MINISTERUL EDUCAȚIEI
UNIVERSITATEA TEHNICĂ
DIN CLUJ-NAPOCA
FACULTATEA DE AUTOMATICĂ ȘI CALCULATOARE

SUPRAVEGHEREA VIDEO A TRAFICULUI

PROIECT DE DIPLOMĂ

Autor: **Ovidiu Vasile Claudiu PLEŞ**

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

2024

MINISTERUL EDUCAȚIEI

UNIVERSITATEA TEHNICĂ
DIN CLUJ-NAPOCA

FACULTATEA DE AUTOMATICĂ ȘI CALCULATOARE

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 _____

MINISTERUL EDUCAȚIEI

UNIVERSITATEA TEHNICĂ
DIN CLUJ-NAPOCA

FACULTATEA DE AUTOMATICĂ ŞI CALCULATOARE

Declaraţie pe proprie răspundere privind

autenticitatea proiectului de diplomă

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

Supravegherea video a traficului

elaborată în vederea susţinerii examenului de finalizare a studiilor de licenţă la **Facultatea de Automatică şi Calculatoare**, specializarea **Automatică şi Informatică Aplicată,** din cadrul Universităţii Tehnice din Cluj-Napoca, sesiunea iulie 2024 a anului universitar 2023-2024, declar pe proprie răspundere, că această lucrare este rezultatul propriei activităţi intelectuale, pe baza cercetărilor mele şi pe baza informaţiilor obţinute din surse care au fost citate, în textul lucrării, şi în bibliografie.

Declar, că această lucrare nu conţine porţiuni plagiate, iar sursele bibliografice au fost folosite cu respectarea legislaţiei române şi a convenţiilor internaţionale privind drepturile de autor.

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

În cazul constatării ulterioare a unor declaraţii false, voi suporta sancţiunile administrative, respectiv, *anularea examenului de licenţă*.

Data

Ovidiu Vasile Claudiu PLEŞ

(semnătura)

MINISTERUL EDUCAȚIEI
UNIVERSITATEA TEHNICĂ
DIN CLUJ-NAPOCA
FACULTATEA DE AUTOMATICĂ ȘI CALCULATOARE

SINTEZA

proiectului de diplomă cu titlul:

Supravegherea video a traficului

Autor: **Ovidiu Vasile Claudiu PLEŞ**

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

1. Cerinţele temei: Proiectul în discuţie are ca şi scop atât identificarea, clasificarea şi numărarea vehiculelor, cât şi identificarea şi citirea automată a numerelor de înmatriculare a autovehiculelor vizibile intr-un format de tip video.

2. Soluţii alese: Se foloseşte limbajul de programare Python pentru dezvoltarea algoritmilor necesari şi doua modele de reţele neuronale pentru detecţia imaginilor şi pentru citirea imaginilor.

3. Rezultate obţinute: S-a obţinut un program capabil să identifice, clasifice şi să numere vehiculele şi numerele de înmatriculare ale acestora într-un timp relativ scurt.

4. Testări şi 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. Contribuţii personale: Contribuţiile personale implică pregatirea datelor folosite la antrenarea modelelor de reţea neuronală şi dezvoltarea algoritmilor şi a programului.

6. Surse de documentare: Bibliografia include articole ştiinţifice, cărţi şi surse on-line.

Semnătura autorului _____

Semnătura conducătorului ştiinţific _____

Cuprins

1 Introducere

1.1 Context general

În ultimul secol, apariţia vehiculelor pe drumurile publice a avut o creştere exponenţială. Odată cu această creştere, controlul traficului public, a autovehiculelor şi a şoferilor de autovehicule a fost din ce în ce mai greu de pus în aplicare.

Creşterea numărului de autovehicule utilizate a avut nenumărate beneficii pentru omenire. Aceasta a favorizat dezvoltarea industriei transportului şi a agriculturii, a creat un număr semnificativ de locuri de muncă, a oferit posibilitatea intervenţiilor de urgenţă să se desfăşoare mai repede şi nu în ultimul rând, a oferit cetăţenilor mobilitate pentru nevoi profesionale şi personale.

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

Dat fiind acest context, tehnologia evoluată în ultimii ani poate fi folosită pentru ameliorarea unor efecte negative ale apariţiei autovehiculelor menţionate mai sus. Motivaţia acestei lucrări este îmbunătăţirea siguranţei de pe drumurile publice şi reducerea aglomeraţiei de trafic. Obiectivele lucrării includ recunoaşterea automată a autovehiculelor şi a numerelor de înmatriculare ale acestora, numărarea şi clasificarea autovehiculelor şi stocarea eficientă a datelor obţinute în urma acestor procese astfel încât aceste date să poată fi folosite cu uşurinţă pentru îmbunătăţirea controlului traficului contemporan.

Această lucrare este importantă şi merită o deosebită atenţie încât contribuie în primul rând la siguranţa tuturor cetăţenilor şi la bunăstarea acestora, încurajând conceptul de automatizare a oraşelor.

Am ales această temă din dorinţa de a ajuta la dezvoltarea rutieră locală şi datorită pasiunii faţă de inovare, dezvoltare tehnologică şi optimizare a resurselor.

În continuare se vor prezenta urmatoarele capitole:

- Studiu bibliografic: Conţine o sinteză a tehnicilor existente de detecţie şi urmărire a obiectelor şi alte aspecte utile pentru dezvoltarea unei aplicaţii.
- Analiza, proiectare, implementare: Sunt prezentate contribuţiile personale, detaliile tehnice şi toate informaţiile necesare pentru reproducerea lucrării.
- Concluzii: Conţine o sinteză a rezultatelor obţinute şi o descriere a viitoarelor direcţii posibile de dezvoltare.
- Bibliografie: Conţine sursele de informaţii necesare pentru dezvoltarea lucrării.

1.2 Obiective

Obiectivele principale ale proiectului sunt detecţia, numărarea şi clasificarea vehiculelor în timp real pentru a ajuta la îmbunătăţirea siguranţei generale de pe drumurile publice şi pentru a optimiza fluxul de trafic.

Principalele obiective din punct de vedere al funcţionalităţilor sunt urmatoarele:

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

1.3 Specificaţii

1.3.1 Antrenarea unui model de reţea neuronală pentru identificarea autovehiculelor şi a plăcilor de înmatriculare ale acestora

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

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

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

1.3.3 Clasificarea autovehiculelor în patru categorii: “Maşină mică”, „Camion”, „Camionetă” şi „Autobuz”

Se va realiza clasificarea autovehiculelor în patru categorii diferite: “Maşină mică”, “Camion”, “Camionetă” şi “Autobuz”. Se va folosi modelul de reţea neuronală folosit şi 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 “intersecţie peste reuniune” şi vor analiza şi compara vehiculele din fiecare imagine corespunzatoare 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 execuţia 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 şi se vor lua în considerare autovehiculele ale căror coordonate intersectează această linie.

1.3.6 Numărarea autovehiculelor în funcţie de categoria autovehiculului

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

1.3.7 Numărarea autovehiculelor în funcţie de banda pe care se află pe şosea

Acest tip de numărare necesită o abordare aparte. Pe lângă folosirea algoritmului de urmărire şi a liniei orizontale de intersecţie, se va identifica marcajul rutier şi se vor lua în considerare atât aceste marcaje cât şi 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 reţea neuronală

Pentru această identificare se va folosi al doilea model de reţea neuronală. Din cauza rezoluţiilor imaginilor, pentru fiecare autovehicul se va genera o nouă imagine care va conţine 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 fiecarei plăci de înmatriculare va genera o nouă imagine care va fi folosită de un model gata antrenat, disponibil public, specializat in 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 algoritmii care contribuie la procesarea imaginilor vor fi optimizați astfel încât aceștia să se execute intr-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 raspuns 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 şi modifica informaţiile găsite în imagini în diferite moduri şi pentru a obţine 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 intersecţia 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 şi operator de convoluţie, matrice de convoluţie sau kernel, este o matrice cu dimensiunea egală cu dimensiunea blocurilor alese. Procesul de filtrare constă in convoluţia fiecărui bloc al matricii de pixel cu matricea de convoluţie, fiecare convoluţie 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 frecvenţelor joase şi atenuează frecvenţele înalte. Acest efect aplicat asupra pixelilor dintr-o imagine rezultă în atenuarea zgomotului şi a detaliilor din imagine.

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

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

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

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

2.1.1.2 Filtrul Laplacian

Conform [32], filtrul Laplacian se foloseşte pentru a identifica contrastele şi muchiile imaginilor, scoţând în evidenţă 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 foloseşte pentru detecţia marginilor şi contururilor obiectelor. Se folosesc două matrici de convoluţie, una pentru identificarea orizontală şi una pentru identificarea verticală. Cele doua filtre sunt de forma:

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

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 întai 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 intensităţii pixelului, noua valoare a pixelului va fi 0 (pixel negru), altfel valoarea lui va fi 255 (pixel alb). [10]

2.1.2 Identificarea proprietăţilor imaginilor

Identificarea proprietăţilor sau a caracteristicilor din imagini presupune identificarea unor puncte distinctive din imagine, numite şi puncte cheie, şi descrierea acesor puncte şi a imprejurimilor lor. Acest tip de procesare a imaginilor este foarte folositoare pentru detecţia 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 detecţie a caracteristicilor imaginilor. Caracteristicile identificate de acest algoritm sunt independente faţă de scara şi rotaţia imaginilor şi parţial independente faţă 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 in 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 gradienți, 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 si 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 neuronala 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 urmatorului strat. Fiecare strat al unei rețele are un rol important și determină performanța și capacitatea totală a rețelei. Tipurile de straturi sunt urmatoarele:

- Straturi de intrare: Acestea sunt primele straturi ale rețelei neuronale și au rolul de a transmite informația inițială spre următoarele straturi. In acest tip de straturi nu există funcții de activare. Numărul de neuroni care compun aceste straturi este egal cu numărule 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.

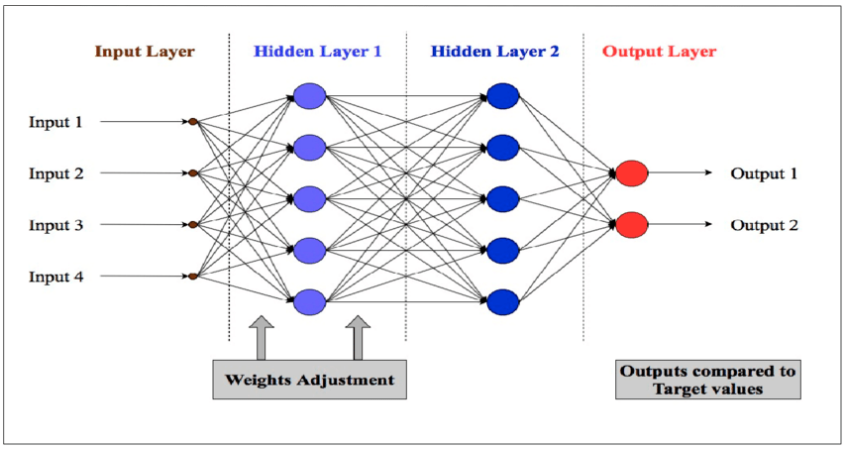


Figura 2.2. Structura de bază a rețelelor neuronale [3]

Alte două componente importante ale structurii modelelor de rețele neuronale sunt greutățile 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 greutăților 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 folosțe pentru recunoașterea imaginilor si 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: Reccurent 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 sa persiste în timp. Un caz de utilizare a acestor rețele este prezicerea unor valori din viitor bazată pe valori anterioare.

2.2.2 Funcţionarea reţelelor neuronale

2.2.2.1 Funcţii de activare

Potrivit [13], funcţiile care procesează semnalul de intrare al unui neuron şi îl transformă în semnal de ieşire se numesc funcţii de activare. Acestea se aleg în funcţie de tipul şi scopul reţelei neuronale, adesea existând reţele cu multiple funcţii de activare. Alegeri populare pentru aceste funcţii sunt următoarele:

- ReLU ("Rectified liniar unit" sau "Unitate liniară rectificată"): Se foloseşte în general pentru straturile ascunse datorită simplităţii şi eficienţei.

$$ReLU(x) = \max(0, x) \tag{2.5}$$

- Sigmoid: Se foloseşte în general pentru reprezentarea probabilităţilor datorită faptului că ieşirea funcţiei este între 1 şi 0.

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

- Tanh (Tangenta hiberbolică): Se foloseşte în general pentru date centrate pe zero, ieşirea funcţiei fiind între -1 şi 1.

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

2.2.2.2 Antrenarea reţelelor neuronale

Conform [14], primul pas în antrenarea reţelelor neuronale este achiziţia datelor. Este necesar un set de date relevant pentru problema care este pusă în discuţie, de exemplu, în cazul detecţiei de obiecte este nevoie de imagini care conţin 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 învăţarea automată a reţelei va fi mai eficientă. De obicei este nevoie de un set relativ mare de date care este împărţit în trei tipuri:

- Date de antrenare: Sunt folosite pentru învăţarea automată a reţelei şi 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 şi pentru a monitoriza procesul de învăţare şi reprezintă aproximativ 15% din totalul de date.
- Date de testare: Aceste date nu sunt folosite deloc în procesul de învăţare automată şi au rolul de a testa şi de a evalua modelul odată ce acesta este gata antrenat. Reprezintă de obicei aproximativ 15% din totalul de date.

Datele colectate sunt adăugate ca şi intrări pentru reţeaua neuronală care le preia, aplică funcţiile de activare pentru fiecare strat, şi ţinând cont de greutăţi şi predispoziţii, generează o ieşire. Pentru a măsura diferenţa dintre ieşirea reţelei şi rezultatul dorit, se foloseşte de obicei eroarea medie pătratică în cazul regresiilor sau pierderea de entropie încrucişată(în engleză: cross-entropy loss) în cazul clasificărilor. Pe baza acestor diferenţe, greutăţile reţelei sunt actualizate. [14]

Un aspect foarte important este antrenarea în buclă. Procesarea datelor de antrenare și actualizarea greutăților rețelei trebuie executată de multiple ori pentru întreg setul de date de antrenare, păstrându-se de fiecare dată greutățile 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 tradiţionale de identificare a obiectelor

Metodele tradiţionale de identificare a obiectelor se referă la tehnici şi algoritmi dezvoltaţi şi utilizaţi înainte de creşterea performanţelor reţelelor neuronale. Aceste metode sunt relevante în contextul în care resursele sistemului de calcul sunt limitate sau procesarea imaginilor şi detecţia obiectelor trebuie sa fie executată intr-un timp cât mai scurt.

2.3.1.1 Cascada Haar

Potrivit [15], identificarea obiectelor folosind cascade Haar funcţionează prin evaluarea unor proprietăţi numite proprietăţi Haar. Aceste proprietăţi sunt identificate prin însumarea intensităţii pixelilor dintr-o regiune a imaginii şi calcularea diferenţei dintre sume. Astfel se obţin proprietăţi care descriu linii sau margini în interiorul imaginii.

Eficienţa acestui algoritm constă în procesul de antrenare, proces similar cu cel folosit în antrenarea reţelelor neuronale. Se foloseşte o tehnică numită Adaboost care combină mai multe proprietăţi Haar pentru a crea o singură proprietate Haar de calitate superioară. În timpul procesului de învăţare, algoritmul Adaboost selectează cele mai distinctive proprietăţi Haar şi le atribuie valori în funcţie de performanţa diferenţierii dintre exemple pozitive şi exemple negative (imagini care conţin obiectul de interes şi imagini care nu conţin obiectul de interes).

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

2.3.1.2 Histogramă de gradienţi orientaţi

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

Imaginea este împărţită în mai multe regiuni numite celule. Pentru fiecare pixel dintr-o celulă este calculată orientarea şi magnitudinea gradientului. Pentru fiecare celulă este construită o histogramă a orientărilor gradientului prin acumularea orientărilor gradientului ale tuturor pixelilor din celulă. Această histogramă reprezintă distribuţia 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 informaţiile despre forma şi proprietăţile obiectelor.

2.3.2 Reţele neuronale convoluţionale pentru detecţia obiectelor

Reţelele neuronale convoluţionale sunt un tip de modele de reţele neuronale dezvoltate în special pentru procesarea imaginilor şi detecţia de obiecte. Aceste tipuri de reţele neuronale oferă performanţe superioare atunci când vine vorba de detecţia obiectelor, fiind capabile să recunoască caracteristici complexe în condiţii variabile.

Un aspect de luat în considerare este faptul că această abordare pentru detecţia 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 acestori 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.

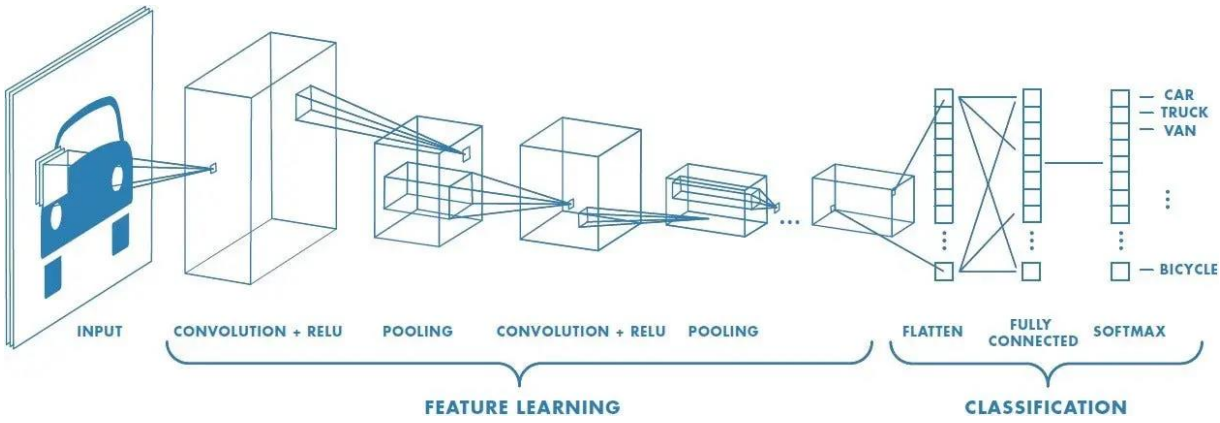


Figura 2.3. Arhitectura unei rețele neuronale convoluționale [5]

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 impărțite 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 şi este intrarea pentru straturile complet conectate care urmează. Aceste straturi sunt rețele neuronale tradiționale şi sunt folosite pentru a genera ieşirea 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 ieşirea şi raspunsul 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 şi reprezentativă a acestui algoritm este proprietatea de detecție intr-un timp foarte scurt, folosind o metodă de divizare a imaginii în mai multe secțiuni, facându-se predicții pentru fiecare secțiune în acelaşi timp. Această proprietate face ca algoritmul sa fie mult mai rapid faţă 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 şi înalțimea dreptunghiului. Pe lângă aceste informații, se regaseşte şi o valoare care indică scorul detecției şi reprezintă probabilitatea ca un obiect să fie înauntrul 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 şi elimină celelalte detecții care se suprapun.

În ultimii ani, algoritmul YOLO a evoluat în diferite versiuni, acurateţea şi rapiditatea fiind îmbunătățite. Fiecare versiune a suferit modificări în structură şi metode de implementare pentru a creşte performanţele.

2.3.3 Identificarea textului

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

Pentru a obține rezultatele dorite, este necesară de obicei o preprocesare a imaginii. Se folosesc operații precum redimensionarea, binarizarea şi aplicarea filtrelor pentru reducerea de zgomote. Regiunea din imagine care conține text este separată de restul imaginii şi împărțită în linii, cuvinte şi 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 şi 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 Euclidiana: 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 obieci 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 şi raportul de intersecţie peste uniune pentru a face asocierile între obiectele din imagini.
- Utilizarea reţelelor neuronale recurente: Se utilizează acest tip de reţele neuronale pentru a prezice traiectoria obiectelor.

2.4.1.1 Kanade-Lucas-Tomasi

Conform [19], acest algoritm funcţionează prin identificarea şi 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 variaţie mare de intensitate a pixelilor (colţurile) şi 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 şi deplasarea obiectelor este relativ mică. Aceste trăsături denotă faptul că algoritmul este fezabil pentru medii stabile şi 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 şi regiunea din jurul acestui punct pe baza orientărilor gradienţilor pixelilor. Acest vector de descriere este invariant la luminozitate şi scară.

Pentru a putea urmări obiectele în imagini consecutive, se compară caracteristicile obiectelor din imagini folosind distanţa Euclidiană a vectorilor de descriere. Pentru o funcţionare 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 poziţiilor viitoare ale obiectelor din imagini bazat pe stările anterioare ale obiectelor şi 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 detecţii ale obiectului urmărit şi conţine de obicei informaţii legate de poziţia şi viteza obiectului.

Pentru a face o predicţie a stării obiectului din viitor, se foloseşte starea curentă a obiectului şi un model de mişcare. Valorile corespunzătoare poziţiei obiectului se actualizează în funcţie de viteza obiectului iar viteza obiectului se recalculează în funcţie de acceleraţie atunci când viteza nu este constantă.

Vectorul de stare este o aproximare a poziţiei obiectului. Pentru a scoate în evidenţă natura lipsită de acurateţe a aproximării, avem nevoie de o matrice de covarianţă a stării. Această matrice reprezintă incertitudinea asociată cu estimarea vectorului de stare şi 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 şi raportul de intersecţie peste uniune pentru a asocia obiectele din

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

Primul pas în asocierea obiectelor a acestui algoritm este calcularea predicţiei poziţiei obiectului din imaginea curentă folosind filtrarea Kalman. Această poziţie reprezintă poziţia 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 intersecţie 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 poziţiilor celor două obiecte. Dacă valoarea raportului este peste un anumit prag, asocierea dintre cele două obiecte este făcută.

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

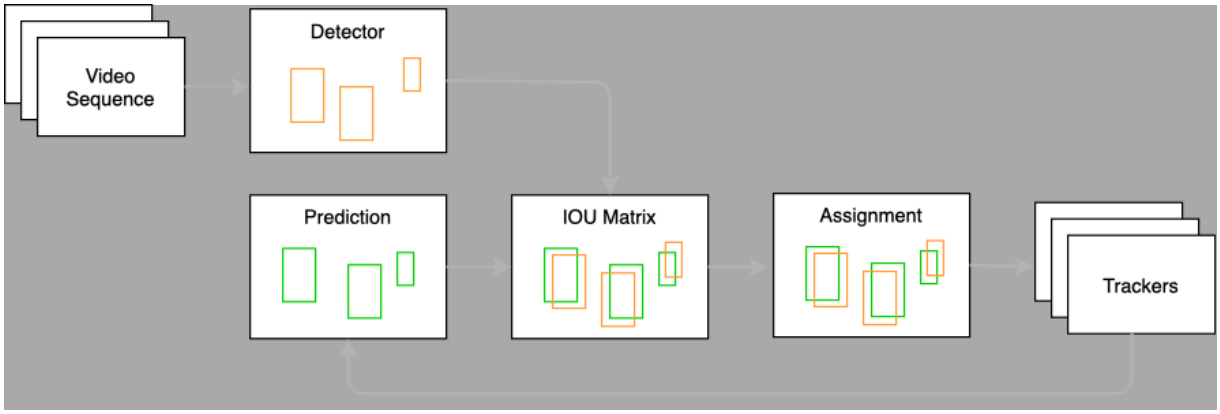


Figura 2.4. Structura algoritmului SORT [6]

2.4.2 Numărarea obiectelor

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

De obicei în contextul procesării videoclipurilor şi mai ales în contextul actual de supraveghere a traficului, numărarea obiectelor se poate face în mai multe feluri şi 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 performanţe scăzute în anumite zone ale imaginii.
- Numărarea în funcţie de zone: Obiectele sunt impărţite în mai multe zone de interes ale imaginii şi se folosesc tehnici bazate pe geometrie pentru a număra obiectele. Această abordare este utilă în contextul actual în care se doreşte numărarea autovehiculelor în funcţie de banda rutieră pe care se află.

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

2.5 Interfeţe de utilizator

Interfeţele pentru utilizator sunt un ansamblu de componente care ajută la comunicarea şi interacţiunea oamenilor cu calculatoarele. Acestea pot să includă ecrane, tastaturi, mouse-uri sau pot să fie ferestre afişate pe ecran care conţin butoane digitale, texte, animaţii şi sunete şi care ajută la interacţiunea şi controlul unei aplicaţii sau a unui site web. [7]

Pentru a face cât mai utile şi mai uşor de utilizat aceste interfeţe, este nevoie de un aspect vizual sugestiv care să permită utilizatorilor să folosească intuitiv aplicaţia şi să gestioneze uşor informaţiile. Acest lucru presupune că interfaţa nu trebuie sa fie prea complicată pentru a nu îngreuna utilizarea acesteia dar în acelaşi timp trebuie să conţină toate elementele necesare pentru un control bun al aplicaţiei.

O interfaţă de utilizator poate să conţină diferite elemente structurate în diferite moduri pentru a obţine diferite funcţionalităţi şi aspecte vizuale. Printre cele mai utilizate elemente sunt:

- Elemente informaţionale: Utilizatorul primeşte informaţii despre aplicaţia cu care interacţionează cu ajutorul componentelor de tip text, notificări şi alte componente vizuale sau auditive sugestive.
- Elemente de intrare: Utilizatorul are posibilitatea să introducă informaţii în aplicaţie modificând datele programului cu care interacţionează prin diferite metode. Se folosesc câmpuri de text, liste predefinite sau butoane.
- Elemente de navigaţie: Informaţiile se organizează şi se aşează în diferite pagini sau zone ale aplicaţiei. Se folosesc componente precum câmpuri de cautare, meniuri sau conceptul de paginare pentru a ajuta utilizatorul să găsească aceste informaţii şi să interacţioneze cu ele.

Conform [7], interfaţa de utilizator este o componentă esenţială a oricărei aplicaţii software, existând diferite tipuri de interfeţe pentru diferite scopuri. Cele mai populare tipuri de interfeţe de utilizator sunt următoarele:

- Interfeţe grafice pentru utilizator: Aceste interfeţe sunt compuse din elemente grafice sugestive cum ar fi butoanele, ferestrele sau meniurile şi sunt de obicei uşor şi intuitiv de folosit.
- Linii de comandă: Sunt interfeţe minimaliste care expun serviciile sistemului de operare. Funcţionează pe bază de comenzi text care sunt introduse în linia de comandă.
- Interfeţe în limbaj natural: Acest tip de interfeţe este creat pentru a înţelege limbajul natural al oamenilor. Pe baza acestui limbaj, interfaţa are scopul de a da comenzi sistemului sau aplicaţiei din care face parte şi de a răspunde utilizatorului într-un limbaj natural.

2.6 Posibile soluții și abordarea aleasă

Pentru a aborda obiectivele lucrării există o multitudine de soluţii pentru fiecare etapă a implementării. În alegerea soluţiilor s-a ţinut cont de următoarele considerente:

- Procesarea imaginilor trebuie să fie cât mai rapidă pentru a putea fi utilizată în contextul de timp real, în acelaşi timp, acurateţea rezultatelor în urma procesării nu trebuie să scadă drastic din acest considerent.
- Urmărirea şi 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.
- Interfaţa de utilizator trebuie să fie una grafică şi minimalistă care să permită controlul asupra procesării imaginilor şi extracţiilor de date.

Astfel, ţinând cont de considerentele listate mai sus, pentru procesarea imaginilor se va folosi algoritmul YOLO în defavoarea metodelor tradiţionale de detecţie a obiectelor iar pentru detecţia textului se va folosi un model de reţea neuronală convolutivă. Se vor folosi tehnici de preprocesare a imaginilor şi filtre pentru a îmbunătăţi 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.

Interfaţa de utilizator va fi una simplă care să permită utilizatorului să controleze procesarea videoclipurilor şi a transmisiilor în direct a videoclipurilor. Vor fi disponibile diferite opţiuni de procesare a videoclipurilor. Nu este nevoie de conceptul de creare a unui cont de utilizator iar accesul la aplicaţie în sine va fi nerestricţionat, considerându-se o restricţie pentru folosirea aplicaţiei 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 catre 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 si 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 fişiere. Acesta este folosit de obicei pentru a gestiona schimbările făcute în fişierele care conţin cod şi pentru a ajuta la dezvoltarea codului în echipă.

Funcţionalităţile acestui sistem se bazează pe depozite. Aceste depozite reprezintă folderele în care se află fişierele 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 fişierelor. Acest lucru asigură posibilitatea de dezvoltare în paralel a proiectelor de programare şi nu numai. Ramurile unui depozit pot să fie concatenate pentru a aduce în aceeaşi versiune modificările de pe ambele ramuri.

3.1.4 Platforma GitHub

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

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

Pe lângă funcţionalităţile de distribuire a fişierelor, 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 facut cererea. Dacă codul este aprobat, modificările de pe o ramură se concatenează cu conţinutul altei ramuri.

3.1.5 Modelul de detecţie YOLOv8

YOLO este un model de reţea neuronală dezvoltat de compania Ultralytics, folosit pentru detecţia de obiecte din imagini şi în special pentru detecţia în timp real datorită performanţelor ridicate. [26]

Modelul folosit în acest proiect este YOLOv8n, un model din versiunea a opta, construit pentru a fi rapid şi în situaţiile în care resursele sunt limitate. Acesta este antrenat pe un set divers de imagini pentru a introduce date fundamentale despre forma şi 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 librarie 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 librarie 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 librarie 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 librărie 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 librăriilor 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 ușoara.

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 parametriilor 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 identificatorului intern.

Modificarea parametriilor de procesare include modificarea scorului modelelor de detecţie, modificarea parametrului de urmărire şi modificarea numărului de încercări de citire a numerelor de înmatriculare.

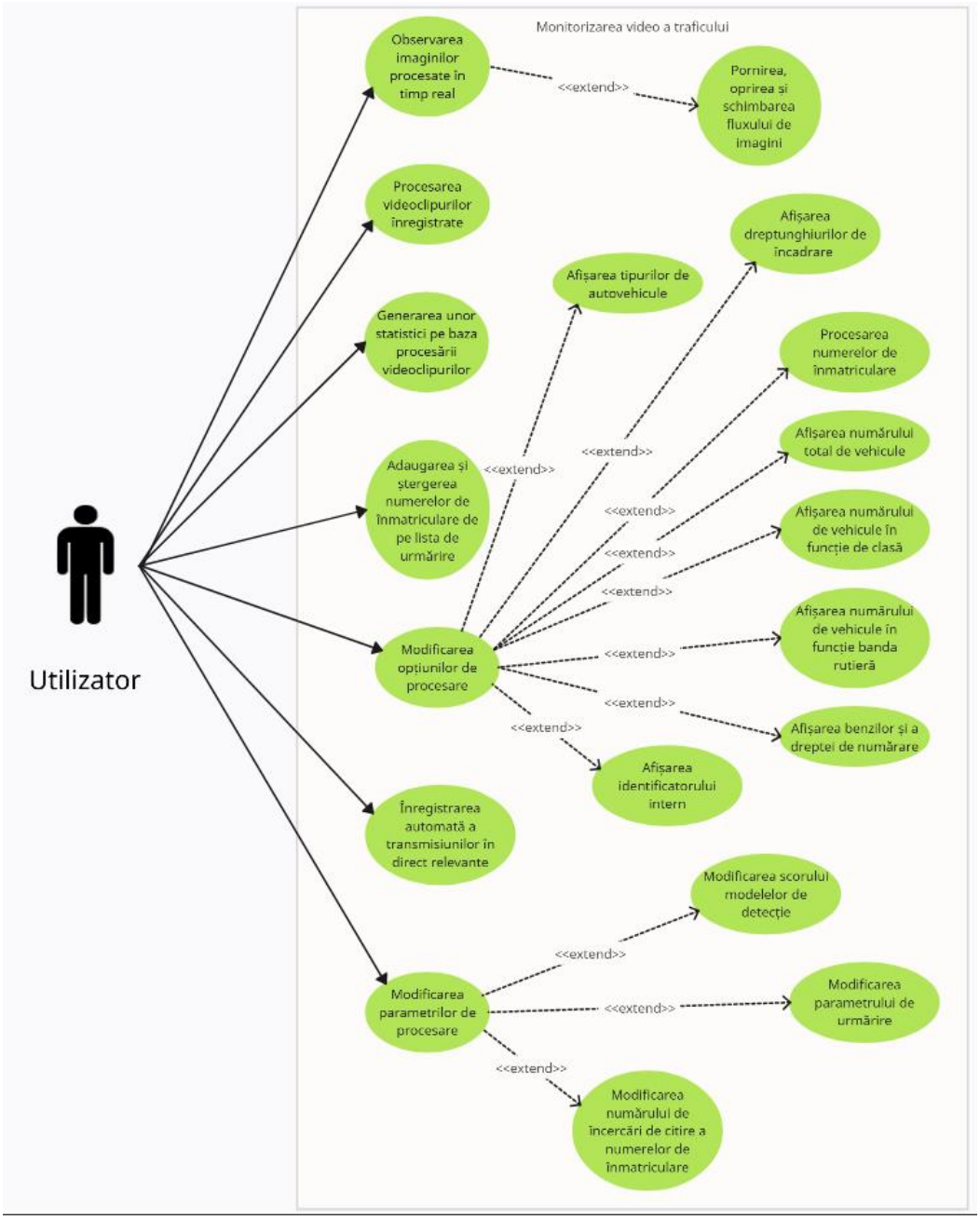


Figura 3.1. Diagrama cazurilor de utilizare

3.2.2 Structura aplicaţiei

Structura aplicaţiei este una relativ simplă, folosindu-se doar limbajul de programare Python atât pentru logica procesării imaginilor cât şi pentru implementarea interfeţei grafice de utilizator. Fişierele Python responsabile pentru afişarea elementelor grafice sunt generate folosind aplicaţia Qt Designer şi sunt introduse în acelaşi folder în care se regăsesc şi fişierele responsabile pentru logica procesării imaginilor. Pentru o

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

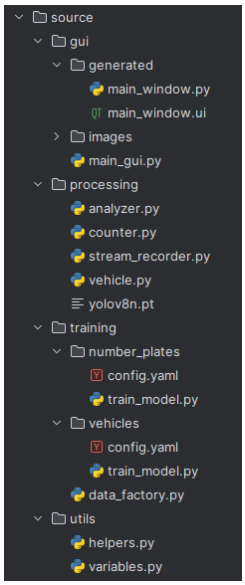


Figura 3.2. Structura fişierelor principale ale aplicaţiei

În Figura 3.2 este prezentată structura fişierelor şi a folderelor care conţin codul sursă principal al aplicaţiei, inclusiv codul folosit pentru antrenarea modelelor YOLO pentru detecţia obiectelor. Aceste fişiere şi 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 fişierul Python principal "main.py" care este responsabil pentru pornirea aplicaţiei, un fişier responsabil pentru generarea unui fişier executabil corespunzător aplicaţiei, folderul "dev_setup" care conţine informaţii despre configurarea mediului de dezvoltare, folderul "documentation" care conţine documentaţia proiectului, folderul "runs" care conţine modelele YOLO antrenate, folderul "dist" care conţine fişierul executabil corespunzător aplicaţiei, fişierul ".venv" care este responsabil pentru funcţionarea mediului virtual Python, fişierul "recordings" în care se află înregistrări ale transmisiilor în direct şi alte fişiere şi foldere care au legătură cu sistemul de control Git, generearea fişierului executabil sau funcţionarea generală a aplicaţiei.

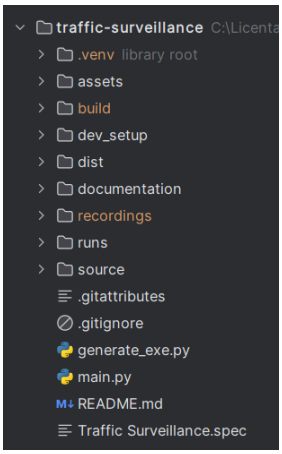


Figura 3.3. Structura proiectului Python

Folderul "source" conţine toate clasele Python folosite şi majoritatea codului Python folosit în dezvoltarea aplicaţiei. În figura următoare este prezentată diagrama claselor care se află în acest folder. Se observă în această diagrama fiecare atribut şi fiecare funcţie a fiecărei clase şi legăturile între clase.

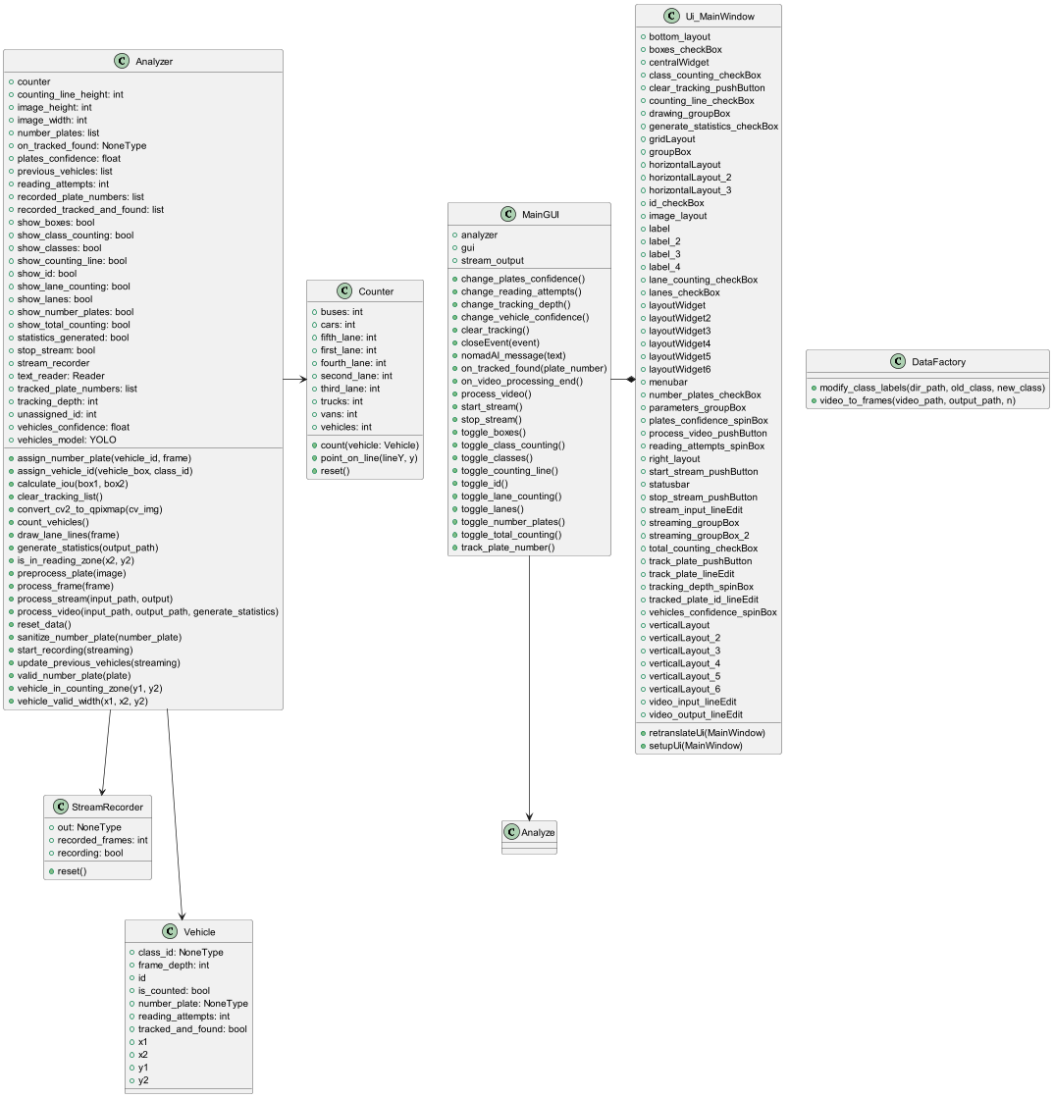


Figura 3.4. Diagrama claselor

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 şi resurselor care vor fi folosite. Acestea sunt următoarele:

- Pachetul Python: Include un interpretor pentru cod Python, câteva pachete standard, un manager de pachete şi altele. Acest pachet se poate descărca de pe site-ul web oficial al organizaţiei 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 şi crearea unui proiect nou. În acest proiect este necesară crearea unui mediu virtual pentru Python. Acest mediu va fi responsabil pentru stocarea şi 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 înlocuieşte cu numele folderului în care se va afla mediul virtual.

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

Ultimul pas în iniţializarea 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 opţiunea "Share project on GitHub". Acest lucru asigură o copie online a proiectului şi posibilitatea de a lucra organizat şi de a stoca online toate versiunile proiectului.

Pentru o organizare mai bună a proiectului, se iniţializează fişierele ".gitignore" şi ".gitattributes". Acestea sunt fişiere de configurare a sistemului de control Git şi sunt create înauntrul folderului de tip depozit Git. Fişierul ".gitignore" specifică toate fişierele şi folderele din proiect care nu trebuie urmărite de Git. În acest caz, în acest fişier 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 şi pachetele folosite nu sunt urmărite de Git, în schimb, acestea sunt specificate într-un fişier text.
- "__pycache__": Aceste foldere sunt create local de Python pentru a stoca fişiere cu cod interpretat pentru o rulare ulterioară a codului mai uşoară.
- "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 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 dupa 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 doua 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 dreptughi 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.

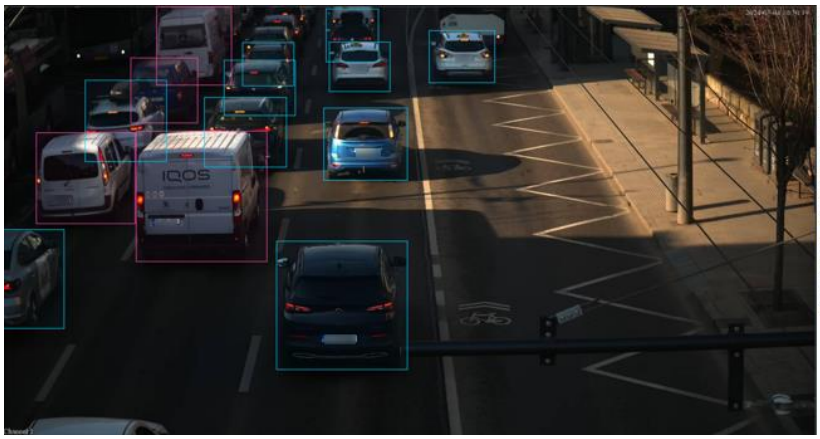


Figura 3.5. Adnotarea unei imagini

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

3.3.2.3 Antrenarea modelului YOLO

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

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

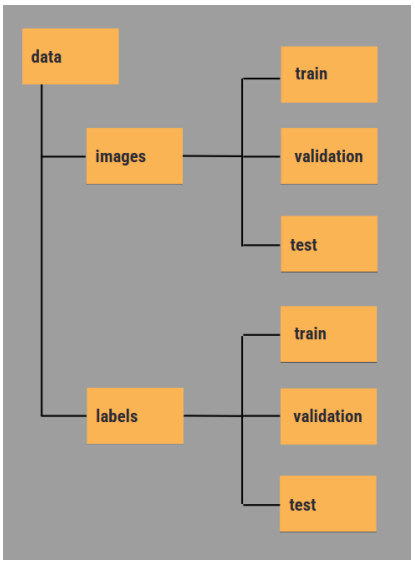


Figura 3.6. Structura setului de date

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

Fişierul de configurare fiind creat, se poate apela funcţia "train" a instanţei modelului YOLO. Pe lângă fişierul de configurare, această funcţie mai primeşte ca parametru şi un număr natural care corespunde cu numărul de repetări pe care modelul îl execută în procesul de învăţare automată pentru întregul set de date. Funcţia de învăţare 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, fişierul de configurare.

```python
from ultralytics import YOLO

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

```yaml
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 şi date diferite până când modelul are performanţe mulţumitoare în detecţia 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 funcţiei 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 esenţial în dezvoltarea şi funcţionarea aplicaţiei. Pentru a putea implementa acest algoritm avem în primul rând nevoie de detecţia de autovehicule. Modelul YOLO returnează multiple date atunci când face o predicţie 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 predicţiei imaginii. Acest atribut se transformă într-o listă şi fiecare element al listei reprezintă un obiect şi conţine următoarele date:

- x_1 şi y_1: Reprezintă coordonatele punctului cel mai apropiat de originea imaginii a dreptunghiului de încadrare (colţul din stânga sus).
- x_2 şi y_2: Reprezintă coordonatele punctului cel mai îndepărtat de originea imaginii a dreptunghiului de încadrare (colţul 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 "assing_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 daca 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.

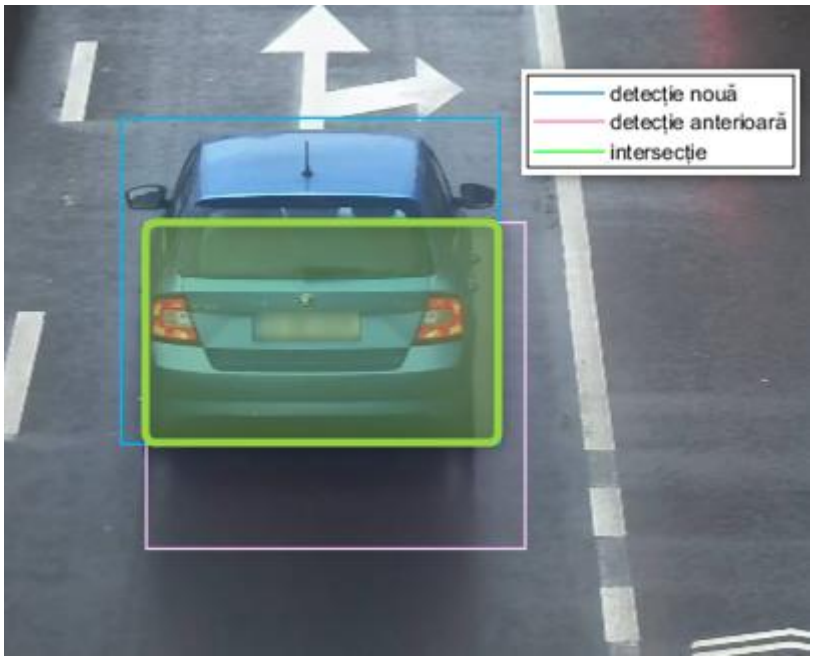


Figura 3.8. Detecţiile unui autovehicul în imagini consecutive

În Figura 3.4 se observă dreptunghiurile de încadrare ale aceluiaşi autovehicul pentru imagini consecutive dintr-un videoclip şi intersecţia acestor dreptunghiuri folosită la calcularea raportului de intersecţie peste uniune.

3.3.3.2 Gestionarea lipsei de detecţii

Deoarece există imperfecţiuni în identificarea autovehiculelor din fiecare imagine care provin din capacitatea limitată a modelului YOLO şi, în special din capacitatea camerei de a transmite imagini, algoritmul de urmărire trebuie îmbunătăţit ţinând cont de aceste aspecte. Camera video de supraveghere transmite adeseori imagini corupte care nu pot fi interpretate iar modelul YOLO poate să proceseze o predicţie greşită, 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 detecţiilor cu o singură imagine anterioară este insuficientă. Dacă pentru o detecţie se caută o corelaţie în imaginea anterioară şi nu se găseşte pentru că imaginea este coruptă sau nu există detecţii, detecţia se consideră una nouă, cu un număr de identificare nou, chiar dacă în imaginea anterioară există un autovehicul care să corespundă cu această nouă detecţie.

Pentru a rezolva această problemă, se consideră compararea detecţiilor cu detecţii din mai multe imagini anterioare. Numărul de imagini anterioare poate să fie ales de utilizator şi 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 detecţie 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 detecţia este făcută pe imaginea curentă şi fiind incrementat la

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

3.3.3.3 Optimizarea algoritmului

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

Pentru a îmbunătăţii algoritmul şi pentru a reduce numărul de operaţii de calcul a raportului de intersecţie peste uniune, dreptunghiurile de încadrare a detecţiilor noi vor fi comparate doar cu dreptunghiurile din apropiere lor. Astfel, se va calcula distanţa între coordonatele celor două dreptunghiuri de încadrare:

- Distanţa pe axa X a imaginii se calculează în funcţie de punctul cel mai apropiat de originea imaginii al celor doua dreptunghiuri (colţul stânga sus). Această distanţă este validă dacă este mai mică decât 300 de pixeli (aproximativ 1/14 din totalul de pixeli de pe axa X).
- Distanţa pe axa Y este calculată în funcţie de acelaşi punct al celor două dreptunghiuri şi 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 distanţe dintre cele două dreptunghiuri de încadrare sunt valide, se poate trece mai departe la calculul raportului de intersecţie peste uniune, astfel se evită calculul inutil al rapoartelor.

Distanţa validă pe axa Y este semnificativ mai mare decât distanţa 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 şi a camioanelor şi datorită frecvenţei reduse de apariţie a acestora, această optimizare nu se aplică pentru aceste două clase de autovehicule.

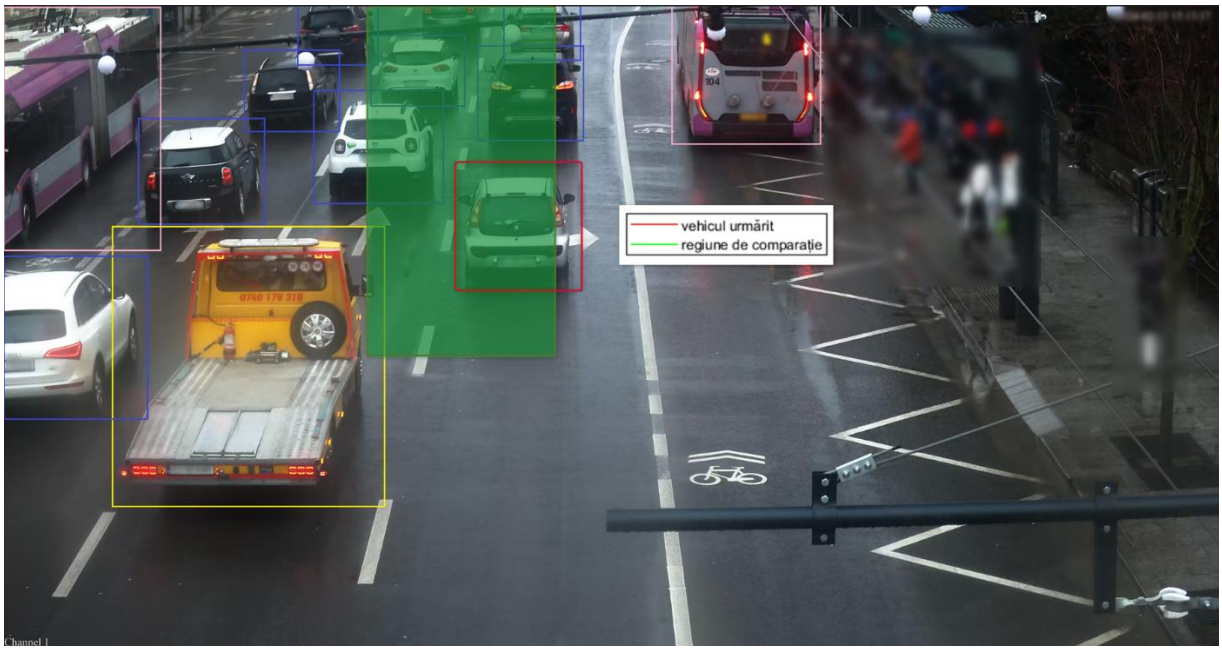


Figura 3.9. Regiunea de comparație a unui autovehicul

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 înauntrul 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 ține 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 dreptei 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 dreptei de numărare corespunzătoare axei Y.

Astfel, ţinând cont de condiţiile de mai sus, dreptunghiul de încadrare a autovehiculului intersectează dreapta de numărare. În continuare se poate apela funcţia "count" a instanţei de tip "Counter" care incrementează o variabilă generală şi 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 şi intersecţia ei cu autovehiculele.

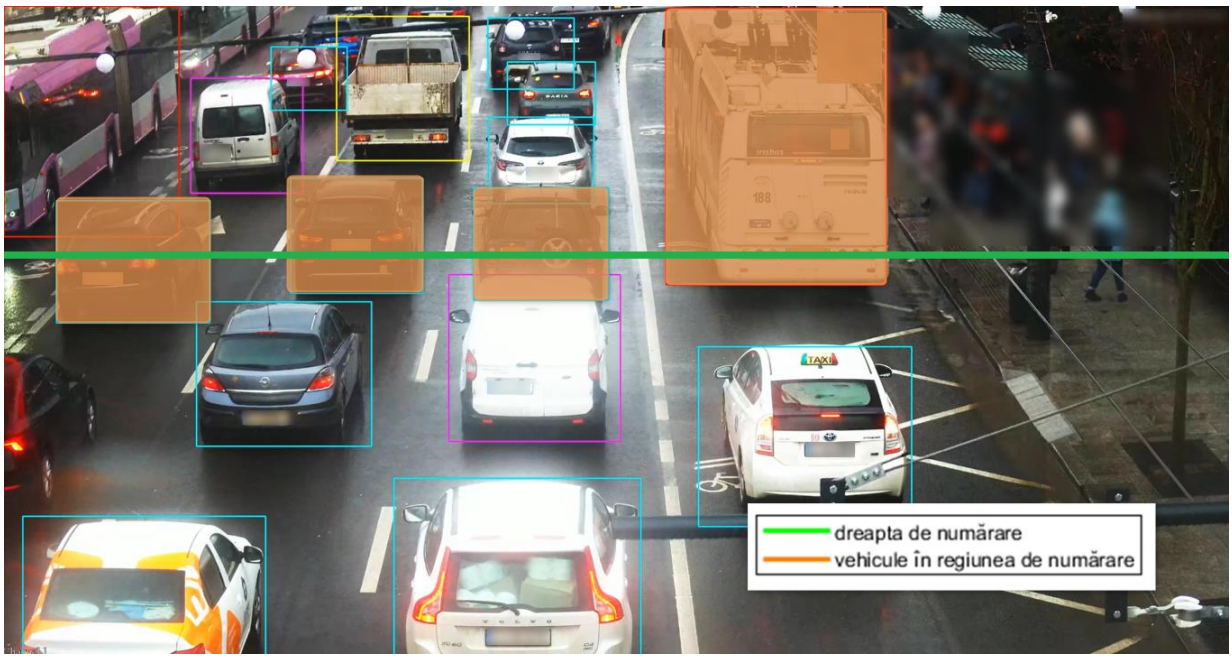


Figura 3.10. Dreapta de numărare şi intersecţia ei cu autovehiculele

3.3.4.2 Numărarea în funcţie de benzile rutiere

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

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

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

- Autovehiculele se consideră pe prima bandă (cea mai din dreapta) dacă colţul din stânga sus a dreptunghiului autovehiculului este mai în dreapta faţă de a doua linie de marcaj rutier.
- Un autovehicul se află pe a doua bandă dacă colţul din stânga sus a dreptunghiului de încadrare se află în apropierea celei de-a treia linii de marcaj rutier pe axa X (mai puţin de 300 de pixeli diferenţă) şi colţul 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 fată 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 şi vecinătăţile liniilor corespunzătoare celei de-a doua benzi pe axa X. Autovehiculul a cărui dreptunghi de încadrare întersectează dreapta de numărare se află în zona de numărare, astfel, funcţia "count" este apelată pentru acest vehicul şi se execută numărarea simplă şi în funcţie de clasă. Se observă că pentru numărarea în funcţie de benzile rutiere, autovehiculul îndeplineşte condiţiile pentru a doua bandă de la stânga la dreapta, colţul din stânga sus fiind în vecinătatea celei de-a treia linii iar colţul din dreapta jos fiind în vecinătatea celei de-a doua linii de marcaj rutier pe axa X.

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

$$m = \frac{y_2 - y_1}{x_1 - x_1} \tag{3.1}$$

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

$$x = \frac{y - y_1}{m} + x_1 \tag{3.2}$$

După calculul coordonatei corespunzătoare axei X a punctului de pe linie care are valoarea coordonatei corespunzătoare axei Y egală cu cea a colţului, se poate face diferenţa între aceste două valori. Această diferenţă se compară cu diferenţa de prag specificată pentru fiecare linie şi fiecare bandă, şi astfel, se pot verifica toate condiţiile necesare pentru a determina poziţia unui autovehicul în funcţie de benzile rutiere.

3.3.5 Detecţia plăcilor de înmatriculare

3.3.5.1 Achiziţia datelor şi antrenarea modelului

Dată fiind rezoluţia 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 detecţia acestora, trebuie prelucrat un nou set de date şi antrenat un nou model de reţea neuronală pentru detecţie.

Pentru a crea setul de date, se decupează din imaginile iniţiale imagini care conţin câte un singur autovehicul. Astfel, raportul între rezoluţia imaginii şi dimensiunea plăcilor de înmatriculare se schimbă, permiţând modelului să recunoască şi să înveţe caracteristici ale plăcilor de înmatriculare. În acest set de date trebuie să se asigure că plăcile de înmatriculare ale autovehiculelor sunt lizibile, pe lângă condiţiile general

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

Se foloseşte o instanţă a modelului YOLOv8n, precum cea folosită pentru detecţia autovehiculelor, deci procesele de adnotare a datelor, organizare a datelor, antrenare şi configurare a modelului sunt similare cu cele descrise la capitolele 3.3.2.2 şi 3.3.2.3.

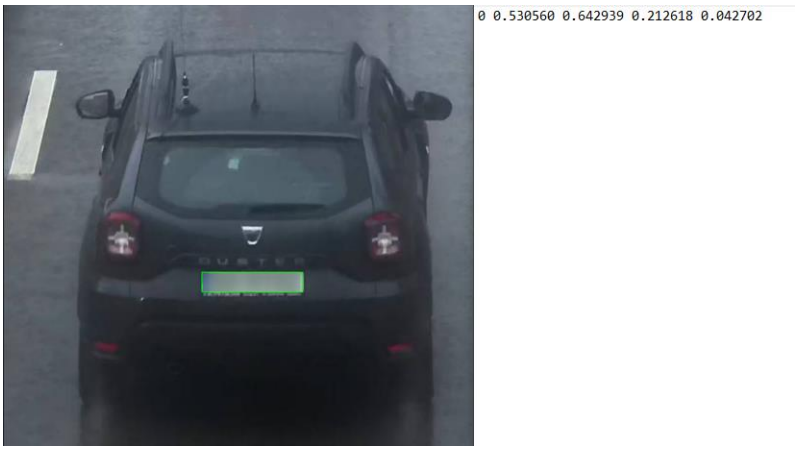


Figura 3.12. Adnotarea datelor pentru modelul de detecţie a plăcilor de înmatriculare

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

3.3.5.2 Recunoaşterea numerelor de înmatriculare

Dat fiind modelul descris la subcapitolul anterior, pentru a detecta plăcile de înmatriculare avem mai întai nevoie de detecţia 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 şi a recunoaşte 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 predicţia, modelul returnează anumite date, printre care şi coordonatele plăcii de înmatriculare identificate. Folosind aceste coordonate, imaginea se decupează din nou, astfel rezultând o imagine care conţine numai placa de înmatriculare şi conţinutul ei.

Identificarea caracterelor necesită în primul rând o preprocesare a imaginii care conţine numărul de înmatriculare. Această preprocesare ajută la acurateţea şi viteza de citire. Dimensiunile imaginii sunt modificate pentru a ajuta la îmbunătăţirea detaliilor şi se foloseşte 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 lățimea imaginii, păstrându-se înălțimea, pentru a înlătura partea plăcii de înmatriculare care nu conține caractere.

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

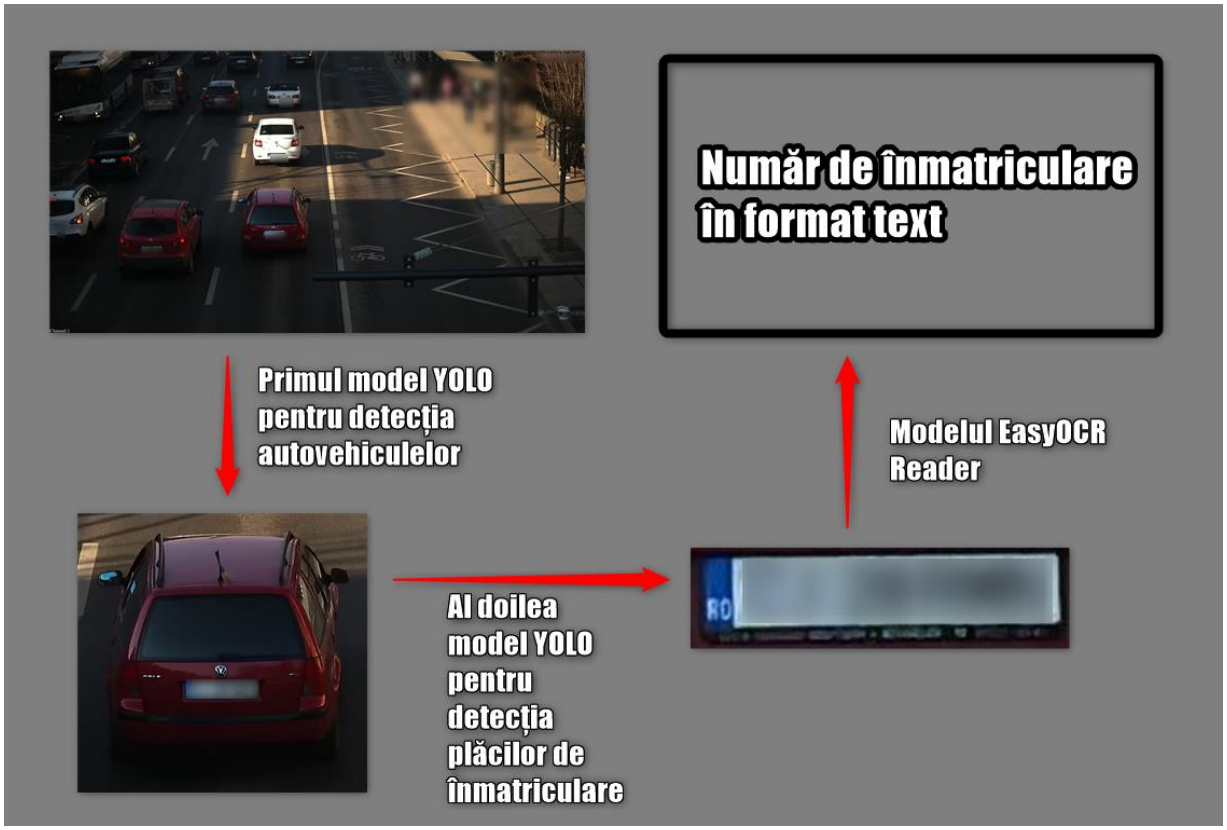


Figura 3.13. Structura procesului de recunoaștere a numerelor de înmatriculare

3.3.5.3 Optimizarea algoritmului

Pentru a crește acuratețea și viteza identificării numerelor de înmatriculare se implementează o serie de îmbunătățiri pentru algoritmul de detecție, ținându-se cont de formatul predominant de numere de înmatriculare, rezoluția și poziționarea camerei de supraveghere video, și nevoia de procesare relativ rapidă a imaginilor astfel încât acestea să poată să fie utilizate într-un context de timp real. Îmbunătățirile aplicate algoritmului sunt următoarele:

- Se transformă toate literele identificate în litere mari pentru a nu produce confuzii și pentru a păstra consistența. Literele pot fi interpretate de către modelul EasyOCR în diferite dimensiuni.
- Condiționarea regiunii de identificare a numerelor de înmatriculare și recunoaștere 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 și a cincea bandă de circulație rutieră. Excluderea tentativelor de identificare în această zonă sporește 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 şi din punct de vedere al acurateţii.
- Se verifică scorul obţinut în urma predicţiei făcute de către modelul EasyOCR şi se exclud rezultatele care au acest scor mai mic decât un scor de prag şi rezultatele a căror text conţine mai puţin de şase caractere.
- Pentru formatul de număr de înmatriculare existent în Romania, se aplică o funcţie de îmbunătăţire care transformă literele în cifre şi invers în funcţie de acest format. Numerele de înmatriculare care nu sunt în acest format sunt ignorate de această funcţie.
- Încercarea de citire a numerelor de înmatriculare, în cazul în care aceasta eşuează sau rezultatul nu este valid, este limitată de un număr implicit sau setat de utilizator. Această limitare ajută semnificativ la creşterea vitezei de procesare a imaginilor.

Pentru a identifica formatul de numere de înmatriculare existent în Romania se verifică în primul rând dacă primul caracter este o literă şi dacă numărul caracterelor care compun textul este egal cu şase sau şapte. În continuare, se verifică fiecare caracter şi se modifică dacă în poziţia în care se află, caracterul este o cifră şi se aşteaptă 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 şi nu se returnează un rezultat valid, placa de înmatriculare a autovehiculului este considerată ilizibilă şi structura de date corelată autovehiculului se schimbă astfel încât să indice faptul ca algoritmul de citire nu trebuie aplicat acestuia. În cazul în care nu se aplică această restricţie, algoritmul de citire a plăcii de înmatriculare se va aplica pentru fiecare imagine în care autovehiculul este existent, fapt care ar creşte semnificativ timpul de procesare al imaginilor.

```
dict_letter_to_figure = {
    '0': '0',
    'I': '1',
    'Z': '2',
    'J': '3',
    'A': '4',
    'S': '5',
    'G': '4',
    'T': '7',
    'B': '8',
    'U': '0',
    'R': '2',
}
```

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

Figura 3.14. Dicţionarele 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.

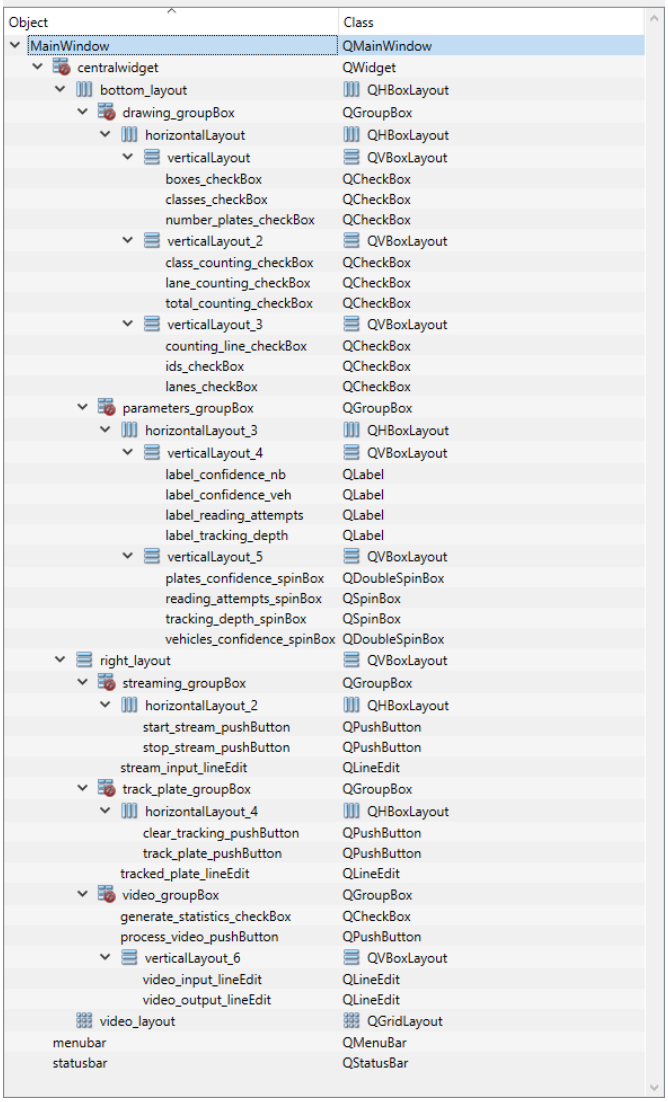


Figura 3.15. Structura și ierarhia elementelor grafice ale ferestrei principale

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 acesteia 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.

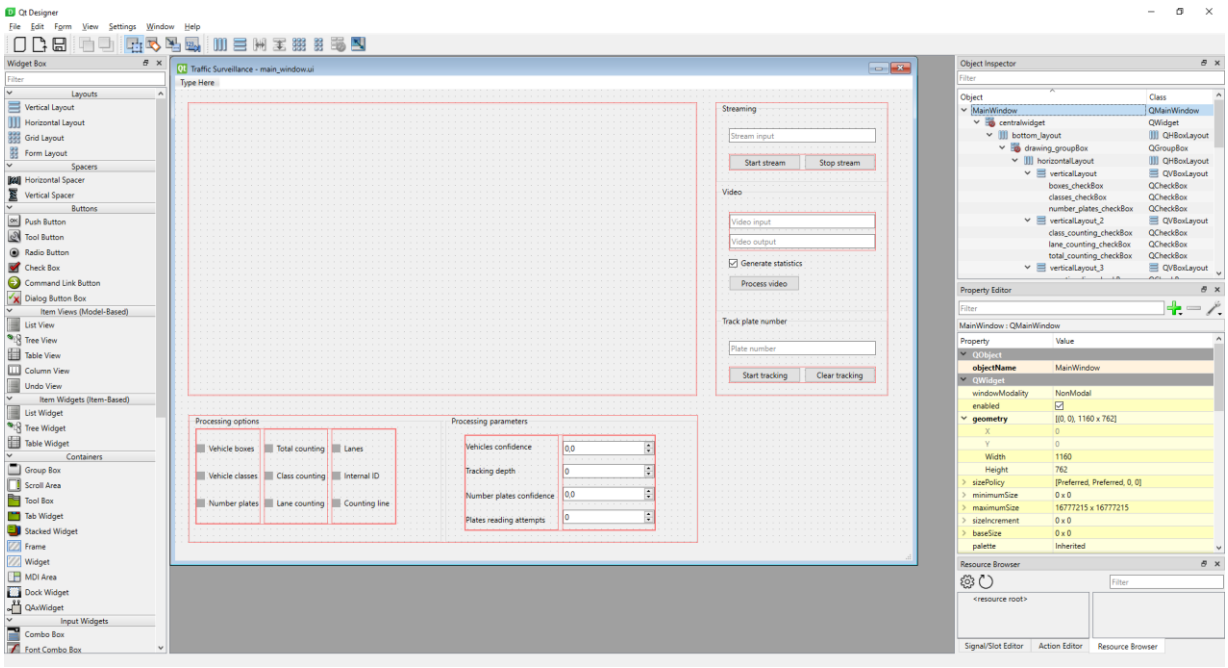


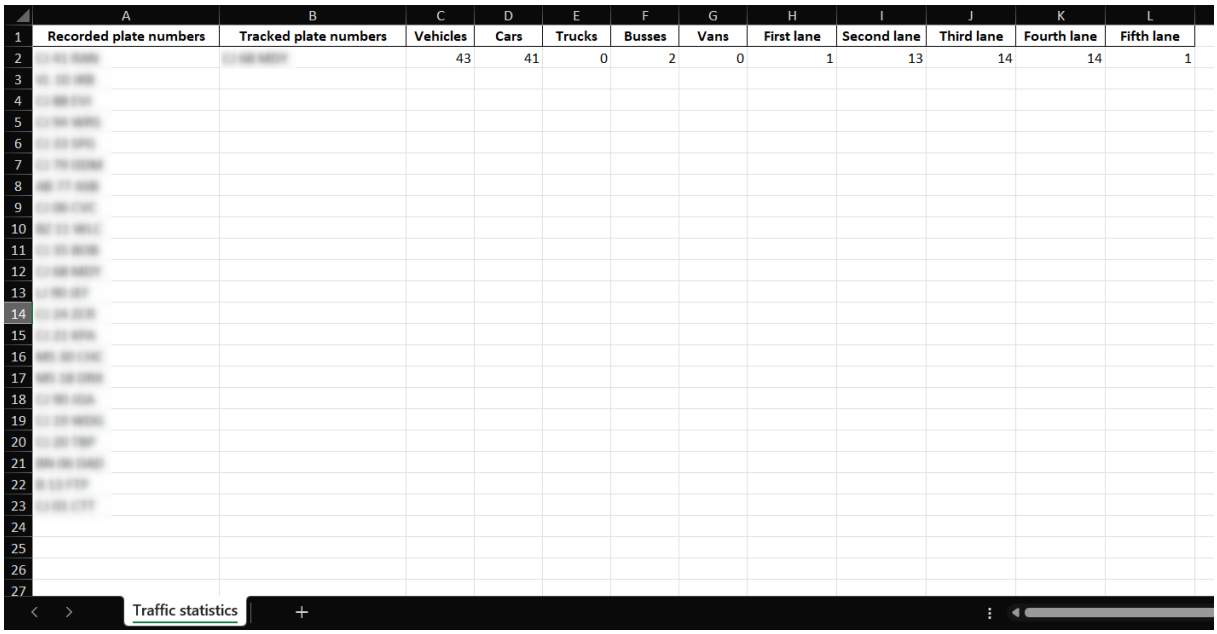
Figura 3.16. Interfața grafică a aplicației Qt Designer

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 algoritmii descrisi 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 şi se află pe lista de căutare.
- Numărul total de autovehicule identificate în videoclip în funcţie de tipul lor şi 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	[redacted]	[redacted]	43	41	0	2	0	1	13	14	14	1

Traffic statistics

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

Pentru a adăuga un număr de înmatriculare în lista de căutare se introduce numărul în secţiunea dedicată şi se apasă butonul “Start tracking”, iar pentru a curăţa 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 primeşte o notificare şi aplicaţia 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 fişierul cu statistici.

Utilizatorul are la dispoziţie un set de opţiuni pentru procesarea imaginilor în partea de jos a ferestrei principale. Aceste opţiuni pot să fie alese în orice fel şi modifică algoritmul de procesare a imaginilor, fiecare opţiune având două stari: activă şi inactivă. Cele nouă opţiuni sunt următoarele:

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

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

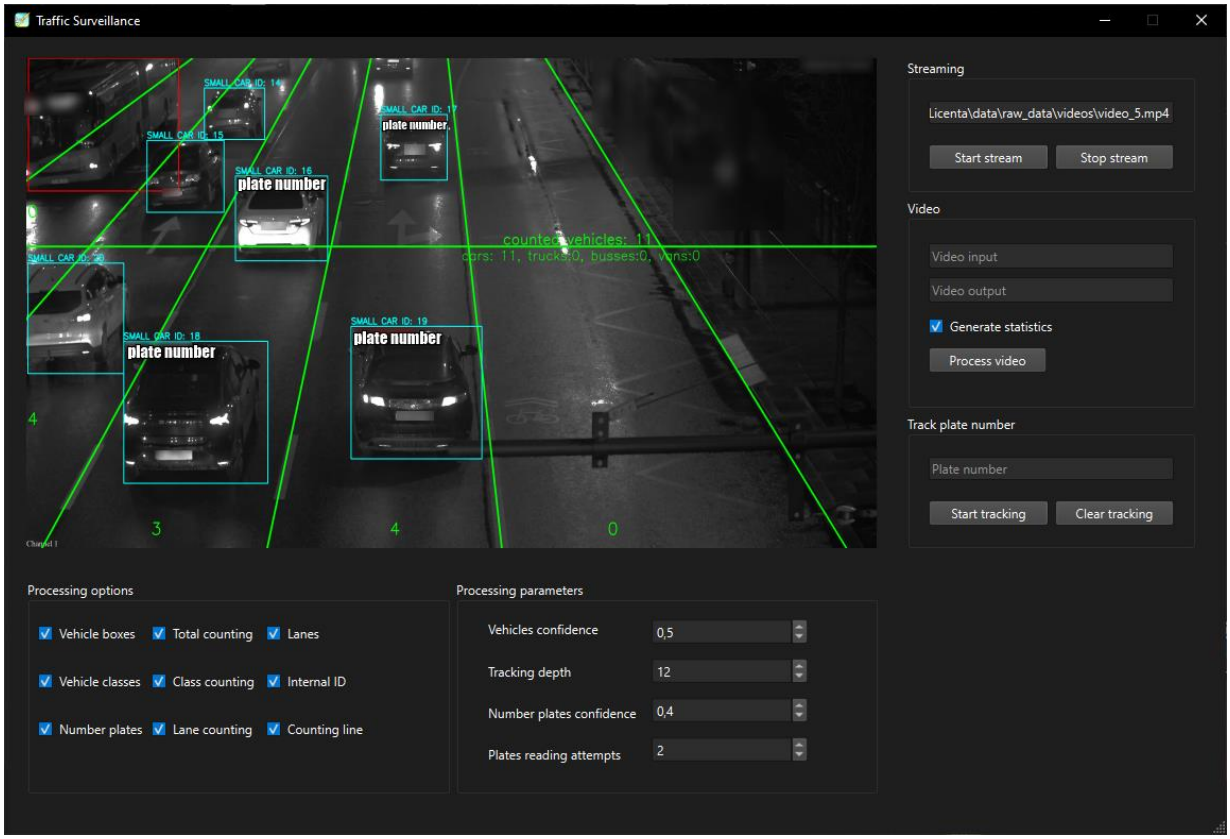


Figura 3.18. Fereastra principală a interfeţei grafice pentru utilizator

Pe lângă opţiunile de procesare, utilizatorul mai are la dispoziţie un set de parametri de procesare care pot să fie modificaţi. Modificarea acestor parametrii presupune un set minimalist de cunoştinţe asupra funcţionării algoritmilor de identificare a obiectelor şi de citire a numerelor de înmatriculare. Cei patru parametrii sunt:

- Scorul autovehiculelor: Determină scorul de prag al modelului YOLO de detecţie 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 detecţiilor. Totuşi, un număr prea mare ar putea duce la performanţe mai scăzute ale algoritmului.
- Numărul de încercări de citire: Numărul maxim de încercări de a identifica şi de a citi numerele de înmatriculare pentru fiecare autovehicul. Un număr mare dat acestui parametru poate creşte semnificativ timpul de procesare al imaginilor.

Pe langa fereastra principală a interfeţei grafice pentru utilizator, există şi 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 aplicaţiei în cazul în care utilizatorul nu deţine o unitate de procesare grafică în sistemul de calcul.

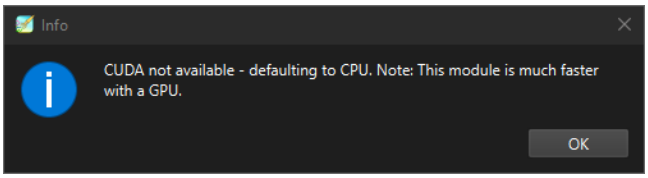


Figura 3.19. Fereastră de dialog pentru informarea utilizatorului

3.3.6.3 Generarea unui fişier executabil

Pentru a face experienţa utilizatorului cât mai plăcută şi pentru a nu-l forţa să configureze mediul de dezvoltare pentru a folosi aplicaţia, se consideră generarea unui fişier executabil. Acest singur fişier porneşte aplicaţia pe sistemele de operare Windows, făcând-o disponibilă utilizatorului.

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

```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 fişier executabil

3.4 Testare şi validare

3.4.1 Validarea detecţiilor şi a algoritmilor

Testarea algoritmilor şi a detecţiilor a fost efectuată folosind imagini şi înregistrări provenite de la camera de supraveghere video a traficului. Pentru modelele de detecţie s-au folosit părţile setului de date responsabile pentru validare şi testare.

În figura următoare se observă testarea modelului de detecţie 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 iteraţii care este folosit pentru antrenarea modelului iar pe axa Y este reprezentată valoarea funcţiei 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 funcţiilor de pierdere ceea ce înseamnă că predicţiile modelului făcute pe datele de validare devin din ce în ce mai acurate pe măsură ce numărul de iteraţii de antrenare creşte. Se observă de asemenea o plafonare a scăderilor valorilor funcţiilor de pierdere ceea ce indică faptul că numărul de iteraţii este potrivit şi o creştere a acestuia ar duce la fenomenul de supraantrenare.

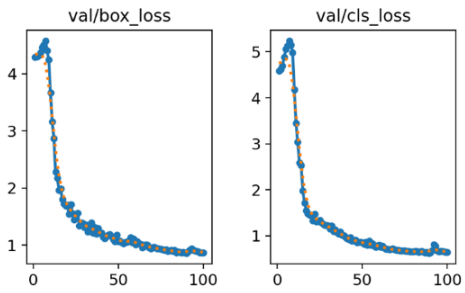


Figura 3.21. Validarea antrenării modelului de detecţie a autovehiculelor

O altă metodă de validare a modelului de detecţie a autovehiculelor este matricea de confuzie generată pe baza adnotărilor şi a detecţiilor 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 corelaţie între două tipuri de autovehicule, unul adnotat iar celălalt identificat. În figura următoare se observă matricea de confuzie a modelului de identificare a autovehiculelor.

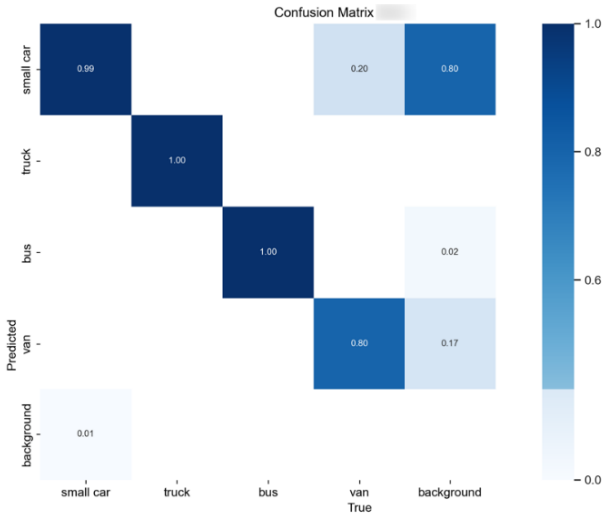


Figura 3.22. Matricea de confuzie a modelului de identificare a autovehiculelor

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

- Maşinile mici sunt identificate corect în proporţie de 99%, restul de 1% sunt identificate greşit drept fundal.
- Toate autobuzele şi camioanele sunt identificate corect.
- Camionetele sunt identificate corect în proporţie de 80%, restul de 20% fiind identificate greşit drept maşini mici.
- Fundalul imaginii este des interpretat de către model ca fiind maşină mică.

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

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

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

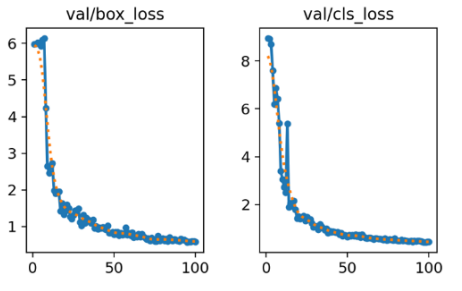


Figura 3.23. Validarea antrenării modelului de detecţie a plăcilor de înmatriculare

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 funcţionează. În cazul în care algoritmul funcţionează, se observă că acest identificator nu se modifică pe parcursul videoclipului pentru un autovehicul dat.

Algoritmii de numărare a autovehiculelor şi de identificare a numerelor de înmatriculare au fost validaţi manual, observând performanţa 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 interfeţei de utilizator şi a funcţionalităţilor

Interfaţa grafică pentru utilizator şi funcţionalităţile acesteia au fost testate manual, verificând fiecare componentă grafică în diferite scenarii şi condiţii. Pentru a verifica funcţionalitatea procesării imaginilor în contextul transmiterii în direct a datelor, s-au folosit înregistrări video şi 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 funcţionalitatea procesării de videoclipuri şi a generării de statistici. Pentru partea de urmărire a numerelor de înmatriculare, s-au observat câteva numere de înmatriculare în înregistrarea video şi s-au adaugat în lista de numere de înmatriculare urmărite. După procesarea videoclipului, numerele de înmatriculare urmărite s-au observat în fişierul cu statistici generate în coloana cu numele “Tracked plate numbers”.

Pentru a testa opţiunile şi parametrii de procesare a imaginilor disponibili în interfaţa vizuală, s-a simulat transmiterea în direct a unor date şi s-au schimbat, pe rând, fiecare parametru şi fiecare opţiune şi s-au observat vizual rezultatele. Pentru a testa parametrul de profunzime a urmăririi, acestuia i se atribuie valoarea zero şi 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 detecţii în interiorul imaginilor. Testarea parametrului responsabil de numărul de încercări de citire a numerelor de înmatriculare poate fi testat fiindu-i atribuită valoarea zero şi observând lipsa identificării de numere de înmatriculare în imagini.

Fiecare metodă manuală descrisă mai sus este repetată în diverse stări ale aplicaţiei şi folosind diferite imagini pentru a acoperi un număr cât mai mare de cazuri particulare de funcţionare a aplicaţiei. În cazul în care aplicaţia nu funcţionează, se identifică problema şi, după ce problema este rezolvată, testarea aplicaţiei se repetă.

4 Concluzii

4.1 Rezultate obţinute

Evidenţiaţi toate rezultatele pe care le-aţi obţinut şi trageţi concluzii din ele. Puteţi prezenta o analiză critică a ceea ce aţi realizat comparativ cu alte lucrări/studii anterioare.

Includeţi o listă a contribuţiilor pe care le-aţi avut în domeniul temei abordate.

4.2 Direcţii de dezvoltare

Descrieţi direcţiile posibile de dezvoltare.

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/Hyun-je/SORT-ros.

[7] F. Churchville, "User Inferface (UI)," 2024. [Online]. Available: https://www.techtarget.com/searchapparchitecture/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.