Локалізація номерних знаків на зображеннях автомобілів

Постановка задачі:

1) Розробити алгоритм пошуку номерних знаків на зображенні (алгоритм для локалізації потенційних номерних знаків) + розпізнавання тексту (для перевірки чи це дійсно номерний знак). Продемонструвати успішні результати та помилкові (якщо такі присутні) з поясненням, чому в такому випадку не спрацював алгоритм.
2) Підготувати вибірку з баундінгбоксами або масками номерних знаків за допомогою попереднього алгоритму, ознайомитись з форматами датасету для тренування мережі для локалізації об’єктів на фотографіях, та обрати один.
3) Подумати над тим, які техніки аугментації будуть використовуватися.

Критерії валідації:

1) Правильно визначені номерні знаки для 80% зображень.

Етап 1: Дослідження датасету

Датасет, взятий із Kaggle (https://www.kaggle.com/datasets/andrewmvd/car-plate-detection), містить 433 зображення автомобілів із номерними знаками (рис. 1).

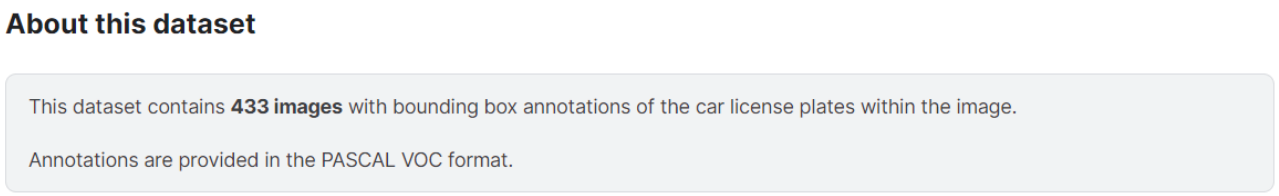


Рис. 1. Інформація про датасет Car License Plate Detection

Варто зазначити, що в датасеті були виявлені однакові зображення автомобілів, тобто повтори (рис. 2).

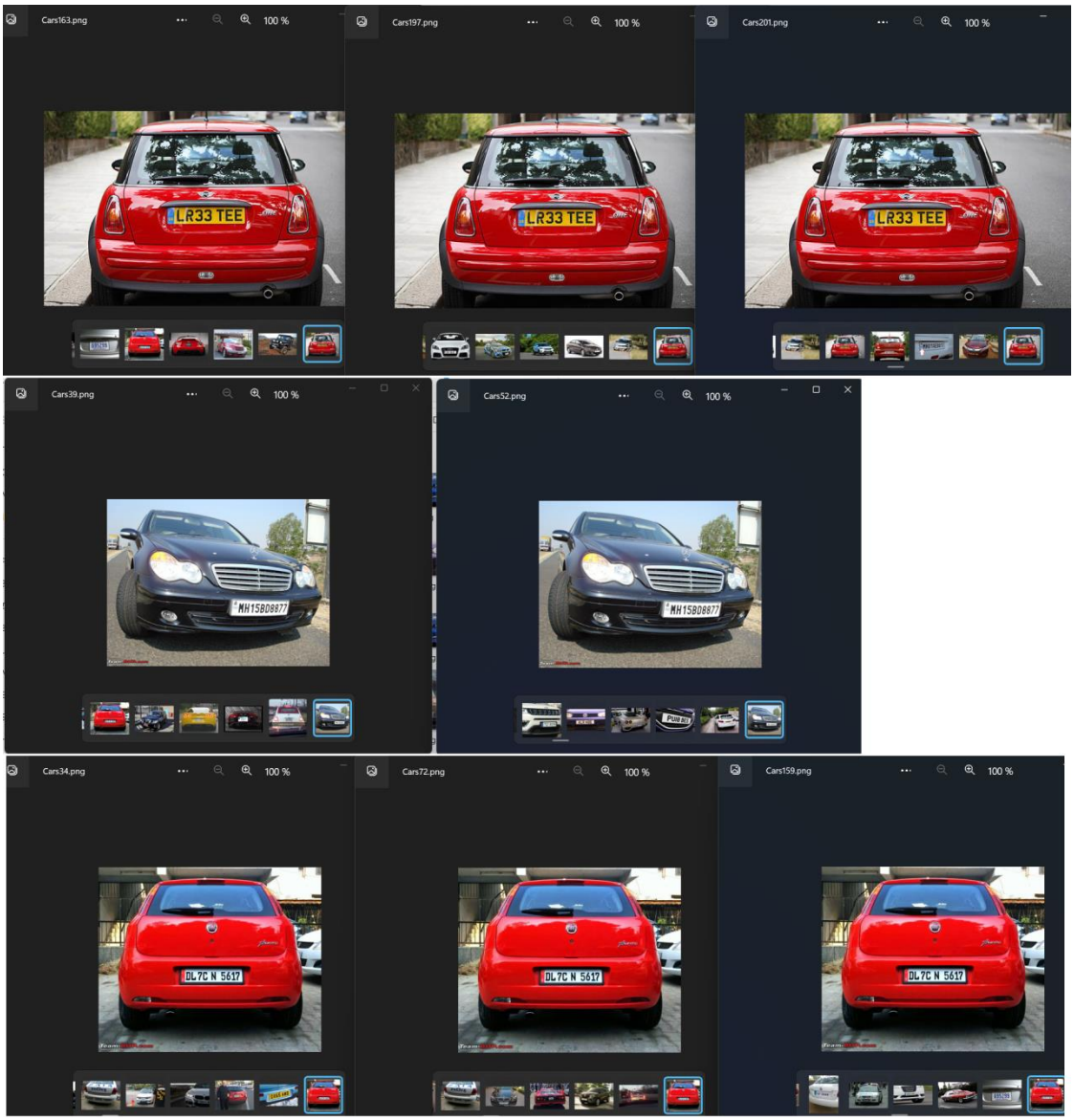


Рис. 2. Повтори автомобілів у досліджуваному датасеті

Також у датасеті наявні зображення, на яких є не лише один автомобіль, а декілька, на яких можна виділити номерні знаки (рис. 3) або декілька номерних знаків на одному автомобілі (рис. 4).

Рис. 3. Зображення із багатьма автомобілями та номерними знаками

Рис. 4. Зображення автомобіля із багатьма номерними знаками

Тому для зручності роботи із зображеннями та перевірки проміжних і отриманих результатів зображення було спочатку поділено на папки "one_car" та "many_cars" із кількістю зображень 389 та 44 відповідно (рис. 5).

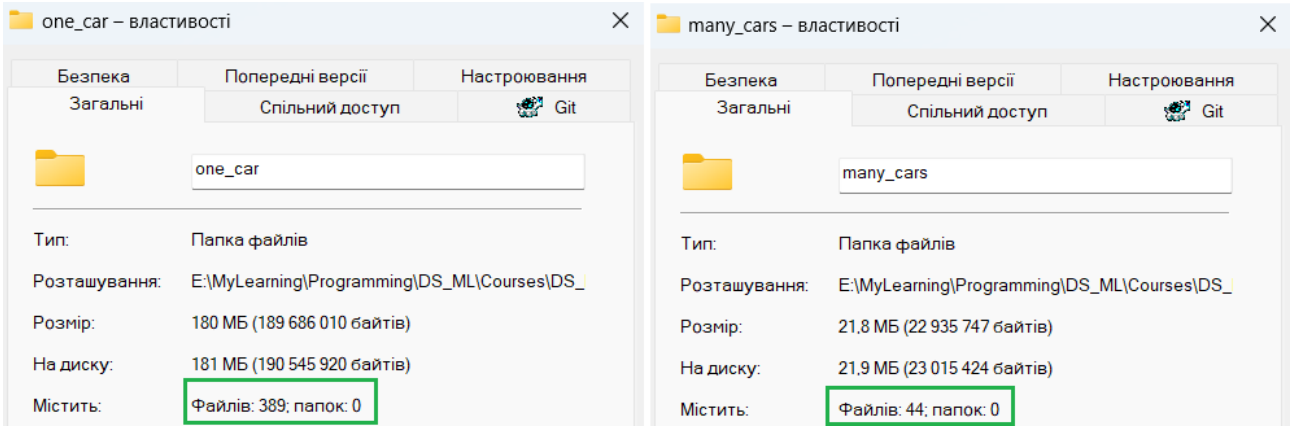


Рис. 5. Поділ зображень на зображення із одним автомобілем та багатьма

Файли папки "many_cars" було також поділено на кілька папок (рис. 6) для обробки зображень порціями (batches), що дозволило значно швидше перевіряти ефективність роботи досліджуваних алгоритмів.

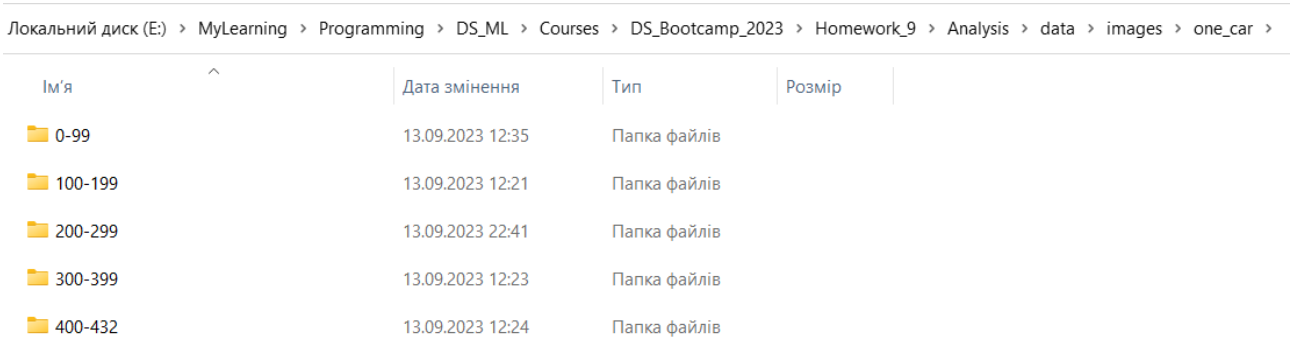


Рис. 6. Поділ зображень із одним автомобілем на порції

Етап 2: Пошук алгоритмів

Початковий алгоритм виділення номерного знаку було взято із ресурсу https://www.youtube.com/watch?v=NApYP_5wlKY&t=68s.

Алгоритм побудований на основі методів бібліотеки OpenCV, тобто передбачає лише використання класичних алгоритмів опрацювання зображень.

Для першого, 'димового' тестування було обрано 4 зображення, наведені у вищезгаданому ресурсі.

Коротко опишемо основні кроки роботи алгоритму та наведемо проміжні результати для тестового зображення:

1) Перетворення зображення на градації сірого (grayscale) (рис. 7).

```
img = cv2.imread('example_data/image4.jpg')
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
plt.imshow(cv2.cvtColor(gray, cv2.COLOR_BGR2RGB))
plt.title('image4.jpg')
plt.show()
```

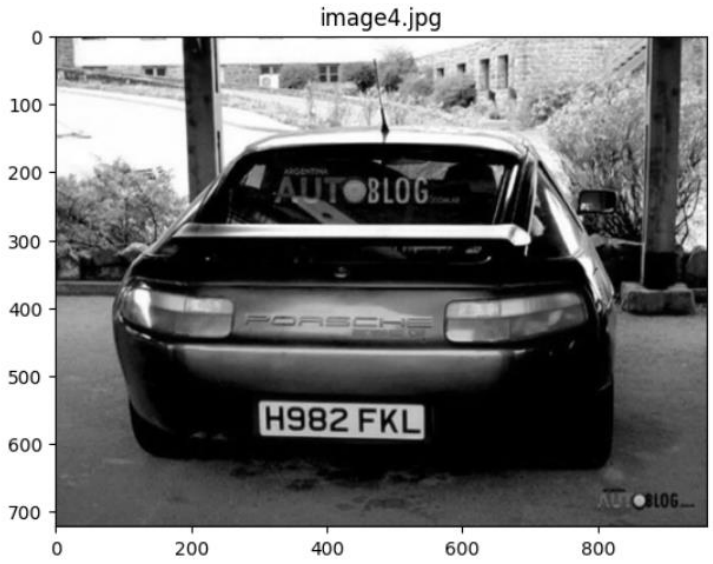


Рис. 7. Результат виконання першого кроку алгоритму

2) Використання двостороннього фільтру (bilateralFilter) для зменшення шумів (Noise reduction) (рис. 3).

```
bfilter = cv2.bilateralFilter(gray, 11, 17, 17) # Noise reduction
plt.imshow(cv2.cvtColor(bfilter, cv2.COLOR_BGR2RGB))
```

```
<matplotlib.image.AxesImage at 0x1e0848f60b0>
```

Рис. 8. Результат зменшення шумів з допомогою двостороннього фільтру

3) Використання оператора Кенні для виділення контурів (рис. 9).

```
edged = cv2.Canny(bfilter, 30, 200) # Edge detection
plt.imshow(cv2.cvtColor(edged, cv2.COLOR_BGR2RGB))
```

```
<matplotlib.image.AxesImage at 0x1e08488d0f0>
```

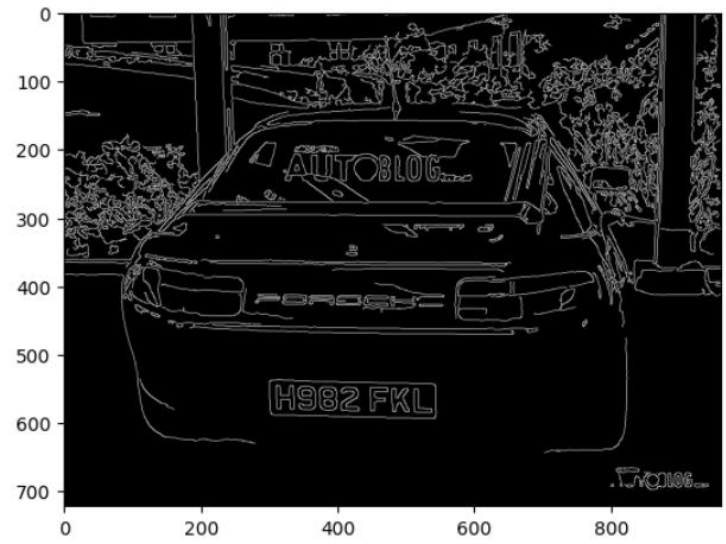


Рис. 9. Результат використання оператора Кенні для виділення контурів

4) Знаходження контурів номерного знаку (у вигляді прямокутника) (рис. 10).

```
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)[:30]

location = None
for contour in contours:
    approx = cv2.approxPolyDP(contour, 10, True)
    if len(approx) == 4:
        location = approx
        break

location
```

```
array([[[300, 540]],

       [[306, 589]],

       [[543, 592]],

       [[538, 543]]], dtype=int32)
```

Рис. 10. Результат знаходження номерного знаку на зображенні автомобіля

5) Використання маски для відображення положення номерного знаку на зображенні (рис. 11) та накладання маски на оригінальне зображення (рис. 12) для перевірки проміжних результатів.

```
mask = np.zeros(gray.shape, np.uint8) # create a blank mask
new_image = cv2.drawContours(mask, [location], 0, 255, -1) # draw contours inside the mask image with location coordinates
plt.imshow(cv2.cvtColor(new_image, cv2.COLOR_BGR2RGB))
```

```
<matplotlib.image.AxesImage at 0x1e08456d600>
```

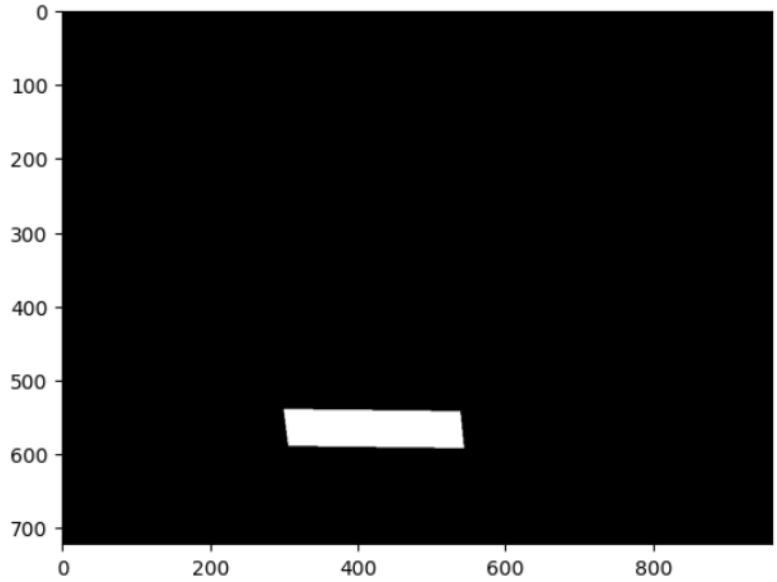


Рис. 11. Використання маски для відображення місця номерного знаку на зображенні

```
new_image = cv2.bitwise_and(img, img, mask=mask) # Applying masks on top of the original image
plt.imshow(cv2.cvtColor(new_image, cv2.COLOR_BGR2RGB))
```

```
<matplotlib.image.AxesImage at 0x23755d97700>
```

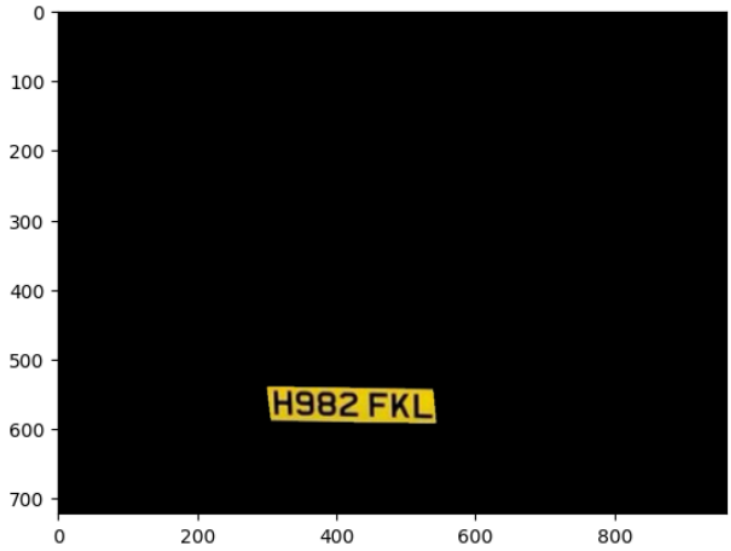


Рис. 12. Накладання маски номерного знаку на оригінальне зображення

6) Виділення номерного знаку із оригінального зображення (рис. 13).

```
(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]

plt.imshow(cv2.cvtColor(cropped_image, cv2.COLOR_BGR2RGB))
```

```
<matplotlib.image.AxesImage at 0x1e084e02dd0>
```

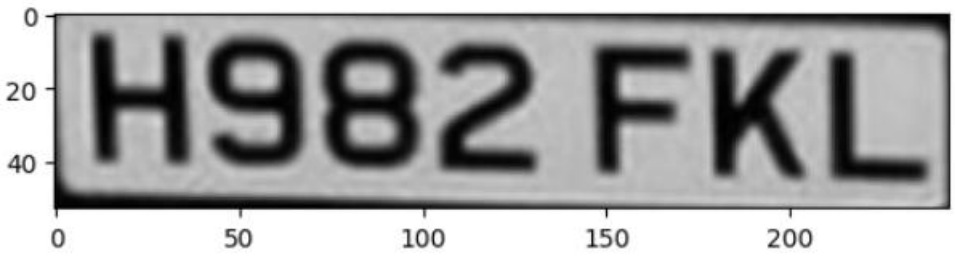


Рис. 13. Виділення номерного знаку із зображення

Окрім алгоритму виділення номерного знаку, у вищезгаданому ресурсі також наводився один із можливих алгоритмів розпізнавання тексту – easyOCR, який показав доволі непогані результати для чотирьох тестових зображень (рис. 14).

Рис. 14. Результати використання easyOCR для розпізнавання тексту на номерних знаках

Дуже схожий алгоритм виділення номерного знаку було запропоновано у статті https://www.makeuseof.com/python-car-license-plates-detect-and-recognize/. Однак тут для розпізнавання тексту номерних знаків використовувався Tesseract OCR, тому я вирішив порівняти результати роботи easyOCR та Tesseract OCR (із використанням бібліотеки pytesseract) для визначення оптимальної технології.

Tesseract OCR показав трохи гірші результати (рис. 15).

Рис. 15. Результати використання Tesseract для розпізнавання тексту на номерних знаках

Порівнявши результати роботи easyOCR (рис. 14) та Tesseract (рис. 15), можна побачити, що easyOCR показав кращі результати. Tesseract для одного із зображень не зумів розпізнати текст, що зумовлено обмеженнями цього алгоритму:

1) Колір тексту: Tesseract може працювати з текстом у різних кольорах, але зазвичай він найкраще працює з висококонтрастним текстом. Чорний текст на білому або світлому фоні є найпоширенішою та рекомендованою конфігурацією (на третьому зображенні навпаки, білий текст на чорному фоні).

2) Розмір тексту: точність Tesseract краща, якщо шрифт має нормальний розмір. Надзвичайно малий (випадок третього зображення) або дуже великий текст може бути важко точно розпізнати.

Оскільки easyOCR показав кращі результати (текст на 4-ому зображенні було точно визначено всі літери), було прийнято рішення використовувати перший алгоритм для розпізнавання літер.

Наступним кроком протестовано алгоритм openCV виділення номерного знаку на зображеннях 1-99 із досліджуваного датасету. Внаслідок виконання алгоритму було отримано різні результати: на деяких зображеннях номерні знаки були виділені доволі точно навіть на складних зображеннях (4-е зображення на рис. 16), в деяких випадках – виділялися інші частини автомобілів, як-от фари або вікна (рис. 17), в деяких випадках замість всього номерного знаку – лише окремі символи (рис. 18), та для декількох зображень взагалі не вдалося виявити номерний знак (рис. 19).

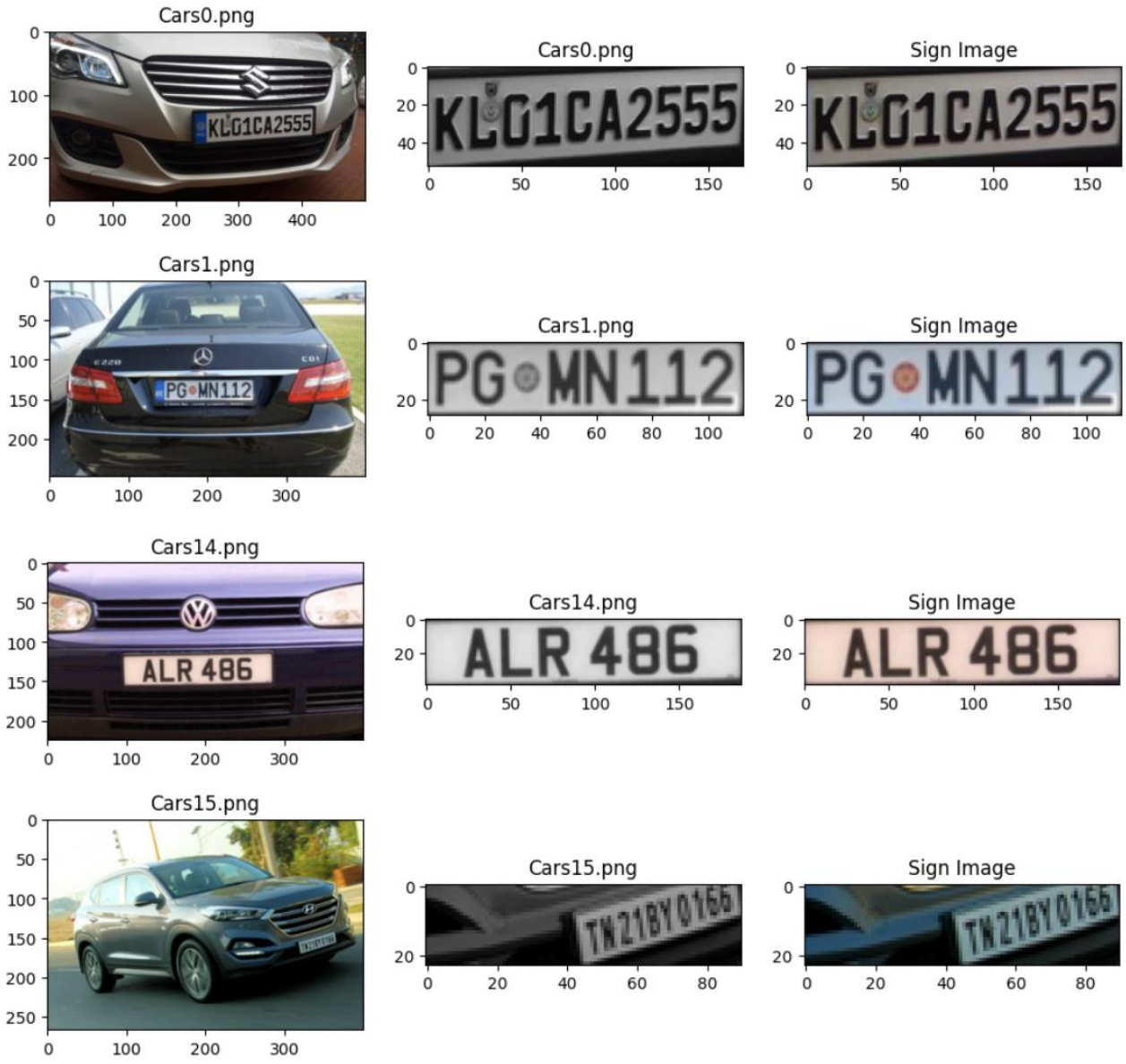


Рис. 16. Позитивні результати виділення номерних знаків за допомогою openCV-алгоритму

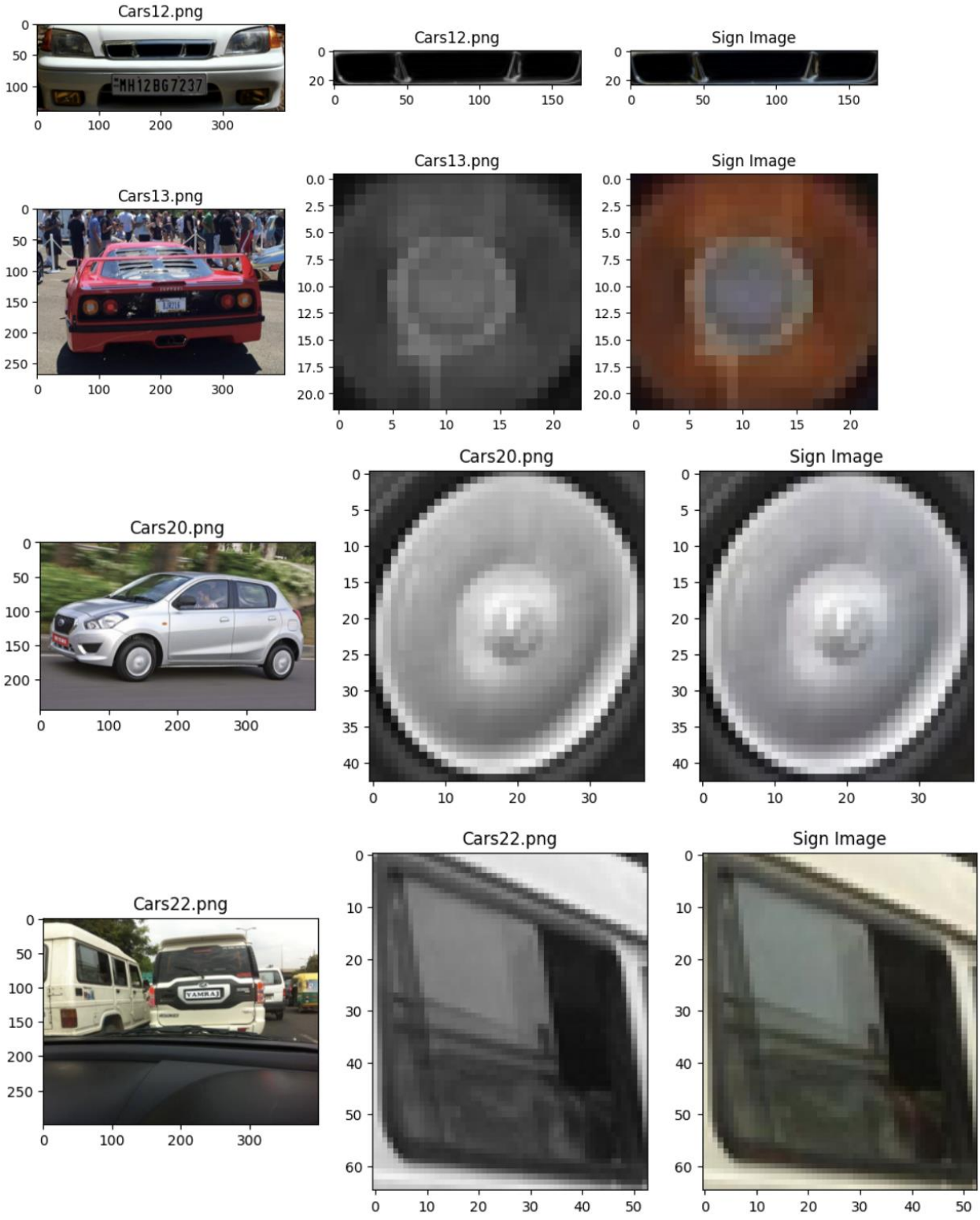


Рис. 17. Негативні результати виділення номерних знаків за допомогою openCV-алгоритму: виділення інших частин автомобілів замість номерних знаків

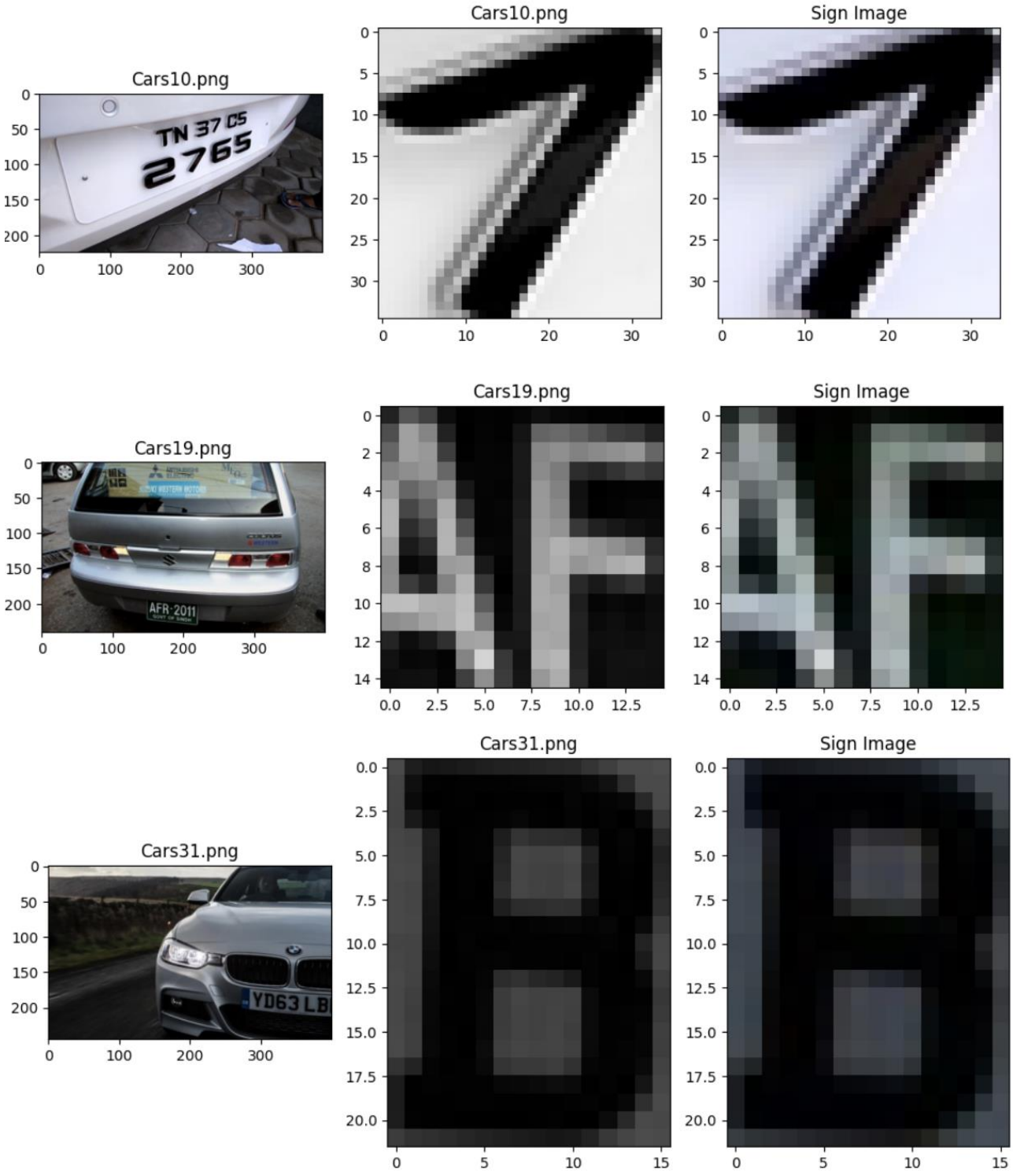


Рис. 18. Негативні результати виділення номерних знаків за допомогою openCV-алгоритму: виділення лише одного-декількох символів замість номерних знаків

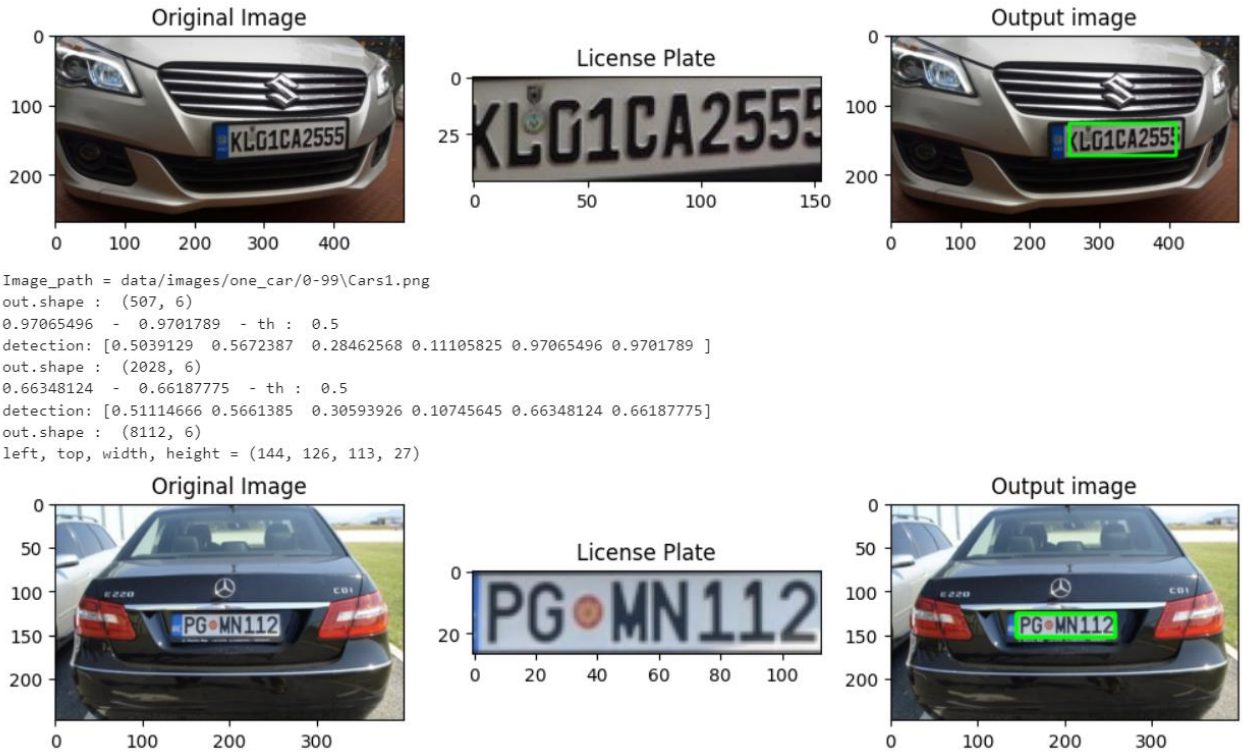
Рис. 19. Невдачі при виділенні номерних знаків

Отримані негативні результати можна пояснити особливостями роботи алгоритму: зокрема, на 4-ому кроці алгоритму з-поміж виділених контурів вибирається 30 найбільших і серед них відбувається пошук контура, який можна інтерпретувати як прямокутник. Перший контур, який задовольняє вказану умову вважатиметься номерним знаком. Тому якщо в ході виконання попередніх кроків алгоритму було виділено великі контури (колеса, фари, вікна – рис. 17), що задовольняють умову прямокутності, і є більшими, ніж прямокутний контур номерного знака, буде отримано не номерний знак. Випадки, коли замість цілого номерного знака було отримано лише окремі символи (рис. 18) можна пояснити кутом нахилу номерного знака.

Додавши до алгоритму openCV розпізнавання тексту за допомогою easyOCR на виділених контурах для перевірки чи це дійсно номерний знак, було відкинуто неправильно виділені частини автомобіля (фари, вікна, колеса і т.д.), однак зображення із одним-декількома символами залишилися.

Оскільки алгоритм OpenCV показав помірні результати, було вирішено використати модель глибокого навчання YOLO (https://youtu.be/NXjCJZxeaQA?si=udsb0Q4_Gzqh5XUh). Результати використання алгоритму YOLO на зображеннях автомобілів 0-99 також виявилися неоднозначними:

1) Більшість номерних знаків, які виділила модель, була правильною (рис. 20).
2) Декілька номерних знаків було виділено неправильно (рис. 21), однак частота помилок є значно меншою, ніж в попереднього алгоритму.
3) На деяких зображеннях модель YOLO не змогла виділити номерні знаки (рис. 22).




```
Image_path = data/images/one_car/0-99\Cars1.png
out.shape :  (507, 6)
0.97065496  -  0.9701789  - th :  0.5
detection: [0.5039129  0.5672387  0.28462568 0.11105825 0.97065496 0.9701789 ]
out.shape :  (2028, 6)
0.66348124  -  0.66187775  - th :  0.5
detection: [0.51114666 0.5661385  0.30593926 0.10745645 0.66348124 0.66187775]
out.shape :  (8112, 6)
left, top, width, height = (144, 126, 113, 27)
```

Рис. 20. Позитивні результати виділення номерних знаків за допомогою моделі YOLO

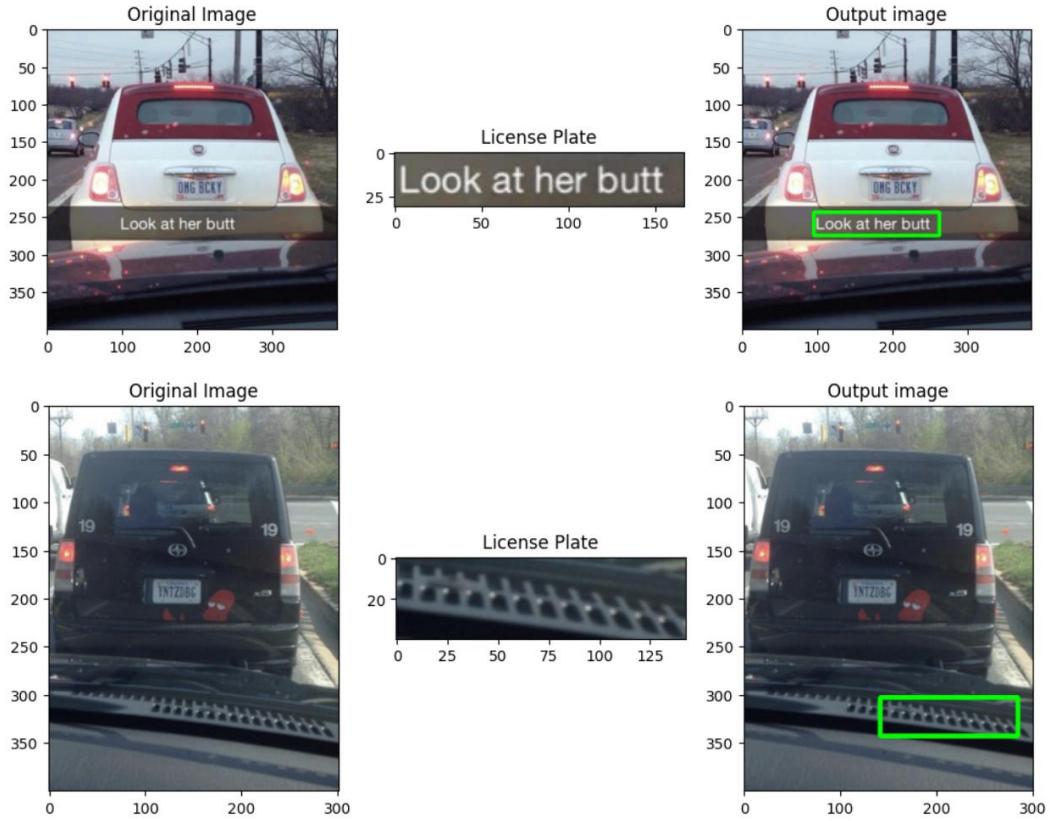


Рис. 21. Негативні результати виділення номерних знаків за допомогою моделі YOLO: виділення інших частин автомобілів або написів замість номерних знаків

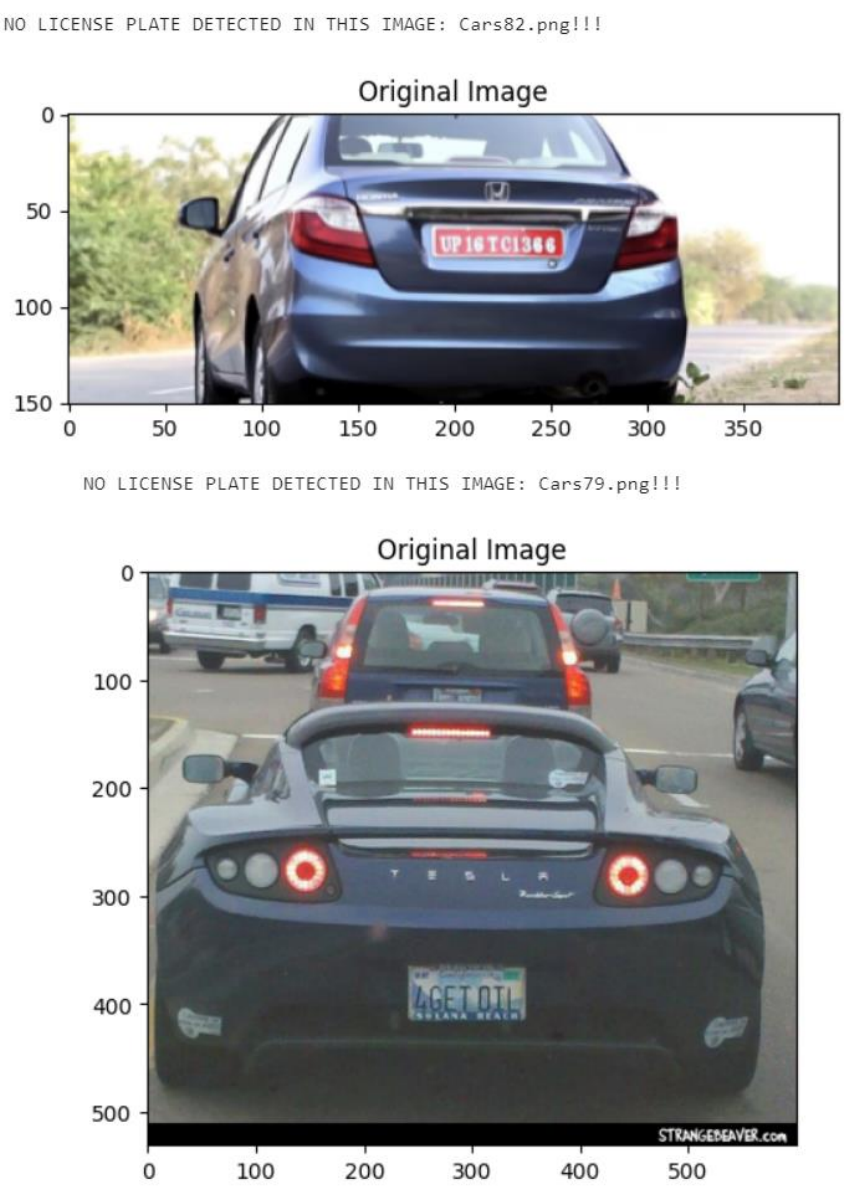


Рис. 22. Невдачі при виділенні номерних знаків за допомогою моделі YOLO

Аналогічно, як і у випадку попереднього алгоритму, додавання розпізнавання тексту за допомогою easyOCR на виділених контурах для перевірки чи це дійсно номерний знак забезпечило відкидання помилково виділених як номерний знак інших частин автомобіля (рис. 23).

NO TEXT WAS DETECTED ON THE SELECTED PART OF THE IMAGE: Cars41.png!!!

Рис. 23. Якщо на виділеній частині не вдалося розпізнати текст, то алгоритм відкидає це зображення

Етап 3: Побудова власного алгоритму

В ході аналізу результатів алгоритму, побудованого на методах openCV, та алгоритму із використанням моделі YOLO зроблено висновок, що ці алгоритми можна об’єднати в один для отримання максимально хороших результатів, оскільки кожен алгоритм дозволяє локалізувати номерні знаки там, де інший не зміг. Наприклад, на рис. 24-25 зображено вдалі кейси локалізації номерних знаків алгоритмом openCV (рис.24) на зображеннях, на яких модель YOLO не справилася із завданням (рис. 25). Однак були і зворотні результати: модель YOLO правильно локалізувала номерні знаки для зображень (рис. 26), на яких алгоритм openCV не реалізував поставленого завдання (рис. 27).

Рис. 24. Позитивні результати виділення номерних знаків за допомогою openCV-алгоритму, які не змогла виявити модель YOLO

Рис. 25. Невдачі при виділенні номерних знаків за допомогою моделі YOLO

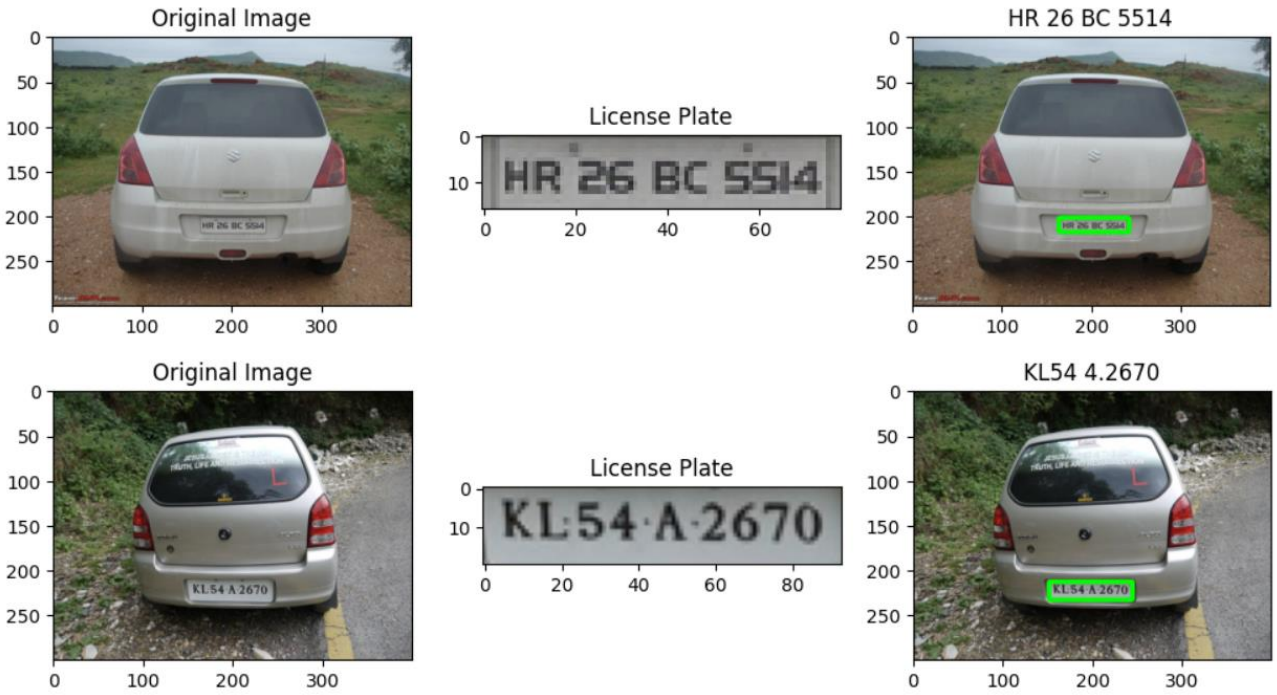


Рис. 26. Позитивні результати виділення номерних знаків за допомогою моделі YOLO, які не зміг виявити алгоритм openCV

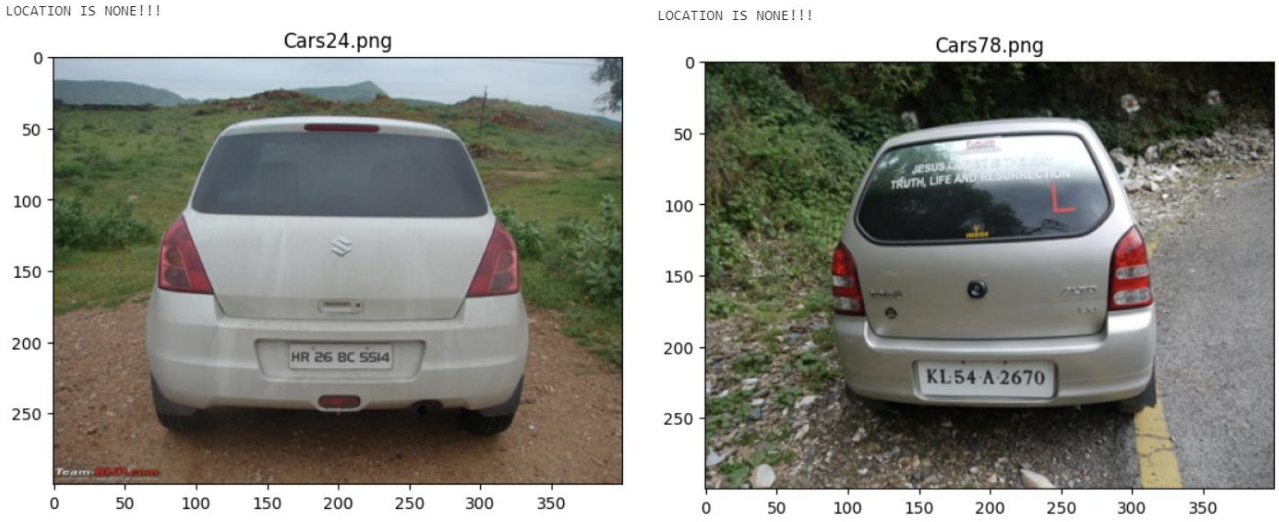


Рис. 27. Невдачі при виділенні номерних знаків за допомогою алгоритму openCV

Врахувавши вищезгадані аспекти було прийнято рішення об’єднати два алгоритми в один для отримання кращих результатів (відкидання меншої кількості зображень).

Алгоритм, отриманий в результаті об’єднання двох попередніх працює за наступним принципом:

Крок 1) Спочатку модель YOLO намагається виділити номерний знак на зображенні автомобіля. Якщо їй це не вдається –> Крок 2, інакше -> Крок 3.

Крок 2) Для виділення номерного знаку використовується алгоритм з методами openCV опрацювання зображень. Якщо номерний знак було виділено вдало –> Крок 3, інакше -> Крок 4.

Крок 3) Для перевірки того, що виділена область і справді є номерним знаком, використовується модель easyOCR. Якщо easyOCR змогла розпізнати текст -> Крок 6, інакше -> Крок 5.

Крок 4) Виведення повідомлення про те, що не вдалося виділити номерний знак та відображення оригінального зображення автомобіля. Вихід із алгоритму.

Крок 5) Виведення повідомлення про те, що на виділеній області не вдалося розпізнати текст та відображення оригінального зображення автомобіля. Вихід із алгоритму.

Крок 6) Зберігання даних про виділений номерний знак та відображення результатів виділення на оригінальному зображенні. Вихід із алгоритму.

В результаті виконання алгоритму наж зображеннями 100-199 було отримано хороші результати як завдяки алгоритму OpenCV (рис. 28), так і завдяки моделі YOLO (рис. 29).

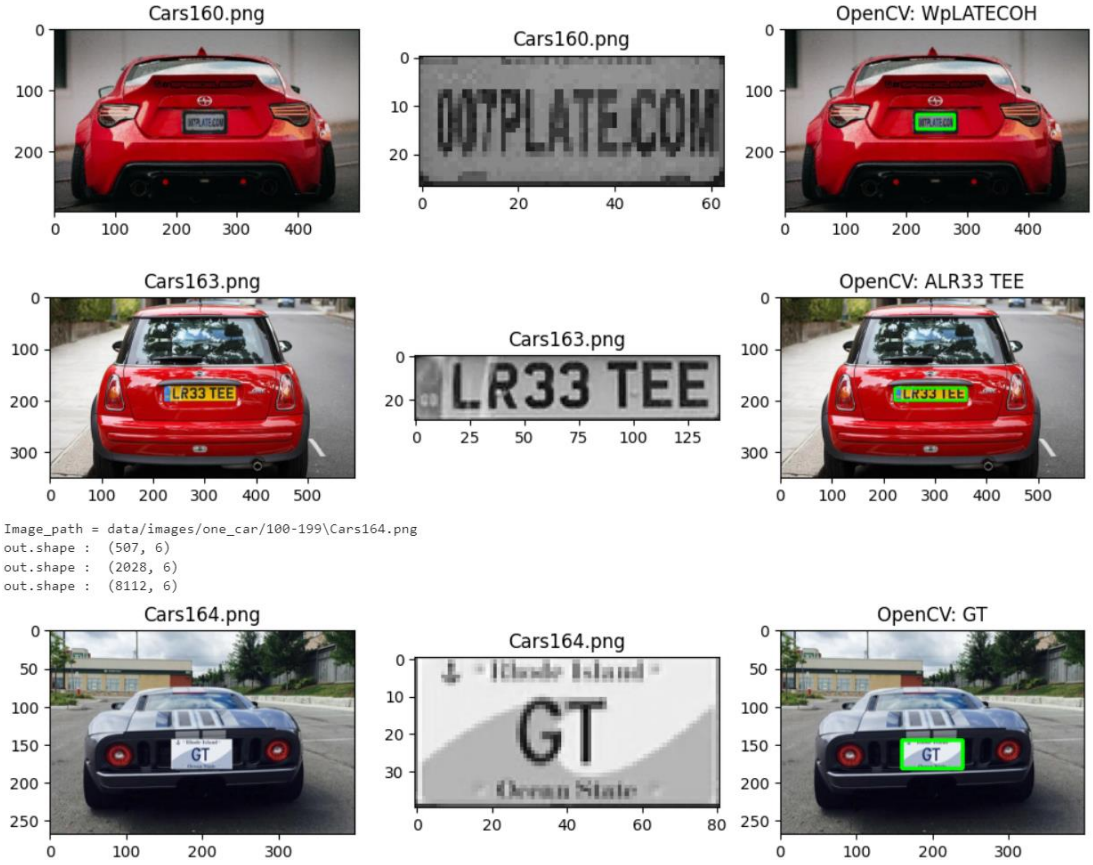


Рис. 28. Позитивні результати виділення номерних знаків за допомогою openCV-алгоритму

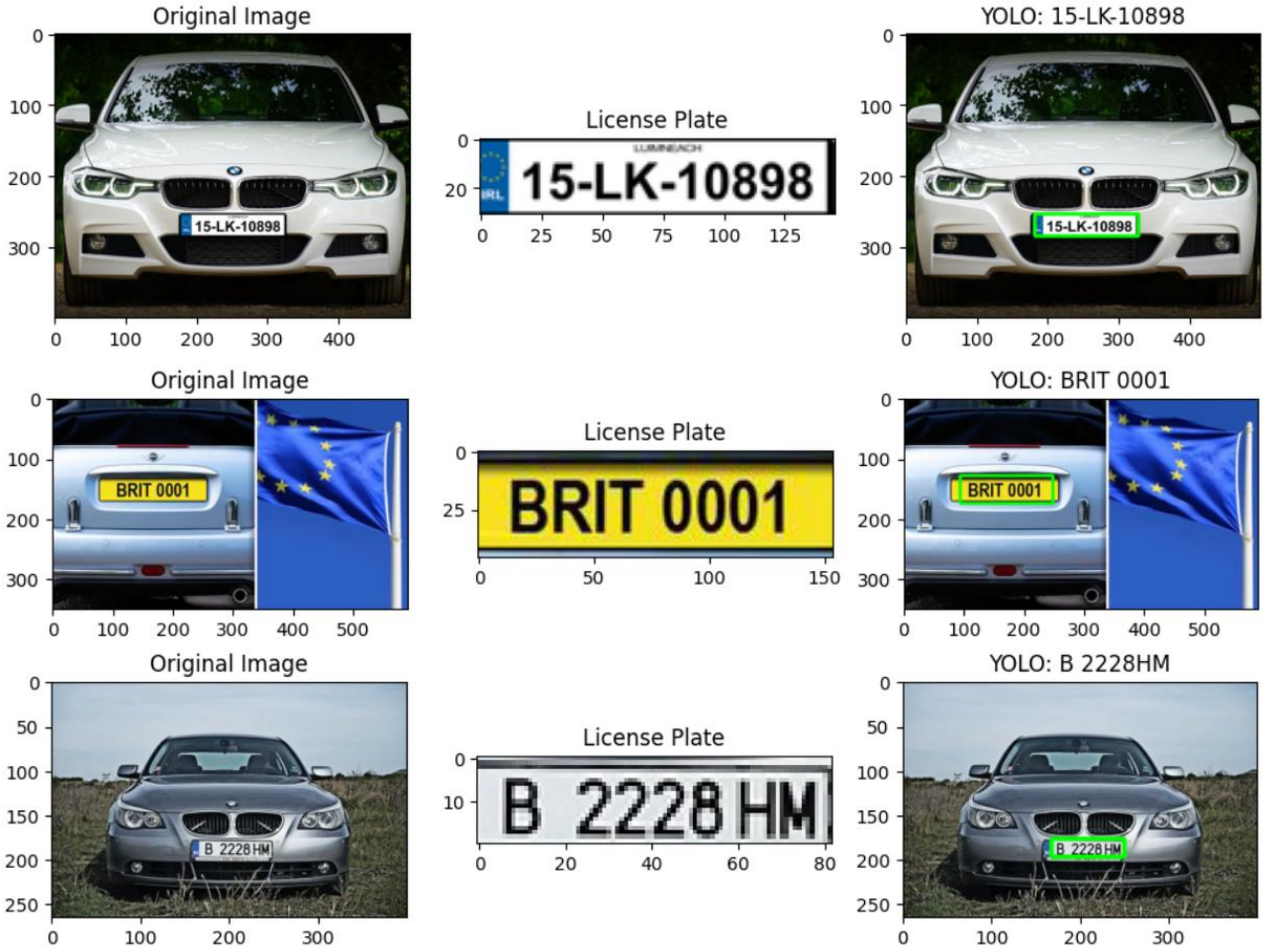


Рис. 29. Позитивні результати виділення номерних знаків за допомогою моделі YOLO

Однак через використання алгоритму openCV наявні випадки виділення одної - декількох літер замість цілого номерного знаку (рис. 30). На деяких зображеннях кінцевий алгоритм теж не зміг виявити номерні знаки (рис. 31).

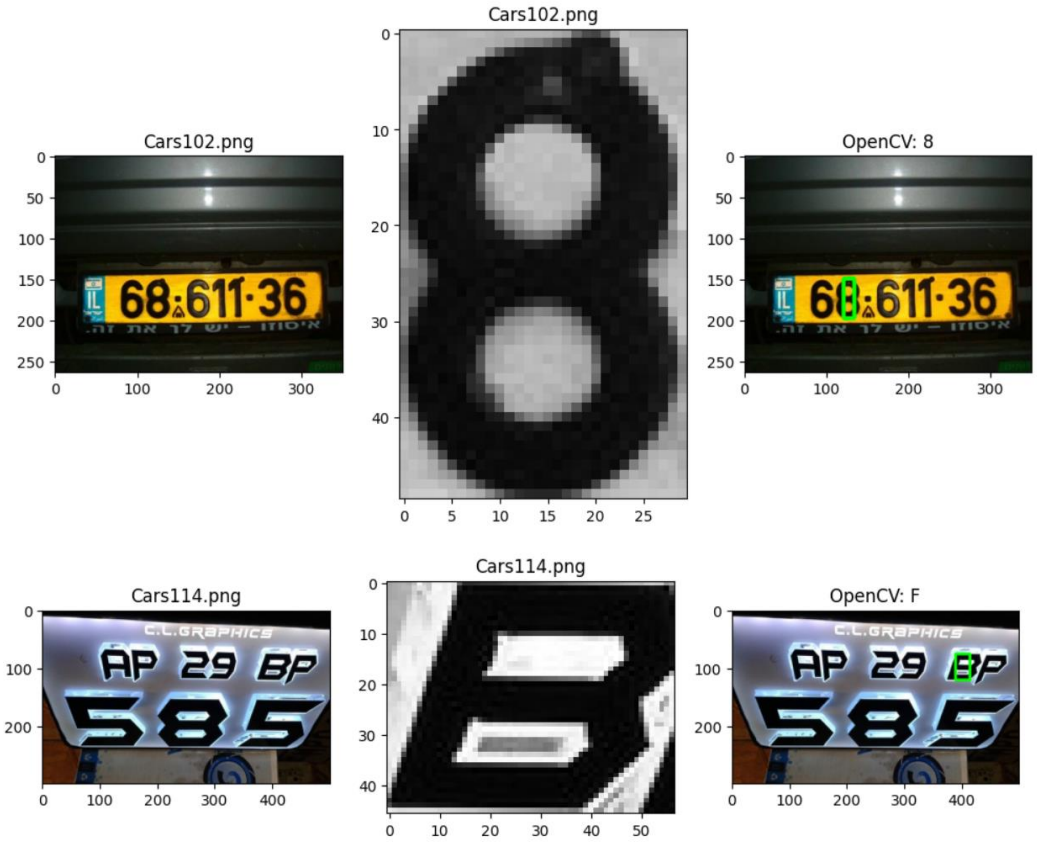


Рис. 30. Негативні наслідки використання алгоритму openCV-алгоритму: виділення лише одного-декількох символів замість номерних знаків

Рис. 31. Невдачі при виділенні номерних знаків за допомогою кінцевого алгоритму

Кінцевий алгоритм було застосовано до всіх зображень досліджуваного датасету, необхідні дані (image_name, x, y, width, height) було збережено в окремий датафрейм – licenses_df, перші 5 елементів якого наведені на рис. 32, а загальна інформація – на рис. 33.

	image_name	x	y	width	height
0	Cars0.png	256	127	153	46
1	Cars1.png	144	126	113	27
2	Cars11.png	130	199	142	38
3	Cars14.png	98	117	187	39
4	Cars18.png	312	281	94	35

Рис. 32. Елементи датафрейму licenses_df

```
licenses_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 233 entries, 0 to 13
Data columns (total 5 columns):
 #   Column      Non-Null Count  Dtype 
---  ------      --------------  ----- 
 0   image_name  233 non-null    object
 1   x           233 non-null    int64 
 2   y           233 non-null    int64 
 3   width       233 non-null    int64 
 4   height      233 non-null    int64 
dtypes: int64(4), object(1)
memory usage: 10.9+ KB
```

Рис. 33. Інформація про датафрейм licenses_df

Із рис. 33 видно, що внаслідок застосування кінцевого алгоритму із 433 зображень автомобілів було отримано лише 233 зображення номерних знаків. Тобто в ході опрацювання датасету залишилося лише 54% від загальної кількості зображень.

Етап 4: Визначення технік аугментації

Проаналізувавши різні умови, при яких може знадобитися розпізнавання номерних знаків, було виділено такі можливі техніки аугментації:

1) Rotation (повороти) - повертання зображення номерного знака під різними кутами, щоб імітувати зміни в його орієнтації.
2) Translation (зміщення) - зміщувати зображення номерного знака по горизонталі та вертикалі, щоб імітувати зміни в положенні номерного знака на зображенні.
3) Centering (Центрування) - штучне маніпулювання розташуванням центру об’єкта на зображенні.
4) Scaling (масштабування) - змінювати масштаб зображення номерного знака, щоб імітувати варіації відстані між камерою та номерним знаком. Це допомагає моделі узагальнити на різні відстані.

5) Shearing (зсув): застосування трансформації зсуву до зображення номерного знака, щоб імітувати спотворення перспективи, які можуть виникнути під час перегляду номерних знаків під різними кутами.
6) Brightness and Contrast Adjustment (налаштування яскравості та контрастності): довільне налаштування яскравості та контрастності зображення номерного знака, щоб імітувати варіації умов освітлення.
7) Data Augmentation with Backgrounds (доповнення даних за допомогою фону): комбінування зображення номерного знака з різними фонами, щоб імітувати варіації в середовищі, де знято номерні знаки.
8) Simulated Environmental Effects (імітація навколишнього середовища): додавання штучних ефектів навколишнього середовища, таких як дощ, сніг або туман, щоб імітувати складні погодні умови.
9) Noise Addition (додавання шуму): додавання різних типів шуму (наприклад, Гауса, salt and pepper) до зображення номерного знака, щоб імітувати шумне середовище або низьку якість зображення.
10) Color Space Transformations (трансформації колірного простору): перетворення зображення номерного знака в інший колірний простір (наприклад, RGB, HSV), щоб імітувати різні умови освітлення та варіації кольорів.
11) Rectification augmentation (ректифікаційне доповнення): штучне маніпулювання перспективою або вирівнюванням об’єктів (у реальних сценаріях номерні знаки можуть відображатися з різним ступенем спотворення перспективи через кут огляду камери), щоб підвищити стійкість моделі до варіацій вирівнювання об’єктів і спотворення перспективи.
12) Mirroring (відзеркалення): створити дзеркальну версію зображення, перевернувши його горизонтально (зліва направо).

Висновки

Реалізований алгоритм для локалізації номерних знаків використовує модель YOLO та алгоритми опрацювання зображень бібліотеки OpenCV, а також easyOCR для перевірки наявності тексту на виділених областях зображень з метою переконатися, що виділені області і справді є номерними знаками. Отриманий алгоритм далекий від ідеалу, оскільки показав хороші результати в більшості випадків лише для горизонтального положення автомобіля і номерного знака, спрямованих перпендикулярно або під невеликим кутом до спостерігача (камери і т.д.). Внаслідок застосування алгоритму із 433 зображень автомобілів було отримано лише 233 зображення номерних знаків, тобто майже половина зображень була відкинута. Окрім того, серед цих результатів є і не зовсім коректні: алгоритмом OpenCV було виділено лише один – декілька символів замість всього номерного знака. Тому алгоритм потребує значної модифікації.

Однією із можливих модифікацій, яка проте зменшить кількість записів вихідного датасету, є встановлення обмеження на кількість символів на номерному знаку: наприклад, якщо модель easyOCR змогла виділити на переданій їй області зображення автомобіля (область номерного знака) більше ніж 3 символи, то такий результат роботи алгоритму можна вважати номерним знаком.

Для збільшення зображень, які використовуватимуться в майбутньому було виділено 12 технік аугментації, які стосуються колірних перетворень, зміни масштабу та кута повороту автомобіля/номерного знаку.