$\Delta w_{ij} = -\eta \frac{\partial E}{\partial w_{ij}}, \quad (9)$$

Де:

η - параметр, який визначає швидкість навчання.

$$\frac{\partial E}{\partial w_{ij}} = \frac{\partial E}{\partial y_i} \frac{dy_i}{dS_j} \frac{\partial S_j}{\partial w_{ij}}, \quad (10)$$

Де:

y_i - значення виходу j-го нейрона,

S_j – зважена сума вхідних сигналів, яка визначається за формулою (4).

При цьому:

$$\frac{\partial S_j}{\partial w_{ij}} = x_i, \quad (11)$$

Де:

x_i - значення і-го входу нейрона.

Далі розглянемо визначення першого множника формули (10).

$$\frac{\partial E}{\partial y_j} = \sum_k \frac{\partial E}{\partial y_k} \frac{dy_k}{dS_k} \frac{\partial S_k}{\partial y_j} = \sum_k \frac{\partial E}{\partial y_k} \frac{dy_k}{dS_k} \omega_{jk}^{(n=1)}, \quad (12)$$

Де:

k – число нейронів у шарі n+1.

Введемо допоміжну змінну

$$\delta_j^{(n)} = \frac{\partial E}{\partial y_j}\frac{dy_j}{dS_j}, \qquad (13)$$

Тоді ми зможемо визначити рекурсивну формулу для визначення n-ного шару, якщо відомо наступного (n+1)-го шару.

$$\delta_j^{(n)} = \left[\sum_k \delta_j^{(n+1)}\, \omega_{jk}^{(n+1)}\right]\frac{dy_j}{dS_j}, \qquad (14)$$

Останній шар визначається за формулою:

$$\delta_j^{(N)} = \left(y_i^{(N)} - d_i\right)\frac{dy_j}{dS_j}\ ,$$

(15)

І нарешті запишемо формулу (9) у розкритому вигляді

$$\Delta w_{ij}^{(n)} = -\eta\delta_j^{(n)} x_i^n\ ,$$

(16)

Приклад коду нейромережі:

```
from numpy import exp, array, random, dot
training_set_inputs = array([[0, 0, 1], [1, 1, 1], [1, 0, 1], [0, 1, 1]])
training_set_outputs = array([[0, 1, 1, 0]]).T
random.seed(1)
synaptic_weights = 2 * random.random((3, 1)) - 1
for iteration in xrange(10000):
    output = 1 / (1 + exp(-(dot(training_set_inputs, synaptic_weights))))
    synaptic_weights += dot(training_set_inputs.T, (training_set_outputs - output) * output * (1 - output))
print 1 / (1 + exp(-(dot(array([1, 0, 0]), synaptic_weights))))
```

Рисунок 12- приклад коду нейромережі.

2.2.2. Рекурентна нейронна мережа

Також я використав рекурентні нейронні мережі. Для цього було задіяно наступні формули:

Формула розрахунку поточного стану:

$$h_t = f(h_{t-1}, x_t), \quad (17)$$

Де:

h_t -> поточний стан

h_{t-1} -> попередній стан

x_t-> вхідний стан

Формула для застосування функції активації (tanh):

$$h_t = \tanh(W_{hh} h_{t-1} + W_{xh} x_t), \quad (18)$$

Де:

W_{hh} -> вага рекурентного нейрона

W_{xh} -> вага на вхідному нейроні

Формула розрахунку виходу:

$$y_t = W_{hy} h_t, \quad (19)$$

Де:

y_t -> вихід

W_{hy}-> вага на вихідному шарі

Навчання я проводив за наступним алгоритмом:

Навчання через RNN

1. У мережу надається одноразовий крок входу.

2. Потім обчисліть його поточний стан, використовуючи набір поточного введення та попереднього стану.
3. Поточний h_t стає h_{t-1} для наступного кроку часу.
4. Можна пройти скільки завгодно часових кроків відповідно до проблеми та об'єднати інформацію з усіх попередніх станів.
5. Після завершення всіх часових кроків кінцевий поточний стан використовується для обчислення виходу.
6. Потім результат порівнюється з фактичним результатом, тобто цільовим результатом, і генерується помилка.
7. Потім помилка передається в мережу для оновлення вагових коефіцієнтів і, отже, мережа (RNN) навчається.

2.3. Білінійна інтерполяція

Для нормалізації зображення я використав білінійну інтерполяцію

Візьмемо зображення та припустимо що в нас є 4 точки з координатами (y1,x1)(y1,x1), (y1,x2)(y1,x2), (y2,x1) , та (y2,x2) і пов'язані з ними значення A, B, C, D

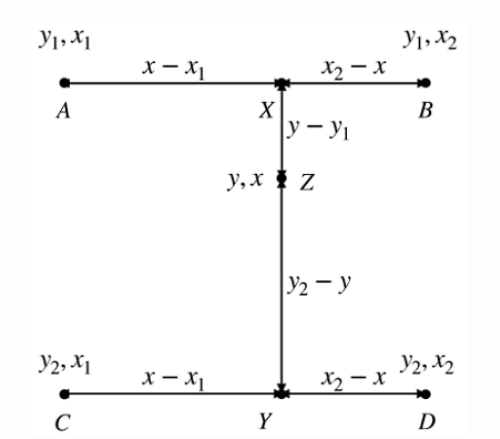

Рисунок 13-Білінійна інтерполяція між чотирма точками

Спочатку ми обчислюємо інтерпольоване значення X та Y в розмірі ширини

$$X = A(1 - \omega_x) + B\omega_x \,, \qquad (20)$$

$$Y = C(1 - \omega_x) + D\,\omega_x\,, \qquad (20)$$

Потім ми виконаємо лінійну інтерполяцію між двома інтерпольованими значеннями X та Y у висотному розмірі

$$Z = X\big(1 - \omega_y\big) + Y\,\omega_y = A(1 - \omega_x)\big(1 - \omega_y\big) + B\,\omega_x\big(1 - y_y\big) + C(1 - \omega_x)\,\omega_y + D\,\omega_x\,\omega_y, \qquad (21)$$

Де:

$$\omega_x = \frac{x - x_1}{x_2 - x_1} \quad \text{та} \quad \omega_y = \frac{y - y_1}{y_2 - y_1}, \qquad (22)$$

РОЗДІЛ 3
КОМП'ЮТЕРНА СИСТЕМА РОЗПІЗНАВАННЯ НОМЕРНОГО ЗНАКУ АВТОМОБІЛІВ ЗА ВІДЕО З ВЕБ-КАМЕРИ

3.1. Опис набору даних

Постановка задачі

Задача полягає в тому, щоб з відеопотоку ір камери як найшвидше розпізнати автономер та передати цю інформацію в до інших систем для аналізу, збрерігання даних та комунікації з кінцевим користувачем, також отримання додаткових відкритих даних про автівку.

Опис даних

Дані були отримані з датасету Autoria.UA та власні відеозаписи з ір камер автомийки Hose. Вони являють собою набір фотографій та кадрів з відеозаписів різної роздільної здатності на яких чітко помітні номери автівок.

3.2. Розв'язок задачі роспізнавання автономерів

Алгоритм роботи системи

Моя система працює за наступним алгаритмом

3.2.1. Версія 1.0

Для реалізації першої версії нашої системи я використовував контурний аналіз для локалізації номерних знаків у кадрі та easyOCR для розпізнавання тексту.

```
import cv2
from matplotlib import pyplot as plt
import numpy as np
import imutils
import easyocr
import time
def detect_carplate(imageSrc):
    start_time = time.time()
    img = cv2.imread(imageSrc)
    gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    bfilter = cv2.bilateralFilter(gray, 11, 17, 17)  # Noise reduction
    edged = cv2.Canny(bfilter, 30, 200)  # Edge detection
    keypoints = cv2.findContours(
        edged.copy(), cv2.RETR_TREE, cv2.CHAIN_APPROX_SIMPLE)
    contours = imutils.grab_contours(keypoints)
    contours = sorted(contours, key=cv2.contourArea, reverse=True)[:10]
    location = None
    for contour in contours:
        approx = cv2.approxPolyDP(contour, 10, True)
        if len(approx) == 4:
            location = approx
            break
    mask = np.zeros(gray.shape, np.uint8)
    new_image = cv2.drawContours(mask, [location], 0, 255, -1)
    new_image = cv2.bitwise_and(img, img, mask=mask)
    plt.imshow(cv2.cvtColor(new_image, cv2.COLOR_BGR2RGB))
    plt.show()
    (x, y) = np.where(mask == 255)
    (x1, y1) = (np.min(x), np.min(y))
    (x2, y2) = (np.max(x), np.max(y))
    cropped_image = gray[x1:x2+1, y1:y2+1]
    reader = easyocr.Reader(['en'], gpu=False, verbose=False)
    result = reader.readtext(cropped_image)
    result
    text = result[-1][1]
    execution_time = time.time() - start_time
    return text, execution_time
if __name__ == "__main__":
    images = [
        "./images/image1.jpeg",
        "./images/image2.jpeg",
```

Рисунок 14-програмний код версії системи 1.0

Вхідні дані:

Рисунок 15 – Вхідне фото image1.jpeg

Рисунок 16 – Вхідне фото image2.jpeg

Рисунок 17 – Вхідне фото image3.jpeg

Результати роботи алгоритму:

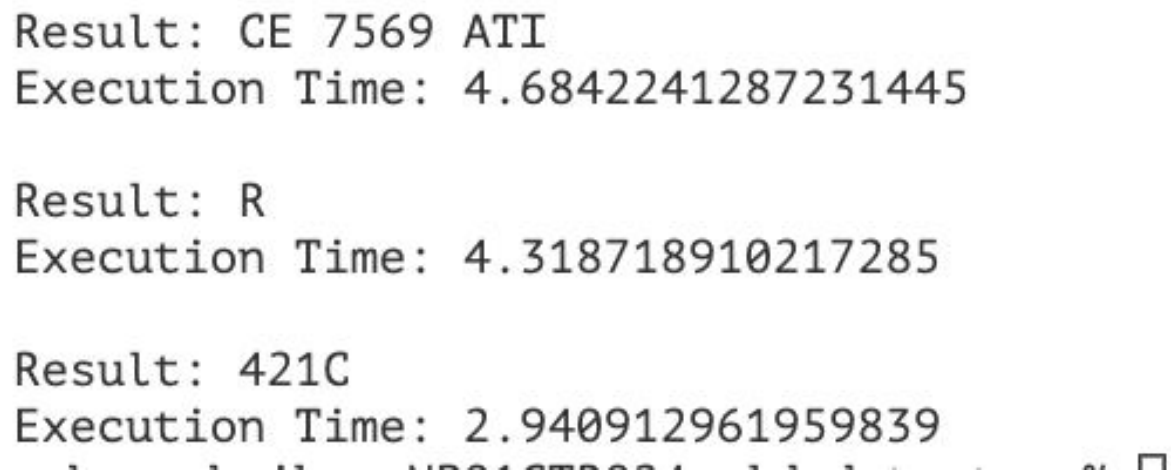

Рисунок 18-результати роботи системи версія 1.0

Як бачимо у другому результаті не правильно локалізувало табличку номера, і не правильно видало результат.А в 3-му варіанті отримали не повний номер, а тільки його частину. Також система працює досить повільно.

Такий результат не задовольнив мене тому я вирішив створити другу версію системи. Для того, щоб досягти кращих результатів, я використовував спеціально навчену нейромережу yolov5 для класифікації номерних знаків.

3.2.2. Версія 2.0

Для версії 2.0 я використав спеціально навчену нейромережу yolov5 для класифікації номерних знаків.

YOLO (You Only Look Once)- один із найпопулярніших на сьогоднішній день алгоритмів виявлення об’єктів у реальному часі

```python
import os
from _paths import nomeroff_net_dir
from nomeroff_net import pipeline
from nomeroff_net.tools import unzip
import time

def create_numberplates_detector():
    number_plate_detection_and_reading = pipeline("number_plate_detection_and_reading", image_loader="opencv")

    def detect_numberplate(imageSrc):
        st = time.time()

        result = number_plate_detection_and_reading(imageSrc if type(imageSrc) is list else [imageSrc])

        (images, images_bboxs,
        images_points, images_zones, region_ids,
        region_names, count_lines,
        confidences, texts) = unzip(result)

        execution_time = time.time()-st

        return texts, execution_time

    return detect_numberplate

if __name__ == '__main__':
    detect_numberplate = create_numberplates_detector()

    images = [
        os.path.join(nomeroff_net_dir, './images/example1.jpeg'),
        os.path.join(nomeroff_net_dir, './images/image1.jpeg'),
        os.path.join(nomeroff_net_dir, './images/image2.jpeg'),
    ]

    for imageSrc in images:
        result, execution_time = detect_numberplate(imageSrc)
        print('Result:', result, '\nExecution time:', execution_time, 'seconds')
```

Рисунок 19 – Програмний код системи версії 2.0

Вхідні дані:

Для вхідних даних ми використали ті ж самі зображення що і для версії 1

Результати роботи алгоритму:

```
Result: (['CE7569AT'],)
Execution time: 0.18280816078186035 seconds

Result: (['AA553XB'],)
Execution time: 0.2436378002166748 seconds

Result: (['AC4921CB'],)
Execution time: 0.24553704261779785 seconds
```

Рисунок 20 – Результат роботи системи версії 2.0

3.3. Аналіз

Для порівняння якості роботи версій визначимо точність розпізнавання та швидкість роботи кожної з них. Для тестування ми використали 200 зображень з датасету Autoria.UA

Перша версія видала наступні результати:

```
Recognition quality: 55.17 %
Total execution time: 85.63059782981873
Average one photo process time: 2.9527792355109908
```

Рисунок 21-результати роботи системи версія 1.0

Як бачимо, в нас всього 55 відсотків точність розпізнавання а також дуже повільна робота системи.

Подивимось на результати другої версії:

```
Recognition quality: 98.57 %
Total execution time: 5.988624095916748
Average one photo process time: 0.21387943199702 67
```

Рисунок 22 – Результат роботи системи версії 2.0

Версія 2.0 на базі yolov5 працює більш ніж в 10 разів швидше порівняно з версією 1.0 а також має точність розпізнавання 98 відсотків, що є дуже гарним результатом.

Також щоб прискорити результат я спробував зробити попередню обробку вхідних даних за допомогою білінійній інтерполяції. Для цього я спочатку обрізаю їх за співвідношенням сторін 1:1 , потім зменшую зображення до 640x640, без значної втрати точності розпізнавання.

Отримали наступний результат

```
Recognition quality: 97.07 %
Total execution time: 4.165619058561325
Average one photo process time: 0.1465056758369718
```

Рисунок 23 - результати після попередньої обробки зображення

ВИСНОВКИ

У роботі створена комп’ютерна система для розпізнавання номерного знаку автомобіля за відео з веб-камери. Побудовано дві версії системи розпізнавання номерних знаків автомобіля.

Версія 1.0 використовувала контурний аналіз для локалізації номерних знаків та easyOCR для розпізнавання тексту виявилась досить повільною у роботі та мала точність розпізнавання всього 55 відсотків

Версію 2.0 на базі моделі комп'ютерного зору YOLOv5 працює більш ніж в 10 разів швидше попередньої версії та має точність розпізнавання 98 відсотків.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Shashirangana J, Padmasiri H, Meedeniya D, Perera C (2020) Automated license plate recognition: a survey on methods and techniques. IEEE Access 9:11203–11225.
2. Roy S (2020) Automatics number plate recognition using convolutional neural network. Azerbaijan Int J High Perform Comput 234–244.
3. Real-Time Automatic Vehicle Management System Using Vehicle Tracking And Car Plate Number Iden- tification / Lee H., Kim D., Kim D., Bang S.Y. // ICME. – 2003. – № 7. – P. 353-356.
4. Acosta B.D. (2004) Experiments in Image Segmentation for Automatic US License Plate Recognition / B.D. Acosta // Thesis submitted to the Faculty of the Virginia Polytechnic Institute and State University in partial fulfill- ment of the requirements for the degree of MASTER OF SCIENCE in Computer Science. – 104 p.
5. Broumandnia A. (2005) Application of pattern recognition for Farsi license plate recognition / A. Broumandnia,M. Fathy // ICGST Int. J. on Graphics, Vision and Image Processing. – Vol. 5, № 2. – P. 25-31.
6. Martin F. (2003) Automatic Car Plate Recognition Using a Partial Segmentation Algorithm / F. Martin, D. Bor- ges // Proc. of Signal Processing, Pattern Recognition, and Applications. – P. 246-249.
7. Paul Viola, Jones Robust. (2001) Real-time object detection / Paul Viola and Michael J. // Proc. of IEEE Workshop on Statistical and Computational Theories of Vision.
8. Menon A, Omman B. (2018) Detection and recognition of multiple license plate from still images. In: 2018 international conference on circuits and systems in digital enterprise technology (ICCSDET). IEEE, pp 1–5.

9. Zheng L, Sayed T, Mannering F. (2020) Modeling traffic conflicts for use in road safety analysis: a review of analytic methods and future directions. Analytik Methods Accident Res 29.
10. Zhong, Zhuoyao, Lei Sun, and Q. Huo. (2019). “Improved Localization Accuracy by LocNet for Faster R-CNN Based Text Detection in Natural Scene Images.” Pattern Recognition 96.
11. A.Conci, J. E. R. de Carvalho, T. W. Rauber. (2009) A Complete System for Vehicle Plate Localization, Segmentation and Recognition in Real Life Scene, Ieee Latin America Transactions, vol. 7, no. 5, September.
12. Liu W, Anguelov D, Erhan D, Szegedy C, Reed S, Fu C, Berg A. (2016) Ssd: single shot multibox detector. European conference on computer vision. Springer, New York, pp 21–37.