

MINISTERUL EDUCAȚIEI



UNIVERSITATEA TEHNICĂ

DIN CLUJ-NAPOCA

FACULTATEA DE AUTOMATICĂ ȘI CALCULATOARE

SUPRAVEGHEREA VIDEO A TRAFICULUI

PROIECT DE DIPLOMĂ

Autor: **Ovidiu Vasile Claudiu PLEȘ**

Conducător științific: **Dr. ing. Mihai HULEA**

2024



Vizat,

DECAN

Prof. dr. ing. Mihaela DÎNȘOREANU

DIRECTOR DEPARTAMENT AUTOMATICĂ

Prof. dr. ing. Honoriu VĂLEAN

Autor: **Ovidiu Vasile Claudiu Pleș**

Supravegherea video a traficului

1. **Enunțul temei:** *Supravegherea video a traficului se concentrează pe identificarea, clasificarea și numărarea vehiculelor și are ca scop îmbunătățirea siguranței rutiere și a fluxului de trafic.*
2. **Conținutul proiectului:** *Pagina de prezentare; Declarație privind autenticitatea proiectului; Sinteza proiectului; Cuprins; Introducere; Studiu bibliografic; Analiză, proiectare, implementare; Concluzii; Bibliografie.*
3. **Locul documentării:** *Universitatea Tehnică din Cluj-Napoca*
4. **Consultanți:** *ing. Mihai Hulea*
5. **Data emiterii temei:**
6. **Data predării:**

Semnătura autorului

Semnătura conducătorului științific

**UNIVERSITATEA TEHNICĂ**

DIN CLUJ-NAPOCA

FACULTATEA DE AUTOMATICĂ ȘI CALCULATOARE

**Declarație pe proprie răspundere privind
autenticitatea proiectului de diplomă**

Subsemnatul(a) **Ovidiu Vasile Claudiu PLEȘ**,
legitimat(ă) cu CI/BI seria XM nr. 177002, CNP 5010409244483,
autorul lucrării:

Supravegherea video a traficului

elaborată în vederea susținerii examenului de finalizare a studiilor de licență la **Facultatea de Automatică și Calculatoare**, specializarea **Automatică și Informatică Aplicată**, din cadrul Universității Tehnice din Cluj-Napoca, sesiunea iulie 2024 a anului universitar 2023-2024, declar pe proprie răspundere, că această lucrare este rezultatul propriei activități intelectuale, pe baza cercetărilor mele și pe baza informațiilor obținute din surse care au fost citate, în textul lucrării, și în bibliografie.

Declar, că această lucrare nu conține porțiuni plagiate, iar sursele bibliografice au fost folosite cu respectarea legislației române și a convențiilor internaționale privind drepturile de autor.

Declar, de asemenea, că această lucrare nu a mai fost prezentată în fața unei alte comisii de examen de licență.

În cazul constatării ulterioare a unor declarații false, voi suporta sancțiunile administrative, respectiv, *anularea examenului de licență*.

Data

Ovidiu Vasile Claudiu PLEȘ

(semnătura)



SINTEZA

proiectului de diplomă cu titlul:

Supravegherea video a traficului

Autor: **Ovidiu Vasile Claudiu PLEȘ**

Conducător științific: **Dr. ing. Mihai HULEA**

1. Cerințele temei: Proiectul în discuție are ca și scop atât identificarea, clasificarea și numărarea vehiculelor, cât și identificarea și citirea automată a numerelor de înmatriculare a autovehiculelor vizibile într-un format de tip video.
2. Soluții alese: Se folosește limbajul de programare Python pentru dezvoltarea algoritmilor necesari și doua modele de rețele neuronale pentru detecția imaginilor și pentru citirea imaginilor.
3. Rezultate obținute: S-a obținut un program capabil să identifice, clasifice și să numere vehiculele și numerele de înmatriculare ale acestora într-un timp relativ scurt.
4. Testări și verificări: Programul a fost testat folosind un set de date preluat de la o camera video de supraveghere a traficului aflată în municipiul Cluj-Napoca.
5. Contribuții personale: Contribuțiile personale implică pregătirea datelor folosite la antrenarea modelelor de rețea neuronală și dezvoltarea algoritmilor și a programului.
6. Surse de documentare: Bibliografia include articole științifice, cărți și surse online.

Semnătura autorului

Semnătura conducătorului științific

Cuprins

1	INTRODUCERE.....	3
1.1	CONTEXT GENERAL	3
1.2	OBIECTIVE.....	4
1.3	SPECIFICAȚII	4
1.3.1	<i>Antrenarea unui model de rețea neuronală pentru identificarea autovehiculelor și a plăcilor de înmatriculare ale acestora</i>	<i>4</i>
1.3.2	<i>Identificarea autovehiculelor dintr-o imagine folosind modelul de rețea neuronală</i>	<i>4</i>
1.3.3	<i>Clasificarea autovehiculelor în patru categorii: "Mașină mică", „Camion”, „Camionetă” și „Autobuz”</i>	<i>5</i>
1.3.4	<i>Implementarea unui algoritm de urmărire a autovehiculelor în contextul unui format video</i>	<i>5</i>
1.3.5	<i>Numărarea autovehiculelor în contextul unui format video</i>	<i>5</i>
1.3.6	<i>Numărarea autovehiculelor în funcție de categoria autovehiculului</i>	<i>5</i>
1.3.7	<i>Numărarea autovehiculelor în funcție de banda pe care se află pe șosea</i>	<i>5</i>
1.3.8	<i>Identificarea plăcilor de înmatriculare pentru fiecare autovehicul folosind modelul de rețea neuronală</i>	<i>5</i>
1.3.9	<i>Identificarea textului de pe plăcile de înmatriculare folosind un model de rețea neuronală</i>	<i>6</i>
1.3.10	<i>Optimizarea cât mai bună a tuturor proceselor pentru a putea fi folosite într-un context de timp real</i>	<i>6</i>
1.3.11	<i>Împlementarea unei interfețe de utilizator</i>	<i>6</i>
2	STUDIUL BIBLIOGRAFIC.....	7
2.1	PROCESAREA IMAGINILOR	7
2.1.1	<i>Filtrarea imaginilor</i>	<i>7</i>
2.1.1.1	<i>Filtrul Gaussian</i>	<i>7</i>
2.1.1.2	<i>Filtrul Laplacian</i>	<i>7</i>
2.1.1.3	<i>Filtrul Sobel</i>	<i>8</i>
2.1.1.4	<i>Binarizarea imaginilor</i>	<i>8</i>
2.1.2	<i>Identificarea proprietăților imaginilor</i>	<i>8</i>
2.1.2.1	<i>Transformarea de caracteristici invariante la scară</i>	<i>8</i>
2.2	MODELE DE REȚELE NEURONALE.....	9
2.2.1	<i>Structura modelelor de rețele neuronale</i>	<i>9</i>
2.2.1.1	<i>Concepte de bază.....</i>	<i>9</i>
2.2.1.2	<i>Tipuri de rețele neuronale</i>	<i>10</i>
2.2.2	<i>Funcționarea rețelelor neuronale</i>	<i>11</i>
2.2.2.1	<i>Funcții de activare.....</i>	<i>11</i>
2.2.2.2	<i>Antrenarea rețelelor neuronale.....</i>	<i>11</i>
2.2.3	<i>Aplicații ale rețelelor neuronale</i>	<i>12</i>
2.3	DETECȚIA ȘI CLASIFICAREA OBIECTELOR	12
2.3.1	<i>Metode tradiționale de identificare a obiectelor</i>	<i>13</i>
2.3.1.1	<i>Cascada Haar.....</i>	<i>13</i>
2.3.1.2	<i>Histogramă de gradienti orientați.....</i>	<i>13</i>
2.3.2	<i>Rețele neuronale convoluționale pentru detecția obiectelor</i>	<i>13</i>
2.3.2.1	<i>Structura rețelelor neuronale convoluționale pentru detecția obiectelor</i>	<i>14</i>
2.3.2.2	<i>Algoritmul YOLO.....</i>	<i>15</i>
2.3.3	<i>Identificarea textului.....</i>	<i>15</i>
2.4	ANALIZA, URMĂRIREA ȘI NUMĂRAREA OBIECTELOR.....	16
2.4.1	<i>Urmărirea obiectelor în contextul formatelor video</i>	<i>16</i>

2.4.1.1	Kanade-Lucas-Tomasi	17
2.4.1.2	Folosirea transformării de caracteristici invariante la scară	17
2.4.1.3	Filtrarea Kalman	17
2.4.1.4	Algoritmul SORT	17
2.4.2	Numărarea obiectelor	18
2.5	INTERFEȚE DE UTILIZATOR	19
2.6	POSIBILE SOLUȚII ȘI ABORDAREA ALEASĂ	20
3	ANALIZĂ, PROIECTARE, IMPLEMENTARE	21
3.1	TEHNOLOGII FOLOSITE ȘI MEDII DE DEZVOLTARE	21
3.1.1	Limbajul de programare Python	21
3.1.2	Mediul de dezvoltare integrat PyCharm	21
3.1.3	Sistemul de control Git	21
3.1.4	Platforma GitHub	22
3.1.5	Modelul de detecție YOLOv8	22
3.1.6	Biblioteca OpenCV	23
3.1.7	Biblioteca EasyOCR	23
3.1.8	Biblioteca Pandas	23
3.1.9	Qt pentru Python	24
3.2	DETALII DE PROIECTARE	24
3.2.1	Cazuri de utilizare	24
3.2.2	Structura aplicației	25
3.3	IMPLEMENTAREA PROIECTULUI	28
3.3.1	Configurarea mediului de dezvoltare	28
3.3.2	Antrenarea modelului de rețea neuronală pentru detecția vehiculelor	29
3.3.2.1	Achiziția datelor	29
3.3.2.2	Anotarea datelor	29
3.3.2.3	Antrenarea modelului YOLO	31
3.3.3	Implementarea algoritmului de urmărire	32
3.3.3.1	Algoritmul de bază	32
3.3.3.2	Gestionarea lipsei de detecții	34
3.3.3.3	Optimizarea algoritmului	35
3.3.4	Numărarea autovehiculelor	36
3.3.4.1	Numărarea simplă și în funcție de clasă	36
3.3.4.2	Numărarea în funcție de benzile rutiere	37
3.3.5	Detecția plăcilor de înmatriculare	39
3.3.5.1	Achiziția datelor și antrenarea modelului	39
3.3.5.2	Recunoașterea numerelor de înmatriculare	40
3.3.5.3	Optimizarea algoritmului	41
3.3.6	Implementarea interfeței grafice pentru utilizator	43
3.3.6.1	Elementele grafice	43
3.3.6.2	Funcționalitățile interfeței	44
3.3.6.3	Generarea unui fișier executabil	47
3.4	TESTARE ȘI VALIDARE	48
3.4.1	Validarea detecțiilor și a algoritmilor	48
3.4.2	Testarea interfeței de utilizator și a funcționalităților	50
4	CONCLUZII	52
4.1	REZULTATE OBTINUTE	52
4.2	DIRECȚII DE DEZVOLTARE	53
5	BIBLIOGRAFIE	54

1 Introducere

1.1 Context general

În ultimul secol, apariția vehiculelor pe drumurile publice a avut o creștere exponențială. Odată cu această creștere, controlul traficului public, a autovehiculelor și a șoferilor de autovehicule a fost din ce în ce mai greu de pus în aplicare.

Creșterea numărului de autovehicule utilizate a avut nenumărate beneficii pentru omenire. Aceasta a favorizat dezvoltarea industriei transportului și a agriculturii, a creat un număr semnificativ de locuri de muncă, a oferit posibilitatea intervențiilor de urgență să se desfășoare mai repede și nu în ultimul rând, a oferit cetățenilor mobilitate pentru nevoi profesionale și personale.

Cu toate acestea, utilizarea autovehiculelor a avut și un impact negativ asupra societății și a mediului global: a crescut nivelul de poluare a aerului prin emanarea dioxidului de carbon, ceea ce a dus la probleme de sănătate în rândul oamenilor, a crescut numărul de accidente de pe drumurile publice și a creat aglomerație de trafic.

Dat fiind acest context, tehnologia evoluată în ultimii ani poate fi folosită pentru ameliorarea unor efecte negative ale apariției autovehiculelor menționate mai sus. Motivația acestei lucrări este îmbunătățirea siguranței de pe drumurile publice și reducerea aglomerației din trafic. Obiectivele lucrării includ recunoașterea automată a autovehiculelor și a numerelor de înmatriculare ale acestora, numărarea și clasificarea autovehiculelor și stocarea eficientă a datelor obținute în urma acestor procese astfel încât aceste date să poată fi folosite cu ușurință pentru îmbunătățirea controlului traficului contemporan.

Această lucrare este importantă și merită o deosebită atenție încât contribuie în primul rând la siguranța tuturor cetățenilor și la bunăstarea acestora, încurajând conceptul de automatizare a orașelor.

Am ales această temă din dorința de a ajuta la dezvoltarea rutieră locală și datorită pasiunii față de inovare, dezvoltare tehnologică și optimizare a resurselor.

În continuare se vor prezenta următoarele capitole:

- Studiu bibliografic: Conține o sinteză a tehnicilor existente de detecție și urmărire a obiectelor și alte aspecte utile pentru dezvoltarea unei aplicații.
- Analiza, proiectare, implementare: Sunt prezentate contribuțiile personale, detaliile tehnice și toate informațiile necesare pentru reproducerea lucrării.
- Concluzii: Conține o sinteză a rezultatelor obținute și o descriere a viitoarelor direcții posibile de dezvoltare.
- Bibliografie: Conține sursele de informații necesare pentru dezvoltarea lucrării.

1.2 Obiective

Obiectivele principale ale proiectului sunt detecția, numărarea și clasificarea vehiculelor în timp real pentru a ajuta la îmbunătățirea siguranței generale de pe drumurile publice și pentru a optimiza fluxul de trafic.

Principalele obiective din punct de vedere al funcționalităților sunt următoarele:

- Antrenarea unui model de rețea neuronală pentru identificarea autovehiculelor și a plăcilor de înmatriculare ale acestora.
- Identificarea autovehiculelor dintr-o imagine folosind modelul de rețea neuronală.
- Clasificarea autovehiculelor în patru categorii: „Mașină mică”, „Camion”, „Camionetă” și „Autobuz”.
- Implementarea unui algoritm de urmărire a autovehiculelor în contextul unui format video.
- Numărarea autovehiculelor în contextul unui format video.
- Numărarea autovehiculelor în funcție de categoria autovehiculului.
- Numărarea autovehiculelor în funcție de banda pe care se află pe șosea.
- Identificarea plăcilor de înmatriculare pentru fiecare autovehicul folosind modelul de rețea neuronală.
- Identificarea textului de pe plăcile de înmatriculare folosind un model de rețea neuronală.
- Optimizarea cât mai bună a tuturor proceselor pentru a putea fi folosite într-un context de timp real.
- Împlementarea unei interfețe de utilizator.

1.3 Specificații

1.3.1 Antrenarea unui model de rețea neuronală pentru identificarea autovehiculelor și a plăcilor de înmatriculare ale acestora

Antrenarea modelelor de rețea neuronală se face folosind date preluate de la camera de supraveghere video a traficului localizată în stația “Regionala CFR” din municipiul Cluj-Napoca. Datorită rezoluției imaginilor provenite de la aceasta camera video, se vor antrena doua modele de rețea neuronală, una pentru identificarea și clasificarea autovehiculelor și una pentru identificarea plăcilor de înmatriculare. Pentru identificarea textului de pe plăcile de înmatriculare se va folosi un model gata antrenat și generalizat, nespecializat pe recunoașterea unui anumit format de date.

1.3.2 Identificarea autovehiculelor dintr-o imagine folosind modelul de rețea neuronală

Se vor folosi modelele antrenate menționate la subcapitolul anterior pentru a determina coordonatele împrejurimilor fiecărui vehicul și a fiecărei plăci de înmatriculare.

1.3.3 Clasificarea autovehiculelor în patru categorii: „Mașină mică”, „Camion”, „Camionetă” și „Autobuz”

Se va realiza clasificarea autovehiculelor în patru categorii diferite: „Mașină mică”, „Camion”, „Camionetă” și „Autobuz”. Se va folosi modelul de rețea neuronală folosit și la identificarea autovehiculelor deoarece acesta a fost antrenat pentru îndeplinirea ambelor scopuri.

1.3.4 Implementarea unui algoritm de urmărire a autovehiculelor în contextul unui format video

Se va implementa un algoritm de tip „intersecție peste reuniune” și vor analiza și compara vehiculele din fiecare imagine corespunzătoare formatului video cu vehicule din imaginea anterioară.

Având în vedere faptul ca transmisia de imagini de la camera video se întrerupe regulat pentru câteva milisecunde, se va adauga un parametru acestui algoritm care va corespunde cu numărul de imagini corespunzătoare formatului video care să fie luate în considerare în execuția algoritmului. Acest fapt va elimina întreruperea urmăririi autovehiculelor.

1.3.5 Numărarea autovehiculelor în contextul unui format video

Numărarea autovehiculelor în contextul unui format video se bazează în primul rând pe algoritmul de urmărire descris la capitolul anterior. Acest algoritm asigură faptul că niciun autovehicul nu va fi numărat de doua ori.

Pentru a număra un vehicul identificat de algoritmul de urmărire, se va alege o linie orizontală specifică fiecărei imagini dintr-un format video și se vor lua în considerare autovehiculele ale căror coordonate intersectează această linie.

1.3.6 Numărarea autovehiculelor în funcție de categoria autovehiculului

Numărarea autovehiculelor în funcție de categorie se bazează și ea pe algoritmul de urmărire. Acest algoritm asigură o structură de date în care este stocată categoria vehiculului identificat, pe lângă alte informații utile. Folosind această structură de date și conceptul de numărare descris la subcapitolul anterior, acest tip de numărare este posibil.

1.3.7 Numărarea autovehiculelor în funcție de banda pe care se află pe șosea

Acest tip de numărare necesită o abordare aparte. Pe lângă folosirea algoritmului de urmărire și a liniei orizontale de intersecție, se va identifica marcajul rutier și se vor lua în considerare atât aceste marcaje cât și dimensiunile autovehiculelor. Acest tip de numărare va ajuta la gestionarea fluxului de trafic.

1.3.8 Identificarea plăcilor de înmatriculare pentru fiecare autovehicul folosind modelul de rețea neuronală

Pentru această identificare se va folosi al doilea model de rețea neuronală. Din cauza rezoluțiilor imaginilor, pentru fiecare autovehicul se va genera o nouă imagine care va conține numai acel vehicul. Cu această nouă imagine se va face identificarea plăcii de înmatriculare.

1.3.9 Identificarea textului de pe plăcile de înmatriculare folosind un model de rețea neuronală

Fiecare identificare a fiecărei plăci de înmatriculare va genera o nouă imagine care va fi folosită de un model gata antrenat, disponibil public, specializat în recunoașterea textului din imagini. După ce acest text a fost identificat, acesta va fi verificat și corectat de un algoritm care recunoaște și corectează formatul numerelor de înmatriculare din România.

1.3.10 Optimizarea cât mai bună a tuturor proceselor pentru a putea fi folosite într-un context de timp real

Se vor alege modele de rețele neuronale care favorizează timpul de execuție peste acuratețea rezultatelor. Acest aspect este foarte important pentru adaptarea sistemului într-un context de timp real. De asemenea, toți algoritmi care contribuie la procesarea imaginilor vor fi optimizați astfel încât aceștia să se execute într-un timp cât mai scurt posibil.

1.3.11 Implementarea unei interfețe de utilizator

Pentru ca toate aceste funcționalități să poată fi folosite, se va implementa o interfață de utilizator. Obiectivul principal al acestei interfețe este de a da posibilitatea utilizatorului să vadă în timp real imaginile procesate primite de la camera de supraveghere, să aleagă opțiunile și parametrii de procesare a imaginilor care vor afecta acuratețea și timpul de răspuns al procesării, să poată urmări în timp real numere specifice de înmatriculare, să poată încărca videoclipuri care să fie procesate și să genereze statistici bazate pe videoclipurile încărcate și procesate. Toate aceste funcționalități vor fi implementate într-un mod în care utilizatorului îi va fi ușor și intuitiv să le folosească.

2 Studiu bibliografic

2.1 Procesarea imaginilor

Procesarea imaginilor se referă la o serie de tehnici utilizate pentru a analiza, manipula și modifica informațiile găsite în imagini în diferite moduri și pentru a obține rezultatele dorite.

2.1.1 Filtrarea imaginilor

Conform [10], filtrarea imaginilor constă în modificarea fiecărui pixel dintr-o imagine pe baza pixelilor dintr-un bloc al matricii de pixeli care formează imaginea. Acest bloc al matricii este de dimensiuni variabile, de obicei pătratic cu dimensiuni impare, elementul de la intersecția diagonalelor având valoarea pixelului care trebuie modificat.

Această modificare a pixelilor este realizată aplicând un filtru tuturor blocurilor din matricea de pixeli care formează imaginea. Acest filtru, numit și operator de convoluție, matrice de convoluție sau kernel, este o matrice cu dimensiunea egală cu dimensiunea blocurilor alese. Procesul de filtrare constă în convoluția fiecărui bloc al matricii de pixel cu matricea de convoluție, fiecare convoluție având ca rezultat noua valoare a pixelului din matricea de pixeli care formează imaginea filtrată.

2.1.1.1 Filtrul Gaussian

Conform [31], filtrul Gaussian este un filtru trece jos care permite trecerea frecvențelor joase și atenuează frecvențele înalte. Acest efect aplicat asupra pixelilor dintr-o imagine rezultă în atenuarea zgomotului și a detaliilor din imagine.

Pentru a aplica acest tip de filtru este necesară generarea unei matrici de convoluție bazată pe funcția Gaussiană continuă reprezentată în relația (2.1). Această funcție se aplică pentru punctele discrete din vecinătatea pixelului central (0, 0).

$$G(x, y) = \frac{1}{2 \times \pi \times \sigma^2} \times e^{-\frac{x^2+y^2}{2 \times \sigma^2}} \quad (2.1)$$

După aplicarea funcției pentru fiecare punct discret, valorile sunt normalizate astfel încât suma lor să fie 1. Acest lucru asigură faptul că imaginea nu o să își schimbe luminozitatea după aplicarea filtrului. [31] Astfel, pentru o matrice de convoluție de dimensiune 3×3 și $\sigma = 1$, obținem un filtru de forma:

$$K = \frac{1}{16} \times \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix} \quad (2.2)$$

2.1.1.2 Filtrul Laplacian

Conform [32], filtrul Laplacian se folosește pentru a identifica contrastele și muchiile imaginilor, scoțând în evidență detaliile. Acest tip de filtru are de obicei forma:

$$K = \begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix} \quad (2.3)$$

2.1.1.3 Filtrul Sobel

Potrivit [11], acest tip de filtru se folosește pentru detecția marginilor și conturilor obiectelor. Se folosesc două matrici de convoluție, una pentru identificarea orizontală și una pentru identificarea verticală. Cele două filtre sunt de forma:

$$K_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}, K_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix} \quad (2.4)$$



Figura 2.1 Aplicarea filtrului Sobel [4]

2.1.1.4 Binarizarea imaginilor

Procesul de binarizare a imaginilor este o tehnică de transformare în care imaginea este mai întâi convertită în tonuri de gri, apoi se alege o valoare de prag. Această valoare poate fi aleasă manual sau calculată folosind diferite metode. Procesarea imaginii constă în modificarea fiecărui pixel în raport cu valoarea de prag. Dacă valoarea de prag este mai mare decât valoarea intensității pixelului, noua valoare a pixelului va fi 0 (pixel negru), altfel valoarea lui va fi 255 (pixel alb). [10]

2.1.2 Identificarea proprietăților imaginilor

Identificarea proprietăților sau a caracteristicilor din imagini presupune identificarea unor puncte distinctive din imagine, numite și puncte cheie, și descrierea acestor puncte și a împrejurimilor lor. Acest tip de procesare a imaginilor este foarte folositoare pentru detecția obiectelor din imagini.

2.1.2.1 Transformarea de caracteristici invariante la scară

Conform [12], transformarea de caracteristici invariante la scară (în engleză: Scale-Invariant Feature Transform sau SIFT) este un algoritm foarte folosit de detecție a caracteristicilor imaginilor. Caracteristicile identificate de acest algoritm sunt independente față de scara și rotația imaginilor și parțial independente față de luminozitatea imaginilor.

Primul pas al acestui algoritm este identificarea punctelor cheie din imagine. Se aplică o serie de filtre Gaussiene cu σ variabil, apoi se face diferența imaginilor rezultate după aplicarea acestor filtre. Potențialele puncte cheie sunt punctele de intensitate minimă sau maximă locală, luând în considerare diferite versiuni ale imaginii. Aceste puncte sunt filtrate, rezultând punctele cheie finale. [12]

Pentru a descrie imaginea din jurul punctelor cheie, se calculează o serie de gradienti, rezultând un vector de descriere pentru fiecare vecinătate a fiecărui punct cheie. [12]

2.2 Modele de rețele neuronale

Modelele de rețele neuronale sunt o clasă de modele de învățare automată inspirată de structura și funcționarea creierului uman. Acestea pot fi folosite pentru diverse aplicații, o aplicație comună fiind detecția de obiecte din imagini.

2.2.1 Structura modelelor de rețele neuronale

2.2.1.1 Concepte de bază

Potrivit [2], unitatea de bază de procesare a unui model de rețea neuronală este neuronul, precum în rețelele neuronale umane. Acesta are o intrare și o ieșire, funcția lui fiind de a procesa intrarea și de a o trimite mai departe la alți neuroni. Procesarea constă în aplicarea unei funcții de activare asupra intrării.

Neuronii sunt grupați în straturi care compun rețeaua neuronală, a căror rol este de a procesa informația de intrare și de a o trimite următorului strat. Fiecare strat al unei rețele are un rol important și determină performanța și capacitatea totală a rețelei. Tipurile de straturi sunt următoarele:

- **Straturi de intrare:** Acestea sunt primele straturi ale rețelei neuronale și au rolul de a transmite informația inițială spre următoarele straturi. În acest tip de straturi nu există funcții de activare. Numărul de neuroni care compun aceste straturi este egal cu numărul de atribute al intrării, de exemplu, pentru o imagine de dimensiune $n \times m$ pixeli și 3 culori, numărul de neuroni din acest strat va fi egal cu $3 \times n \times m$.
- **Straturi ascunse:** În aceste straturi se execută majoritatea calculelor, fiecare neuron aplicând o funcție de activare pentru intrarea primită de la stratul anterior. Numărul de neuroni care compun aceste straturi variază în funcție de complexitatea problemelor.
- **Straturi de ieșire:** Aceste straturi produc rezultatul final al rețelei neuronale, numărul de neuroni corespunzând cu numărul de clase atunci când se dorește clasificarea intrării sau cu numărul de valori de ieșire în cazul regresiei.

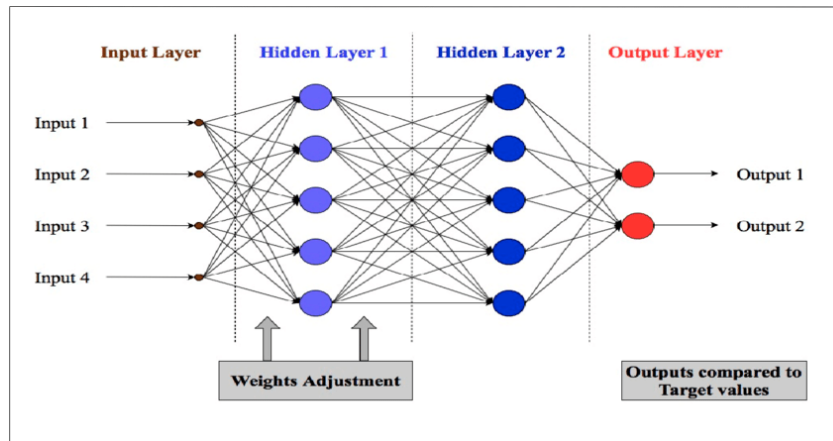


Figura 2.2. Structura de bază a rețelelor neuronale [3]

Alte două componente importante ale structurii modelelor de rețele neuronale sunt greutatea și predispozițiile (în engleză: “weights” și “biases”). [3] Acestea sunt valori atribuite legăturilor dintre neuroni, respectiv neuronilor fără de care nu ar fi posibilă învățarea automată a rețelei. Greutățile atribuite conexiunilor dintre neuroni reprezintă importanța contribuției fiecărei intrări la ieșirea unui neuron. Aceste valori se ajustează pe parcursul procesului de învățare automată. Predispozițiile sunt valori atribuite fiecărui neuron și au rolul de prag de activare. Dacă suma greutateilor intrărilor unui neuron minus predispoziția neuronului este mai mare decât zero, neuronul este activ, altfel acesta rămâne inactiv.

2.2.1.2 Tipuri de rețele neuronale

Conform [1], există multe tipuri de rețele neuronale și diferite implementări pentru acestea, fiecare având un scop bine definit. Printre cele mai populare tipuri se află:

- Rețele Neuronale Feedforward (în engleză: Feedforward Neural Networks sau FNN): Au cea mai simplă structură, datele de intrare parcurgând o singură direcție, din stratul de intrare prin straturile ascunse, ajungând la stratul de ieșire. Fiecare neuron dintr-un strat este conectat cu toți neuronii din stratul următor. Se folosește pentru recunoașterea imaginilor și probleme simple de clasificare.
- Rețele Neuronale Convoluționale (în engleză: Convolutional Neural Networks sau CNN): Sunt specializate pentru detecția de imagini, clasificarea imaginilor și detecția de obiecte din imagini. Se folosesc straturi ascunse convoluționale care execută operații de înmulțire element cu element a matricilor, lucru care permite aplicarea filtrelor asupra imaginii de intrare și recunoașterea de caracteristici din imagine.
- Rețele Neuronale Recurente (în engleză: Recurrent Neural Networks sau RNN): Aceste tipuri de rețele au fost implementate pentru cazurile în care intrările rețelei sunt influențate de intrări anterioare. Se folosesc conexiuni recurente de neuroni care permit informației să persiste în timp. Un caz de utilizare a acestor rețele este prezicerea unor valori din viitor bazată pe valori anterioare.

2.2.2 Funcționarea rețelelor neuronale

2.2.2.1 Funcții de activare

Potrivit [13], funcțiile care procesează semnalul de intrare al unui neuron și îl transformă în semnal de ieșire se numesc funcții de activare. Acestea se aleg în funcție de tipul și scopul rețelei neuronale, adesea existând rețele cu multiple funcții de activare. Alegeri populare pentru aceste funcții sunt următoarele:

- ReLU (“Rectified liniar unit” sau “Unitate liniară rectificată”): Se folosește în general pentru straturile ascunse datorită simplității și eficienței.

$$ReLU(x) = \max(0, x) \quad (2.5)$$

- Sigmoid: Se folosește în general pentru reprezentarea probabilităților datorită faptului că ieșirea funcției este între 1 și 0.

$$S(x) = \frac{1}{1 + e^{-x}} \quad (2.6)$$

- Tanh (Tangenta hiberbolică): Se folosește în general pentru date centrate pe zero, ieșirea funcției fiind între -1 și 1.

$$\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \quad (2.7)$$

2.2.2.2 Antrenarea rețelelor neuronale

Conform [14], primul pas în antrenarea rețelelor neuronale este achiziția datelor. Este necesar un set de date relevant pentru problema care este pusă în discuție, de exemplu, în cazul detecției de obiecte este nevoie de imagini care conțin acel tip de obiect. Cu cât setul de date este mai diversificat, respectiv cu cât imaginile surprind tipul respectiv de obiect în diferite ipostaze, cu atât învățarea automată a rețelei va fi mai eficientă. De obicei este nevoie de un set relativ mare de date care este împărțit în trei tipuri:

- Date de antrenare: Sunt folosite pentru învățarea automată a rețelei și constituie cea mai mare parte din cele trei tipuri, de obicei aproximativ 70% din totalul de date.
- Date de validare: Aceste date sunt folosite pentru validarea procesului de antrenare și pentru a monitoriza procesul de învățare și reprezintă aproximativ 15% din totalul de date.
- Date de testare: Aceste date nu sunt folosite deloc în procesul de învățare automată și au rolul de a testa și de a evalua modelul odată ce acesta este gata antrenat. Reprezintă de obicei aproximativ 15% din totalul de date.

Datele colectate sunt adăugate ca și intrări pentru rețeaua neuronală care le preia, aplică funcțiile de activare pentru fiecare strat, și ținând cont de greutatea și predispoziții, generează o ieșire. Pentru a măsura diferența dintre ieșirea rețelei și rezultatul dorit, se folosește de obicei eroarea medie pătratică în cazul regresiiilor sau pierderea de entropie încrucișată (în engleză: cross-entropy loss) în cazul clasificărilor. Pe baza acestor diferențe, greutatea rețelei sunt actualizate. [14]

Un aspect foarte important este antrenarea în buclă. Procesarea datelor de antrenare și actualizarea greutateilor rețelei trebuie executată de multiple ori pentru întreg setul de date de antrenare, păstrându-se de fiecare dată greutateile rezultate la iterația precedentă. Periodic, se va folosi setul de date de validare pentru a monitoriza procesul de învățare și pentru a preveni fenomenul de supraantrenare. [14]

Fenomenul de supraantrenare se referă la modul în care rețeaua este supusă procesului de învățare descris mai sus de prea multe ori pentru același set de date. În acest caz, rețeaua va performa excelent pentru setul de date de antrenare dar nu va putea să recunoască trăsăturile unui nou set de date, cum ar fi cel de testare.

După ce procesul de antrenare este finalizat, rețeaua este evaluată folosind setul de date de testare pentru a estima performanța generală a acesteia. Pentru rezultate cât mai bune, este recomandată monitorizarea constantă a performanței modelului de rețea și adăugarea de date noi și diversificate pentru a menține acuratețea.

2.2.3 Aplicații ale rețelelor neuronale

Modelele de rețele neuronale au o gamă largă de aplicații în diferite domenii datorită abilității acestora de a procesa și recunoaște structuri și tipare complexe în diferite tipuri de date. [14] Câteva dintre cele mai des întâlnite aplicații sunt următoarele:

- Recunoașterea imaginilor: În special rețelele neuronale convoluționale sunt folosite pentru detecția și clasificarea obiectelor din imagini sau pentru detecția bolilor și a tumorilor în cazul radiografiilor medicale.
- Procesarea limbajului natural: Rețelele neuronale recurente se folosesc pentru traducerea textului dintr-o limbă în alta și analizarea limbajului comun pentru a obține diferite informații. De asemenea, acestea se pot folosi pentru transformarea limbii vorbite în text.
- Aplicații în securitate: Rețelele neuronale pot fi folosite pentru a identifica diferite tipare de pe camerele de supraveghere pentru a preveni acțiuni neautorizate sau diferite comportamente ale programelor în cazul securității digitale.
- Aplicații în educație: Examenele și lucrările de control din cadrul sistemului de învățământ se pot corecta automat cu ajutorul rețelelor neuronale. De asemenea se pot genera lecții și cursuri personalizate pentru studenți și elevi în funcție de gradul de dificultate și preferințe personale.

2.3 Detecția și clasificarea obiectelor

Detecția obiectelor se referă la identificarea și localizarea regiunilor dintr-o imagine care conțin un anumit tip de obiect. Scopul este delimitarea acestor regiuni de restul imaginii pentru ca acestea să fie folosite mai departe într-o multitudine de aplicații cum ar fi supravegherea video, asistența medicală sau conducerea autonomă.

Clasificarea obiectelor constă în atribuirea unei etichete obiectelor identificate și diferențierea clară între diferite tipuri de obiecte. Acest aspect ajută la gestionarea mai bună și mai eficientă a obiectelor dintr-o imagine.

2.3.1 Metode tradiționale de identificare a obiectelor

Metodele tradiționale de identificare a obiectelor se referă la tehnici și algoritmi dezvoltați și utilizați înainte de creșterea performanțelor rețelelor neuronale. Aceste metode sunt relevante în contextul în care resursele sistemului de calcul sunt limitate sau procesarea imaginilor și detecția obiectelor trebuie să fie executată într-un timp cât mai scurt.

2.3.1.1 *Cascada Haar*

Potrivit [15], identificarea obiectelor folosind cascade Haar funcționează prin evaluarea unor proprietăți numite proprietăți Haar. Aceste proprietăți sunt identificate prin însumarea intensității pixelilor dintr-o regiune a imaginii și calcularea diferenței dintre sume. Astfel se obțin proprietăți care descriu linii sau margini în interiorul imaginii.

Eficiența acestui algoritm constă în procesul de antrenare, proces similar cu cel folosit în antrenarea rețelelor neuronale. Se folosește o tehnică numită Adaboost care combină mai multe proprietăți Haar pentru a crea o singură proprietate Haar de calitate superioară. În timpul procesului de învățare, algoritmul Adaboost selectează cele mai distinctive proprietăți Haar și le atribuie valori în funcție de performanța diferențierii dintre exemple pozitive și exemple negative (imagini care conțin obiectul de interes și imagini care nu conțin obiectul de interes).

Aranjarea proprietăților este făcută într-o structură de cascadă care permite rejectarea rapidă a imaginilor și a regiunilor care au o probabilitate mică de a conține obiectul de interes, evaluând eficient fiecare regiune a unei imagini.

2.3.1.2 *Histogramă de gradienti orientați*

Conform [16], histograma de gradienti orientați (în engleză: Histogram of Oriented Gradients sau HOG) constă în calcularea gradientelor intensităților pixelilor din imagine care reprezintă rata de modificare a intensităților în plan vertical sau orizontal. Pentru acest calcul se poate folosi operatorul Sobel.

Imaginea este împărțită în mai multe regiuni numite celule. Pentru fiecare pixel dintr-o celulă este calculată orientarea și magnitudinea gradientului. Pentru fiecare celulă este construită o histogramă a orientărilor gradientului prin acumularea orientărilor gradientului ale tuturor pixelilor din celulă. Această histogramă reprezintă distribuția orientărilor gradientului.

Celulele adiacente sunt grupate pentru a forma blocuri iar histogramele din fiecare bloc sunt concatenate pentru a forma un vector de trăsături. Acest vector de trăsături reprezintă informațiile despre forma și proprietățile obiectelor.

2.3.2 Rețele neuronale convoluționale pentru detecția obiectelor

Rețelele neuronale convoluționale sunt un tip de modele de rețele neuronale dezvoltate în special pentru procesarea imaginilor și detecția de obiecte. Aceste tipuri de rețele neuronale oferă performanțe superioare atunci când vine vorba de detecția obiectelor, fiind capabile să recunoască caracteristici complexe în condiții variabile.

Un aspect de luat în considerare este faptul că această abordare pentru detecția obiectelor presupune utilizarea mai intensă a sistemului de calcul. Astfel, pentru a folosi

această abordare în contextul identificării obiectelor într-un timp scurt, este necesar un sistem de calcul performant.

2.3.2.1 Structura rețelelor neuronale convoluționale pentru detecția obiectelor

Potrivit [33], structura rețelelor neuronale convoluționale este adaptată pentru detecția și localizarea de obiecte din imagini digitale prin utilizarea unei combinații de straturi specializate:

- Straturi convoluționale: Acestea sunt straturile de bază ale acestor tipuri de rețele și sunt responsabile de aplicarea filtrelor pentru imaginile de intrare și extragerea caracteristicilor relevante ale imaginilor.
- Straturi de agregare (în engleză: pooling layers): Aceste straturi ajută la reducerea dimensiunilor arhitecturii rețelelor prin reducerea dimensiunii hărților de caracteristici identificate, păstrând doar caracteristicile esențiale. De asemenea, aceste straturi ajută și la prevenirea fenomenului de supraînvățare.

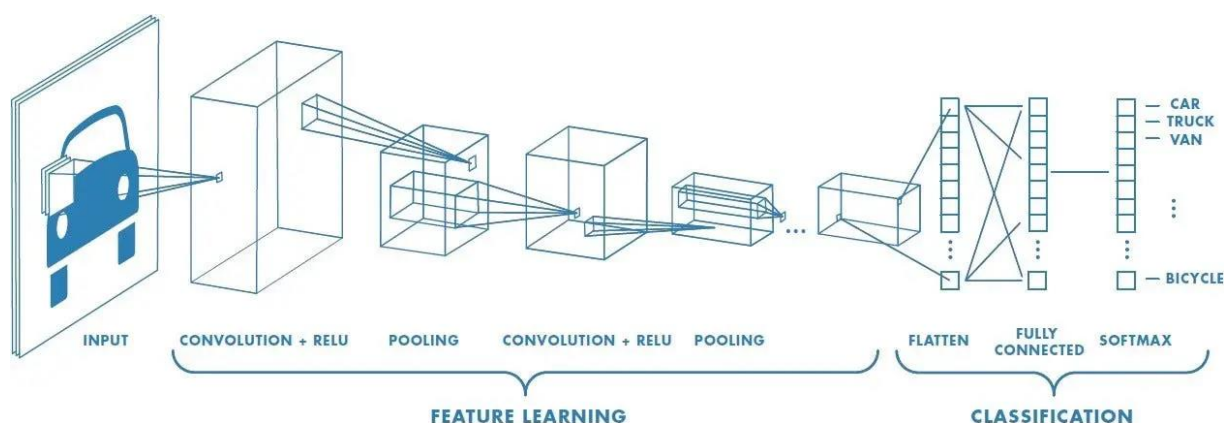


Figura 2.3. Arhitectura unei rețele neuronale convoluționale [5]

Pentru ca un obiect dintr-o imagine să fie identificat, este nevoie de mai multe etape. În primul rând, în straturile de convoluție se vor aplica diferite tipuri de filtre pentru extragerea de caracteristici. După executarea operațiilor de convoluție, se aplică o funcție de activare, de obicei funcția ReLU pentru a introduce neliniarități, ajutând rețeaua să identifice structuri complexe de caracteristici.

În straturile de agregare se execută operații de reducere a dimensiunilor imaginilor. O operație comună este agregarea maximă (în engleză: max pooling) în care imaginile sunt împărțite în grupuri de pixeli, păstrându-se doar pixelul cu valoarea intensității cea mai mare. Acest lucru asigură reducerea controlată a dimensiunii păstrând caracteristicile esențiale ale imaginilor.

Procesul de filtrare urmat de aplicarea funcției de activare este repetat de mai multe ori utilizând diferite filtre pentru a scoate în evidență diferite caracteristici ale imaginii. După ce repetarea acestui proces este executată, procesul de agregare se execută pentru fiecare imagine în parte.

După mai multe etape de filtrare și agregare, urmează procesul de aplatizare unde valorile intensităților fiecărui pixel din fiecare imagine obținută sunt grupate organizat

într-un vector unidimensional pentru a permite introducerea în straturile următoare ale rețelei. [33]

Vectorul unidimensional rezultat în urma procesului de aplatizare reprezintă o sinteză a caracteristicilor imaginilor și este intrarea pentru straturile complet conectate care urmează. Aceste straturi sunt rețele neuronale tradiționale și sunt folosite pentru a genera ieșirea rețelei neuronale pe baza informațiilor provenite de la straturile convoluționale.

Stratul final al acestei structuri este o funcție de activare de tip softmax care are ca rezultat o probabilitate distribuită pentru diferite clase. Clasa cu cea mai mare probabilitate este ieșirea și răspunsul rețelei neuronale pentru imaginea de intrare. [13]

2.3.2.2 Algoritmul YOLO

Potrivit [17], YOLO este un algoritm popular de detecție a obiectelor bazat pe rețele neuronale convoluționale. Acesta a fost prezentat pentru prima dată în anul 2016. O proprietate foarte importantă și reprezentativă a acestui algoritm este proprietatea de detecție într-un timp foarte scurt, folosind o metodă de divizare a imaginii în mai multe secțiuni, făcându-se predicții pentru fiecare secțiune în același timp. Această proprietate face ca algoritmul să fie mult mai rapid față de alți algoritmi de detecție.

Pentru fiecare secțiune a imaginii, algoritmul prezice dreptunghiuri de încadrare a obiectelor care includ coordonatele a două colțuri în raport cu secțiunea din care face parte, lățimea și înălțimea dreptunghiului. Pe lângă aceste informații, se regasește și o valoare care indică scorul detecției și reprezintă probabilitatea ca un obiect să fie înăuntrul dreptunghiului de încadrare. Aceste preziceri se fac cu ajutorul rețelelor neuronale convoluționale.

Pentru a evita fenomenul în care un obiect este identificat de mai multe ori, algoritmul păstrează detecția cu cel mai bun scor și elimină celelalte detecții care se suprapun.

În ultimii ani, algoritmul YOLO a evoluat în diferite versiuni, acuratețea și rapiditatea fiind îmbunătățite. Fiecare versiune a suferit modificări în structură și metode de implementare pentru a crește performanțele.

2.3.3 Identificarea textului

Identificarea textului din imagini este procedura în care textul scris sau tipărit prezent într-o imagine este recunoscut și transformat într-un format de text editabil. Această tehnologie este utilă pentru digitalizarea documentelor tipărite și pentru extragerea informațiilor din imagini.

Pentru a obține rezultatele dorite, este necesară de obicei o preprocesare a imaginii. Se folosesc operații precum redimensionarea, binarizarea și aplicarea filtrelor pentru reducerea de zgomote. Regiunea din imagine care conține text este separată de restul imaginii și împărțită în linii, cuvinte și caractere individuale. Această secționare este posibilă cu ajutorul histogramelor de proiecție care reprezintă suma valorilor intensității pixelilor de pe o linie sau coloană de pixeli, identificându-se astfel spațiile dintre linii, cuvinte și caractere.

Conform [18], recunoașterea fiecărui caracter în parte se poate realiza folosind modele de rețele neuronale convoluționale sau algoritmi de asociere cu șabloane. Pentru metoda de recunoaștere folosind șabloane, fiecare imagine a fiecărui caracter este adusă la o dimensiune și o formă standardizată și este comparată cu fiecare șablon disponibil. Fiecare comparație generează un scor iar caracterul cu scorul cel mai mare este ales ca rezultat.

Pentru realizarea comparațiilor cu șabloanele se pot folosi multiple metode în funcție de cerințele specifice ale sistemelor. Printre cele mai utilizate metode sunt:

- Distanța Euclidiană: Se calculează diferența între vectorii de valori de intensități ai pixelilor. Cu cât diferențele sunt mai mici, cu atât asemanările dintre imagini sunt mai mari.
- Distanța Manhattan: Se calculează suma valorilor absolute ale diferențelor dintre coordonatele pixelilor din cele două imagini.
- Corelația: Se evidențiază similaritatea dintre două imagini luând în considerare media și deviațiile intensității pixelilor din fiecare imagine.

După ce procesarea textului este executată, există posibilitatea de postprocesare. Această metodă reprezintă corectarea automată a textului identificat bazată pe un set de reguli. Aceste reguli pot să descrie procesul de înlocuire a unei litere acolo unde este de așteptat să fie o cifră sau invers, procesul de înlocuire a cuvintelor greșite sau metodologii mai avansate de corectare a greșelilor gramaticale sau de ortografie.

2.4 Analiza, urmărirea și numărarea obiectelor

Analiza, urmărirea și numărarea obiectelor sunt concepte de bază și foarte folosite în procesarea videoclipurilor pentru diferite scopuri și în diferite domenii precum securitatea și supravegherea, producția industrială sau domeniul autovehiculelor autonome. Urmărirea și numărarea obiectelor în contextul unui format video se poate implementa utilizând diverse metode și diferiți algoritmi.

2.4.1 Urmărirea obiectelor în contextul formatelor video

Urmărirea obiectelor în contextul formatelor video reprezintă procesul de a identifica și de a asocia datele găsite între două sau mai multe imagini consecutive corespunzătoare videoclipului. Aceste asocieri reprezintă de fapt asocieri ale obiectelor, fiind identificat exact același obiect în imagini diferite, obiect căruia i se va atribui aceeași etichetă pe tot parcursul videoclipului.

Urmărirea obiectelor presupune de obicei că identificarea obiectelor este deja făcută și poate fi implementată folosind diferite tehnici:

- Algoritmi bazați pe detecția caracteristicilor: Se utilizează metode precum Kanade-Lucas-Tomasi sau transformarea de caracteristici invariante la scară pentru a face asocieri între caracteristici distinctive ale obiectelor.
- Filtrarea Kalman: Se calculează o estimare a poziției obiectului din imaginea următoare pe baza pozițiilor din trecut și a poziției din imaginea curentă.

- Algoritmul SORT: Utilizează filtrarea Kalman și raportul de intersecție peste uniune pentru a face asocierile între obiectele din imagini.
- Utilizarea rețelelor neuronale recurente: Se utilizează acest tip de rețele neuronale pentru a prezice traiectoria obiectelor.

2.4.1.1 *Kanade-Lucas-Tomasi*

Conform [19], acest algoritm funcționează prin identificarea și urmărirea unor puncte specifice din imagini. Se utilizează metode de identificare a unor puncte de interes cum ar fi punctele în jurul cărora există o variație mare de intensitate a pixelilor (colțurile) și se caută puncte de interes asemănătoare în imagini succesive.

Pentru a aplica acest algoritm, se consideră că intensitatea pixelilor din jurul punctelor de interes nu se modifică și deplasarea obiectelor este relativ mică. Aceste trăsături denotă faptul că algoritmul este fezabil pentru medii stabile și nu performează în medii în care luminozitatea este volatilă sau deplasarea obiectelor raportată la dimensiunea imaginii este mare.

2.4.1.2 *Folosirea transformării de caracteristici invariante la scară*

Potrivit [20], primul pas în aplicarea acestui algoritm este identificarea trăsăturilor obiectelor cu ajutorul transformării de caracteristici invariante la scară. Fiecărui punct de interes îi este asociat un vector de descriere care reprezintă acest punct și regiunea din jurul acestui punct pe baza orientărilor gradientilor pixelilor. Acest vector de descriere este invariant la luminozitate și scară.

Pentru a putea urmări obiectele în imagini consecutive, se compară caracteristicile obiectelor din imagini folosind distanța Euclidiană a vectorilor de descriere. Pentru o funcționare optimă este necesară o metodă eficientă de asociere a acestor caracteristici.

2.4.1.3 *Filtrarea Kalman*

Conform [21], filtrarea Kalman este un algoritm folosit pentru estimarea pozițiilor viitoare ale obiectelor din imagini bazat pe stările anterioare ale obiectelor și starea curentă a obiectelor care se doresc a fi urmărite.

Starea curentă a unui obiect este reprezentată printr-un vector de stare care este actualizat pe măsură ce se fac noi detecții ale obiectului urmărit și conține de obicei informații legate de poziția și viteza obiectului.

Pentru a face o predicție a stării obiectului din viitor, se folosește starea curentă a obiectului și un model de mișcare. Valorile corespunzătoare poziției obiectului se actualizează în funcție de viteza obiectului iar viteza obiectului se recalculează în funcție de accelerație atunci când viteza nu este constantă.

Vectorul de stare este o aproximare a poziției obiectului. Pentru a scoate în evidență natura lipsită de acuratețe a aproximării, avem nevoie de o matrice de covarianță a stării. Această matrice reprezintă incertitudinea asociată cu estimarea vectorului de stare și este folosită pentru a calcula următorul vector de stare.

2.4.1.4 *Algoritmul SORT*

Potrivit [22], algoritmul SORT (Simple Online and Realtime Tracking) utilizează filtrarea Kalman și raportul de intersecție peste uniune pentru a asocia obiectele din

imagini consecutive. Pentru ca doua obiecte din doua imagini consecutive să fie asociate, acestea trebuie mai întâi identificate.

Primul pas în asocierea obiectelor a acestui algoritm este calcularea predicției poziției obiectului din imaginea curentă folosind filtrarea Kalman. Această poziție reprezintă poziția unde este probabil ca obiectul să fie în imaginea următoare.

Pentru a asocia obiectul din imaginea curentă cu obiectul din imaginea succesivă se calculează raportul de intersecție peste uniune a dreptunghiului de încadrare a obiectului estimat cu dreptunghiul de încadrare a obiectului detectat în imaginea succesivă. Valoarea acestui raport reprezintă gradul de potrivire a pozițiilor celor două obiecte. Dacă valoarea raportului este peste un anumit prag, asocierea dintre cele două obiecte este făcută.

Obiectele care sunt identificate și nu sunt asociate cu obiecte urmărite se consideră obiecte noi iar obiectele pentru care se fac aproximări și nu se găsesc asocieri cu identificări noi sunt șterse după un anumit număr de cicluri.

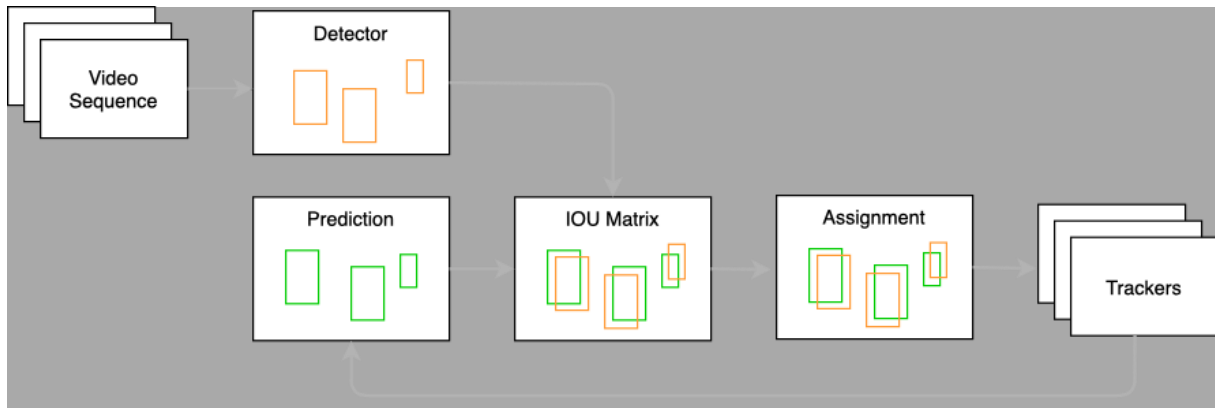


Figura 2.4. Structura algoritmului SORT [6]

2.4.2 Numărarea obiectelor

Numărarea obiectelor presupune că identificarea și urmărirea obiectelor sunt deja implementate. Acest lucru asigură ca un obiect nu este numărat de mai multe ori.

De obicei în contextul procesării videoclipurilor și mai ales în contextul actual de supraveghere a traficului, numărarea obiectelor se poate face în mai multe feluri și cu diferite scopuri. Printre metodele de numărare se regăsesc:

- Numărarea simplă: Se incrementează o entitate din structura de date de fiecare dată când apare un obiect nou.
- Numărarea într-o anumită zonă: Numărarea se face doar în cazul în care un obiect ajunge într-o anumită zonă. Această abordare poate fi utilă în cazul în care identificarea obiectelor are performanțe scăzute în anumite zone ale imaginii.
- Numărarea în funcție de zone: Obiectele sunt împărțite în mai multe zone de interes ale imaginii și se folosesc tehnici bazate pe geometrie pentru a număra obiectele. Această abordare este utilă în contextul actual în care se dorește numărarea autovehiculelor în funcție de banda rutieră pe care se află.

- Numărarea în funcție de clase: Obiectele sunt numărate la fel ca la numărarea simplă sau numărarea într-o anumită zonă și sunt împărțite în funcție de categoria de obiect din care fac parte.

2.5 Interfețe de utilizator

Interfețele pentru utilizator sunt un ansamblu de componente care ajută la comunicarea și interacțiunea oamenilor cu calculatoarele. Acestea pot să includă ecrane, tastaturi, mouse-uri sau pot să fie ferestre afișate pe ecran care conțin butoane digitale, texte, animații și sunete și care ajută la interacțiunea și controlul unei aplicații sau a unui site web. [7]

Pentru a face cât mai utile și mai ușor de utilizat aceste interfețe, este nevoie de un aspect vizual sugestiv care să permită utilizatorilor să folosească intuitiv aplicația și să gestioneze ușor informațiile. Acest lucru presupune că interfața nu trebuie să fie prea complicată pentru a nu îngreuna utilizarea acesteia dar în același timp trebuie să conțină toate elementele necesare pentru un control bun al aplicației.

O interfață de utilizator poate să conțină diferite elemente structurate în diferite moduri pentru a obține diferite funcționalități și aspecte vizuale. Printre cele mai utilizate elemente sunt:

- Elemente informaționale: Utilizatorul primește informații despre aplicația cu care interacționează cu ajutorul componentelor de tip text, notificări și alte componente vizuale sau auditive sugestive.
- Elemente de intrare: Utilizatorul are posibilitatea să introducă informații în aplicație modificând datele programului cu care interacționează prin diferite metode. Se folosesc câmpuri de text, liste predefinite sau butoane.
- Elemente de navigație: Informațiile se organizează și se așează în diferite pagini sau zone ale aplicației. Se folosesc componente precum câmpuri de cautare, meniuri sau conceptul de paginare pentru a ajuta utilizatorul să găsească aceste informații și să interacționeze cu ele.

Conform [7], interfața de utilizator este o componentă esențială a oricărei aplicații software, existând diferite tipuri de interfețe pentru diferite scopuri. Cele mai populare tipuri de interfețe de utilizator sunt următoarele:

- Interfețe grafice pentru utilizator: Aceste interfețe sunt compuse din elemente grafice sugestive cum ar fi butoanele, ferestrele sau meniurile și sunt de obicei ușor și intuitiv de folosit.
- Linii de comandă: Sunt interfețe minimaliste care expun serviciile sistemului de operare. Funcționează pe bază de comenzi text care sunt introduse în linia de comandă.
- Interfețe în limbaj natural: Acest tip de interfețe este creat pentru a înțelege limbajul natural al oamenilor. Pe baza acestui limbaj, interfața are scopul de a da comenzi sistemului sau aplicației din care face parte și de a răspunde utilizatorului într-un limbaj natural.

2.6 Posibile soluții și abordarea aleasă

Pentru a aborda obiectivele lucrării există o multitudine de soluții pentru fiecare etapă a implementării. În alegerea soluțiilor s-a ținut cont de următoarele considerente:

- Procesarea imaginilor trebuie să fie cât mai rapidă pentru a putea fi utilizată în contextul de timp real, în același timp, acuratețea rezultatelor în urma procesării nu trebuie să scadă drastic din acest considerent.
- Urmărirea și identificare obiectelor trebuie să fie făcută în diferite medii de luminozitate iar obiectele, în acest caz autovehiculele, nu au o viteză deosebit de mare în raport cu imaginea din care fac parte.
- Există disponibilitatea unui sistem de calcul performant, de preferat un sistem de calcul cu unitate de procesare grafică.
- Interfața de utilizator trebuie să fie una grafică și minimalistă care să permită controlul asupra procesării imaginilor și extracțiilor de date.

Astfel, ținând cont de considerentele listate mai sus, pentru procesarea imaginilor se va folosi algoritmul YOLO în defavoarea metodelor tradiționale de detecție a obiectelor iar pentru detecția textului se va folosi un model de rețea neuronală convolutivă. Se vor folosi tehnici de preprocesare a imaginilor și filtre pentru a îmbunătăți rezultatele acestor tehnici.

Pentru urmărirea obiectelor se va implementa un algoritm SORT simplificat, fără filtrare Kalman, deoarece deplasarea obiectelor în videoclip nu este una foarte semnificativă de la o imagine la alta, camera de supraveghere fiind amplasată înaintea unei treceri de pietoni cu semafor.

Interfața de utilizator va fi una simplă care să permită utilizatorului să controleze procesarea videoclipurilor și a transmisiilor în direct a videoclipurilor. Vor fi disponibile diferite opțiuni de procesare a videoclipurilor. Nu este nevoie de conceptul de creare a unui cont de utilizator iar accesul la aplicație în sine va fi nerestricționat, considerându-se o restricție pentru folosirea aplicației accesul la imaginile transmise de camera de supraveghere.

3 Analiză, proiectare, implementare

3.1 Tehnologii folosite și medii de dezvoltare

3.1.1 Limbajul de programare Python



Potrivit [8], Python este un limbaj de programare foarte folosit în prezent, fiind ușor de utilizat și de învățat. O caracteristică a acestui limbaj este că acesta este interpretat, fiind executat linie cu linie de către un interpretor. Acest interpretor transformă codul Python într-un cod intermediar, independent de platformă, apoi mașina virtuală Python execută codul intermediar generat de interpretor folosind codul sursă. Codul intermediar este salvat local pentru a mări viteza de execuție pentru executări viitoare.

Acest limbaj de programare are la dispoziție o colecție mare de librării și poate fi extins cu module de cod scrise în limbajele C sau C++. Acesta are de asemenea o procedură automată de gestionare a memoriei care alocă și eliberează memoria folosită după nevoie. [9]

3.1.2 Mediul de dezvoltare integrat PyCharm



PyCharm este un mediu de dezvoltare integrat, creat specific pentru limbajul de programare Python. Acesta permite scrierea, interpretarea și executarea de cod Python într-un mod ușor și intuitiv de folosit. [23]

Pe lângă editorul de cod, PyCharm are mai multe componente integrate care ajută la gestionarea proiectelor și a codului, testarea și repararea codului, gestionarea pachetelor, gestionarea bazelor de date și multe altele, oferind suport dezvoltatorilor de cod Python.

3.1.3 Sistemul de control Git



Coform [24], Git este un sistem distribuit de control al versiunilor de fișiere. Acesta este folosit de obicei pentru a gestiona schimbările făcute în fișierele care conțin cod și pentru a ajuta la dezvoltarea codului în echipă.

Funcționalitățile acestui sistem se bazează pe depozite. Aceste depozite reprezintă folderele în care se află fișierele care se doresc a fi controlate. Depozitele pot să aibă mai multe versiuni numite ramuri. Aceste ramuri sunt identificate prin noduri care reprezintă schimbările făcute asupra fișierelor. Acest lucru asigură posibilitatea de dezvoltare în paralel a proiectelor de programare și nu numai. Ramurile unui depozit pot să fie concatenate pentru a aduce în aceeași versiune modificările de pe ambele ramuri.

3.1.4 Platforma GitHub



GitHub este o platformă web folosită pentru distribuirea fișierelor folosind sistemul de control Git. Această platformă este utilizată în general pentru distribuirea fișierelor care conțin cod sursă, facilitând astfel controlul proiectelor și munca în echipă făcută pentru aceste proiecte. [25]

Pentru ca distribuirea fișierelor să funcționeze, este făcută o clonă a depozitului local pe platforma web sau invers. Astfel, fiecare membru al echipei va avea o clonă a aceluiași depozit iar modificările se pot trimite sau prelua de pe platformă.

Pe lângă funcționalitățile de distribuire a fișierelor, GitHub are multe componente care ajută la gestionarea organizată a unui proiect, una dintre cele mai importante astfel de componente fiind cererea de concatenare a ramurilor (în engleză: Pull Request). Această cerere dă posibilitatea întregii echipe de a verifica codul scris de membrul care a făcut cererea. Dacă codul este aprobat, modificările de pe o ramură se concatenează cu conținutul altei ramuri.

3.1.5 Modelul de detecție YOLOv8



YOLO este un model de rețea neuronală dezvoltat de compania Ultralytics, folosit pentru detecția de obiecte din imagini și în special pentru detecția în timp real datorită performanțelor ridicate. [26]

Modelul folosit în acest proiect este YOLOv8n, un model din versiunea a opta, construit pentru a fi rapid și în situațiile în care resursele sunt limitate. Acesta este antrenat pe un set divers de imagini pentru a introduce date fundamentale despre forma și caracteristicile obiectelor. Pe lângă aceste date, modelul poate fi antrenat utilizând un set de date specific pentru adaptarea lui în contexte concrete.

3.1.6 Biblioteca OpenCV



Potrivit [27], OpenCV este o bibliotecă publică, creată și folosită pentru sarcini care includ procesarea de imagini. Printre cele mai des întâlnite funcționalități din această librerie sunt:

- Manipularea imaginilor prin modificarea proprietăților precum dimensiunea, culoarea sau locația acestora.
- Aplicarea diferitelor tipuri de filtre asupra imaginilor.
- Aplicarea metodelor clasice de detecție a obiectelor.

3.1.7 Biblioteca EasyOCR

Conform [28], EasyOCR este o bibliotecă publică disponibilă pentru limbajul de programare Python care aduce o soluție eficientă și ușor de folosit pentru problema recunoașterii de caractere dintr-o imagine. Aceasta folosește modele de rețele neuronale convoluționale și recurente pentru a recunoaște și pentru a citi textul din imagini în mai multe limbi.

Modelul “Reader” disponibil în această librerie este ușor de folosit, fiind deja antrenat pe un set larg și diversificat de date. Acest model nu presupune că utilizatorul trebuie să achiziționeze un set nou de date pentru a rezolva problema recunoașterii de caractere din imagini, deși acest lucru este posibil. Pe lângă acuratețea rezultatelor și ușurința de folosire a bibliotecii, aceasta este potrivită pentru procesarea imaginilor în timp real, modelele de detecție având un timp de procesare scăzut.

3.1.8 Biblioteca Pandas



Pandas este o librerie publică disponibilă pentru limbajul de programare Python și folosită pentru manipularea și analizarea datelor. Aceasta conține structuri de date și funcții care ajută la procesarea eficientă a informațiilor atunci când vine vorba de lucrul cu multe date. [29]

Există două structuri de date principale:

- “Series”: Este un vector unidimensional capabil să stocheze orice tip de date și să le eticheteze.
- “DataFrame”: Este o matrice bidimensională capabilă să stocheze și să eticheteze diferite tipuri de date similar unui tabel dintr-o bază de date.

Această librerie suportă citirea și scrierea datelor în diverse formate precum CSV, Excel, SQL sau JSON și oferă soluții de detectare sau autocompletare a datelor lipsă.

Operațiile folosite sunt optimizate pentru performanță, deseori fiind executate folosind limbajul de programare C.

3.1.9 Qt pentru Python



Potrivit [30], Qt pentru Python sau PySide6 este setul oficial de legături Python pentru folosirea librărilor Qt disponibile în limbajul de programare C++. Aceste librării sunt folosite pentru crearea și dezvoltarea aplicațiilor care conțin elemente vizuale complexe într-o interfață grafică pentru utilizator.

Această bibliotecă are multe avantaje, printre cele mai populare fiind următoarele:

- Dispune de un set larg și variat de elemente grafice complexe.
- Aplicațiile sunt compatibile cu diverse sisteme de operare: Windows, macOS, Linux, iOS și Android.
- Include Qt Designer, o aplicație pentru a crea interfețele de utilizator vizual și intuitiv.
- Este eficientă și rapidă, având o documentație bine pusă la punct și o comunitate mare. Acest lucru face ca găsirea soluțiilor la anumite probleme să fie mai ușoară.

3.2 Detalii de proiectare

3.2.1 Cazuri de utilizare

Pentru a implementa aplicația de supraveghere video a traficului, avem în primul rând nevoie de un set de cazuri clare de utilizare. Posibilitățile de utilizare din punct de vedere al utilizatorului trebuie clar definite. Astfel, pentru orice utilizator al aplicației există următoarele cazuri de utilizare:

- Observarea imaginilor procesate în timp real
- Procesarea videoclipurilor înregistrate
- Generarea unor statistici pe baza procesării videoclipurilor
- Adăugarea și ștergerea numerelor de înmatriculare de pe lista de urmărire
- Modificarea opțiunilor de procesare
- Modificarea parametrilor de procesare
- Înregistrarea automată a transmisiunilor în direct relevante

Modificarea opțiunilor de procesare include afișarea dreptunghiurilor de încadrare, afișarea tipurilor de autovehicule, procesarea numerelor de înmatriculare, afișarea numărului total de vehicule, afișarea numărului de vehicule în funcție de clasă, afișarea vehiculelor în funcție de banda rutieră, afișarea benzilor și a dreptei de numărare și afișarea identicatorului intern.

Modificarea parametrilor de procesare include modificarea scorului modelelor de detecție, modificarea parametrului de urmărire și modificarea numărului de încercări de citire a numerelor de înmatriculare.

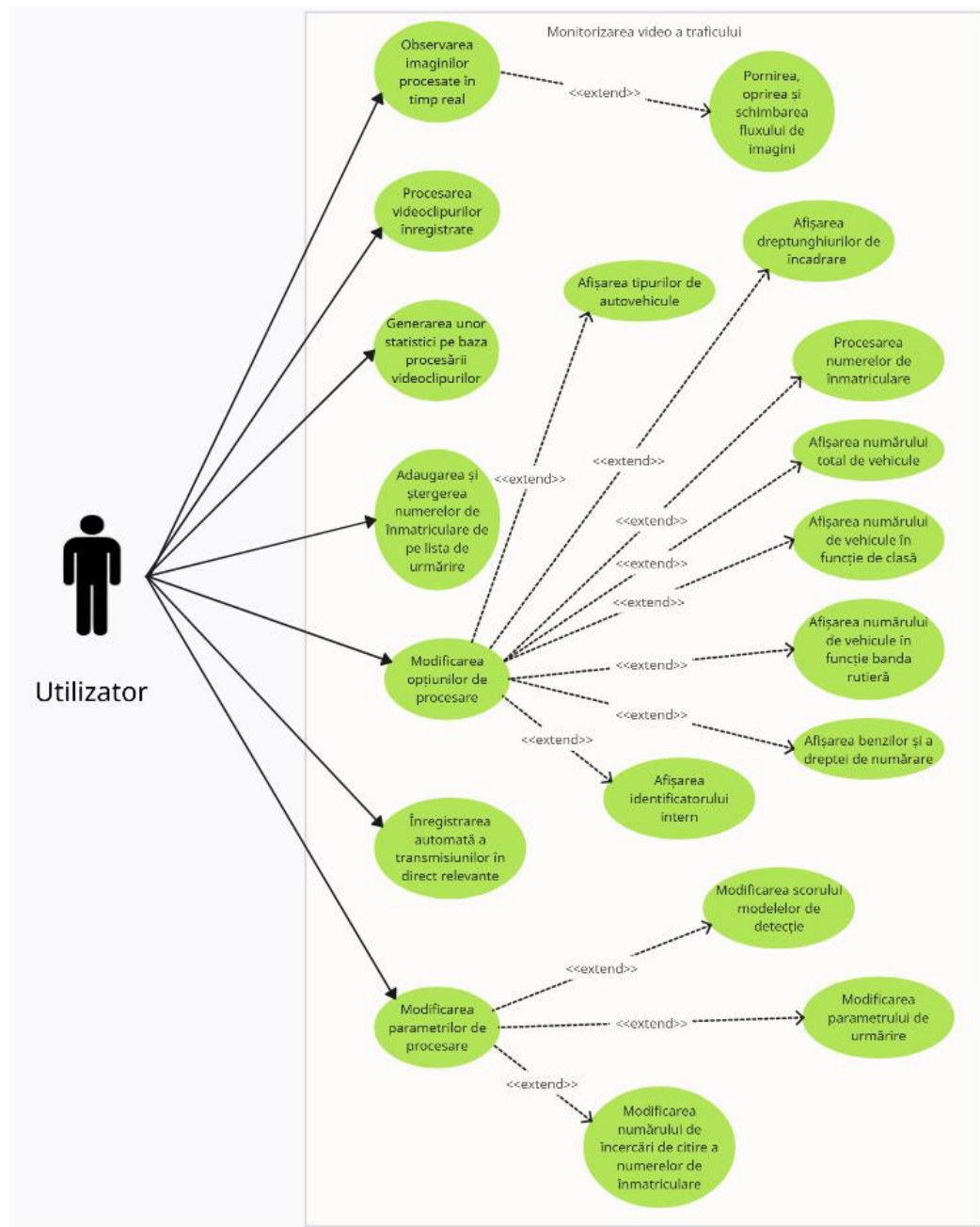


Figura 3.1. Diagrama cazurilor de utilizare

3.2.2 Structura aplicației

Structura aplicației este una relativ simplă, folosindu-se doar limbajul de programare Python atât pentru logica procesării imaginilor cât și pentru implementarea interfeței grafice de utilizator. Fișierele Python responsabile pentru afișarea elementelor grafice sunt generate folosind aplicația Qt Designer și sunt introduse în același folder în care se regăsesc și fișierele responsabile pentru logica procesării imaginilor. Pentru o

structură organizată, se folosește o ierarhie a folderelor care este prezentată în figura următoare.

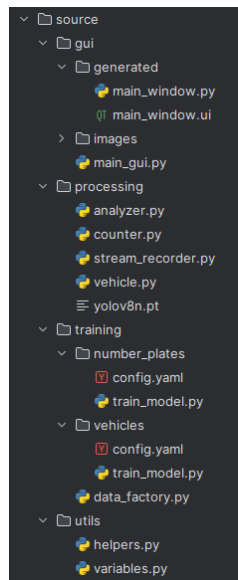


Figura 3.2. Structura fișierelor principale ale aplicației

În Figura 3.2 este prezentată structura fișierelor și a folderelor care conțin codul sursă principal al aplicației, inclusiv codul folosit pentru antrenarea modelelor YOLO pentru detecția obiectelor. Aceste fișiere și foldere sunt în interiorul folderului “source”, folder care se află în folderului principal al proiectului Python.

Pe lângă folderul “source”, în folderul principal al proiectului mai există fișierul Python principal “main.py” care este responsabil pentru pornirea aplicației, un fișier responsabil pentru generarea unui fișier executabil corespunzător aplicației, folderul “dev_setup” care conține informații despre configurarea mediului de dezvoltare, folderul “documentation” care conține documentația proiectului, folderul “runs” care conține modelele YOLO antrenate, folderul “dist” care conține fișierul executabil corespunzător aplicației, fișierul “.venv” care este responsabil pentru funcționarea mediului virtual Python, fișierul “recordings” în care se află înregistrări ale transmisiilor în direct și alte fișiere și foldere care au legătură cu sistemul de control Git, generearea fișierului executabil sau funcționarea generală a aplicației.

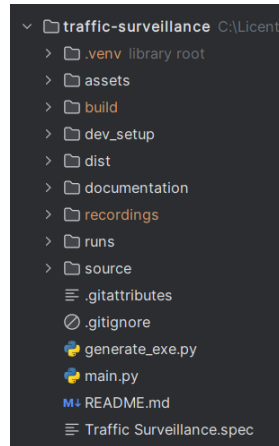


Figura 3.3. Structura proiectului Python

Folderul “source” conține toate clasele Python folosite și majoritatea codului Python folosit în dezvoltarea aplicației. În figura următoare este prezentată diagrama claselor care se află în acest folder. Se observă în această diagramă fiecare atribut și fiecare funcție a fiecărei clase și legăturile între clase.

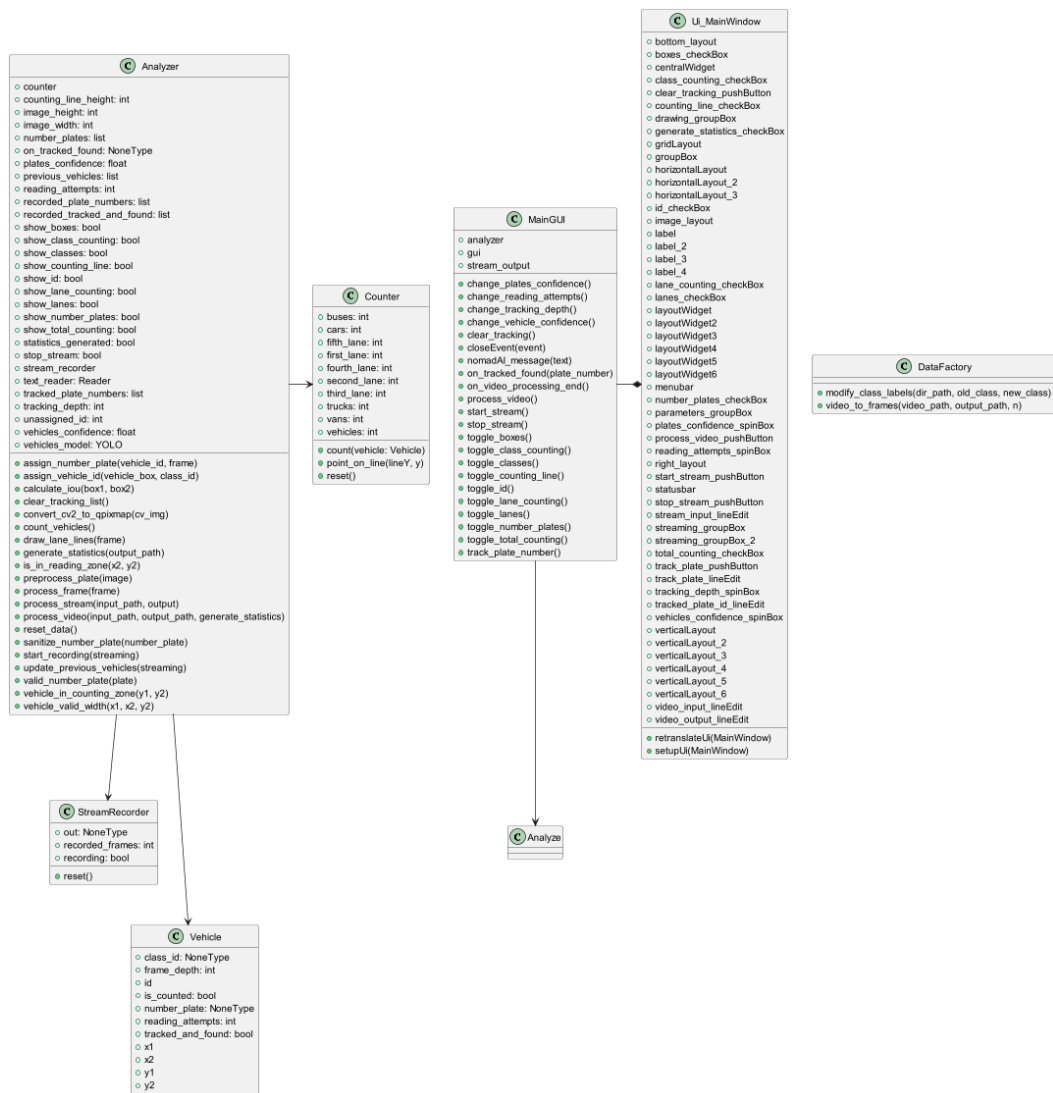


Figura 3.4. Diagrama claselor

3.3 Implementarea proiectului

3.3.1 Configurarea mediului de dezvoltare

Primul pas în implementarea proiectului este configurarea mediului de dezvoltare. Pentru asta, este nevoie în primul rând de descărcarea programelor și resurselor care vor fi folosite. Acestea sunt următoarele:

- Pachetul Python: Include un interpretor pentru cod Python, câteva pachete standard, un manager de pachete și altele. Acest pachet se poate descărca de pe site-ul web oficial al organizației Python.
- Mediul de dezvoltare integrat PyCharm: Acesta se poate descărca de pe site-ul web oficial al companiei JetBrains.
- Sistemul de control Git: Se poate descărca de pe site-ul web oficial Git.

Următorul pas în configurarea proiectului, după ce programele de care este nevoie sunt instalate, este deschiderea mediului de dezvoltare PyCharm și crearea unui proiect nou. În acest proiect este necesară crearea unui mediu virtual pentru Python. Acest mediu va fi responsabil pentru stocarea și gestionarea pachetelor care vor fi instalate folosind managerul de pachete Python. Pentru a crea un mediu virtual Python se execută în terminal comanda `python -m venv numevenv` unde `numevenv` se înlocuiește cu numele folderului în care se va afla mediul virtual.

Pentru a face posibilă interpretarea și rularea codului Python, este necesară configurarea unui interpretor. Acesta se selectează din mediul de dezvoltare, din fereastra responsabilă de setări și este aleasă opțiunea `Virtual Environment`.

Ultimul pas în inițializarea proiectului este configurarea controlului de versiune cu Git. Acesta se poate face în mai multe moduri, cel mai simplu fiind direct din mediul de dezvoltare. Din fereastra `VCS` (Version Control System), se selectează opțiunea `Share project on GitHub`. Acest lucru asigură o copie online a proiectului și posibilitatea de a lucra organizat și de a stoca online toate versiunile proiectului.

Pentru o organizare mai bună a proiectului, se inițializează fișierele `.gitignore` și `.gitattributes`. Acestea sunt fișiere de configurare a sistemului de control Git și sunt create înăuntrul folderului de tip depozit Git. Fișierul `.gitignore` specifică toate fișierele și folderele din proiect care nu trebuie urmărite de Git. În acest caz, în acest fișier se specifică următoarele foldere:

- `.idea`: Acesta este un folder generat de mediul de dezvoltare PyCharm pentru a stoca diferite configurări locale ale proiectului.
- `.venv`: Acesta este numele mediului virtual Python. Librăriile și pachetele folosite nu sunt urmărite de Git, în schimb, acestea sunt specificate într-un fișier text.
- `__pycache__`: Aceste foldere sunt create local de Python pentru a stoca fișiere cu cod interpretat pentru o rulare ulterioară a codului mai ușoară.
- `recordings`: În acest folder se vor salva automat înregistrări relevante ale transmisiunii în direct a fluxului video.

- “build”: Folderul acesta este folosit în procesul de creare a unui executabil pentru aplicație.

Fișierul “.gitattributes” se folosește pentru a specifica cum sunt tratate diferite fișiere sau foldere de către Git. În cazul de față, în acest folder se menționează stocarea separată a executabilului din moment ce acesta are o dimensiune considerabil mai mare decât restul fișierelor. Pentru acest lucru se folosește Git LFS (Large File Storage), o extensie a sistemului Git folosită pentru gestionarea și stocarea fișierelor de dimensiuni mari.

3.3.2 Antrenarea modelului de rețea neuronală pentru detecția vehiculelor

Pentru a putea implementa detecția obiectelor avem mai întâi nevoie de un model de rețea neuronală antrenat. Se folosește modelul YOLOv8n care se descarcă folosind comanda “pip install ultralytics” și se creează o instanță a acestui model. Comanda trebuie folosită în consolă după activarea mediului virtual Python pentru a instala pachetele local, doar pentru acest proiect.

3.3.2.1 Achiziția datelor

Modelul descărcat trebuie antrenat folosind un set specific de date. Pentru a crea acest set de date se folosesc înregistrări capturate de camera de supraveghere video. Din aceste videoclipuri se extrag imagini.

Pentru extragerea imaginilor din videoclipuri se folosește funcția “video_to_frames” din clasa “DataFactory”. Aceasta primește ca parametri calea către video, calea către folderul unde se doresc stocate imaginile și un număr natural care reprezintă frecvența de extragere a imaginilor din video. Acest ultim parametru este necesar pentru a nu extrage imagini foarte similare dintr-o înregistrare cu trafic aglomerat. Funcția verifică cele două căi iar apoi folosește biblioteca OpenCV pentru încărcarea videoclipului și pentru salvarea fiecărei imagini dorite.

Indiferent de tipul de date, în format video sau imagini, acestea nu vor fi urmărite de sistemul de control Git, deci vor avea o locație diferită față de proiect sau vor fi incluse în fișierul “.gitignore”. Acest lucru se datorează în primul rând confidențialității imaginilor care conțin numere de înmatriculare a mașinilor existente în Cluj-Napoca, dar și mărimii datelor.

Pentru a obține rezultate cât mai bune, se asigură o varietate largă de date, în diferite condiții. Imaginile trebuie să conțină diferite tipuri de autovehicule de mărimi, culori și forme variate. De asemenea, înregistrările trebuie să fie capturate în diferite momente ale zilei și condiții de vreme diferite pentru a capta cât mai multe variații posibile ale imaginilor. Pe cât posibil, se vor introduce și imagini în care nu apar deloc autovehicule pentru a aproviziona modelul și cu exemple negative.

3.3.2.2 Adnotarea datelor

Odată ce imaginile sunt pregătite, fiecare obiect din fiecare imagine trebuie identificat manual și adnotat conform standardului YOLO. Adnotarea constă în procesul în care pentru fiecare imagine se creează un fișier text și pentru fiecare detecție din imagine se creează o linie în fișierul text cu următoarea structură:

- Primul număr este un număr natural care reprezintă clasa obiectului. Acest număr este același pentru toate obiectele de același tip.
- Următoarele două numere semnifică coordonatele centrului dreptunghiului de încadrare a obiectului pe axa x și respectiv y. Aceste coordonate sunt normalizate la lățimea și înălțimea imaginii, deci valorile vor fi cuprinse între 0 și 1.
- Ultimele două numere reprezintă lățimea respectiv înălțimea dreptunghiului de încadrare a obiectului. Și aceste două valori sunt normalizate în funcție de lățimea și înălțimea imaginii.

Pentru a executa cu ușurință acest proces, există diferite aplicații ajutătoare care generează aceste fișiere text pentru imagini folosind o interfață simplă de utilizator care permite trasarea dreptunghiurilor de încadrare direct pe imagini. În acest caz, s-a folosit aplicația web “Computer Vision Annotation Tool”.

Pentru a evita introducerea de zgomot în parametrii modelului responsabili de detecția obiectelor, adnotarea trebuie făcută folosind anumite reguli care se respectă pe tot parcursul procesului. Aceste reguli sunt următoarele:

- Se adnotează fiecare autovehicul din fiecare imagine. În cazul în care tipul autovehiculului nu se încadrează în lista predefinită de autovehicule, acesta nu se adnotează.
- Dacă cel puțin jumătate din autovehicul este vizibil, acesta se ia în considerare. Vehiculele ascunse, care se văd în imagine mai puțin de jumătate din mărimea lor completă, se ignoră.
- Se include în dreptunghiul de încadrare tot obiectul, cu cât mai puțin spațiu în jurul obiectului. Dacă obiectul este ascuns, se încadrează o aproximare a întregului obiect.
- În cazul în care doar o parte a autovehiculului este vizibilă și această parte este îndeajuns de mare încât să fie luată în considerare, restul autovehiculului fiind în afara imaginii, se încadrează doar partea disponibilă a obiectului.
- La fiecare încadrare a unui autovehicul într-un dreptunghi se ține cont de tipul acestuia.

Tipurile de autovehicule alese pentru această lucrare sunt patru la număr: mașină mică, camion, camionetă și autobuz. Delimitarea între aceste tipuri este foarte importantă și poate introduce zgomot în sistem dacă nu este făcută. Pentru a delimita aceste tipuri, se folosesc caracteristicile vizuale ale autovehiculelor precum mărimea sau forma. Mașinile mici au cea mai mică dimensiune, urmate de camionete, care au de obicei și o lipsă a geamului din partea din spate a autovehiculului. Camioanele au cea mai mare dimensiune, având diferite forme, iar autobuzele au o formă specifică, o dimensiune mare, și au în general multe geamuri.



Figura 3.5. Adnotarea unei imagini

În Figura 3.2 se observă în partea stângă imaginea provenită de la camera de supraveghere care conține dreptunghiurile de încadrare pentru fiecare autovehicul. În partea dreaptă a figurii este reprezentat fișierul text corespunzător imaginii, în formatul specific adnotării YOLO. Se observă că numărul de rânduri din fișierul text corespunde cu numărul de autovehicule adnotate în imagine. Pe prima coloană a fișierului text sunt doar două valori distincte, fapt care arată că în imagine există doar două tipuri de autovehicule. Pe următoarele coloane se regăsesc coordonatele și dimensiunile normalizate ale fiecărui dreptunghi de încadrare.

3.3.2.3 Antrenarea modelului YOLO

Odată ce achiziția și adnotarea datelor a fost făcută și se asigură un set de date destul de mare și diversificat, se poate face antrenarea modelului.

Primul pas în antrenarea modelului este împărțirea setului de date în trei părți, cea mai mare parte a datelor fiind în setul de antrenare, urmat de cel de validare și testare. Imaginile trebuie de asemenea separate de fișierele text, fiecare imagine având exact același nume ca și fișierul text corespunzător. În figura următoare se va prezenta structura folderelor necesară pentru ca modelul YOLO să poată fi antrenat.

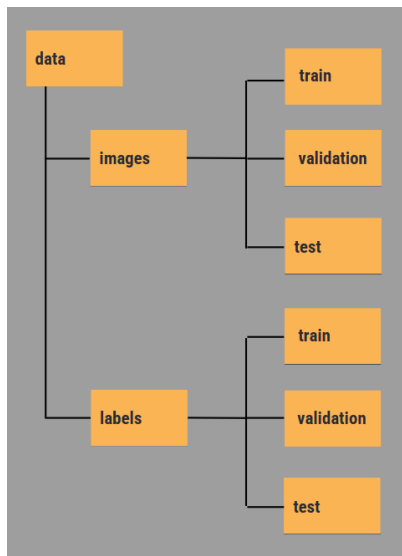


Figura 3.6. Structura setului de date

După ce datele sunt împărțite după exemplul prezentat în Figura 3.2, următorul pas este crearea unui fișier de configurare YAML care va fi transmis ca parametru funcției de antrenare a modelului. În acest fișier se specifică calea absolută către setul de date și căile relative la calea absolută către folderele care conțin imaginile de antrenare și imaginile de validare. Pe lângă aceste date, tot în acest fișier se specifică și corelația între numărul întreg de pe prima coloană a fișierelor text și numele tipurilor de autovehicule.

Fișierul de configurare fiind creat, se poate apela funcția “train” a instanței modelului YOLO. Pe lângă fișierul de configurare, această funcție mai primește ca parametru și un număr natural care corespunde cu numărul de repetări pe care modelul îl execută în procesul de învățare automată pentru întregul set de date. Funcția de învățare a modelului generează un model antrenat pe setul de date specificat. În figura următoare se observă în partea stângă scriptul de antrenare a modelului, iar în partea dreaptă, fișierul de configurare.



```

from ultralytics import YOLO

model = YOLO("yolov8n.yaml")
model.train(data="config.yaml", epochs=100)

```

```

path: C:/Licenta/data
train: images/train
val: images/val

# vehicle classes
names:
  0: small car
  1: truck
  2: bus
  3: van

```

Figura 3.7. Antrenarea modelului YOLO

Procesul de antrenare se execută iterativ cu parametri și date diferite până când modelul are performanțe mulțumitoare în detecția de autovehicule. Pentru a face acest proces mai rapid, din moment ce poate să consume multe resurse ale sistemului de calcul, se poate utiliza parametrul funcției de antrenare “device” cu valoarea “gpu” pentru a rula procesul de antrenare folosind unitatea de procesare grafică, în cazul în care aceasta este disponibilă.

3.3.3 Implementarea algoritmului de urmărire

3.3.3.1 Algoritmul de bază

Algoritmul de urmărire a autovehiculelor este esențial în dezvoltarea și funcționarea aplicației. Pentru a putea implementa acest algoritm avem în primul rând nevoie de detecția de autovehicule. Modelul YOLO returnează multiple date atunci când face o predicție asupra unei imagini. Pentru a accesa datele de interes, în acest caz datele corespunzătoare dreptunghiurilor de încadrare a obiectelor, vom accesa atributul “data” al atributului “boxes” al obiectului returnat în urma predicției imaginii. Acest atribut se transformă într-o listă și fiecare element al listei reprezintă un obiect și conține următoarele date:

- x_1 și y_1 : Reprezintă coordonatele punctului cel mai apropiat de originea imaginii a dreptunghiului de încadrare (colțul din stânga sus).
- x_2 și y_2 : Reprezintă coordonatele punctului cel mai îndepărtat de originea imaginii a dreptunghiului de încadrare (colțul din dreapta jos).

- Scorul: Este un număr cuprins între 0 și 1 și reprezintă probabilitatea ca obiectul să existe în imagine, respectiv rezultatul aplicării funcției de tip softmax din ultimul strat al rețelei neuronale.
- Identificatorul de clasă: Este un număr natural corelat cu numerele atribuite fiecărei clase de obiecte, în acest caz, fiecărei clase de autovehicule.

Pentru a atribui un identificator fiecărui obiect detectat de modelul YOLO se folosește funcția “assign_vehicle_id”. Înainte de acest proces, sunt verificate scorul și lățimea fiecărui obiect. Scorul trebuie să fie mai mare decât un scor de prag setat de utilizator iar lățimea dreptunghiului de încadrare trebuie să fie mai mare decât o lățime de prag. Această lățime de prag este aleasă în funcție de poziționarea obiectului în imagine. Dacă aceste două atribute ale obiectului nu corespund cerințelor, detecția se consideră una invalidă și algoritmul de identificare pentru obiectul respectiv se oprește.

Funcția “assign_vehicle_id” primește ca parametri o listă care conține coordonatele dreptunghiului de încadrare a autovehiculului și numărul corespunzător clasei autovehiculului. Pentru fiecare dreptunghi de încadrare se calculează raportul de intersecție peste uniune cu fiecare dreptunghi detectat în imaginea anterioară. Raportul cu cea mai mare valoare, mai mare decât o valoare de prag, corespunde asocierii celor două dreptunghiuri de încadrare, respectiv celor două obiecte.

După ce asocierea celor două dreptunghiuri de încadrare a fost făcută, se adaugă în lista de autovehicule anterioare un nou obiect de tip “Vehicle” creat folosind coordonatele detecției și atributele obiectului care este asociat cu această detecție. Obiectul asociat este șters din lista de obiecte anterioare, astfel făcându-se înlocuirea lui cu noua detecție din noua imagine.

Dacă nu-i este găsită nicio asociere detecției, respectiv dacă raportul maxim de intersecție peste uniune este prea mic sau nu există, se consideră că detecția nu aparține niciunui autovehicul anterior. Se adaugă în lista de autovehicule anterioare un nou obiect de tip “Vehicle” creat folosind coordonatele detecției, un număr de identificare nefolosit și valorile implicite ale celorlalte atribute ale clasei.

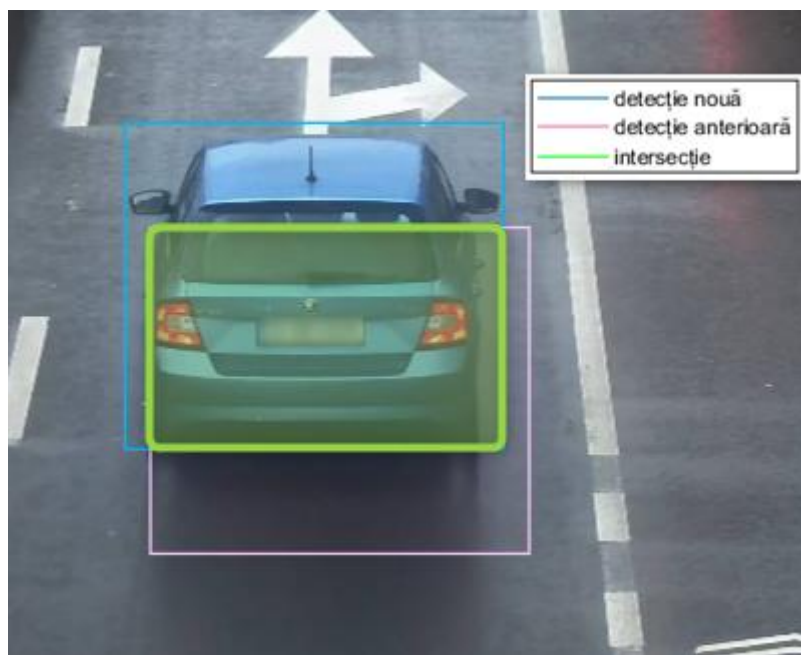


Figura 3.8. Detectiile unui autovehicul în imagini consecutive

În Figura 3.4 se observă dreptunghiurile de încadrare ale aceluiași autovehicul pentru imagini consecutive dintr-un videoclip și intersecția acestor dreptunghiuri folosită la calcularea raportului de intersecție peste uniune.

3.3.3.2 Gestionarea lipsei de detecții

Deoarece există imperfecțiuni în identificarea autovehiculelor din fiecare imagine care provin din capacitatea limitată a modelului YOLO și, în special din capacitatea camerei de a transmite imagini, algoritmul de urmărire trebuie îmbunătățit ținând cont de aceste aspecte. Camera video de supraveghere transmite adeseori imagini corupte care nu pot fi interpretate iar modelul YOLO poate să proceseze o predicție greșită, neidentificând un autovehicul.

Astfel, este posibil ca identificarea să nu fie făcută pentru fiecare autovehicul din fiecare imagine. Acest lucru determină faptul că, în implementarea algoritmului de urmărire, compararea detecțiilor cu o singură imagine anterioară este insuficientă. Dacă pentru o detecție se caută o corelație în imaginea anterioară și nu se găsește pentru că imaginea este coruptă sau nu există detecții, detecția se consideră una nouă, cu un număr de identificare nou, chiar dacă în imaginea anterioară există un autovehicul care să corespundă cu această nouă detecție.

Pentru a rezolva această problemă, se consideră compararea detecțiilor cu detecții din mai multe imagini anterioare. Numărul de imagini anterioare poate să fie ales de utilizator și are implicit valoarea 10 care corespunde cu aproximativ o secundă dintr-o înregistrare transmisă de camera de supraveghere. Acest interval de timp este intervalul maxim de timp observat în care camera de supraveghere transmite imagini corupte.

Fiecare detecție este salvată în structura de date, mai exact în lista de autovehicule anterioare, alături de numărul imaginii anterioare din care provine, acest număr având valoarea zero atunci când detecția este făcută pe imaginea curentă și fiind incrementat la

fiecare nouă detecție. Dacă o detecție nouă este corelată cu o astfel de detecție salvată în structura de date, numărul de imagini anterioare corespunzător detecției se resetează. Atunci când valoarea numărului de imagini anterioare a unei detecții este mai mare sau egală cu valoarea maximă aleasă de utilizator sau valoarea maximă implicită, detecția este ștearsă din structura de date. Doar în acest caz se consideră că autovehiculul nu mai este urmărit de algoritmul de detecție.

3.3.3.3 Optimizarea algoritmului

În algoritmul de bază, fiecare detecție nouă a unui autovehicul este comparată cu dreptunghiurile de încadrare a tuturor detecțiilor salvate în structura de date. Această abordare implică faptul că vehiculele foarte îndepărtate în imagine sunt comparate, chiar dacă o corelație între cele două autovehicule nu este posibilă. Compararea acestor autovehicule înseamnă consumarea inutilă a resurselor sistemului de calcul.

Pentru a îmbunătăți algoritmul și pentru a reduce numărul de operații de calcul a raportului de intersecție peste uniune, dreptunghiurile de încadrare a detecțiilor noi vor fi comparate doar cu dreptunghiurile din apropiere lor. Astfel, se va calcula distanța între coordonatele celor două dreptunghiuri de încadrare:

- Distanța pe axa X a imaginii se calculează în funcție de punctul cel mai apropiat de originea imaginii al celor două dreptunghiuri (colțul stânga sus). Această distanță este validă dacă este mai mică decât 300 de pixeli (aproximativ $1/14$ din totalul de pixeli de pe axa X).
- Distanța pe axa Y este calculată în funcție de același punct al celor două dreptunghiuri și este validă doar dacă este mai mică decât 600 de pixeli (aproximativ $1/4$ din totalul de pixeli de pe axa Y).

În cazul în care cele două distanțe dintre cele două dreptunghiuri de încadrare sunt valide, se poate trece mai departe la calculul raportului de intersecție peste uniune, astfel se evită calculul inutil al rapoartelor.

Distanța validă pe axa Y este semnificativ mai mare decât distanța validă pe axa X deoarece pe axa Y există în general o deplasare mai mare a autovehiculelor care trebuie luată în considerare.

Datorită marimilor mari ale dreptunghiurilor de încadrare a autobuzelor și a camioanelor și datorită frecvenței reduse de apariție a acestora, această optimizare nu se aplică pentru aceste două clase de autovehicule.

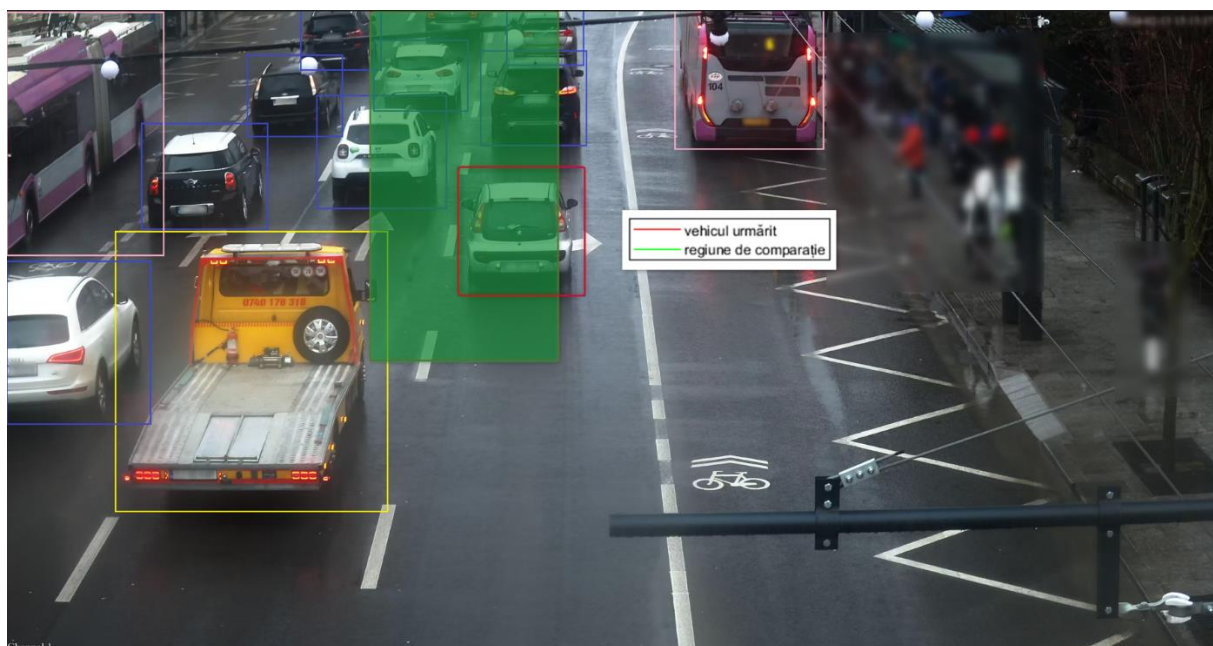


Figura 3.9. Regiunea de comparație a unui autovehicul

În Figura 3.5 este reprezentată regiunea de comparație a unui autovehicul marcată de distanțele maxime valide pe axele X și Y. Se observă că punctul cel mai apropiat de originea imaginii a dreptunghiului de încadrare a autovehiculului (colțul stânga sus) este în mijlocul dreptunghiului. Coordonatele autovehiculului urmărit se vor compara doar cu detecțiile anterioare care au punctul corespunzător colțului din stânga sus înăuntrul regiunii de comparație. Se observă că se iau în considerare pentru calculul raportului de intersecție peste uniune decât patru autovehicule dintr-un total de 12.

3.3.4 Numărarea autovehiculelor

3.3.4.1 Numărarea simplă și în funcție de clasă

Pentru a implementa numărarea autovehiculelor din imagini, se alege o regiune a imaginii care este intersectată de traiectoria fiecărui autovehicul. Această regiune este determinată de o dreaptă orizontală de numărare care se găsește puțin mai sus față de mijlocul imaginii și este așezată astfel încât să se intersecteze cu fiecare autovehicul de pe fiecare bandă de circulație.

Înainte de analizarea și detecția fiecărei imagini se face numărarea autovehiculelor. Aceasta este făcută pe baza autovehiculelor anterioare stocate în structura de date. Pentru a evita numărarea multiplă a unui autovehicul și pentru a menține consistența, se țin cont de următoarele condiții:

- Autovehiculul trebuie să fie nenumărat, fapt indicat de un atribut al instanței clasei de tip "Vehicle".
- Distanța între originea axei Y și punctul cel mai apropiat de originea axei Y al dreptunghiului de încadrare al autovehiculului trebuie să fie mai mică decât valoarea coordonatelor drepte de numărare corespunzătoare axei Y.
- Distanța între originea axei Y și punctul cel mai îndepărtat de originea axei Y al dreptunghiului de încadrare trebuie să fie mai mare decât valoarea coordonatelor drepte de numărare corespunzătoare axei Y.

Astfel, ținând cont de condițiile de mai sus, dreptunghiul de încadrare a autovehiculului intersectează dreapta de numărare. În continuare se poate apela funcția “count” a instanței de tip “Counter” care incrementează o variabilă generală și una specifică tipului de autovehicul care este numărat. Atributul obiectului de tip “Vehicle” care indică starea de numărare a autovehiculului trebuie să fie actualizat. În figura următoare se prezintă ilustrarea dreptei de numărare și intersecția ei cu autovehiculele.

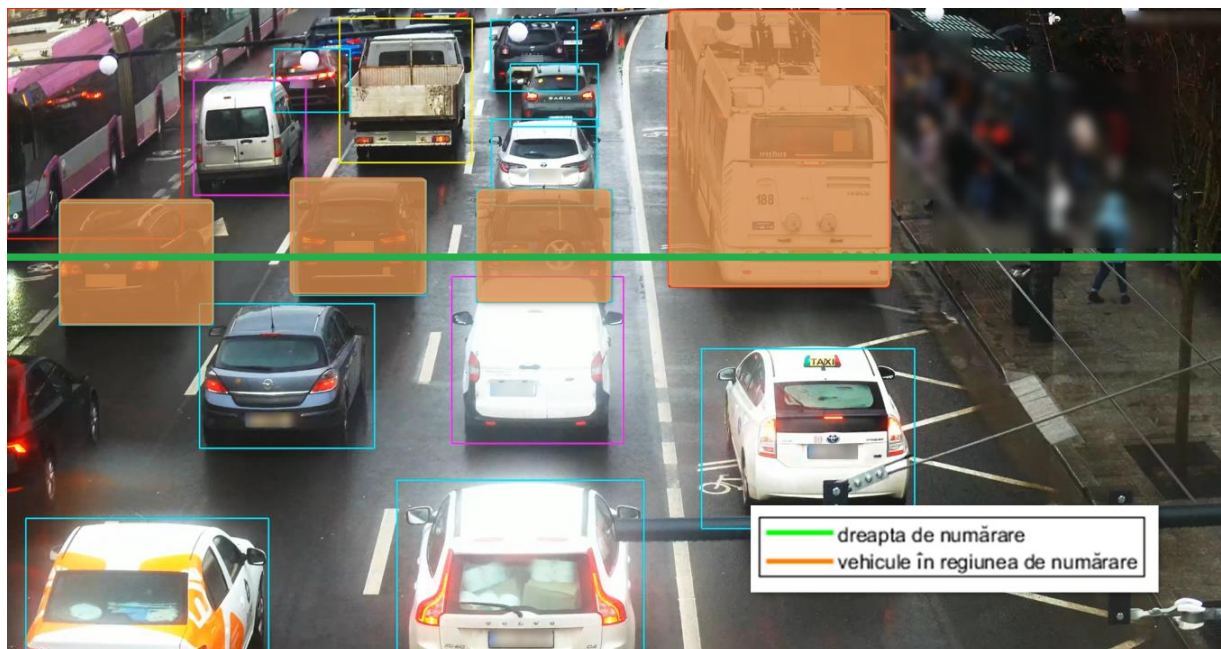


Figura 3.10. Dreapta de numărare și intersecția ei cu autovehiculele

3.3.4.2 Numărarea în funcție de benzile rutiere

Pentru numărarea autovehiculelor de pe fiecare bandă rutieră, avem mai întâi nevoie de numărarea simplă. Astfel, se iau în considerare criteriile și condițiile de numărare prezente la numărarea simplă iar apelarea funcției “count” nu se modifică, ci doar se adaugă o nouă parte acesteia.

Numărarea autovehiculelor în funcție de benzile rutiere implică identificarea benzilor rutiere din imagine. Această identificare este făcută experimental și se consideră validă pentru toate înregistrările provenite de la camera de supraveghere, fiind invalidă doar în cazul în care unghiul sau rezoluția camerei de supraveghere se schimbă.

Odată ce sunt definite șase drepte corespunzătoare marcajelor rutiere, respectiv sunt identificate coordonatele a două puncte de pe aceste drepte, este nevoie de crearea unor condiții de numărare pentru fiecare bandă în parte, ținând cont de unghiul camerei de supraveghere, respectiv:

- Autovehiculele se consideră pe prima bandă (cea mai din dreapta) dacă colțul din stânga sus a dreptunghiului autovehiculului este mai în dreapta față de a doua linie de marcaj rutier.
- Un autovehicul se află pe a doua bandă dacă colțul din stânga sus a dreptunghiului de încadrare se află în apropierea celei de-a treia linii de marcaj rutier pe axa X (mai puțin de 300 de pixeli diferență) și colțul din

dreapta jos se află în apropierea celei de-a doua linii de marcaj rutier pe axa Y (mai puțin de 300 de pixeli diferență).

- Autovehiculul este pe a treia bandă de la stânga la dreapta dacă colțul din stânga sus al autovehiculului este în apropierea relativă a celei de-a patra linii de marcaj rutier pe axa X: mai în dreapta cu cel mult 200 de pixeli sau mai în stânga cu cel mult 900 de pixeli. Distanța spre stânga este mult mai mare datorită unghiului camerei de supraveghere și existenței mașinilor cu dimensiuni mari. Pe lângă această condiție, colțul din dreapta jos trebuie să fie în apropierea celei de-a treia linii de marcaj rutier (mai puțin de 300 de pixeli diferență).
- Pentru a patra bandă, vehiculele se consideră aparținând acesteia dacă colțul din stânga sus este mai în stânga decât cea de-a cincea linie și colțul din dreapta jos este în apropierea celei de-a patra linii de marcaj rutier pe axa X (mai puțin de 200 de pixeli diferență). Această diferență de pixeli este mai mică în acest caz pentru a nu interfera cu detecțiile autovehiculelor cu dimensiuni mari de pe banda a cincea.
- Autovehiculele se consideră pe banda a cincea, ultima de la stânga la dreapta, dacă colțul din stânga sus al dreptunghiului de încadrare este mai în stânga față de a șasea linie de marcaj rutier. Această condiție este suficientă deoarece autovehiculele de pe a șasea bandă rutieră, circulând în sens invers, intră în zona de numărare mult mai târziu în traiectoria lor față de autovehiculele care circulă pe banda a cincea.

De menționat este că aceste condiții se aplică doar în momentul în care dreptunghiul de încadrare al autovehiculului intersectează dreapta de numărare și funcția “count” se apelează.



Figura 3.11. Liniile de marcaj rutier și vecinătățile lor

În Figura 3.7 se observă dreapta orizontală de numărare, dreptele care aproximează liniile de marcaj rutier și vecinătățile liniilor corespunzătoare celei de-a doua benzi pe axa X. Autovehiculul a cărui dreptunghi de încadrare intersectează dreapta de numărare se află în zona de numărare, astfel, funcția “count” este apelată pentru acest vehicul și se execută numărarea simplă și în funcție de clasă. Se observă că pentru numărarea în funcție de benzile rutiere, autovehiculul îndeplinește condițiile pentru a doua bandă de la stânga la dreapta, colțul din stânga sus fiind în vecinătatea celei de-a treia linii iar colțul din dreapta jos fiind în vecinătatea celei de-a doua linii de marcaj rutier pe axa X.

Pentru a determina distanța între colțul dreptunghiului de încadrare a autovehiculului și punctul de pe dreapta care aproximează marcajul rutier, și pentru a verifica dacă această distanță este mai mică decât distanța de prag astfel încât colțul să se considere în apropierea liniei de marcaj rutier, avem nevoie de coordonata colțului corespunzătoare axei Y.

$$m = \frac{y_2 - y_1}{x_1 - x_1} \quad (3.1)$$

Știind coordonata colțului corespunzătoare axei Y, se poate calcula panta dreptei care aproximează marcajul rutier folosind relația (3.1) unde (x_1, y_1) și (x_2, y_2) sunt coordonatele celor două puncte care determină dreapta. După ce panta dreptei este calculată, se folosește relația (3.2) pentru a calcula coordonata corespunzătoare axei X a punctului de pe dreapta care are valoarea coordonatei corespunzătoare axei Y egală cu cea a colțului.

$$x = \frac{y - y_1}{m} + x_1 \quad (3.2)$$

După calculul coordonatei corespunzătoare axei X a punctului de pe linie care are valoarea coordonatei corespunzătoare axei Y egală cu cea a colțului, se poate face diferența între aceste două valori. Această diferență se compară cu diferența de prag specificată pentru fiecare linie și fiecare bandă, și astfel, se pot verifica toate condițiile necesare pentru a determina poziția unui autovehicul în funcție de benzile rutiere.

3.3.5 Detecția plăcilor de înmatriculare

3.3.5.1 Achiziția datelor și antrenarea modelului

Dată fiind rezoluția de dimensiuni relativ mari a imaginilor provenite de la camera video de supraveghere (4096×2160), plăcile de înmatriculare nu se pot trata ca un obiect separat în interiorul imaginii, fiind prea mici. Astfel, pentru detecția acestora, trebuie prelucrat un nou set de date și antrenat un nou model de rețea neuronală pentru detecție.

Pentru a crea setul de date, se decupează din imaginile inițiale imagini care conțin câte un singur autovehicul. Astfel, raportul între rezoluția imaginii și dimensiunea plăcilor de înmatriculare se schimbă, permițând modelului să recunoască și să învețe caracteristici ale plăcilor de înmatriculare. În acest set de date trebuie să se asigure că plăcile de înmatriculare ale autovehiculelor sunt lizibile, pe lângă condițiile general

valabile care se aplică unui set de date și anume să fie suficient de mare și suficient de diversificat.

Se folosește o instanță a modelului YOLOv8n, precum cea folosită pentru detecția autovehiculelor, deci procesele de adnotare a datelor, organizare a datelor, antrenare și configurare a modelului sunt similare cu cele descrise la capitolele 3.3.2.2 și 3.3.2.3.

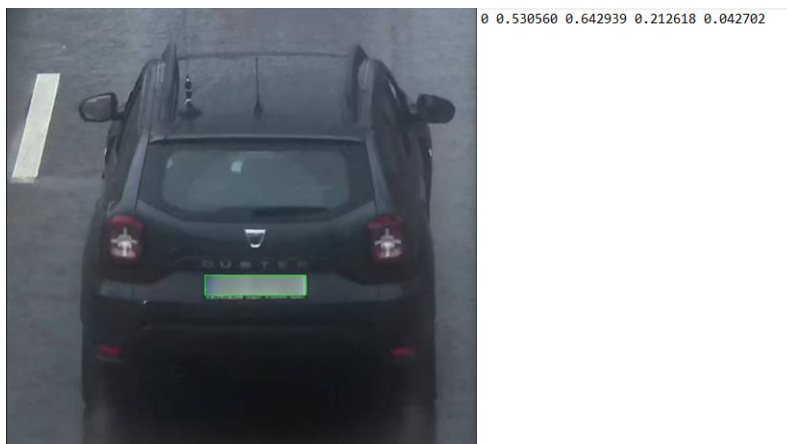


Figura 3.12. Adnotarea datelor pentru modelul de detecție a plăcilor de înmatriculare

În Figura 3.8 se observă în partea stângă o imagine a setului de date și un dreptunghi de încadrare a unei plăci de înmatriculare. În partea dreaptă este fișierul text corespunzător adnotării imaginii care conține coordonatele și dimensiunile plăcii de înmatriculare. Numărul asociat cu clasa obiectului este mereu același pentru orice obiect din orice fișier text, existând o singură clasă de obiecte în setul de date.

3.3.5.2 Recunoașterea numerelor de înmatriculare

Dat fiind modelul descris la subcapitolul anterior, pentru a detecta plăcile de înmatriculare avem mai întâi nevoie de detecția autovehiculelor din imaginile provenite de la camera de supraveghere video. Numărul de înmatriculare identificat este salvat în structura de date asociată autovehiculului respectiv, procesul de identificare a numerelor de înmatriculare făcându-se o singură dată pentru fiecare autovehicul.

Pentru a identifica numerele de înmatriculare și a recunoaște caracterele care se află pe acestea, fiecare autovehicul identificat va genera o copie a imaginii originale, decupată utilizând coordonatele dreptunghiului de încadrare a autovehiculului. Această imagine decupată este intrarea modelului pentru identificarea plăcilor de înmatriculare. După ce se face predicția, modelul returnează anumite date, printre care și coordonatele plăcii de înmatriculare identificate. Folosind aceste coordonate, imaginea se decupează din nou, astfel rezultând o imagine care conține numai placa de înmatriculare și conținutul ei.

Identificarea caracterelor necesită în primul rând o preprocesare a imaginii care conține numărul de înmatriculare. Această preprocesare ajută la acuratețea și viteza de citire. Dimensiunile imaginii sunt modificate pentru a ajuta la îmbunătățirea detaliilor și se folosește biblioteca OpenCV pentru a transforma imaginea color într-o imagine alb-negru, din moment ce culorile nu sunt relevante în contextul identificării de caractere. Ultimul pas al preprocesării este decuparea din partea stânga a imaginii. Se elimină a

zecea parte din lăţimea imaginii, păstrându-se înălţimea, pentru a înlătura partea plăcii de înmatriculare care nu conţine caractere.

După ce procesarea imaginii este finalizată, se foloseşte o instanţă a unui obiect de tip "Reader" a bibliotecii EasyOCR pentru a extrage informaţia din imagine într-un format text. Acest text reprezintă numărul de înmatriculare al autovehiculului şi este salvat în structura de date corespunzătoare acestuia.



Figura 3.13. Structura procesului de recunoaştere a numerelor de înmatriculare

3.3.5.3 Optimizarea algoritmului

Pentru a creşte acurateţea şi viteza identificării numerelor de înmatriculare se implementează o serie de îmbunătăţiri pentru algoritmul de detecţie, ţinându-se cont de formatul predominant de numere de înmatriculare, rezoluţia şi poziţionarea camerei de supraveghere video, şi nevoia de procesare relativ rapidă a imaginilor astfel încât acestea să poată să fie utilizate într-un context de timp real. Îmbunătăţirile aplicate algoritmului sunt următoarele:

- Se transformă toate literele identificate în litere mari pentru a nu produce confuzii şi pentru a păstra consistenţa. Literele pot fi interpretate de către modelul EasyOCR în diferite dimensiuni.
- Condiţionarea regiunii de identificare a numerelor de înmatriculare şi recunoaştere a caracterelor în interiorul imaginii este necesară pentru ca identificarea caracterelor să fie făcută. Numerele de înmatriculare sunt ilizibile în partea stângă a imaginii care corespunde cu a patra şi a cincea bandă de circulaţie rutieră. Excluderea tentativelor de identificare în această zonă sporeşte semnificativ procesarea totală a imaginilor.

- Se exclude din regiunea de identificare a numerelor de înmatriculare regiunea imaginii care se află deasupra de dreapta de numărare. Această excludere afectează doar primele imagini ale transmisiunii video, fiind eficientă și din punct de vedere al acurateții.
- Se verifică scorul obținut în urma predicției făcute de către modelul EasyOCR și se exclud rezultatele care au acest scor mai mic decât un scor de prag și rezultatele a căror text conține mai puțin de șase caractere.
- Pentru formatul de număr de înmatriculare existent în România, se aplică o funcție de îmbunătățire care transformă literele în cifre și invers în funcție de acest format. Numerele de înmatriculare care nu sunt în acest format sunt ignorate de această funcție.
- Încercarea de citire a numerelor de înmatriculare, în cazul în care aceasta eșuează sau rezultatul nu este valid, este limitată de un număr implicit sau setat de utilizator. Această limitare ajută semnificativ la creșterea vitezei de procesare a imaginilor.

Pentru a identifica formatul de numere de înmatriculare existent în România se verifică în primul rând dacă primul caracter este o literă și dacă numărul caracterelor care compun textul este egal cu șase sau șapte. În continuare, se verifică fiecare caracter și se modifică dacă în poziția în care se află, caracterul este o cifră și se așteaptă să fie o literă, sau invers.

Fiecare număr de înmatriculare poate să fie citit de un număr maxim de ori pentru a nu scădea semnificativ viteza de procesare a transmisiunii de imagini în cazul în care algoritmul de citire a numărului de înmatriculare returnează un rezultat invalid. Dacă se execută acest număr maxim de aplicări ale algoritmului de citire și nu se returnează un rezultat valid, placa de înmatriculare a autovehiculului este considerată ilizibilă și structura de date corelată autovehiculului se schimbă astfel încât să indice faptul că algoritmul de citire nu trebuie aplicat acestuia. În cazul în care nu se aplică această restricție, algoritmul de citire a plăcii de înmatriculare se va aplica pentru fiecare imagine în care autovehiculul este existent, fapt care ar crește semnificativ timpul de procesare al imaginilor.

```
dict_letter_to_figure = {
    '0': '0',
    '1': '1',
    '2': '2',
    '3': '3',
    '4': '4',
    '5': '5',
    '6': '4',
    '7': '7',
    '8': '8',
    '9': '0',
    'R': '2',
}

dict_figure_to_letter = {
    '0': '0',
    '1': '1',
    '2': '2',
    '3': '3',
    '4': '6',
    '5': 'S',
    '6': '6',
    '7': 'T',
    '8': 'B',
}
```

Figura 3.14. Dicționarele de conversie a caracterelor

3.3.6 Implementarea interfeței grafice pentru utilizator

3.3.6.1 Elementele grafice

Pentru a crea interfața pentru utilizator se folosește biblioteca Qt disponibilă pentru limbajul de programare Python și aplicația Qt Designer pentru a ușura procesul de adăugare și așezare a elementelor grafice. Se folosește ca element vizual principal o instanță a clasei “QMainWindow” și mai multe instanțe ale obiectelor de tip “QCheckBox”, “QSpinBox”, “QLineEdit” și altele. Aceste elemente sunt așezate în diferite structuri oferite de biblioteca Qt pentru a oferi un aspect vizual plăcut interfeței grafice. Un exemplu pentru o astfel de structură este “QVBoxLayout” care este folosită pentru alinierea obiectelor pe verticală.

Object	Class
MainWindow	QMainWindow
centralwidget	QWidget
bottom_layout	QHBoxLayout
drawing_groupBox	QGroupBox
horizontalLayout	QHBoxLayout
verticalLayout	QVBoxLayout
boxes_checkBox	QCheckBox
classes_checkBox	QCheckBox
number_plates_checkBox	QCheckBox
verticalLayout_2	QVBoxLayout
class_counting_checkBox	QCheckBox
lane_counting_checkBox	QCheckBox
total_counting_checkBox	QCheckBox
verticalLayout_3	QVBoxLayout
counting_line_checkBox	QCheckBox
ids_checkBox	QCheckBox
lanes_checkBox	QCheckBox
parameters_groupBox	QGroupBox
horizontalLayout_3	QHBoxLayout
verticalLayout_4	QVBoxLayout
label_confidence_nb	QLabel
label_confidence_veh	QLabel
label_reading_attempts	QLabel
label_tracking_depth	QLabel
verticalLayout_5	QVBoxLayout
plates_confidence_spinBox	QDoubleSpinBox
reading_attempts_spinBox	QSpinBox
tracking_depth_spinBox	QSpinBox
vehicles_confidence_spinBox	QDoubleSpinBox
right_layout	QVBoxLayout
streaming_groupBox	QGroupBox
horizontalLayout_2	QHBoxLayout
start_stream_pushButton	QPushButton
stop_stream_pushButton	QPushButton
stream_input_lineEdit	QLineEdit
track_plate_groupBox	QGroupBox
horizontalLayout_4	QHBoxLayout
clear_tracking_pushButton	QPushButton
track_plate_pushButton	QPushButton
tracked_plate_lineEdit	QLineEdit
video_groupBox	QGroupBox
generate_statistics_checkBox	QCheckBox
process_video_pushButton	QPushButton
verticalLayout_6	QVBoxLayout
video_input_lineEdit	QLineEdit
video_output_lineEdit	QLineEdit
video_layout	QGridLayout
menubar	QMenuBar
statusbar	QStatusBar

Figura 3.15. Structura și ierarhia elementelor grafice ale ferestrei principale

Elementele grafice sunt așezate vizual în fereastră folosind aplicația Qt Designer care permite plasarea obiectelor folosind mouse-ul. După ce ferestrele au fost create, aplicația le salvează sub forma unor fișiere specifice acestora care au extensia “ui”. Pentru a transforma aceste fișiere în fișiere Python, se rulează în consolă comanda “pyside6-uic source/window.ui -o source/window.py” unde “source/window.ui” este calea către

fișierul generat de aplicația Qt Designer iar “source/window.py” este calea către fișierul Python care este generat.

Fișierul Python generat este folosit pentru a accesa și a controla comportamentul componentelor grafice. Se folosește o instanță a clasei “mkQApp” care aparține librăriei “pyqtgraph” pentru afișarea componentelor grafice. Această instanță are ca atribut o instanță a clasei principale de control a interfeței de utilizator numită “MainGUI”. Clasa “MainGUI” conține elementele grafice preluate de la fișierul Python generat și o instanță a clasei “Analyzer” care conține toate funcțiile necesare pentru procesarea transmisiilor de imagini. Pe lângă aceste instanțe, în această clasă se află logica de control a tuturor elementelor grafice și ale funcționalităților acestora.

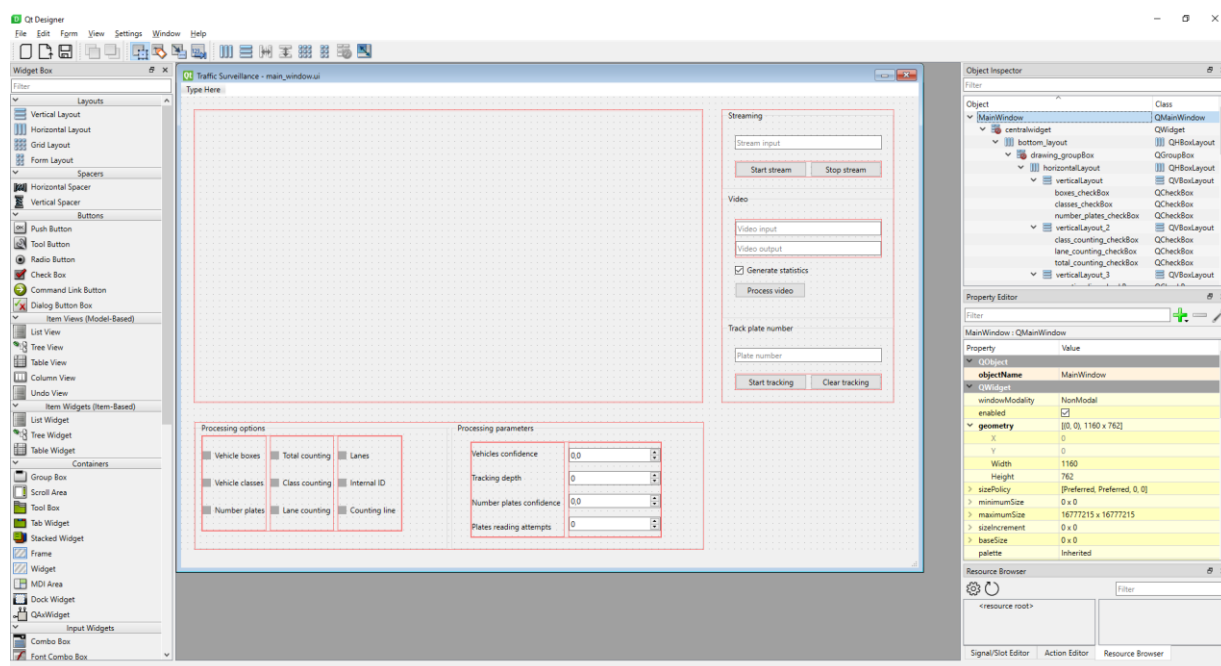


Figura 3.16. Interfața grafică a aplicației Qt Designer

3.3.6.2 Funcționalitățile interfeței

Prima parte a funcționalităților interfeței pentru utilizator este cea de procesare a unui flux de imagini care sunt transmise în direct de la camera de supraveghere. Pentru a începe procesarea se introduce adresa URL a fluxului video sau calea către un fișier video pentru a simula o transmitere în direct de imagini. La apăsarea butonului de start se afișează în fereastra principală pe rând, fiecare imagine, după ce tot algoritmul de procesare a fost aplicat acesteia. Se folosește biblioteca OpenCV pentru a captura imaginile și se aplică pentru fiecare imagine algoritmi descriși la subcapitolele anterioare în funcție de alegerile utilizatorului. Procesarea imaginilor se continuă până la oprirea transmisiunii sau până la apăsarea butonului de stop.

A doua parte a funcționalităților constă în procesarea fișierelor video. Se introduce o cale către fișierul video care se dorește a fi procesat și o cale către locația unde va fi salvat fișierul după procesare. Asemănător cu cazul transmisiunii în direct, se folosește biblioteca OpenCV pentru a procesa fiecare imagine a videoclipului în funcție de alegerile utilizatorului. Există opțiunea de a genera un fișier cu statistici în același folder cu videoclipul procesat, folosind biblioteca Pandas, care să conțină următoarele informații:

- Numerele de înmatriculare ale tuturor autovehiculelor care au fost identificate în videoclip.
- Numerele de înmatriculare ale autovehiculelor care au fost identificate în videoclip și se află pe lista de căutare.
- Numărul total de autovehicule identificate în videoclip în funcție de tipul lor și banda rutieră pe care se află.

	A	B	C	D	E	F	G	H	I	J	K	L
1	Recorded plate numbers	Tracked plate numbers	Vehicles	Cars	Trucks	Busses	Vans	First lane	Second lane	Third lane	Fourth lane	Fifth lane
2	11-12-10000	11-12-10000	43	41	0	2	0	1	13	14	14	1
3												
4												
5												
6												
7												
8												
9												
10												
11												
12												
13												
14												
15												
16												
17												
18												
19												
20												
21												
22												
23												
24												
25												
26												
27												

Figura 3.17. Fișierul cu statistici generat în urma procesării unui videoclip

Pentru a adăuga un număr de înmatriculare în lista de căutare se introduce numărul în secțiunea dedicată și se apasă butonul “Start tracking”, iar pentru a curăța lista de căutare, se utilizează butonul “Clear tracking”. În cazul în care un număr de înmatriculare este identificat în timpul unei transmisiuni în direct, utilizatorul primește o notificare și aplicația începe înregistrarea automată a imaginilor pentru câteva secunde. În cazul în care numărul este identificat în timpul procesării unui video, acesta se adaugă în lista de numere de înmatriculare căutate din fișierul cu statistici.

Utilizatorul are la dispoziție un set de opțiuni pentru procesarea imaginilor în partea de jos a ferestrei principale. Aceste opțiuni pot să fie alese în orice fel și modifică algoritmul de procesare a imaginilor, fiecare opțiune având două stări: activă și inactivă. Cele nouă opțiuni sunt următoarele:

- Dreptunghiuri de încadrare a vehiculelor: Determină dacă dreptunghiurile de încadrare sunt prezente în imaginea procesată sau nu. Indiferent de această opțiune, procesul de identificare a autovehiculelor este activ și nu poate fi oprit.
- Clasa autovehiculelor: Afișează tipul autovehiculelor identificate în imagini sau nu.

- Numere de înmatriculare: Determină dacă procesul de identificare și citire a numerelor de înmatriculare este activ sau nu. Activarea acestei opțiuni poate crește semnificativ timpul de procesare a imaginilor.
- Numărare totală: Afișează în imagine sau în fișierul de statistici numărul total de vehicule identificate.
- Numărare în funcție de clasă: Afișează în imagine sau în fișierul de statistici numărul total de vehicule identificate de un anumit tip.
- Numărarea în funcție de benzi: Afișează în imagine sau în fișierul de statistici numărul total de vehicule identificate în funcție de benzile rutiere.
- Benzi rutiere: Afișează în imaginile procesate aproximarea marcajelor rutiere sau nu.
- Identificator intern: Afișează pentru fiecare autovehicul un număr natural corespunzător cu identificatorul intern al acestuia. Acest identificator se atribuie la identificarea vehiculului și este unic pentru fiecare vehicul din imagine.
- Dreapta de numărare: Afișează sau nu în imaginile procesate dreapta care delimitează regiunea de numărare a autovehiculelor.

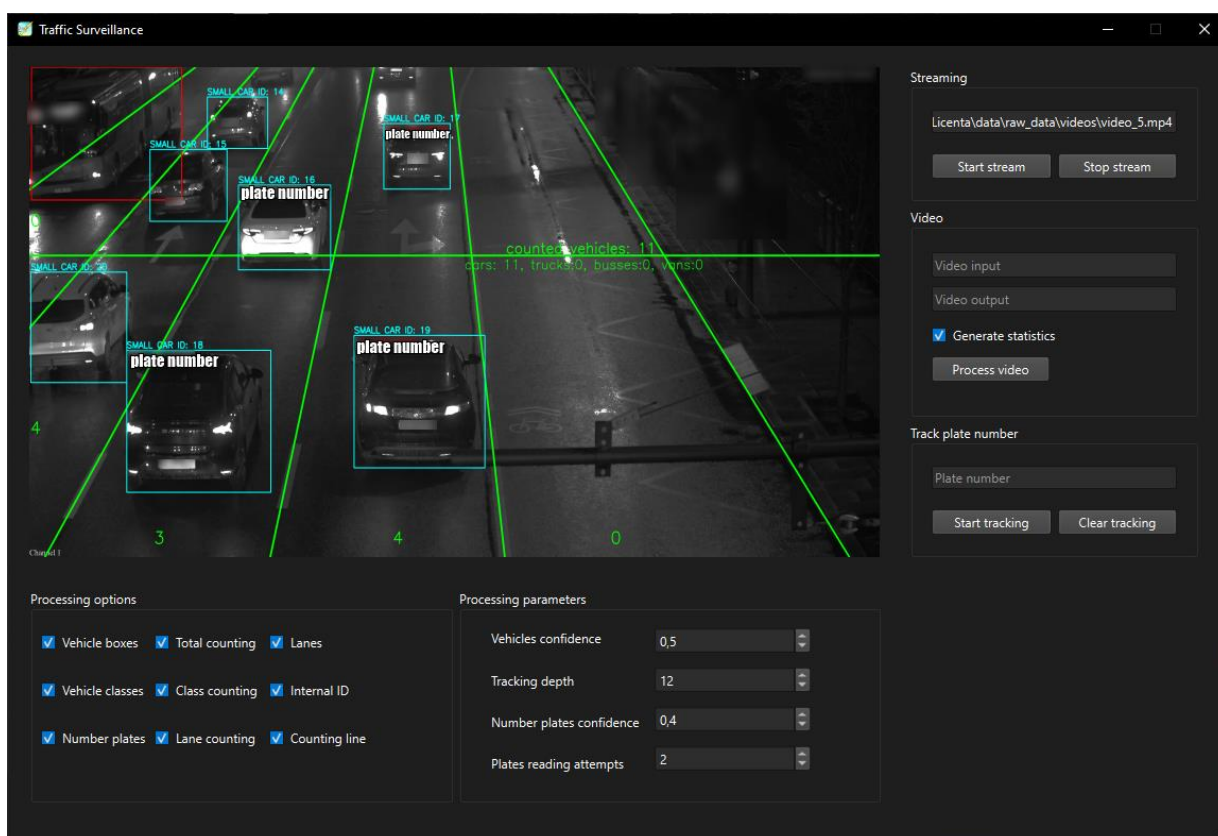


Figura 3.18. Fereastra principală a interfeței grafice pentru utilizator

Pe lângă opțiunile de procesare, utilizatorul mai are la dispoziție un set de parametri de procesare care pot să fie modificați. Modificarea acestor parametri presupune un set minimalist de cunoștințe asupra funcționării algoritmilor de identificare a obiectelor și de citire a numerelor de înmatriculare. Cei patru parametri sunt:

- Scorul autovehiculelor: Determină scorul de prag al modelului YOLO de detecție a autovehiculelor.
- Scorul numerelor de înmatriculare: Determină scorul de prag al modelului EasyOCR de citire a numerelor de înmatriculare.
- Profunzimea urmăririi: Determină numărul de imagini anterioare pe care algoritmul de urmărire să le considere. Un număr relativ mare dat acestui parametru ajută la continuitatea algoritmului în cazul în care există o lipsă a detecțiilor. Totuși, un număr prea mare ar putea duce la performanțe mai scăzute ale algoritmului.
- Numărul de încercări de citire: Numărul maxim de încercări de a identifica și de a citi numerele de înmatriculare pentru fiecare autovehicul. Un număr mare dat acestui parametru poate crește semnificativ timpul de procesare al imaginilor.

Pe langa fereastra principală a interfeței grafice pentru utilizator, există și diferite ferestre de dialog pentru a ajuta utilizatorul să ia decizii sau pentru a-l informa cu privire la diferite aspecte. În figura următoare este prezentată fereastra de dialog care apare la pornirea aplicației în cazul în care utilizatorul nu deține o unitate de procesare grafică în sistemul de calcul.

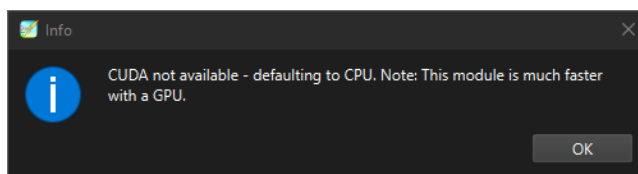


Figura 3.19. Fereastră de dialog pentru informarea utilizatorului

3.3.6.3 Generarea unui fișier executabil

Pentru a face experiența utilizatorului cât mai plăcută și pentru a nu-l forța să configureze mediul de dezvoltare pentru a folosi aplicația, se consideră generarea unui fișier executabil. Acest singur fișier pornește aplicația pe sistemele de operare Windows, făcând-o disponibilă utilizatorului.

Se folosește biblioteca PyInstaller pentru generarea unui singur fișier executabil care nu are consolă. Având în vedere faptul că este vorba de un singur fișier, este nevoie ca toate datele utilizate de aplicație să fie incluse în acest fișier. Se includ manual modelele YOLO create și antrenate pentru detecții și biblioteca Ultralytics din cauza unei erori de includere automată. Pentru ușurința utilizării comenzii de creare a fișierului executabil, se scrie un fișier Python care folosește biblioteca “subprocess” pentru a rula comanda PyInstaller în consolă. În figura următoare se poate observa conținutul acestui fișier Python.

```

import os
import subprocess

from source.utils.variables import STREAM_BACKGROUND_PATH

# extra data
icon_path = os.path.join('.', 'source', 'gui', 'images', 'app_icon.ico')
# we need to add this manually: https://github.com/ultralytics/ultralytics/tree/main/ultralytics
ultralytics = os.path.join('.', 'dev_setup', 'exe_gen_extra_files', 'ultralytics')
runs = os.path.join('.', 'runs') # yolo detection models
main_script = os.path.join('.', 'main.py')

pyinstaller_cmd = [
    'pyinstaller',
    '--noconfirm',
    '--onefile',
    '--windowed',
    '--icon', f'{icon_path}',
    '--name', "Traffic Surveillance",
    '--add-data', f'{ultralytics};ultralytics/',
    '--add-data', f'{runs};runs/',
    '--add-data', f'{STREAM_BACKGROUND_PATH};.',
    '--distpath', os.path.join(os.getcwd(), 'dist'),
    f'{main_script}'
]

subprocess.run(pyinstaller_cmd, shell=True)

```

Figura 3.20. Comanda de generare a unui fișier executabil

3.4 Testare și validare

3.4.1 Validarea detecțiilor și a algoritmilor

Testarea algoritmilor și a detecțiilor a fost efectuată folosind imagini și înregistrări provenite de la camera de supraveghere video a traficului. Pentru modelele de detecție s-au folosit părțile setului de date responsabile pentru validare și testare.

În figura următoare se observă testarea modelului de detecție a autovehiculelor făcută automat de către modelul YOLO în timpul antrenării folosind setul de date de validare. Pe axa X a graficelor este reprezentat numărul de iterații care este folosit pentru antrenarea modelului iar pe axa Y este reprezentată valoarea funcției de pierdere. În graficul din stânga este reprezentată validarea dreptunghiurilor de încadrare iar în partea dreaptă este graficul pentru validarea clasificărilor de autovehicule. În ambele cazuri se observă o scădere a valorilor funcțiilor de pierdere ceea ce înseamnă că predicțiile modelului făcute pe datele de validare devin din ce în ce mai acurate pe măsură ce numărul de iterații de antrenare crește. Se observă de asemenea o plafonare a scăderilor valorilor funcțiilor de pierdere ceea ce indică faptul că numărul de iterații este potrivit și o creștere a acestuia ar duce la fenomenul de supraantrenare.

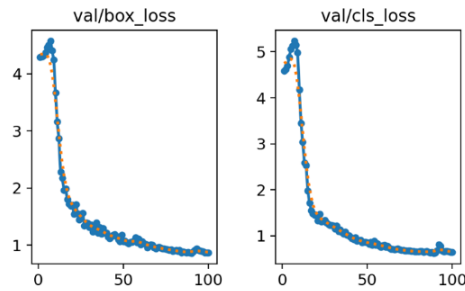


Figura 3.21. Validarea antrenării modelului de detecție a autovehiculelor

O altă metodă de validare a modelului de detecție a autovehiculelor este matricea de confuzie generată pe baza adnotărilor și a detecțiilor făcute pentru setul de date de validare. În partea de jos este notat pentru fiecare coloană tipul autovehiculului adnotat iar în partea stângă este notat pentru fiecare linie tipul autovehiculului identificat. Fiecare valoare din matrice corespunde cu o corelație între două tipuri de autovehicule, unul adnotat iar celălalt identificat. În figura următoare se observă matricea de confuzie a modelului de identificare a autovehiculelor.

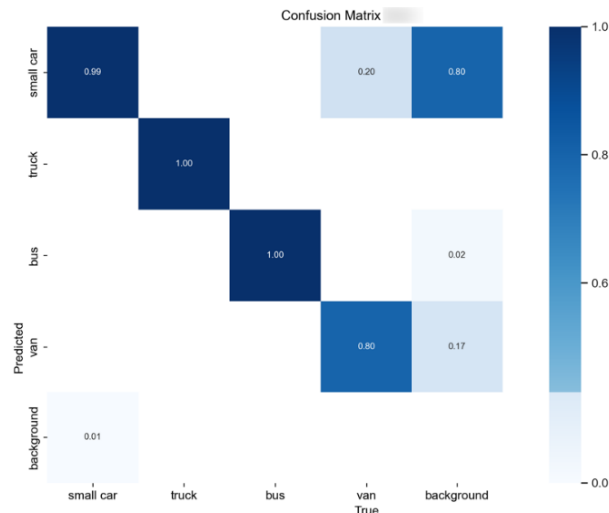


Figura 3.22. Matricea de confuzie a modelului de identificare a autovehiculelor

Pe baza setului de date de validare, din matricea de confuzie prezentată mai sus reies următoarele performanțe:

- Mașinile mici sunt identificate corect în proporție de 99%, restul de 1% sunt identificate greșit drept fundal.
- Toate autobuzele și camioanele sunt identificate corect.
- Camionetele sunt identificate corect în proporție de 80%, restul de 20% fiind identificate greșit drept mașini mici.
- Fundalul imaginii este des interpretat de către model ca fiind mașină mică.

Este de luat în considerare faptul că setul de date de validare și setul de date generale sunt restrânse, astfel, performanțele modelului vizibile în matricea de confuzie sunt o aproximare a performanțelor reale. De asemenea, fundalul imaginii este adesea

interpretat greșit de către model din cauza neexistenței de imagini în setul de date în care nu sunt vizibile autovehicule.

În continuare, în figura următoare sunt prezentate graficele de performanțe ale învățării modelului pentru detecția plăcilor de înmatriculare. Se observă performanțe asemănătoare primului model YOLO, diferența majoră între cele două modele fiind faptul că în acest model există o singură clasă de obiecte.

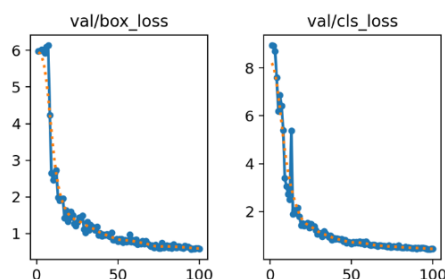


Figura 3.23. Validarea antrenării modelului de detecție a plăcilor de înmatriculare

Testarea algoritmului de urmărire s-a făcut folosind conceptul de identificator intern. Fiecărui autovehicul îi este atribuit un număr natural unic în imagine pentru a observa dacă algoritmul de urmărire funcționează. În cazul în care algoritmul funcționează, se observă că acest identificator nu se modifică pe parcursul videoclipului pentru un autovehicul dat.

Algoritmii de numărare a autovehiculelor și de identificare a numerelor de înmatriculare au fost validați manual, observând performanța acestora pentru setul de date de testare. Rezultatele algoritmilor au fost comparate cu datele vizibile în înregistrările provenite de la camera de supraveghere.

3.4.2 Testarea interfeței de utilizator și a funcționalităților

Interfața grafică pentru utilizator și funcționalitățile acesteia au fost testate manual, verificând fiecare componentă grafică în diferite scenarii și condiții. Pentru a verifica funcționalitatea procesării imaginilor în contextul transmiterii în direct a datelor, s-au folosit înregistrări video și s-au tratat precum un flux de date transmis în direct, neavând acces la camera de supraveghere video.

S-au folosit înregistrări provenite de la camera de supraveghere video pentru a testa funcționalitatea procesării de videoclipuri și a generării de statistici. Pentru partea de urmărire a numerelor de înmatriculare, s-au observat câteva numere de înmatriculare în înregistrarea video și s-au adăugat în lista de numere de înmatriculare urmărite. După procesarea videoclipului, numerele de înmatriculare urmărite s-au observat în fișierul cu statistici generate în coloana cu numele "Tracked plate numbers".

Pentru a testa opțiunile și parametrii de procesare a imaginilor disponibili în interfața vizuală, s-a simulat transmiterea în direct a unor date și s-au schimbat, pe rând, fiecare parametru și fiecare opțiune și s-au observat vizual rezultatele. Pentru a testa parametrul de profunzime a urmăririi, acestuia i se atribuie valoarea zero și se observă cum identificatorul intern al vehiculelor este schimbat la fiecare imagine. Parametrii responsabili pentru scorul modelelor de identificare se testează atribuindu-le valoarea

maxima, astfel, se observă lipsa de detecții în interiorul imaginilor. Testarea parametrului responsabil de numărul de încercări de citire a numerelor de înmatriculare poate fi testat fiindu-i atribuită valoarea zero și observând lipsa identificării de numere de înmatriculare în imagini.

Fiecare metodă manuală descrisă mai sus este repetată în diverse stări ale aplicației și folosind diferite imagini pentru a acoperi un număr cât mai mare de cazuri particulare de funcționare a aplicației. În cazul în care aplicația nu funcționează, se identifică problema și, după ce problema este rezolvată, testarea aplicației se repetă.

4 Concluzii

4.1 Rezultate obținute

Obiectivul principal al acestui proiect este îmbunătățirea siguranței rutiere și a traficului rutier. Acest lucru este posibil datorită tehnologiilor de detecție a obiectelor din imagini digitale și datorită tehnicilor de urmărire și analizare a obiectelor. Implementarea proiectului în discuție a presupus dezvoltarea unei aplicații software care să fie capabilă să detecteze și să clasifice autovehiculele și numerele de înmatriculare ale acestora în timp real.

Obiectivele și specificațiile proiectului au fost îndeplinite în totalitate, realizându-se astfel următoarele sarcini:

- Antrenarea unui model de rețea neuronală pentru identificarea autovehiculelor și a plăcilor de înmatriculare ale acestora.
- Identificarea autovehiculelor dintr-o imagine folosind modelul de rețea neuronală.
- Clasificarea autovehiculelor în patru categorii: „Mașină mică”, „Camion”, „Camionetă” și „Autobuz”.
- Implementarea unui algoritm de urmărire a autovehiculelor în contextul unui format video.
- Numărarea autovehiculelor în contextul unui format video.
- Numărarea autovehiculelor în funcție de categoria autovehiculului.
- Numărarea autovehiculelor în funcție de banda pe care se află pe șosea.
- Identificarea plăcilor de înmatriculare pentru fiecare autovehicul folosind modelul de rețea neuronală.
- Identificarea textului de pe plăcile de înmatriculare folosind un model de rețea neuronală.
- Optimizarea cât mai bună a tuturor proceselor pentru a putea fi folosite într-un context de timp real.
- Împlementarea unei interfețe de utilizator.

În concluzie, îndeplinind toate obiectivele de mai sus, s-a obținut un sistem capabil să identifice, clasifice și să numere autovehiculele și numerele de înmatriculare ale acestora într-un timp relativ scurt. Acest sistem poate să fie folosit de către un utilizator obișnuit, fără cunoștințe tehnice, datorită interfeței grafice pentru utilizator, și poate să fie folosit pentru a ajuta la gestionarea traficului și la îmbunătățirea siguranței rutiere.

Comparativ cu alte lucrări, acest proiect a constituit o dificultate în special datorită nevoii de procesare cât mai rapidă a datelor. Acesta a fost de asemenea primul proiect în care am avut ocazia să învăț fundamentele procesării imaginilor și să aplic aceste cunoștințe în contextul formatelor video.

Din punct de vedere al contribuțiilor personale asupra domeniului temei abordate, acestea sunt următoarele:

- Învățarea și aplicarea cunoștințelor de bază despre procesarea imaginilor și funcționarea modelelor de rețele neuronale.
- Antrenarea și pregătirea modelelor de rețele neuronale pentru detecția obiectelor.
- Dezvoltarea algoritmilor necesari urmăririi și numărării obiectelor.
- Dezvoltarea aplicației software care să permită utilizatorilor folosirea cu ușurință a modelelor de detecție și a algoritmilor.

4.2 Direcții de dezvoltare

În dezvoltarea acestui proiect s-au îndeplinit toate obiectivele inițiale. Totuși, performanțele și funcționalitățile obținute sunt departe de a fi perfecte, astfel, există multe îmbunătățiri care ar putea fi făcute. Având în vedere atât optimizarea funcționalităților actuale cât și posibile dezvoltări ale unor noi funcționalități, direcțiile de dezvoltare propuse sunt următoarele:

- Îmbunătățirea modelelor de detecție.
- Adăugarea claselor pentru detecția bicicletelor și a motocicletelor.
- Găsirea unei soluții eficiente pentru identificarea aglomerației traficului.
- Optimizarea algoritmilor de procesare a imaginilor.
- Procesarea mai puținor imagini din transmisiunea în direct a datelor pentru a crește numărul de imagini afișate într-o secundă.
- Afișarea listei de urmărire a numerelor de înmatriculare în interfața grafică pentru utilizator.
- Antrenarea modelului EasyOCR folosind un set specific de date de la camera de supraveghere video.
- Adăugarea opțiunilor de folosire a modelelor de detecție mai performante pentru cazurile în care sistemul de calcul permite acest lucru.
- Identificarea autovehiculelor care încalcă linia rutieră continuă.
- Implementarea filtrării Kalman pentru algoritmul de urmărire.
- Implementarea unui sistem de securitate care să nu permită accesul neautorizat la aplicație.
- Îmbunătățirea vizualizării datelor și a statisticilor în urma procesării imaginilor.
- Extinderea sistemului pentru mai multe camere de supraveghere.

Pentru îmbunătățirea modelului de detecție și adăugarea unor noi tipuri de vehicule este nevoie de mărirea semnificativă a setului de date, fapt care include accesul la camera de supraveghere video a traficului. De asemenea, extinderea sistemului pentru mai multe camere de supraveghere presupune mai întâi accesul la aceste camere.

Antrenarea modelului EasyOCR poate fi făcută construind un set de date specific din setul inițial de date iar adăugarea opțiunilor de folosire a modelelor de detecție mai performante poate fi făcută folosind alte modele YOLO, modele mai performante dar care au un timp de răspuns mai mare.

5 Bibliografie

- [1] Goodfellow, Y. Bengio, and A. Courville, "Chapter 6: Deep Learning Models," in *Deep Learning*, MIT Press, 2016.
- [2] LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436-444, 2015.
- [3] "Basic structure of neural network," 2020. [Online]. Available: https://www.researchgate.net/figure/Basic-structure-of-neural-network_fig1_341716390.
- [4] A. Dougherty, "Magic of the Sobel Operator," 2020. [Online]. Available: <https://towardsdatascience.com/magic-of-the-sobel-operator-bbbcb15af20d>.
- [5] S. Saha, "A Comprehensive Guide to Convolutional Neural Networks — the ELI5 way," 2018. [Online]. Available: <https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53>.
- [6] "SORT-ros," 2021. [Online]. Available: <https://github.com/Hyunje/SORT-ros>.
- [7] F. Churchville, "User Interface (UI)," 2024. [Online]. Available: <https://www.techtarget.com/searcharchitecture/definition/user-interface-UI>.
- [8] "Despre limbajul de programare Python," [Online]. Available: <https://www.pythonisti.ro/despre-limbajul-de-programare-python.php>.
- [9] Python Software Foundation, "Execution model," 2024. [Online]. Available: <https://docs.python.org/3/reference/executionmodel.html>.
- [10] A. Bovik, "Chapter 14: Edge Detection," in *Handbook of Image and Video Processing*, 2005.
- [11] J. Canny, "A computational approach to edge detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vols. PAMI-8, no. 6, pp. 679-698, 1986.
- [12] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International Journal of Computer Vision*, vol. 60, no. 2, pp. 91-110, 2004.
- [13] S. Haykin, "Chapter 3: Activation Functions," in *Neural Networks and Learning Machines*, 2008.

- [14] M. Nielsen, "Neural Networks and Deep Learning," 2015. [Online]. Available: <http://neuralnetworksanddeeplearning.com/>.
- [15] P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features," in *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2001.
- [16] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2005.
- [17] J. Redmon, S. Divvala, R. Girshick and A. Farhadi, "You Only Look Once: Unified, Real-Time Object Detection," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016.
- [18] S. M. Lucas, "Robust Reading Competitions: Entries, Results, and Future Directions," *International Journal on Document Analysis and Recognition*, vol. 7, no. 2-3, pp. 105-122, 2005.
- [19] B. D. Lucas and T. Kanade, "An iterative image registration technique with an application to stereo vision," in *Proceedings of the 7th International Joint Conference on Artificial Intelligence*, 1981.
- [20] D. Comaniciu, V. Ramesh and P. Meer, "Kernel-based object tracking," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 25, no. 5, pp. 564-577, 2003.
- [21] E. R. Kalman, "A new approach to linear filtering and prediction problems," *Journal of Basic Engineering*, vol. 82, no. 1, pp. 35-45, 1960.
- [22] A. Bewley, Z. Ge, L. Ott, F. Ramos and B. Upcroft, "Simple Online and Realtime Tracking," in *Proceedings of the IEEE International Conference on Image Processing*, 2016.
- [23] JetBrains, "PyCharm: The Python IDE for data science and web development," JetBrains, [Online]. Available: <https://www.jetbrains.com/pycharm/>.
- [24] S. Chacon and B. Straub, "Pro Git," Apress, [Online]. Available: <https://git-scm.com/book/en/v2>.
- [25] GitHub, "Let's build from here," GitHub, [Online]. Available: <https://github.com/>.
- [26] A. Bochkovskiy, "Ultralytics YOLOv8," Ultralytics, [Online]. Available: <https://github.com/ultralytics/ultralytics>.
- [27] O. Team, "OpenCV is the world's biggest computer vision library," OpenCV.org, [Online]. Available: <https://opencv.org/>.

- [28] J. Han, "Ready-to-use OCR," [Online]. Available: <https://github.com/JaidedAI/EasyOCR>.
- [29] T. P. D. Team, "Pandas: Powerful Python Data Analysis Toolkit," The Pandas Development Team, [Online]. Available: <https://pandas.pydata.org/>.
- [30] T. Q. Company, "Qt for Python (PySide6)," The Qt Company, [Online]. Available: <https://www.qt.io/qt-for-python>.
- [31] M. S. Seung and K. J. Lee, "Gaussian filter design for image smoothing and its impact on subsequent processing," *IEEE Transactions on Signal Processing*, vol. 68, no. 2, pp. 345-358, 2020.
- [32] A. A. Arifi and B. G. Liu, "Efficient implementation of Gaussian and Laplacian filters on FPGA," in *IEEE International Conference on Image Processing*, 2019.
- [33] A. Krizhevsky, I. Sutskever and G. E. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks," *Advances in Neural Information Processing Systems*, vol. 25, pp. 1097-1105, 2012.