Universitatea
Transilvania
din Brașov

ȘCOALA DOCTORALĂ INTERDISCIPLINARĂ

Facultatea: Inginerie Electrică și Știința Calculatoarelor

Ing. MARINA Liviu Alexandru

# Prelucrarea statistică a datelor senzoriale locale şi geospațiale în sistemele de navigație autonomă

# Statistical Processing of Local and Geospatial Data within Autonomous Driving Systems

REZUMAT / ABSTRACT

Conducător științific

Prof. dr. ing. MOLDOVEANU Florin Dumitru

BRAȘOV, 2022

Universitatea Transilvania din Brașov

D-lui (D-nei) .....

**COMPONENȚA**

**Comisiei de doctorat**

Numită prin ordinul Rectorului Universității Transilvania din Braşov

Nr. ..... din .....

| | |
|---|---|
| PREŞEDINTE: | - Prof. univ. dr. ing. MORARU Sorin-Aurel<br>Director de departament<br>Universitatea Transilvania din Brașov |
| CONDUCĂTOR ŞTIINȚIFIC: | - Prof. univ. dr. ing. MOLDOVEANU Florin Dumitru<br>Universitatea Transilvania din Brașov |
| REFERENȚI: | - Prof. univ. dr. ing. POPESCU Dumitru<br>Universitatea Politehnica din București |
| | - Prof. univ. dr. ing. COJOCARU Dorian<br>Universitatea din Craiova |
| | - Prof. univ. dr. ing. GRIGORESCU Sorin Mihai<br>Universitatea Transilvania din Brașov |

Data, ora şi locul susținerii publice a tezei de doctorat: 08 Iulie 2022, ora 11:00, sala .....

Eventualele aprecieri sau observații asupra conținutului lucrării vor fi transmise electronic, în timp util, pe adresa marina.liviu.alexandru@unitbv.ro

Totodată, vă invităm să luați parte la şedința publică de susținere a tezei de doctorat.

Vă mulțumim.

# Cuprins

# 1. Introducere

## 1.1 Oportunitatea și aplicabilitatea tezei de doctorat

Vehiculele autonome reprezintă unul dintre domeniile care au beneficiat și beneficiază de resurse importante. Inteligența artificială (IA), pe de altă parte are un rol important în dezvoltarea funcțiilor de conducere autonomă cum ar fi: examinarea mediului în care un vehicul navighează, identificarea participanților la trafic, sau controlul deplasării vehiculului.

Conform studiilor realizate [1], 98% din totalul accidentelor produse în trafic sunt cauzate de erori umane. Pentru a se reduce acest procent, au fost dezvoltate sisteme de conducere automată al căror rol este de a preveni accidentele rutiere, de a reduce emisiile de carbon, sau de a reduce stresul provocat de conducerea unui automobil [2].

În ceea ce privește gradul de autonomie al unui autovehicul, au fost definite șase niveluri, denumite *Society of Automotive Engineers (SAE) Levels*. Standardul *SAE J3016* [3] introduce o clasificare a gradului de automatizare cu valori cuprinse între 0 și 5. Nivelurile SAE cuprinse între 0 și 2 oferă funcționalități de bază pentru asistența șoferului, pe când, următoarele niveluri se îndreaptă către o automatizare care necesită din ce în ce mai puțină interacțiune cu șoferul. Așadar, în nivelul 0 se încadrează în prezent automobilele, care sunt controlate manual de către un operator uman și nu dispun de nici o funcționalitate de conducere autonomă. Nivelul 1 se referă la autovehiculele care oferă funcții de asistență, cum ar fi păstrarea distanței în raport cu mașina din față, sau asistență la parcare. Autovehiculele cu nivelul 2 de automatizare sunt capabile să controleze atât virarea cât și accelerarea sau frânarea. Cu toate acestea, șoferul trebuie să mențină mâinile pe volan tot timpul. Un exemplu pentru acest nivel de automatizare poate fi parcarea automată sau menținerea autovehiculului pe banda de circulație.

În cadrul nivelului 3 SAE, monitorizarea permanentă din partea șoferului nu este obligatorie, autovehiculului putând conduce, naviga și lua decizii critice de unul singur. Șoferul poate prelua însă controlul când autovehiculul cere intervenția acestuia, sau când dorește să conducă. Nivelul 4 este cu un pas mai aproape de autonomia completă. În marea majoritate a situațiilor și mediilor, automobilul nu necesită control din partea șoferului, acesta putându-se deplasa și lua decizii salvatoare de viață fără intervenție umană. Desigur, se permite preluarea controlului, dacă se dorește acest lucru. Nivelul 5 reprezintă viitorul tehnologiei de conducere autonomă. Vehiculele cu nivelul 5 de automatizare vor fi capabile să navigheze în orice tip de mediu sau trafic, să îndeplinească sarcini complexe și să poate lua decizii indiferent de complexitatea sau urgența situațiilor. Autovehiculele din această categorie nu vor necesita nici un fel de asistență din partea unui operator uman.

Pentru a se atinge nivelurile de automatizare superioare este necesar ca algoritmii clasici să fie înlocuiți cu algoritmi ce se pot adapta situațiilor neprevăzute și care pot folosi volumul mare de date diponibil. Tehnicile de inteligență artificială pot fi utilizate pentru a se crea modele complexe ce pot îndeplini cu succes sarcini de percepție sau control. În ultimii ani, datorită avansului tehnologic, algoritmii creați prin metode specifice de *machine learning* (ML) au un rol din ce în ce mai important în progresul conducerii autonome. Chiar dacă acest tip de algoritmi prezintă o complexitate ridicată, datorită componentelor hardware proiectate special pentru accelerarea calculelor necesare rețelelor

neurale, pot fi integrați în sistemele din componența autovehiculelor. În ceea ce privește antrenarea modelelor de tip deep learning, dezvoltarea procesoarelor grafice a permis utilizarea unor arhitecturi din ce în ce mai complexe, care pot să atingă performanțe ridicate dacă sunt utilizate împreună cu seturi de date cuprinzătoare și variate.

Deoarece accesul la date de antrenare de calitate reprezintă o problemă atunci când resursele financiare sunt limitate, este important să existe metode pentru a se genera date sintetice, care să simuleze condițiile de trafic ce pot fi întâlnite. În acest scop se pot utiliza medii de simulare, unde senzori precum LIDAR sau RADAR, dinamica unui vehicul, sau participanți la trafic se pot replica într-un mod realist [4].

Chiar dacă în ultimii ani s-au făcut eforturi importante pentru a se rezolva problemele de percepție și înțelegere a mediului în care un autovehicul se deplasează, funcționalitățile de asistență a șoferului necesită încă expertiză și inovație pentru a devini robuste. Așa cum a fost amintit, fiecare senzor utilizat are avantajele și dezavantajele sale. Din acest motiv fuziunea datelor provenite din mai multe surse poate ajuta un sistem să primească informații mai detaliate și mai exacte, pe baza cărora să poată lua cele mai bune decizii. De asemenea, detectarea în timp-real a contextului în care un vehicul se deplasează este încă o problemă nerezolvată, deoarece hărțile de navigare de înaltă rezoluție necesită mult efort pentru a fi implementate, iar semnalul GPS (eng. *Global Positioning System*) are erori ce nu pot fi acceptabile în cazul unui vehicul autonom.

Nivelul 5 SAE promite o automatizare totală, așa cum s-a mai menționat, prin care autovehicul să fie capabil să ia decizii în situații limită, asemenea unui utilizator uman. În această direcție, viitoarele cercetări se vor concentra pe obținerea unui algoritm care să simuleze un comportament cât mai apropiat de cel al unui șofer, eliminându-se însă factorii care duc la accidentele rutiere din ziua de astăzi. Acest obiectiv este departe de a fi realizat, însă, această teză de doctorat demonstrează că există posibilitatea de a simula un astfel de comportament atunci când se dorește controlul deplasării unui autovehicul.

## 1.2 Obiectivele tezei de doctorat

Avansul tehnologiilor bazate pe inteligență artificială a impactat decisiv dezvoltarea autovehiculelor autonome, intensificând dezvoltarea capabilităților de percepție și control. Dezvoltarea procesoarelor grafice performante, a arhitecturilor hardware ce pot rula cu succes algoritmi de ML și a platformelor cloud ce pot stoca seturi de date impresionante și pot să ofere capacități de calcul sporite, a condus ca în prezent, în automobile, să existe funcționalități dezvoltate prin tehnica de IA.

Având în vedere că multe dintre funcțiile de asistență ale șoferului și cele de control ale autovehiculului încă se bazează pe algoritmi clasici, domeniul de ML poate aduce îmbunătățiri semnificative în ceea ce privește performanța acestora. Așadar, această teză de doctorat contribuie la dezvoltarea unor noi algoritmi pentru percepția mediului din jurul unui autovehicul , prin interpretarea statistică a datelor provenite de la senzori, și demonstrează aplicabilitatea algoritmilor de tip ML pentru sarcini de control. De asemenea, se prezintă și posibilitatea de industrializare a modelelor generate, pentru a fi testate și evalute în sistemele integrate în autovehicule.

În concluzie, s-au avut în vedere următoarele obiective principale de cercetare:

- Percepția mediului din jurul unui autovehicul:
  - Dezvoltarea unei platforme software open-source pentru detecția în timp-real a participanților la trafic în cadrul unui simulator de curse de mașini;
  - Implementarea unor mecanisme pentru generarea datelor sintetice utilizate în procesul de antrenare;

  - Reprezentarea spațială sub formă de griduri de ocupanță și utilizarea acestei reprezentări ca intrare într-o rețea neurală destinată clasificării mediului în care se deplasează un vehicul;
  - Implementarea unei rețele neurale de convoluție de complexitate redusă pentru clasificarea scenariilor de trafic;
  - Implementarea unui mecanism pentru optimizarea rețelelor neurale prin ajustarea automată a setului de hiperparametri.

- Industrializarea modelelor de machine learning:
  - Implementarea unui cadru pentru industrializarea modelelor de ML și dezvoltarea acestora conform standardelor de calitate acceptate de industria constructoare de autovehicule;
  - Evaluarea modelelor dezvoltate pentru percepția mediului din jurul unui vehicul prin integrarea acestora în arhitecturi software utilizate în producția de serie, în machete ale unor vehicule complet echipate pentru conducere autonomă și, de asemenea, în automobile care au fost echipate în prealabil cu senzorii necesari algoritmilor de percepție.

- Prelucrarea statistică a datelor senzoriale pentru generarea traiectorie și controlul unui vehicul autonom:
  - Definirea unei arhitecturi de rețea neurală capabilă să controleze și să genereze o traiectorie a vehiculelor autonome prin simularea unui comportament asemănător celui uman;
  - Utilizarea unei abordări de neuro-evoluție pentru antrenarea sistemelor de conducere autonomă bazată pe optimizarea Pareto multi-obiectiv.

## 1.3 Organizarea tezei de doctorat

Teza de doctorat este structurată după cum urmează:

**Capitolul 1** conține o prezentare generală a domeniului în care s-a efectuat cercetarea, definindu-se contextul și justficându-se relevanța științifică. Acest capitol subliniază problemele ce se doresc a fi rezolvate, principalele obiective, împreună cu metodologiile utilizate și experimentele efectuate în timpul cercetărilor. În finalul capitolului este expusă structura lucrării și conținutul tezei.

În **capitolul 2** este prezentat stadiul actual al cercetărilor privind aplicarea algoritmilor de tip deep learning în domeniul vehiculelor autonome. Se prezintă cercetările efectuate pentru îmbunătățirea procesului de percepție a mediului în care se deplasează un autovehicul, prin aplicarea tehnicilor de detecție și recunoaștere a obiectelor bidimensionare și tridimensionale, de segmentare semantică, și de reprezentare a spațiului prin griduri de ocupanță. Acest capitol cuprinde și o analiză a metodelor disponibile pentru controlul și localizarea autovehiculelor autonome, alături de un studiu al algoritmilor de neuro-evoluție ce pot fi utilizați pentru căutarea unui set optim de hiperparametri sau pentru determinarea ponderilor unei rețele neurale. Se menționează, de asemenea, și platformele de accelerare hardware disponibile pentru inferența algoritmilor dezvoltați.

În **capitolul 3** este prezentat procesul de dezvoltare și evaluare a unei platforme software pentru implementarea și testarea algoritmilor de percepție, aplicați domeniului conducerii autonome. Platforma are la bază un simulator de curse de mașini, open-source, care permite integrarea algoritmilor de machine learning. Acest simulator oferă posibilitatea de a se achiziționa date despre vehiculul ce se dorește a fi controlat, precum și date despre mediul în care acesta se deplasează.

În **capitolul 4** este prezentat un algoritm inovativ dezvoltat pentru clasificarea contextului în care un autovehicul se deplasează. Pentru a se determina dacă un autovehicul rulează într-un oraș,

pe autostradă, pe un drum județean, într-o parcare, sau într-o zonă cu ambuteiaj, sunt calculate hărți de ocupanță utilizându-se date de la sezorii LIDAR și RADAR montați pe autovehicul. Sistemul implementat se bazează pe puterea și performanța arhitecturilor de rețele neurale pentru a învăța o reprezentare de tip grid a scenelor de trafic. Se preferă utilizarea gridurilor de ocupanță în locul imaginilor, deoarece pot face față cu succes incertitudinilor prezente în scenele de trafic, cum ar fi schimbări în calibrarea senzorilor, poziția, timpul și latența. Sistemul proiectat este capabil să clasifice, în timp-real contextul în care se deplasează un autovehicul utilizându-se o rețea neurală de adâncime convolutivă. Acest model preia ca date de intrare gridurile de ocupanță exportate sub formă de imagini și furnizează ca ieșire probabilitatea ca autovehiculul să se afle într-una dintre situațiile de trafic menționate anterior. Pentru a se reduce complexitatea modelului s-au utilizat algoritmi genetici pentru a se determina arhitectura ideală a rețelei și setul optim de hiperparametri. Pentru a se testa aplicabilitatea algoritmului într-un scenariu de trafic real, acesta a fost integrat într-o arhitectură funcțională care gestionează complexitatea sarcinilor de conducere autonomă și care poate fi instalată într-un autovehicul pentru evaluarea datelor provenite de la senzori. Capitolul se încheie prin prezentarea rezultatelor obținute în urma evaluării și prin expunerea dificultăților și a punctelor deschise ce urmează a fi rezolvate.

**Capitolul 5** este o continuare a cercetărilor realizate în capitolul 4, și propune o platformă software ce poate fi utilizată pentru a se genera motoare de inferență bazate pe algoritmi de machine learning. Pentru generarea modelelor destinate inferenței, ce conțin funcții de conducere autonomă, se utilizează metodologii de dezvoltare software standardizate și acceptate de industria constructoare de autovehicule. Pentru a se demonstra oportunitatea dezvoltării unei astfel de platforme, se generează motoare de inferență ce pot fi instalate într-un mediu embedded și care pot rula într-o configurație hardware cu capacitate redusă de procesare. În acest scop se dezvoltă un algoritm pentru clasificarea în timp-real a spațiilor interioare ale unei clădiri de birouri. Pentru achiziția de date și testarea modelului s-a folosit o machetă a unui vehicul, echipată pentru îndeplinirea sarcinilor de conducere autonomă. Validarea modelelor generate prin platforma propusă s-a realizat prin integrarea acestora ca module independente într-un framework care este utilizat în producția de serie a componentelor software de companii importante din industrie. În cadrul acestui capitol se demonstrează și posibilitatea de a se instala modelele generate în Cloud, prezentându-se astfel flexibilitatea în alegerea tipului de hardware și de sistem de operare pentru inferență.

În **capitolul 6** se prezintă o abordare bazată pe neuro-evoluție, capabilă să genereze traiectoria unui vehicul autonom prin învățarea de tip multi-obiectiv. În acest scop se utilizează algoritmi de tip deep learning, mai exact o rețea neurală convoluțională-recurentă profundă, care este antrenată printr-o tehnică inovativă de neuro-evoluție, bazată pe optimizarea Pareto multi-obiectiv și pe algoritmi genetici. În acest capitol s-a pus accentul pe interpretarea statistică a datelor senzoriale, procesate în scopul unei reprezentări a mediului din jurul unui vehicul într-un spațiu de căutare redus. S-a ales utilizarea straturilor de tip *Long Short-Time Memory* (LSTM) care au capacitatea de a aproxima secvențele de date temporale. Datele de intrare utilizate pentru antrenarea algoritmului sunt compuse dintr-o secvență de griduri de ocupanță, poziția vehiculului și coordonatele destinației, sarcina rețelei neurale fiind de a estima strategia optimă pentru ca vehiculul să se deplaseze la coordonatele destinației. Procesul de antrenare utilizează un vector care conține mai multe obiective: calea parcursă de agent, viteza laterală a acestuia și viteza longitudinală. Una dintre secțiunile acestui capitol este dedicată rezultatelor experimentale obținute prin utilizarea unui set de date achiziționat cu un automobil complet echipat pentru conducerea autonomă, alături de un set de date sintetice generate prin intermediul unui mediu de simulare.

În final, **capitolul 7** evidențiază concluziile generale și rezumă noi oportunități de cercetare. De asemenea, se menționează contribuțiile personale și se prezintă rezultatele obținute în perioada studiilor doctorale.

# 2. Stadiul actual al cercetărilor în domeniu

Pe parcursul ultimelor decenii inteligența artificială și tehnica deep learning au devenit principalele tehnologii din spatele multor descoperiri din domeniul procesării imaginilor [5], roboticii [6] și înțelegerii și procesării limbajului natural [7]. Aceste tehnologii au, de asemenea, un impact major în revoluția autovehiculelor autonome, prezentă astăzi atât în mediul academic cât și în cel industrial. Autovehiculele autonome migrează din interiorul laboratoarelor către testarea pe drumurile publice. Lansarea vehiculelor autonome promite o reducere a numărului de accidente și ambuteiaje și, de asemenea, îmbunătățirea mobilității în orașele foarte aglomerate.

Chiar dacă până la 95% dintre scenariile de trafic pot fi rezolvate relativ simplu cu algoritmii clasici de percepție, de control al mișcării sau de planificare a traiectoriei, rămân 5% care sunt cazuri speciale pe care metodele clasice nu reușesc să le îndeplinească. Rezolvarea acestor cazuri speciale reprezintă soluția pentru lansarea pe scară largă a vehiculelor autonome.

## 2.1 Percepția mediului în care se deplasează autovehiculele

Tehnologia conducerii autonome permite vehiculelor să opereze în mod autonom, observând mediul înconjurător și instrumentând un răspuns reactiv. O cerință esențială pentru un vehicul autonom este abilitatea sa de a înțelege mediul înconjurător în timp-real. O cantitate mare de resurse a fost alocată în ultimii ani pentru a se rezolva provocările percepției mediului înconjurător. Rețelele neurale artificiale au devenit soluția principală pentru a analiza cantitatea considerabilă de date provenită de la diferiți senzori montați pe o mașină fără șofer.

În cele ce urmează se va prezenta starea actuală a cercetărilor în domeniu, având în vedere tematici ca detecția de obiecte, segmentarea semantică și înțelegerea scenei utilizându-se griduri de ocupanță. Studii dedicate percepției mediului înconjurător pot fi găsite în referințele [8] și [9].

### 2.1.1 Detecția și recunoașterea obiectelor

O metodă de detecție de obiecte robustă reprezintă o funcționalitate esențială a vehiculelor inteligente. O mașină autonomă trebuie să dețină capabilitatea de a detecta participanții la trafic, în mod particular în zonele urbane unde o mare varietate de obiecte și ocluzii pot să apară. Rețelele neurale de convoluție au devenit standardul de-facto în recunoașterea și detecția obiectelor, obținându-se rezultate remarcabile în competiții cum ar fi, de exemplu, *ImageNet Large Scale Visual Recognition Challenge* [10].

O analiză cuprinzătoare asupra framework-urilor generale de detecție de obiecte poate fi găsită în [11]. Autorii furnizează o scurtă descriere a metodelor bazate pe propunerea de regiuni de interes și o evaluare experimentală a fiecărei metode. Cele mai populare arhitecturi care sunt, de asemenea, utilizate în aplicațiile destinate conducerii autonome, sunt detectoarele într-o singură etapă (eng. *single stage*) sau în două etape (eng. *double stage*). În [12], autorii fac o prezentare extinsă a cercetărilor recente în domeniul detecției obiectelor, incluzând și o descriere a diferitelor topologii de

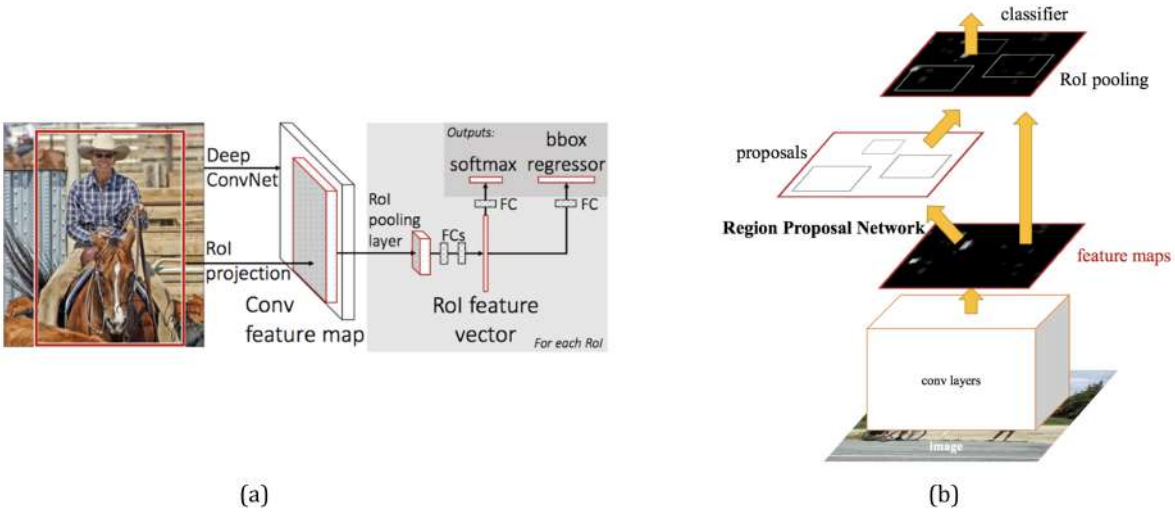



Figura 2.1: Prezentare arhitecturi algoritmi: (a) Fast R-CNN și (b) Faster R-CNN. Imaginile au fost preluate din [17], respectiv [18].

rețele neurale utilizate pentru îndeplinirea acestor sarcini.

În general, algoritmii prin intermediul cărora se realizează detecția într-o singură etapă nu furnizează aceleași performanțe ca cei ce realizează detecția în două etape, dar sunt semnificativ mai rapizi. Două dintre cele mai populare abordări ale algoritmilor de tip single stage sunt YOLO (eng. *You Only Look Once*) [13] și SSD (eng,. *Single Shot Multibox Detector*) [14]. YOLO are capacitatea de a calcula coordonatele casetelor de încadrare și probabilitatea de a aparține unei anumite clase de obiecte pentru fiecare detecție, într-un timp relativ redus. Variante îmbunătățite ale algoritmului YOLO au fost publicate în [15] și [16] unde au fost aduse îmbunătățiri în termeni de acuratețe și viteză de detecție, prin normalizarea lotului de eșantioane, predicția locațiilor sau prin variația dimensiunilor casetelor de detecție. YOLO poate fi utilizat cu succes în aplicații de percepție a mediului înconjurător, putând fi optimizat pentru detecția participanților la trafic din zonele urbane, cum ar fi alte vehicule sau pietoni.

Algoritmii de detecție în două etape, precum R-CNN (eng. *Regions with CNN features*) [19], împart procesul de detecție de obiecte în două faze: propuneri ale regiunilor de interes ale candidaților și clasificarea cadrelor de determinare. Una dintre cercetările care au impactat domeniul detecției obiectelor a fost referită în lucrarea [17], unde a fost introdus algoritmul Fast R-CNN pentru a rezolva o parte din inconvenientele prezente în R-CNN [19], unul dintre acestea fiind viteza redusă de detecție. Acest articol demonstrează că metoda Fast R-CNN este de nouă ori mai rapidă în comparație cu R-CNN. Acest algoritm este compus din trei module. Primul realizează propunerea de regiuni, al doilea este rețeaua neurală care îndeplinește rolul de extractor de caracteristici, iar al treilea este modulul ce realizează clasificarea. Modulul de propuneri de regiuni selectează un set de posibile cadre de încadrare iar apoi, utilizând caracteristicile extrase de rețeaua neurală, clasificatorul produce ca rezultat clasa corespunzătoare fiecărui cadru.

Faster R-CNN [18] este un algoritm de detecție eficient care extinde cercetările realizate în [17] cu o rețea neurală de convoluție foarte adâncă antrenată într-o singură etapă, utilizându-se o funcție de cost de tip multiobiectiv. În [18] autorii au demonstrat că utilizând arhitectura algoritmului Fast R-CNN [17] se pot genera propunerile de regiuni de interes, reducând în acest mod timpul de procesare și apropiindu-se de cerințele rulării în timp-real. În figura 2.1 sunt prezentate arhitecturile algoritmilor Fast R-CNN, respectiv Faster R-CNN, arhitecturi care au fost preluate din articolele de referință corespunzătoare.

### 2.1.2 Segmentarea semantică

În afară de clasificarea imaginilor și detecția obiectelor, cercetările din domeniul sistemelor de vedere artificială au abordat și problema segmentării imaginilor. Segmentarea semantică a scenei reprezintă etichetarea în categorii presetate a fiecărui pixel dintr-o imagine. În contextul conducerii autonome, pixelii pot fi etichetați ca suprafețe carosabile, pietoni, participanți la trafic, clădiri, ș.a.m.d..

Segmentarea semantică este una dintre tehnicile de nivel înalt care pavează calea către înțelegerea completă a scenei, fiind utilizată în aplicații cum ar fi conducerea autonomă, navigarea în spații interioare sau în aplicații de realitate virtuală sau augmentată. Un studiu amplu asupra acestei metodologii poate fi găsit în [20], unde autorii fac o trecere în revistă exhaustivă a metodelor de tip deep learning utilizate pentru segmentarea semantică și aplicabilitatea acestora în domenii variate. În cele ce urmează se va realiza o trecere în revistă a cercetărilor actuale privind segmentarea semantică, punându-se accent pe aplicabilitatea acestora în domeniul conducerii autonome a autovehiculelor.

Ca și în cazul detecției și recunoașterii obiectelor, rețelele neurale adânci au adus contribuții semnificative în domeniu. Există o serie de arhitecturi de rețele neurale care sunt utilizate în mod uzual ca bază pentru algoritmii destinați segmentării semantice. Dintre acestea se pot aminti: AlexNet [21], VGG-16 [22], GoogLeNet [23], ResNet [24] sau SegNet [25].

### 2.1.3 Percepția utilizându-se griduri de ocupanță

O hartă de ocupanță, cunoscută și sub numele de grid de ocupanță (eng. *Occupancy Grid* (OG)), este o reprezentare a mediului înconjurător care împarte spațiul într-un set de celule și calculează probabilitatea de ocupare a fiecărei celule. În figura 2.2 pot fi observate reprezentări ale unor scene de trafic sub forma unor hărți de ocupanță.

Gridurile de ocupanță au fost utilizate cu succes în cazul navigării agenților autonomi. În [26] rețele neurale de convoluție au fost antrenate utilizându-se date bidimensionale provenite de la senzori de distanță pentru etichetarea semantică a diferitelor spații în medii necunoscute. Această abordare permite unui robot să utilizeze un senzor de tip LIDAR pentru clasificarea spațiilor.

Conducerea într-un mediu exterior implică, de asemenea, și interacțiunea cu obiecte dinamice. Rețelele neurale recurente au fost utilizate de Ondruska [27] pentru urmărirea și clasificarea mediului din jurul unui robot plasat într-un mediu dinamic și parțial observabil. O RNR filtrează fluxul de intrare compus din măsurătorile brute ale unui laser pentru a deduce locațiile obiectelor, împreună cu identitatea acestora, atât în ariile vizibile cât și în cele ocluzionate. Algoritmul descris în [27] se inspiră din Deep Tracking [28], care este un sistem de tip DL ce se bazează pe rețele neurale pentru urmărirea de la un cap la altul (eng. *end-to-end*) a obiectelor.

Chiar dacă hărțile de ocupanță sunt instrumente comune în domeniul roboticii, cazurile în care au fost utilizate împreună cu metode DL pentru aplicații în timp-real sunt reduse. Această nișă nu a fost suficient studiată până în acest moment și poate furniza multe posibilități de cercetare. Utilizându-se date recepționate de la un senzor LIDAR, în [29] este propusă o versiune îmbunătățită a regulii *Redistribuției Conflictului Proporțional* (eng. *Proportional Conflict Redistribution '#'6 - PCR6*), luându-se în considerare teoria lui Zhang privind gradul de intersecție a elementelor focale [30].

În afara funcționalităților de detecție, gridurile de ocupanță pot fi utilizate și pentru a clasifica și eticheta contextul în care un autovehicul se deplasează. Pornind de la metodele dezvoltate în [31], unde OG și rețelele neurale au fost utilizate pentru a clasifica scenele de trafic, în timpul cercetărilor pentru această teză, această idee a fost dezvoltată, perfecționată și aplicată în sisteme ce rulează în timp-real. În [31] hărțile de ocupanță au fost create prin acumularea informațiilor de-a lungul timpului

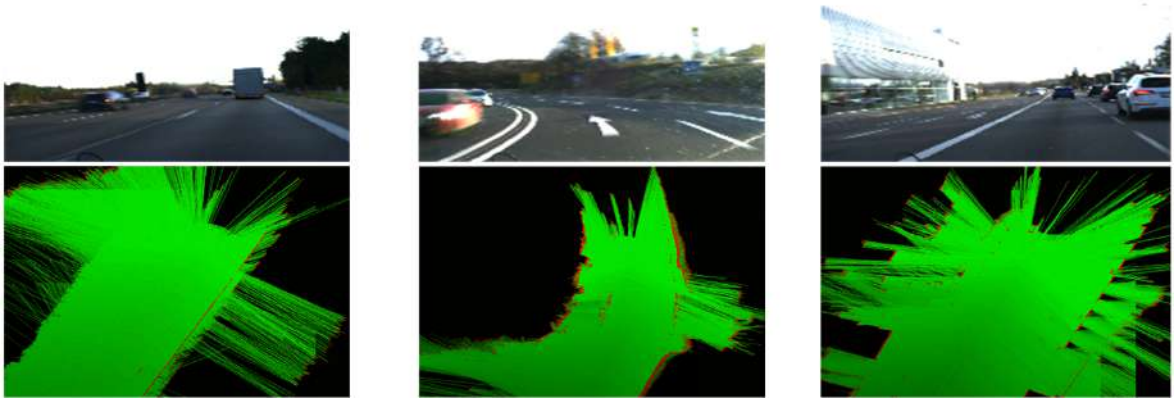


Figura 2.2: Exemplu de reprezentare a scenelor de trafic ca hărți de ocupanță.

și au fost utilizate pentru a se eticheta scenariile de trafic în anumite clase, cum ar fi zona de parcare, mediul urban sau autostrada.

În această teză gridurile de ocupanță au fost folosite intens pentru a se studia noi modalități de a se detecta scenariile de trafic în timp-real, informație utilizată ca intrare pentru algoritmii de arbitrare a comportamentului vehiculelor autonome.

## 2.2 Controlul și localizarea autovehiculelor autonome utilizându-se algoritmi de deep learning

### 2.2.1 Localizarea autovehiculelor autonome

Una dintre problemele fundamentale în conducerea autonomă este abilitatea de a localiza cât mai precis poziția autovehiculului, în timp-real. Localizarea exterioară se bazează masiv pe tehnologia GPS. Datorită variațiilor privind acuratețea și momentele când semnalul GPS lipsește (în interiorul tunelurilor sau în zonele fără acoperire), această abordare nu este recomandată în cazul vehiculelor autonome. Pentru a îmbunătății localizarea au fost propuse metode alternative ce se bazează pe tehnologia deep learning.

Capabilitatea de a localiza un vehicul estimând poziția camerei dintr-o imagine este o sarcină importantă în aplicațiile de navigație autonomă a vehiculelor autonome bazate pe vederea artificială. PoseNet [32], este un sistem de relocalizare monocular, cu șase grade de libertate ce funcționează robust în aplicațiile de timp-real, și care antrenează o rețea neurală de convoluție pentru a învăța maparea prin regresie a poziției unei camere video cu șase grade de libertate dintr-o singură imagine color (RGB). LTSM-Pose [33] îmbunătățește acuratețea localizării utilizând unități LSTM (eng. *Long Short-Term Memory*) [34] aplicate stratului complet conectat de ieșire din PoseNet, rezolvând în acest mod problema fenomenului de overfitting din [32].

În lucrarea [35] este propus un sistem de localizare bazat pe LIDAR, denumit $L^3$ - Net, capabil să obțină o acuratețe în centimetri a localizării. $L^3$ - Net învață descriptori locali, optimizați în mod specific să fie compatibili în scenarii de trafic diferite. Algoritmul utilizează rețele neurale recurente, acestea fiind eficiente în modelarea dinamicii vehiculelor, oferind o mai bună acuratețe și fluență temporală.

### 2.2.2 Învățarea prin întărire

Paradigma de învățarea prin întărire (eng. *reinforcement learning*) este considerată a fi una dintre cele mai puternice din domeniul inteligenței artificiale, aplicându-se pentru a antrena și învăța autovehiculele să se comporte prin interacțiunea cu mediul.

Recent, conceptul de *Deep Reinforcement Learning (DRL)* a fost introdus și testat cu succes de către compania DeepMind din Londra în jocuri ca Atari 2600 sau Go [36], demonstrându-se capabilitatea de a învăța cu succes reprezentarea mediului. DRL este o metodă de învățare nesupervizată, în care avantajele conceptului de învățare prin întărire sunt combinate cu versatilitatea și capabilitatea de generalizare a rețelelor neurale artificiale adânci.

Un algoritm DL pentru planificarea traiectoriei pentru conducerea autonomă este propus în [37]. Metoda adresează problema modelării erorii și dependențele necesare urmăririi traiectoriei. Găsirea celei mai bune traiectorii către o locație țintă din interiorul unui grid este analizată în [38] cu metode de planificare cunoscute în literatură, corelate cu abordările bazate pe rețele neurale.

Un alt studiu despre reinforcement learning și aplicabilitatea sa în domeniul conducerii autonome a fost efectuat în lucrarea [39] unde s-a sintetizat stadiul actual al acestei metodologii.

## 2.3 Neuro-evoluția

### 2.3.1 Introducere în neuro-evoluție

Cercetările curente privind rețelele neurale se concentrează cu precădere pe domeniile de deep learning și reinforcement learning. În aceste domenii metoda dominantă de antrenare este reprezentată de propagarea înapoi a erorii, un algoritm eficient pentru calcularea gradientului funcției de cost.

O abordare alternativă, care se inspiră din procesul biologic care stă la baza funcționării creierul uman, este antrenarea rețelelor neurale utilizându-se algoritmi de evoluție. Acest domeniu este denumit neuro-evoluție (eng. *Neuro Evolution*) și activează capacități ale rețelelor neurale artificiale care, în mod tipic, nu sunt disponibile abordărilor bazate pe minimizarea gradientului. Astfel de capacități pentru rețelele neurale includ învățarea funcțiilor de activare, hiperparametrilor, topologiei sau chiar a regulilor de auto-învățare.

Chiar dacă neuro-evoluția a existat pentru o perioadă mare de timp ca o arie de cercetare independentă, apare o tendință ca linia de separație dintre aceasta și DL, ambele referindu-se la rețelele neurale, să se estompeze treptat. Ideile din neuro-evoluție sunt infuzate în DL, atât prin hibridizarea evoluției și a descreșterii gradientului, cât și din punctul de vedere al transferului de perspective conceptuale de la o paradigmă la alta. Această tendință este accelerată în special datorită extinderii resurselor de calcul, deschizând noi perspective pentru algoritmii de evoluție, care s-au concentrat în trecut cu precădere pe modelele de mici dimensiuni.

## 2.4 Medii de simulare

Deoarece un vehicul autonom își bazează deciziile pe informațiile primite de la senzori, este evident că virtualizarea diferiților senzori este cheia pentru orice platformă de simulare utilizată pentru conducerea autonomă. În orice caz, pentru ca o simulare să fie cât mai apropiată de realitate, sunt necesare modele perfecte pentru a se minimiza diferența rezultată [51, 52].

Simulatoarele moderne tind să aibă următoarele funcționalități:

Tabela 2.1: Sumar funcționalități simulatoare destinate testării și implementării algoritmilor de conducere autonomă.

| Simulator | Licență | Motor fizic | Limbaj de programare | Agent extern |
|---|---|---|---|---|
| Carla [40] | GPL/Open-Source | Unreal Engine | Python | Da |
| AirSim [41] | GPL/Open-Source | Unreal Engine | C++, Python, C#, Java | Da |
| DeepDrive [42] | GPL/Open-Source | Unreal Engine | C++, Python | Da |
| Udacity [43] | GPL/Open-Source | Unity | C++, Python | n |
| Constellation [44] | Restricționată | PhysX/CUDA | C/C++, Python | Da |
| Carcraf/Waymo [45] | Restricționată | n | Python | Da |
| SIMLidar [46] | GPL/Open-Source | n | C++ | Da |
| Helios [47] | GPL/Open-Source | JMonkey Engine | Java | Da |
| GLIDAR [48] | GPL/Open-Source | n | C++ | Da |
| RADSim [49] | Comercială | n | Matlab | n |
| TORCS [50] | GPL/Open-Source | Da | C++ | n |

- prototipare rapidă;
- motoare fizice pentru deplasările reale;
- randare 3D realistă; instrumente standard de modelare 3D și instrumente terțe pot fi utilizate pentru a construi mediile simulate;

În tabelul 2.1 sunt comparate principalele caracteristici ale simulatoarelor utilizate în mod uzual în cercetare.

## 2.5 Concluzii

În acest capitol a fost prezentat stadiul actual al cercetărilor privind algoritmii de machine learning aplicați domeniului conducerii vehiculelor autonome. Această temă s-a impus în literatura de specialitate datorită tendinței marilor producători de autovehicule de a fabrica autovehicule cu un grad cât mai mare de autonomie.

După o scurtă introducere în domeniu s-au prezentat metodele specifice percepției mediului înconjurător. Acest subiect a fost intens dezbătut și cercetat de-a lungul anilor deoarece un pas esențial în avansul privind conducerea autovehiculelor autonome este reprezentat de înțelegerea mediului unde acestea navighează. Au fost prezentate rezultatele obținute pentru detecția și recunoașterea obiectelor, percepția utilizându-se griduri de ocupanță sau segmentarea semnatică.

Una dintre limitările importante în cercetările privind autovehiculele autonome este lipsa unor seturi de date care să permită antrenarea și evaluarea algoritmilor. De aceea, în această lucrare cercetările au fost efectuate utilizându-se baze de date open-source și, de asemenea, folosindu-se o bază de date creată prin achiziționarea informațiilor cu ajutorul unui vehicul echipat cu o gama largă de senzori. O abordare care a permis extinderea datelor pentru o mai bună generalizare a algoritmilor a fost utilizarea unor medii de simulare, medii din care informația necesară a putut fi extrasă cu ușurință. De asemenea, utilizarea unor medii de simulare duce la creșterea variației setului de date artificiale, deoarece pot fi create noi scenarii.

Așa cum a fost prezentat în acest capitol, pentru a se optimiza arhitectura unei rețele neurale este nevoie de tehnici inovative pentru a se găsi topologia și setul de hiperparametri ideale. Tehnica de neuro-evoluție poate fi folosită cu succes în antrenarea algoritmilor deep learning, selectând cei mai performanți indivizi ai unei populații, indivizi care sunt apoi testați intensiv pentru a se valida rezultatele obținute.

# 3. Platformă software pentru implementarea și testarea algoritmilor de percepție

În capitolul anterior s-au menționat diferite metode utilizate pentru percepția mediului încojurător și, de asemenea, aplicabilitatea acestora în domeniul conducerii autonome a autovehiculelor. S-au prezentat diverse medii de simulare unde algoritmii pot fi antrenați și validați. În cadrul acestui capitol se vor introduce conceptele teoretice ale algoritmilor utilizați pentru detecția și recunoașterea obiectelor. Algoritmi de acest tip se vor utiliza într-un mediu de simulare a curselor de mașini, cunoscut sub numele de TORCS [50]. Deoarece algoritmii realizați se bazează pe conceptele paradigmei de machine learning, sunt expuse și conceptele teoretice care stau la baza acesteia.

Pentru dezvoltarea și aplicarea algoritmilor de percepție, în cadrul studiilor doctorale s-a dezvoltat o platformă software open-source, care permite integrarea acestora cu mediul de simulare TORCS. Prin această platformă se efectuează o achiziție facilă a datelor vizuale ce descriu mediul din jurul vehiculului controlat și, de asemenea, sunt furnizate informații cum ar fi viteza de deplasare, poziția autovehiculului pe calea de rulare, distanța dintre participanții la trafic sau panta drumului. Aceste date pot fi stocate și folosite ulterior pentru etapele de antrenare și validare ale algoritmilor dezvoltați.

## 3.1 Noțiuni introductive

În cazul unui vehicul autonom, scopul unui algoritm de percepție este de a furniza informațiile necesare controlului acestuia. Pentru a se lua cele mai bune decizii este foarte important ca obiectele statice sau dinamice din jurul vehiculului controlat să fie detectate și clasificate. De asemenea, în cazul obiectelor dinamice, este util să se anticipeze poziția viitoare a acestora în planul de acțiune.

În momentul realizării cercetărilor privind aplicabilitatea algoritmilor implementați pentru detecția și recunoașterea obiectelor în domeniul autovehiculelor autonome s-a constatat lipsa unei platforme software care să permită antrenarea și validarea algoritmilor menționați anterior în timp-real, acesta fiind motivul principal pentru care a fost realizat studiul publicat prin lucrarea [53]. În cadrul platformei dezvoltate au fost cercetate și optimizate conceptele introduse prin lucrarea [18].

Obiectivul lucrării [53] a fost furnizarea către comunitatea științifică a unui framework software unde avantajele și dezavantajele conceptelor noi de detecție de obiecte sau cele deja publicate pot fi testate și validate. Această platformă furnizează o interfață către mediul de simulare TORCS, în care metodele lansate pot fi evaluate. Pentru a se crea topologia rețelei neurale a fost utilizată biblioteca open-source Caffe [54].

În [18] a fost introdusă o rețea neurală utilizată pentru a genera propuneri de obiecte, denumită *Region Proposal Network* (RPN), rețea care partajează caracteristicile extrase dintr-o imagine de intrare cu o rețea neurală folosită pentru detecție. În acest mod timpul de calcul al propunerilor de obiecte a fost redus în mod semnificativ. Algoritmul Faster R-CNN [18] poate fi aplicat cu succes pe imaginile care provin din lumea reală, cu o distribuție spectrală naturală.

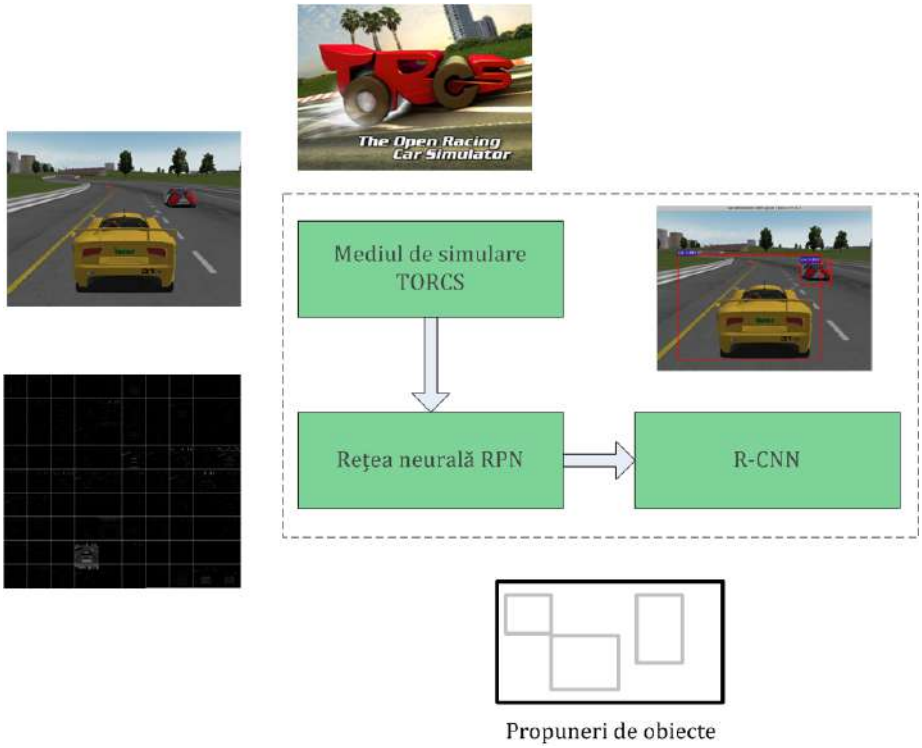



Figura 3.1: Diagramă-bloc a platformei software propusă pentru elaborarea, testarea și validarea algoritmilor de detecție a obiectelor.

Chiar dacă algoritmul Faster R-CNN poate să realizeze detecția de obiecte în timp-real, acesta a fost testat doar pe fluxuri de imagini, fără a fi integrat într-un mediu de simulare sau aplicație din lumea reală, unde avantajele și dezavantajele acestui concept pot fi puse în evidență.

Principalele contribuții care sunt evidențiate pe parcursul acestui capitol pot fi sumarizate după cum urmează:

- dezvoltarea unei platforme software care să ruleze în timp-real, platformă prin intermediul căreia pot fi antrenați, validați și testați algoritmi pentru detecția și recunoașterea obiectelor; acest framework este furnizat ca soluție open-source pentru a facilita dezvoltarea de algoritmi specifici percepției mediului înconjurător;
- îmbunătățirea timpului de execuție necesar algoritmilor de percepție prin paralelizarea sarcinilor.

## 3.2 Formularea problemei, metodologia de antrenare și optimizarea algoritmilor

În cadrul acestui capitol s-a efectuat o analiză privind aplicarea și validarea metodelor de machine learning pentru detecția obiectelor în mediul de simulare TORCS. O platformă software open-source, a cărei concept este prezentat în figura 3.1, a fost dezvoltată și furnizată comunității științifice în scopul de a se concepe și antrena algoritmi de tip deep learning pentru rezolvarea diverselor probleme din domeniul vehiculelor autonome. Sistemul propus se bazează pe arhitecturile de rețele neurale cunoscute ca RPN si R-CNN pentru o detecție robustă și eficientă a participanților la trafic.

### 3.2.1 Introducere în machine learning

Paradigma învățării automate (eng. *Machine Learning* - ML) permite abordarea sarcinilor de complexitate ridicată care nu pot fi indeplinite cu succes de algoritmii clasici. Instrumentele de machine learning oferă alternativa la rigiditatea instrumentelor programate, care odată instalate într-un sistem rămân neschimbate, chiar dacă multe sarcini se pot modifica în timp.

Metodele de ML se bazează pe căutarea unui model $f$, provenit dintr-o familie $\mathscr{F}$ de modele, ideal pentru aproximarea unor comportamente dorite. Acestea se pot baza atât pe modele parametrizate cât și pe modele non-parametrizate pentru a realiza maparea dorită. Un clasificator non-parametrizat este, de exemplu, algoritmul celui mai apropiat vecin (eng. *Nearest Neighbour*) [55], care utilizează ca date de intrare vecinii cunoscuți ai obiectului actual pentru a prezice clasa corespunzătoare a acestuia. De asemenea, pot fi menționate și modele, ca de exemplu, arborii de decizie sau rețelele neurale artificiale.

### 3.2.2 Metodologia utilizată pentru etapa de antrenare a algoritmului propus pentru detecția și recunoașterea obiectelor

Metoda pentru detecția obiectelor prezentată în acest capitol este o combinație dintre o rețea neurală de adâncime utilizată pentru propunerea de regiuni ale obiectelor (eng. *Region Proposal Network* RPN [18]) și conceptul de Fast-RCNN [17]. Un avantaj în utilizarea acestei arhitecturi este partajarea calculelor necesare între cele două rețele neurale RPN și Fast R-CNN folosindu-se un set comun de straturi de convoluție.

Antrenarea algoritmului pentru detecția obiectelor se realizează utilizându-se metoda propagării înapoi a erorii. Funcția de cost, denumită și funcția obiectiv, utilizată pentru a se minimiza eroarea calculată la ieșirea rețelei neurale este definită prin relația următoare:

$$L(\{p_i\},\{t_i\}) = \frac{1}{N_{cls}} \sum_i L_{cls}(p_i, p_i^*) + \lambda \frac{1}{N_{reg}} \sum_i p_i^* L_{reg}(t_i, t_i^*), \tag{3.1}$$

unde: $i$ este indexul ancorei; $p_i$ reprezintă probabilitatea ca o ancoră $i$ să fie un obiect sau nu; $p_i^*$ este eticheta observațiilor cunoscute care vor fi adresate de acum ca ground-truth, etichetă a cărei valoare este 1 dacă ancora este pozitivă și 0 dacă este negativă; $t_i$ este vectorul de coordonate parametrizat al cadrelor prezise; $t_i^*$ este cadrul utilizat ca ground-truth, care este asociat cu o ancoră cu eticheta egală cu 1; $L_{cls}$ este funcția de cost utilizată pentru clasificare (funcție de cost logaritmică pentru două clase, și anume obiect sau non obiect).

$L_{reg}$ este funcția de cost utilizată pentru regresie, fiind definită astfel:

$$L_{reg}(t_i, t_i^*) = R(t_i - t_i^*), \tag{3.2}$$

unde $R$ este așa-numita funcție obiectiv robustă, introdusă și definită în lucrarea [18].

Termenul $p_i^* L_{reg}$ subliniază faptul că funcția de cost pentru regresie este activată doar în cazul în care ancora este pozitivă ($p_i^* = 1$), fiind dezactivată în caz contrar (($p_i^* = 0$)).

Ieșirile straturilor de regresie și clasificare sunt normalizate utilizându-se $N_{reg}$ și $N_{cls}$. Parametrul $\lambda$ este folosit pentru a se balansa calculul ponderilor.

Pentru reglajul fin al detecției, procedeul de propagare înapoi al erorii este activat prin stratul de pooling ROI. Luându-se în considerare $x_i \in \mathscr{R}$ pentru activarea intrării $i$ din stratul ROI respectiv $y_{rj}$ ieșirea $j$ a stratului ROI notat cu $r$, acesta calculează $y_{rj} = x_{i^*(rj)}$, unde $i*(r,j) = argmax_{i' \in \mathscr{R}(ij)} x_{i'}$.

Funcția de propagare înapoi definită pentru stratul de pooling ROI calculează o derivată parțială a funcției de cost, luându-se în considerare fiecare variabilă de intrare $x_i$ și utilizând relația:

$$\frac{\partial L}{\partial x_i} = x_r x_j [i = i^*(r,j)] \frac{\partial L}{\partial y_{rj}}. \tag{3.3}$$

Derivata parțială $\partial L / \partial x_i$ este acumulată pentru fiecare ROI $r$ și pentru fiecare unitate de pooling de ieșire $y_{rj}$, dacă $i$ este $argmax$ selectat pentru $y_{rj}$.

Pentru antrenare s-au utilizat seturile de date open-source Pascal VOC2012 și Pascal VOC2007 [56, 57]. Imaginile din aceste seturi de date au fost scalate astfel încât dimensiunea minimă, fie pe lățime, fie pe înălțime să fie de 600 de pixeli. Pentru ancore, s-au folosit trei scalări, și anume $128^2$, $256^2$, $512^2$. Cadrele ancorelor care depășesc marginile imaginilor sunt ignorate în timpul antrenării, aceste informații nefiind utilizate în funcția de cost. De exemplu, pentru o imagine de dimensiune 1000×600, sunt în total 20 000 (60×40×9) de ancore. Prin ignorarea celor care depășesc marginile imaginii, mai rămân aproximativ 6 000 de ancore pentru antrenare.

Contribuția principală adusă prin acest studiu este proiectarea și dezvoltarea unui framework cu o arhitectură flexibilă și robustă, capabil să ruleze algoritmi de percepție a mediului înconjurător, în timp-real. Funcționare al sistemului utilizează mai multe fire de execuție în paralel.

S-a pornit de la metoda Faster-RCNN [18] pentru a se realiza detecția și recunoașterea de obiecte într-un mediu de simulare al curselor de mașini, denumit TORCS. În figura 3.1 este ilustrat modul în care mediul de simulare și algoritmul de detecția de obiecte sunt fuzionate într-un singur framework. A fost dezvoltat un mediu care rulează pe mai multe fire de execuție pentru a se reduce timpul necesar detecției și pentru a se înlătura posibilele întreruperi ce pot să apară în timpul rulării mediului de simulare.

Așa cum se poate observa în figura 3.1, o imagine cu o scenă de trafic este furnizată ca intrare într-o rețea neurală ce propune posibile regiuni ale obiectelor. Această rețea denumită RPN este antrenată pentru a genera propuneri de regiuni cât mai exacte, propuneri care sunt utilizate în procesul de detecție de către o rețea neurală bazată pe regiuni ale obiectelor, rețea denumită R-CNN.

Astfel, imaginea recepționată din TORCS este mai întâi convertită într-o imagine de tip OpenCV [58] de dimensiune 500×375 și apoi trimisă către RPN pentru procesare. Deoarece precesarea este realizată asincron, pe un fir de execuție separat, timpul necesar pentru generarea de propuneri de regiuni este redus, algoritmul de detecție fiind capabil să ruleze într-un mediu cu cerințe de execuție în timp-real. Odată ce propunerile de regiuni au fost generate, devin date de intrare în algoritmul de detecție de obiecte. Obiectele care au scorul de potrivire cu clasa etalon cel mai mare sunt detectate și trimise către mediul de simulare pentru a fi marcate pe imagine.

Topologia rețelei neurale este una complexă și necesită o putere de calcul semnificativă. Stratul de intrare, care aparține rețelei RPN, este urmat de un număr de cinci straturi de convoluție. Primele două straturi de convoluție sunt urmate de straturi de pooling care au ca scop reducerea dimensiunii hărților de caracteristici rezultate. Numărul de filtre kernel aplicate straturilor de convoluție este mai mare cu cât se avansează în adâncimea rețelei neurale. Dacă pentru primul strat de convoluție este setat un număr de 96 de filtre, acesta ajunge la 512 pentru al patrulea strat de convoluție, scăzând apoi la 256 pentru ultimul strat convolutiv. Prima parte a arhitecturii formează rețeaua RPN, ale cărei rezultate sunt trimise către un strat personalizat, dezvoltat în Python, care preia propunerile de regiuni și le transmite mai departe către stratul de pooling ROI. Acestea sunt redimensionate, și procesate rând pe rând pentru a se determina care este scorul cel mai mare de potrivire pentru o clasă anume.

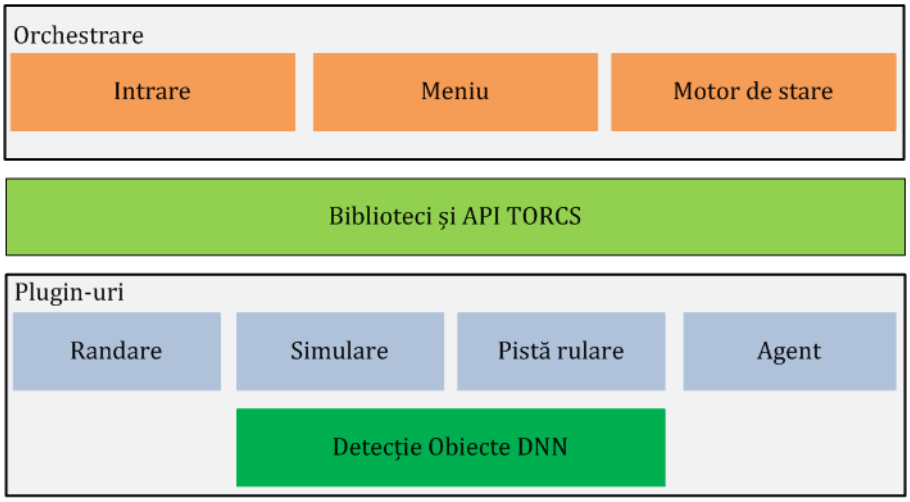



Figura 3.2: Arhitectura generală a mediului de simulare TORCS. Modulul "Detecție Obiecte DNN" a fost dezvoltat în cadrul acestui studiu pentru a realiza detecția și recunoașterea obiectelor.

## 3.3 Rezultate experimentale

Pentru evaluarea framework-ului dezvoltat pentru percepția mediului înconjurător, s-au folosit date de evaluare achiziționate din mediul de simulare TORCS. Acest mediu de simulare a fost utilizat pentru a se măsura performanța algoritmului de detecție de obiecte. Pentru a se obține acest lucru a fost dezvoltată o aplicație capabilă să ruleze pe mai multe fire de execuție, fiind utilizată pentru măsurarea performanței atât în cazul folosirii procesorului grafic, cât și în cazul folosirii unității centrale de procesare.

### 3.3.1 Integrarea mediului de simulare TORCS

În cadrul acestei teze, așa cum a fost menționat a priori, s-a utilizat un mediu de simulare al curselor de mașini, denumit TORCS. Aceasta s-a bucurat de o popularitate ridicată în comunitatea științifică, fiind o platformă portabilă, open-source și cu acces facil la resursele și datele despre mașina simulată și mediul în care se desfășoară cursa.

Un alt avantaj al acestui simulator este capabilitatea de a rula pe o varietate mare de sisteme de operare: rulează pe Linux (toate arhitecturile, pe 32 sau 64 de biți, configurație little sau big endian), FreeBSD, OpenSolaris, MacOSX și Windows (32 sau 64 biți).

Din nucleul jocului se pot colecta imagini și, de asemenea, indicatori critici pentru controlul mașinii, cum ar fi viteza acesteia, poziția relativă față de centrul drumului sau distanța până la mașina din față.

Metodele bazate pe rețelele neurale de adâncime pot fi integrate în TORCS, prin intermediul unui plugin. Așa cum se poate observa în figura 3.2, *Detecție Obiecte DNN* este plugin-ul creat și introdus în arhitectura generală a simulatorului. Se pot evidenția o serie de avantaje care decurg din utilizarea acestei platforme pentru algoritmi de tip machine learning, dintre care se pot menționa: marea varietate de tipuri de obiecte care pot fi detectate, un mod facil de a se construi noi circuite care să acopere toate cerințele necesare unor situații speciale sau posibilitatea de a se dezvolta noi agenți, bazați inclusiv pe tehnici de inteligență artificială, care pot fi antrenați pentru îndeplinirea a numeroase sarcini, cum ar fi percepția mediului sau planificarea traiectoriei. Un alt avantaj important este accesul la datele de antrenare și validare prin intermediul sistemului modular bazat pe plugin-uri.

Figura 3.3: Exemple de detecție de obiecte în cadrul simulatorului TORCS.

### 3.3.2 Rezultatele obținute în urma validării și testării platformei dezvoltate

Pentru validarea algoritmului de detecție de obiecte și integrarea acestuia în platforma software dezvoltată, se propune utilizarea unui sistem ce rulează pe mai multe fire de execuție. Chiar dacă mediul de simulare TORCS este compatibil cu o varietate de sisteme de operare, în acest studiu s-a utilizat exclusiv Windows.

Scene de trafic reprezentând starea mediului, sunt captate din TORCS și trimise spre a fi procesate, această operațiune efectuându-se într-un fir de execuție separat de cel al simulatorului, pentru a se reduce timpul necesar detecției și pentru a se fluidiza execuția aplicației TORCS. Metoda raportată în lucrarea Faster-RCNN [18] a fost preluată și, de asemenea, integrată și optimizată în platforma dezvoltată.

Procesul de validare s-a realizat prin crearea unei interfețe de comunicare între simulator și plugin-ul *Detecție Obiecte DNN*. Această interfață a fost dezvoltată utilizându-se Python Embedded, care permite o legătură facilă între scripturile dezvoltate în Python și bibliotecile C++.

Arhitectura rețelei neurale utilizată pentru detecția de obiecte descrisă în secțiunea 3.2.2. Pentru antrenarea algoritmului s-a utilizat baza de date open-source Pascal VOC2012. Acest set de date a fost prelucrat astfel încât să fie extrase doar clasele de interes pentru aplicația dezvoltată.

Pentru a se minimiza eroarea, antrenarea a fost realizată pe parcusul a 150 de epoci, cu dimensiunea unui lot de eșantioane setată la valoarea de 4, datorită memoriei reduse a procesorului grafic. Antrenarea s-a desfășurat pe parcursul a aproximativ 30 de ore, fiecare epocă având o durată de aproximativ 700 de secunde, fiind realizată pe GPU.

Odată modelul antrenat, acesta a fost testat prin rularea în TORCS, așa cum poate fi observat în figura 3.3, în care se prezintă exemple de detecții ale participanților la cursa de mașini. De asemenea, atașat acestei lucrări este și un clip video, în care este exemplificată modalitatea de detecție a obiectelor.

Deoarece platforma dezvoltată poate fi configurată să ruleze atât în modul CPU, cât și în modul GPU, a fost realizată o analiză a performanțelor măsurate în urma utilizării celor două configurații. A fost calculat așadar intervalul de timp necesar pentru detecția de obiecte din imagine pentru care a fost generat un anumit număr de propuneri de obiecte. Numărul maxim de propuneri generate este setat la valoarea de 300. Acest interval a fost calculat în interiorul simulatorului și include timpul necesar pentru achiziția unei imagini și afișarea cadrelor de încadrare pe imaginea analizată.

Așa cum este afișat în figura 3.4, utilizarea unității centrale de procesare nu reprezintă o soluție optimă pentru a rula rețele neurale și pentru calculele complexe generate de acestea. Intervalul de timp mediu obținut în timpul testelor a fost de 10 secunde, nepermis de mare. Desigur, timpul de procesare este direct proporțional cu numărul de propuneri de obiecte procesate. Atunci când numărul acestora este scăzut, timpul necesar detecției este, desigur, mai redus. În figura 3.4 se poate observa o diferență semnificativă de performanță dintre cele două configurații. Prin utilizarea unui GPU, fie și unul cu performanțe scăzute, se reduce timpul necesar procesării, în medie, de 15 ori. În cazul descris, timpul necesar este în medie de 0.51 secunde. Un alt factor important este acela că numărul de propuneri de regiuni nu influențează în mod semnificativ timpul de procesare atunci când

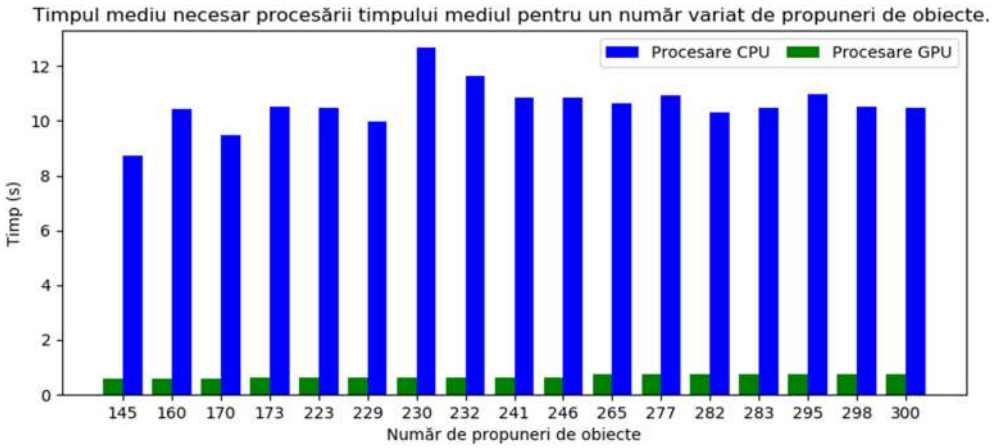



Figura 3.4: Timpul necesar procesării unui anumit număr de propuneri de regiuni din imagini captate în TORCS, utilizându-se ambele configurații: GPU și CPU.

se rulează pe GPU, lucru foarte important pentru o aplicație care trebuie să ruleze în timp-real și să furnizeze răspunsuri la perioade constante de timp.

În figura 3.5 este reprezentată diferența dintre evoluția timpului de procesare înregistrat pentru 100 de imagini consecutive, pentru cele două configurații utilizate. Timpul de procesare utilizându-se CPU-ul este considerabil mai mare. De asemenea, se observă că timpul de procesare variază semnificativ de la o imagine la alta, chiar dacă acestea au dimensiuni egale. Pe de altă parte, evoluția în cazul utilizării unui GPU este mult mai constantă, fără a avea vârfuri semnificative. Vizualizând și acest grafic, se poate constata că utilizându-se o structură hardware cu performanțe reduse nu s-a reușit să se obțină o rulare în timp-real. Însă această limitare poate fi ușor eliminată prin înlocuirea procesorului grafic cu unul ce dispune de mai multă capacitate de memorare și de capabilitați de calcul sporite.

Pentru a se măsura acuratețea algoritmului de detecție de obiecte s-a calculat metrica mAP (media preciziei medii), așa cum se poate observa în tabelul 3.1. mAP este o metrică populară utilizată pentru a se calcula precizia algoritmilor de detecție de obiecte. A priori, este necesar să se determine parametri de precizie și rata de regăsire (eng. *recall*).

Precizia măsoară acuratețea predicției ca, de exemplu, ce procent din predicții sunt corecte. Recall reprezintă abilitatea modelului de a găsi toate punctele de interes sau cazuri relevante. Altfel spus, recall măsoară cât de capabil este modelul de a deduce adevărat pozitivele (exemplare care aparțin clasei (pozitiv) fiind recunoscute ca aparținând clasei (adevăr)).

Pentru a se calcula mAP se utilizează așa-numita intersecție asupra reuniunii IoU (eng. *Intersection over Union*), ce reprezintă raportul dintre aria de uniune și aria de suprapunere:

$$IoU = \frac{\text{aria de suprapunere}}{\text{aria de uniune}}. \tag{3.4}$$

Metrica mAP se va calcula pentru IoU setat ca prag la valoarile 0.5 și 0.75, prin relația de mai jos:

$$mAP = \frac{\sum_{q=1}^{Q} AveP(q)}{Q}, \tag{3.5}$$

unde $Q$ reprezintă numărul de clase, iar $AveP()$ calculează precizia medie a eșantioanelor care au IoU mai mare față de valoarea de prag setată.

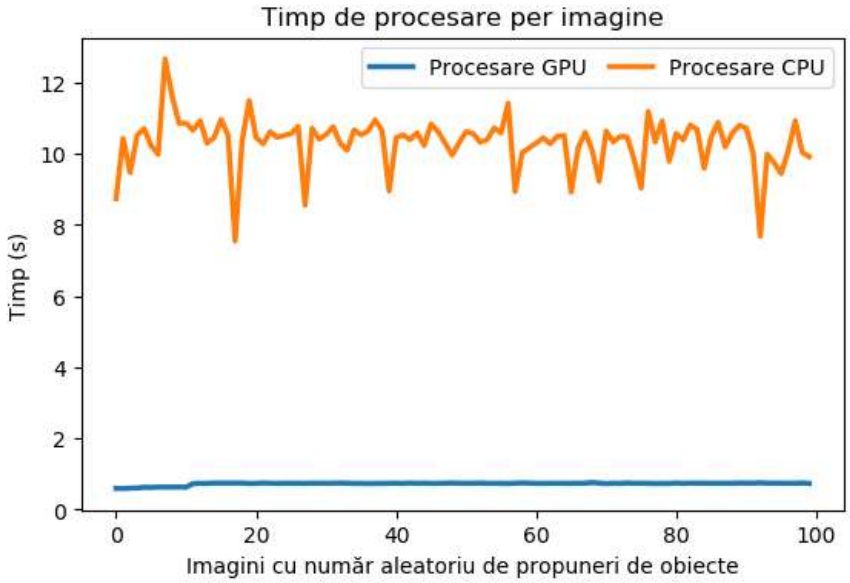



Figura 3.5: Comparație între timpii necesari de procesare pentru o serie de 100 de imagini consecutive captate în TORCS, utilizându-se ambele configurații: GPU și CPU.

Tabela 3.1: Calcul mAP pentru IoU setat la valoarea 0.5, respectiv 0.75.

| Metodă | Bază de date utilizată pentru antrenare | $mAP_{50}$ | $mAP_{75}$ | FPS |
|---|---|---|---|---|
| Faster RCNN în TORCS | Pascal VOC 2012 | 56.2% | 33.2% | 2 |

## 3.4 Concluzii

Platforma software open-source introdusă prin acest studiu poate fi utilizată pentru a se valida diferite tipuri de rețele neurale dezvoltate în biblioteca Caffe. Chiar dacă activitatea desfășurată s-a bazat pe algoritmi de detecție de obiecte aplicați pe imaginile captate din TORCS, această platformă poate fi utilizată și pentru alte tipuri de sarcini, cum ar fi cele de clasificare sau planificare de traiectorii, doar prin simpla modificare a topologiei rețelei neurale și prin adăugarea unui nou modul pentru interpretarea rezultatelor de ieșire.

Datorită existenței posibilității de a se utiliza procesarea în paralel prin integrarea limbajului CUDA, este asigurată o rulare optimă pe un procesor grafic ce dispune mai multe nuclee CUDA și o memorie de capacitate extinsă.

Concluzia este că această platformă poate fi utilizată de comunitatea științifică pentru studiul metodelor și algoritmilor de machine learning, deoarece furnizează un framework complet, ce poate rula în sistemul de operare Windows și se poate utiliza pentru dezvoltare, testare și validare. Deoarece librăria Caffe a fost dezvoltată pentru a fi utilizată în Linux, compilarea și rularea acesteia în Windows a presupus foarte multă muncă de configurare, integrare și recompilare dependențe. Un alt avantaj important este faptul că dezvoltând modulul de detecție de obiecte ca un plugin, s-a demonstrat că un modul extern, personalizat, poate fi atașat cu ușurință acestei platforme, accesul la date fiind facil.

Performanța acestui framework poate fi îmbunătățită prin rescrierea straturilor dezvoltate în Python în limbajul C++, sau prin utilizarea altor metode pentru detecția de obiecte, cum ar fi cele de tip single stage, ca de exemplu Yolo [15, 16].

# 4. Detectarea în timp-real a scenelor de trafic, aplicată domeniului de conducere autonomă

În acest capitol se prezintă una dintre cele mai importante și cuprinzătoare cercetări efectuate pe parcursul studiilor doctorale. Deoarece percepția mediului înconjurător realizată pe baza imaginilor achiziționate de la senzorii video montați pe automobile poate avea numeroase dezavantaje, în acest capitol s-a introdus o metodă de percepție care clasifică scenele de trafic prin utilizarea unor hărți de ocupanță, construite pe baza semnalelor primite de la o serie de senzori precum LIDAR sau RADAR.

Se introduce algoritmul *Deep Grid Net* (DGN) [59], un algoritm de tip deep learning proiectat pentru a se înțelege contextul în care se deplasează un autovehicul. DGN încorporează o reprezentare învățată a mediului, bazată pe griduri de ocupanță obținute din date brute provenite de la un senzor LIDAR și sunt construite pe baza teoriei dezvoltate de Dempster-Shafer (DS) [60]. Datele au fost obținute înregistrându-se secvențe din lumea reală cu ajutorul unui autovehicul echipat pentru această sarcină.

Sistemul proiectat este capabil să previzioneze în timp-real dacă un vehicul se deplasează pe o autostradă, un drum județean, în interiorul unui oraș, într-o parcare sau este prins într-un ambuteiaj. Contextul prezis este apoi utilizat ca intrare într-un algoritm responsabil cu schimbarea strategiilor de condus, algoritm implementat în EB Robinos [61], platforma software pentru conducerea autonomă (eng. *Autonomous Driving* (AD)) dezvoltată de compania Elektrobit.

Se propune o abordare bazată pe neuro-evoluție pentru a se căuta hiperparametri optimi ai algoritmului DGN. Algoritmii genetici (AG) au fost selectați datorită capacității demonstrate de a evolua rețele neurale de adâncime cu o acuratețe sporită și o viteză de procesare superioară. Performanța algoritmului dezvoltat a fost comparată cu aceea obținută cu o serie de clasificatori similari.

## 4.1 Noțiuni introductive

Așa cum a fost specificat anterior, o cerință esențială pentru vehiculele autonome este abilitatea de a percepe și întelege în timp-real mediul în care se află vehiculului pentru a se instrumenta un răspuns corespunzător. Rețelele neurale au devenit cea mai viabilă soluție pentru realizarea percepției datorită capacității de generalizare și adaptare la medii noi.

Contextul în care un vehicul se deplasează reprezintă o informație extrem de importantă pentru un sistem HAD, pentru a se aplica strategia de conducere ideală în cazul în care un autovehicul se află în interiorul unui oraș, față de strategia necesară, de exemplu, în cazul deplasării pe un drum județean. Sunt multiple scenarii în care clasificarea contextului de trafic poate fi utilă, unul dintre acestea fiind momentul în care semnalul GPS lipsește sau precizia nu este satisfăcătoare.

În acest capitol sunt dezvoltate cercetările prezentate în lucrările [59] și [62] unde algoritmul DGN, ilustrat în figura 4.1, a fost introdus. Un video care prezintă funcționalitatea DGN poate fi accesat utilizându-se acest link. Se vor adăuga noi considerații teoretice și se va explica în detaliu metodologia

utilizată.

Algoritmul DGN prezice contextul în care se află un vehicul analizând griduri de ocupanță locale, construite prin fuziunea datelor senzoriale neprocesate. Gridurile de ocupanță sunt preferate imaginilor datorită spațiului de căutare foarte redus determinat de informația reprezentată minimal. Aceste griduri descriu mediul de deplasare prin celule distribuite spațial care pot fi libere sau ocupate, cum va fi explicat mai în detaliu pe parcursul acestui capitol.

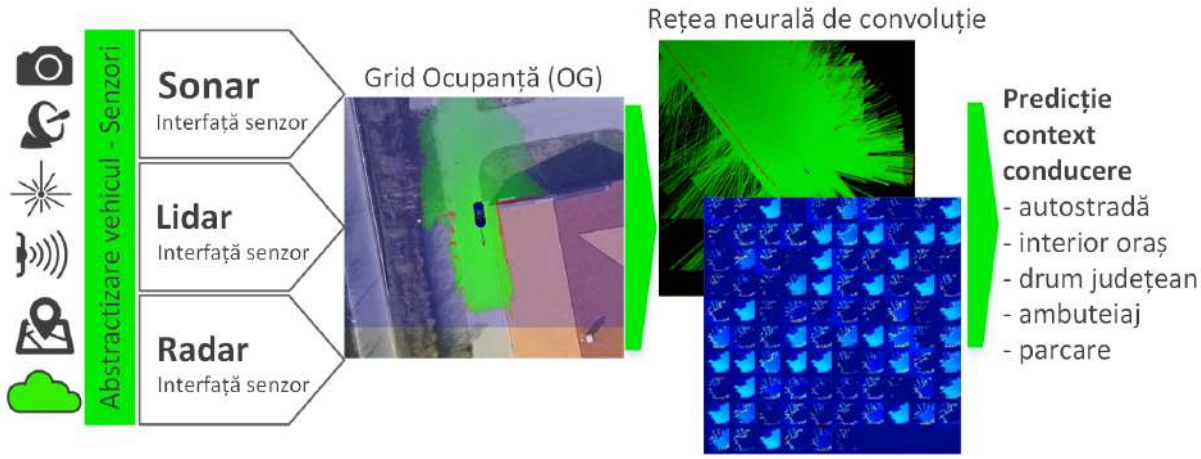



Figura 4.1: Arhitectura DGN [59]. Fluxul de date de la senzorii LIDAR este convertit în OG, care sunt mai departe procesate de o rețea neurală de convoluție. Ultimul strat al acestei rețele furnizează o ieșire probabilistică, și anume, probabilitatea ca vehiculul să se deplaseze pe autostradă, în interiorul orașului, pe drumurile județene, în parcare sau în arii cu ambuteiaje.

Algoritmul DGN este integrat în platforma software HAD dezvoltată de compania Elektrobit, cunoscută sunt numele de *EB Robinos*. DGN oferă estimare în timp-real a contextului de trafic în care se află un autovehicul, mapat la cinci clase principale: deplasarea pe autostradă, deplasarea în interiorul orașului, deplasarea pe drumurile județene, deplasarea în parcare și deplasarea în arii cu ambuteiaje. Această informație este utilizată de un algoritm de *Behavior Arbitration* pentru planificarea diferitelor strategii de conducere autonomă.

S-a utilizat informația provenită de la aceste griduri de ocupanță ca intrare într-o rețea neurală artificială de convoluție, astfel obținându-se o perspectivă *bird's-eye view* a scenei de trafic, așa cum se poate observa în figura 4.2. Algoritmul DGN poate să distingă între următoarele scenarii: autostradă (a), ambuteiaj (b), drum județean (c), parcări (d) sau conducere în interiorul orașului (e).

Principalele realizări care au fost aduse în urma cercetărilor efectuate în teză sunt sumarizate în cele ce urmează:

- introducerea și optimizarea arhitecturii DGN, care învață o reprezentare a scenelor de trafic sub forma unor griduri de ocupanță. Această arhitectură este una optimizată, care face posibilă utilizarea sa în aplicații de timp-real;
- ajustarea și selectarea hiperparametrilor utilizându-se concepul de neuro-evoluție. Prin utilizarea algoritmilor genetici s-a asigurat obținerea unei arhitecturi optime din punct de vedere cost-performanță, și, de asemenea, s-au selectat funcțiile de cost și de optimizare ideale pentru problema studiată;
- integrarea algoritmului DGN în platforma software destinată conducerii autonome EB Robinos și evaluarea acestuia utilizându-se înregistrări efectuate pe parcursul deplasărilor cu autovehiculul de test.

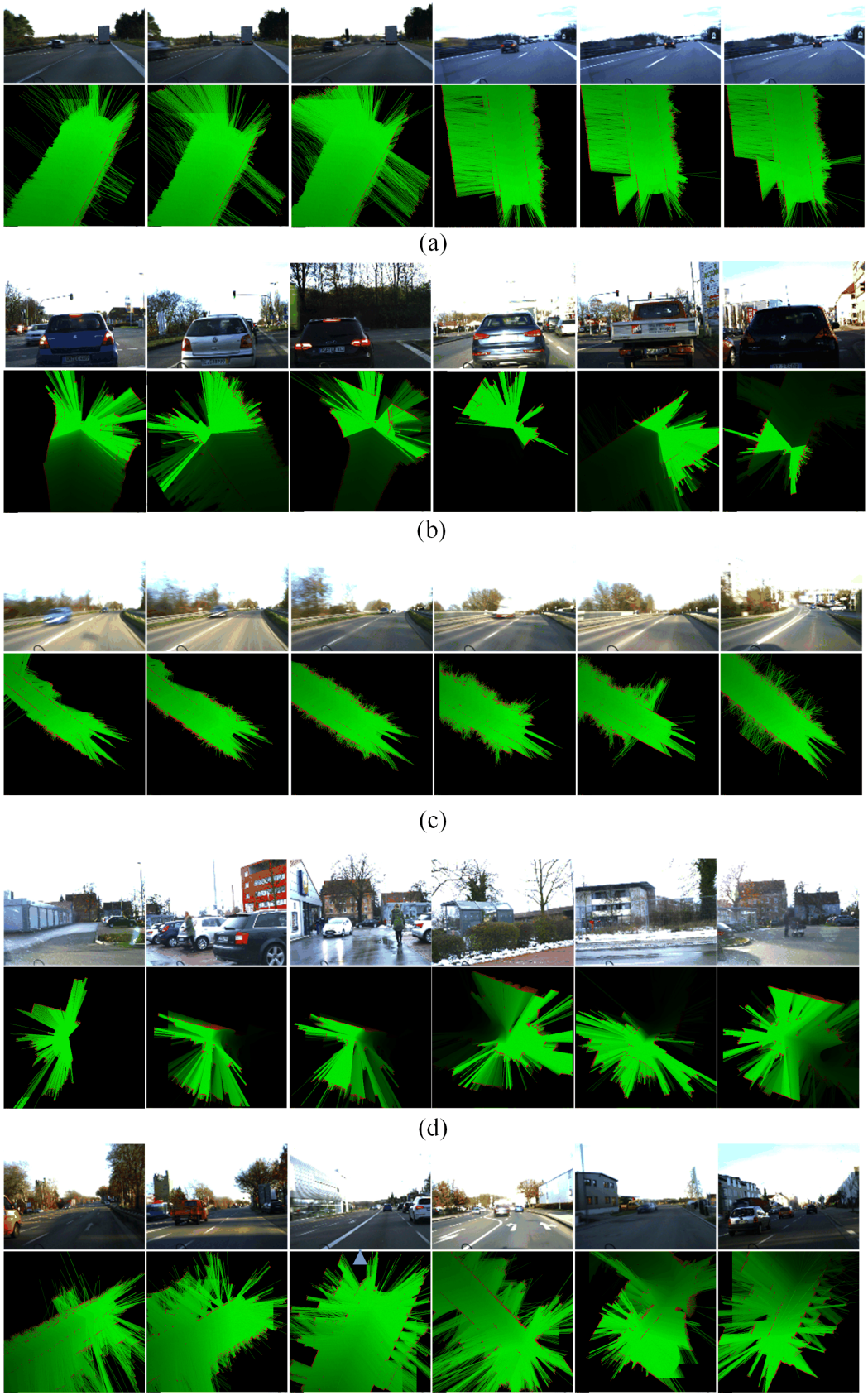



Figura 4.2: Exemple de griduri de ocupanță capturate din diverse scene de trafic.

## 4.2 Metodologia utilizată pentru obținerea modelului de clasificare

### 4.2.1 Contextul problemei de clasificare a scenariilor de trafic

Algoritmul proiectat pentru clasificarea scenelor de trafic, denumit *Deep Grid Net* (DGN), este compus din trei componente principale:

- un algoritm creat pentru fuziunea datelor într-un grid de ocupanță;
- o rețea neurală convolutivă de adâncime utilizată pentru analiza gridurilor în timp-real;
- un algoritm de neuro-evoluție utilizat pentru selectarea setului optim de hiperparametri ai rețelei.

Ieșirea furnizată de algoritmul DGN reprezintă o clasificare a contextului deplasării autovehiculului, mapat la 5 clase: *inner city* (interior oraș), *country road* (drum județean), *parking lot* (parcare), *highway* (autostradă) și *traffic jam* (ambuteiaj).

Setul de date ce conține griduri de ocupanță $D$ este utilizat pentru a se identifica ipoteza optimă $h_{DGN}$ a DGN, care codifică structura și ponderile rețelei neurale de adâncime. Ipoteza $h_{DGN}$ reprezintă capacitatea modelului de a determina una dintre situațiile de trafic în care se află un autovehicul, și anume: deplasarea pe autostradă, deplasarea în interiorul orașului, deplasarea pe drumurile județene, deplasarea în parcare și deplasarea în arii cu ambuteiaje. Contextul problemei este definit în următorul cadru Bayesian:

$$P(h|D) = \frac{P(D|h)P(h)}{P(D)}, \tag{4.1}$$

unde $P(h)$ este probabilitatea a priori asupra $h$, $P(D) = \int_h P(D|h)P(h)$ este probabilitatea datelor de antrenare, independentă de $h$ și $P(h|D)$ este probabilitatea, denumită în continuare *likelihood*, a $h$ pentru setul de date $D$ furnizat. $P(D|h)$ este likelihoodul datelor asupra unei ipoteze date.

Ipoteza MAP (eng. *maximum a posteriori*) $h_{MAP}$ este definită mai jos:

$$h_{MAP} = \underset{h \in \mathscr{H}}{\operatorname{argmax}} P(h|D). \tag{4.2}$$

Utilizându-se teorema lui Bayes, ecuația (v .ec. (4.2)) poate fi scrisă după cum urmează:

$$h_{MAP} = \underset{h \in \mathscr{H}}{\operatorname{argmax}} \frac{P(D|h)P(h)}{P(D)}, \tag{4.3}$$

unde $P(D)$ nu influențează problema maximizării, reducând ecuația (v. ec. (4.3)) la:

$$h_{MAP} = \underset{h \in \mathscr{H}}{\operatorname{argmax}} P(D|h)P(h). \tag{4.4}$$

Asumându-se faptul că toate ipotezele au același grad de probabilitate, se poate alege abordarea *Maximum Likelihood* (ML) pentru antrenarea rețelei neurale de adâncime:

$$h_{ML} = \underset{h \in \mathscr{H}}{\operatorname{argmax}} P(D|h) = \underset{h \in \mathscr{H}}{\operatorname{argmax}} L(h). \tag{4.5}$$

Eșantioanele de antrenare sunt considerate a fi distribuite în mod identic, acest lucru satisfăcând următoarea relație:

$$P(D|h) = \prod_{i=1}^{m} P(\langle x_i, y_i \rangle | h) = \prod_{i=1}^{m} P(y_i|x_i; h)P(x_i). \tag{4.6}$$

Maximizarea ecuației (4.6) este echivalentă cu maximizarea funcției logaritmice $\log L(h)$:

$$\log L(h) = \sum_{i=1}^{m} \log P(y_i|x_i; h) + \sum_{i=1}^{m} \log P(x_i). \tag{4.7}$$

Termenul $\sum_{i=1}^{m} \log P(x_i)$ depinde de setul de date $D$, dar nu și de ipoteza $h$. Așadar, poate fi ignorat în momentul căutării ipotezei optime a algoritmului DGN:

$$\log L(h_{DGN}) = \sum_{i=1}^{m} \log P(y_i|x_i; h). \tag{4.8}$$

### 4.2.2 Griduri de ocupanță

Gridurile de ocupanță reprezintă sursa de date pentru calcularea ipotezei optime $h_{DGN}$ a algoritmului DGN. OG sunt folosite adesea pentru percepția mediului înconjurător și navigarea vehiculelor autonome, aplicații care necesită tehnici pentru fuziune de date și evitarea obstacolelor. În cercetările realizate pe parcursul studiilor doctorale, gridurile utilizate pentru a se clasifica contextul în care un vehicul se deplasează au fost construite utilizându-se teoria lui *Dempster-Shafer* (DS), cunoscută și ca *Teoria Evidenței*, dezvoltată de Shafer în 1976 și sintetizată în lucrarea [63]. Această teorie este caracterizată de patru funcții: FoD (eng. *Frame of Discernment*), Bel (eng. *Belief*), BBA (eng. *Basic Belief Assignment*) și Pl (eng. *Plausibility*).

Un exemplu pedagogic al utilizării teoriei evidenței este ilustrat în figura 4.3, unde se poate observa cum, prin măsurători repetate, crește confidența ca o celulă să fie ocupată sau liberă, fiind marcată cu codul de culoare corespunzător. Inițial, la momentul de timp $t_0$, celula este mascată ca și spațiu liber (culoarea verde). În momentul în care mașina începe să se apropie de obstacol, credibilitatea de ocupanță a celulei respective crește, modificându-și în acest sens culoarea în roșu la momentul de timp $t_3$.

Fie $\Omega$ o funcție FoD finită discretă :

$$\Omega = \{w_1, w_2, w_3, \ldots, w_n\}, \tag{4.9}$$

unde $n > 1$ pentru problema de fuziune considerată și spațiul său de fuziune $G^{\Omega}$. $D^{\Omega}$ reprezintă setul hiper-putere, iar $S^{\Omega}$ setul de super-putere, în funcție de problema considerată [64]. O funcție *Basic Belief Assignment* (BBA) asociată cu setul de evidențe dat este definită ca maparea spațiului de fuziune la valorile din intervalul închis [0,1]:

$$m(.) : G^{\Omega} \rightarrow [0, 1], \tag{4.10}$$

satisfăcându-se relațiile:

$$m(\Phi) = 0, \quad \Sigma_{A \in G^{\Omega}} m(A) = 1, \tag{4.11}$$

unde $m(\Phi)$ este denumită *funcția masă de probabilitate*, $\Phi$ reprezintă mulțimea vidă, și $A$ este orice submulțime a lui $G^{\Omega}$. În consecință, bazat pe evidența existentă, o valoare numerică în intervalul

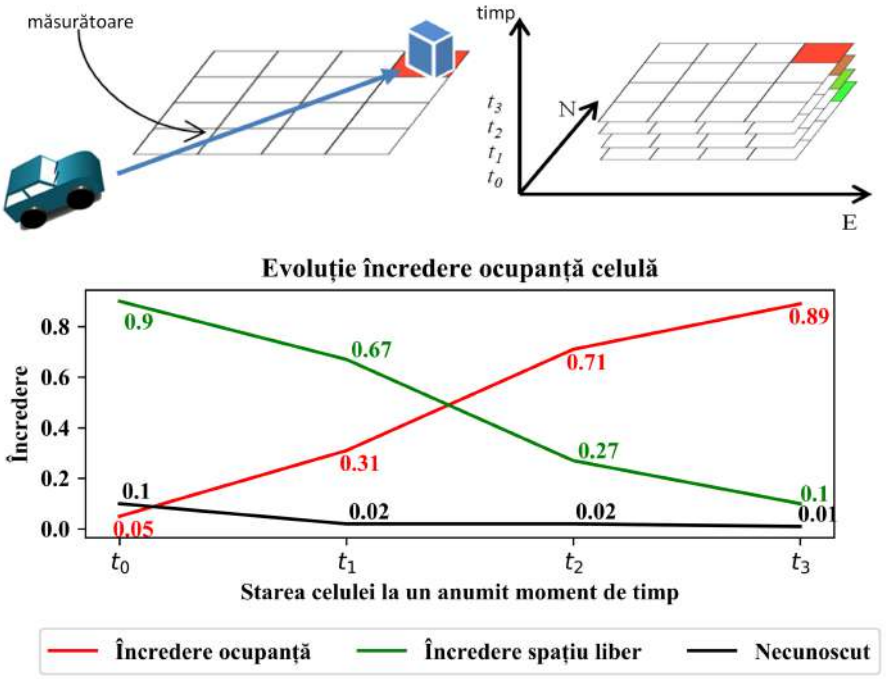



Figura 4.3: Exemplu pedagogic ilustrând comportamentul algoritmului DS. Mașina controlată întâlnește un obstacol când se deplasează în direcția nord-est (NE).

închis [0,1] este atribuită fiecărui element al setului $G^{\Omega}$ reprezentând probabilitatea asociată fiecărui element al mulțimii.

Setul de fuziune $G^{\Omega}$, denumit și set de puteri, este setul tuturor cadrelor de discernământ, acesta reprezentând toate ipotezele și toate uniunile posibile ale ipotezelor. Setul hiper-putere $D^{\Omega}$ și setul super-putere $S^{\Omega}$ sunt definite ca setul format de toate reuniunile și intersecțiile ipotezelor, respectiv setul format din toate reuniunile, intersecțiile și complementele ipotezelor [65].

Funcțiile credibilitate (eng. *Belief*) $Bel$ și plauzibilitate $Pl$ sunt definite după cum urmează:

$$Bel(A) = \sum_{\substack{B \subseteq A \\ B \in G^{\Omega}}} m(B), \quad Pl(A) = \sum_{\substack{B \cap A \neq \phi \\ B \in G^{\Omega}}} m(B) = 1 - Bel(\overline{A}). \tag{4.12}$$

Funcția $Bel(A)$ reprezintă suportul evidențial pentru ipoteza $A$, evidență care arată aceea că starea adevărată, de fapt, susține această ipoteză. Funcția $Pl(A)$ reprezintă suportul evidențial posibil pentru $A$. Intervalul $[Bel(A), Pl(A)]$ poate fi definit ca suportul total care sprijină ipoteza $A$, iar limitele acestui interval pot fi considerate ca limita minimă și maximă a probabilității cu care ipoteza $A$ este sprijinită prin evidențele considerate. $\overline{A}$ reprezintă complementul lui $A$.

Masa comună este calculată din seturile de masă $m_1(.)$ și $m_2(.)$. Combinația specifică regulii lui Dempster-Shafer este definită luându-se în considerare $m_{1,2}^{DS}(\Phi) = 0$ pentru toate $X \neq \Phi$:

$$m_{1,2}^{DS} \triangleq \frac{1}{1 - m_{1,2}(\phi)} \sum_{\substack{X_1, X_2 \in 2^{\Omega} \\ X_1 \cap X_2 \neq \phi}} \prod_{i=1}^{2} m_i(X_i), \tag{4.13}$$

unde $m_{1,2}(\phi)$ măsoară cantitatea de conflict dintre cele două seturi de masă. Termenul $1 - m_{1,2}(\phi)$ reprezintă constanta de normalizare. Gradul total de conflict dintre două surse de evidență este definit ca:

$$m_{1,2}(\phi) \triangleq \sum_{\substack{X_1, X_2 \in 2^{\Omega} \\ X_1 \cap X_2 = \phi}} \prod_{i=1}^{2} m_i(X_i). \tag{4.14}$$

În conformitate cu Shafer [63], regula combinării nu poate fi aplicată când cele două surse de evidență sunt într-un conflict total. Se spune că două surse sunt în conflict total dacă se respectă $m_{1,2}(\phi) = 1$. Detalii suplimentare legate de regula combinației dezvoltată de Dempster pot fi găsite în [60].

Gridurile de ocupanță furnizează o perspectivă de tipul *bird's-eye view* asupra scenei de trafic. Ideea de bază din spatele acestor hărți de ocupanță este divizarea mediului în celule bidimensionale, fiecare celulă reprezentând probabilitatea, sau credibilitatea, de ocupanță. Fiecare celulă are un cod de culoare, pixelii verzi reprezentând spațiul liber, cei roșii marcând celulele ocupate (care pot fi denumite și obstacole), iar modelele de culoare neagră sunt utilizate pentru a reprezenta zonele despre care nu există informație necesară pentru a se stabili dacă sunt sau nu ocupate. Intensitatea culorii reprezintă gradul de ocupanță. Cu cât culoarea verde, de exemplu, este de o intensitate mai mare, cu atât probabilitatea ca celula să fie liberă este mai mare.

În acest studiu, conținutul gridurilor se degradează în timp, prin reducerea graduală, în timp, a încrederii privind ocupanța fiecarei celule din grid. Conținutul gridurilor este actualizat continuu, în timp-real, cu fiecare ciclu de măsurare al senzorilor.

Gridurile de ocupanță utilizate pentru antrenare au fost etichetate manual prin inspecție vizuală a imaginilor provenite de la senzorul video. Exemple de etichetări pot fi vizualizate în figura 4.2.

## 4.3 Antrenarea algoritmului DGN

Pentru a se obține o cât mai mare acuratețe a modelului DGN, au fost căutate soluții inovative care să conducă la rezultate cât mai bune în ceea ce privește clasificarea scenelor de trafic. Deoarece setul de date disponibil pentru antrenare a fost limitat, utilizându-se un autovehicul echipat cu senzori de tip LIDAR, ultrasonici și video pentru achiziție, a fost necesară o tehnică prin care mai multe modele pot fi antrenate și evaluate cu setul de date avut la dispoziție și care să furnizeze cea mai bună variantă pentru problema studiată.

Cercetându-se literatura de specialitate, s-a constatat că au fost utilizate cu precădere două metode pentru obținerea unui model ideal al unei rețele neurale: neuro-evoluția și căutarea de tip grilă (metodă cunoscută sub numele de *Grid Search*). Aceste metode au fost utilizate pentru a se selecta cel mai bun set de hiperparametri pentru topologia algoritmului DGN. În cele ce urmează se vor prezenta detaliile de implementare ale metodei de neuro-evoluție.

### 4.3.1 Selectarea hiperparametrilor prin neuro-evoluție

Algoritmii genetici [66] reprezintă o metodă de optimizare metaheuristică, aparținând unei categorii mai largi de algoritmi evolutivi. Procesul de antrenare evolutiv începe de la un set inițial de soluții, denumit și populație, $\mathscr{P}$, unde fiecare soluție este dată de un set de proprietăți denumite gene. O soluție este cunoscută și sub numele de individ, care aparține populației, notat $h_{DGN} \in \mathscr{P}$.

Algoritmii genetici sunt utilizați în acest studiu pentru a se găsi setul optim de hiperparametri codificând $h^*_{DGN}$. Este utilizat același set ca în secțiunea **??**, și anume: numărul optim de neuroni din fiecare strat al rețelei, cel mai potrivit algoritm de optimizare, numărul și dimensiunea filtrelor Kernel utilizate în cadrul operației de convoluție și funcția de cost folosită în procesul de propagare înapoi a erorii.

**Algoritm 4.1** Procedura de antrenare a rețelei neurale adânci de convoluție Deep Grid Net, utilizându-se metodologia de neuro- evoluție, afișată în pseudo-cod.

**procedure** Antrenare($\mathscr{P}$)
  **for** $(\theta, h_{DGN}, t) \in \mathscr{P}$ **do**
    **while** nu este încheiată antrenarea **do**
      $\theta \leftarrow backprop(\theta | h_{DGN})$ ▷ antrenare utilizându-se propagarea înapoi a erorii pentru setul curent de hiperparametri în $h_{DGN}$
      $h_{DGN} \leftarrow eval(h_{DGN})$ ▷ evaluarea modelului curent
      $h_{DGN} \leftarrow explore(h_{DGN}, \mathscr{P})$ ▷ selectarea noului set de hiperparametri $p$
      actualizare $\mathscr{P}$ cu noii ($h_{DGN}$, $\theta$, $t+1$) ▷ actualizarea populație $\mathscr{P}$
    **end while**
  **end for**
  **return** top 5 $h_{DGN}$ indivizi în $\mathscr{P}$ ▷ returnează modelele cu cea mai mare valoare a funcție de fitness
**end procedure**

Prin utilizarea acestor algoritmi, se poate determina structura rețelei neurale cu un număr minim de neuroni care poate să livreze rezultate de o acuratețe sporită, structură care să poată permite utilizarea algoritmului DGN în aplicații ce trebuie să ruleze în timp-real [67]. A fost definită o funcție $eval$ pentru evaluarea unei funcții cunoscute în literatura de specialitate ca funcție de potrivire (eng. *fitness function*). Căutarea setului optim de parametri este reprezentată prin relația:

$$\theta^* = \underset{\theta \in \Theta}{\operatorname{argmax}}\, eval(\theta). \tag{4.15}$$

Metoda de antrenare propusă optimizează hiperparametri din spațiul soluțiilor, având ca scop calcularea indivizilor de top, $h_{DGN} \in \mathscr{P}$, bazându-se pe valoarea de fitness furnizată de aceștia, valoare care este definită aici ca acuratețea rețelei neurale:

$$h^*_{DGN} = \underset{h_{DGN} \in \mathscr{P}}{\operatorname{argmax}}\, eval(backprop(\theta | h_{DGN})). \tag{4.16}$$

De îndată ce antrenarea este completă, setul de hiperparametri este evaluat pe baza $h_{DGN}$ utilizându-se funcția de fitness $eval(\cdot)$. Noul set de hiperparametri este calculat explorându-se spațiul soluției cu ajutorul procedurii $explore(h_{DGN}, \mathscr{P})$. Bucla de training se oprește după 20 de epoci de antrenare și returnează top cinci indivizi, care au cea mai mare valoare de fitness. Pseudo-codul pentru metoda definită în acest studiu este afișat prin algoritmul 4.1.

## 4.4 Arhitectura modelului DGN

Deoarece scenele de trafic din cadrul acestei cercetări au fost reprezentate sub forma unor griduri de ocupanță, a fost definită o topologie de rețea neurală de convoluție care preia ca date de intrare aceste griduri, exportate ca imagini cu dimensiunea 128×128, ce au fost calculate utilizându-se metodologia definită în secțiunea 4.2.2.

Arhitectura algoritmului DGN este ilustrată în figura Fig. 4.4 și a fost dezvoltată cu scopul de a fi integrată în EB Robinos (v. 4.5.1), în cadrul căruia sunt necesare hărți de activare de dimensiuni reduse, și, de asemenea, o complexitate redusă a rețelei, pentru ca acest algoritm să fie compatibil cu cerințele unei aplicații ce rulează în timp-real, pe un hardware care dispune de un procesor grafic. Topologia rețelei neurale conține două straturi de convoluție, având ca funcție de activare ReLU

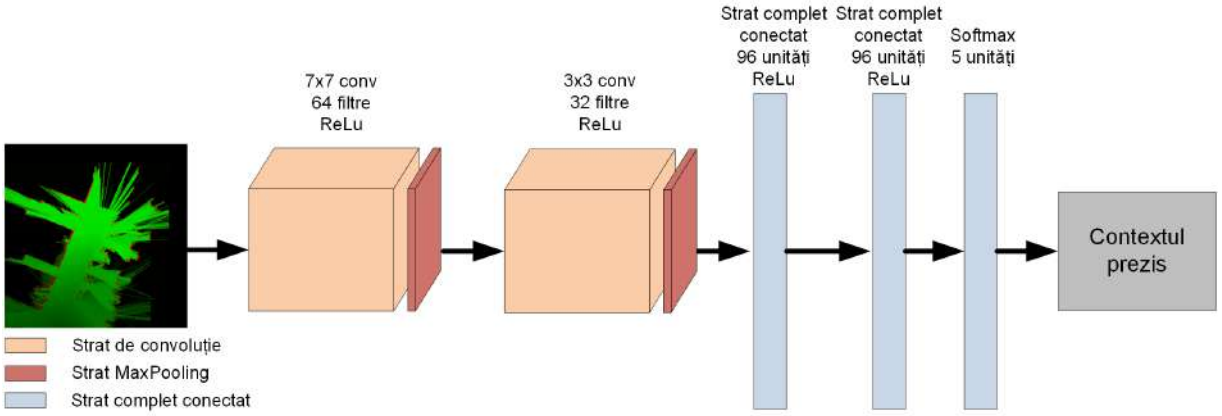



Figura 4.4: Arhitectură generală DGN. Conține două straturi de convoluție filtrate de funcții de activare ReLu. Operațiile de pooling și normalizare urmează fiecare strat de convoluție. Modelul conține trei straturi complet conectate ce sunt legate la o funcție de activare softmax.

(*Rectified Linear Unit*). Pentru determinarea numărului optim de filtre Kernel și dimensiunea acestora, s-au furnizat intervale de valori algoritmului de neuro-evoluție descris în sub-capitolul 4.3.1. Fiecare strat de convoluție este urmat de un strat de pooling și un strat de normalizare (eng. *Batch Normalization*). Rețeaua conține, de asemenea, și trei straturi complet conectate, ultimul fiind conectat la funcția de activare Softmax, care calculează probabilitatea ca scena de trafic să fie inclusă într-una din clasele definite.

## 4.5 Rezultate experimentale

În această secțiune a tezei de doctorat vor fi prezentate rezultatele obținute în urma integrării algoritmului DGN în platforma software dedicată conducerii autonome a autovehiculelor, EB Robinos [61].

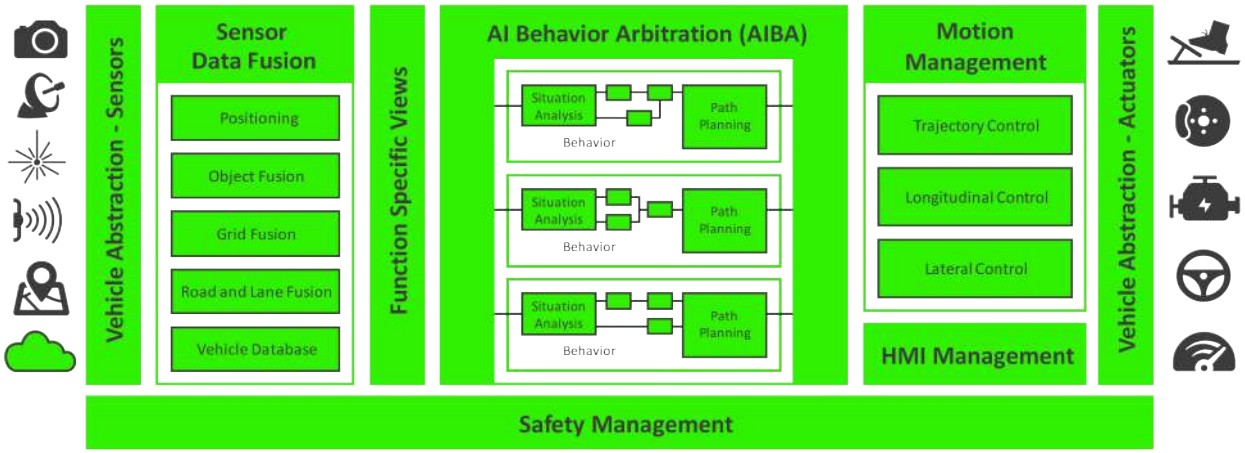



Figura 4.5: Arhitectura de referință a platformei software EB Robinos (www.elektrobit.com).

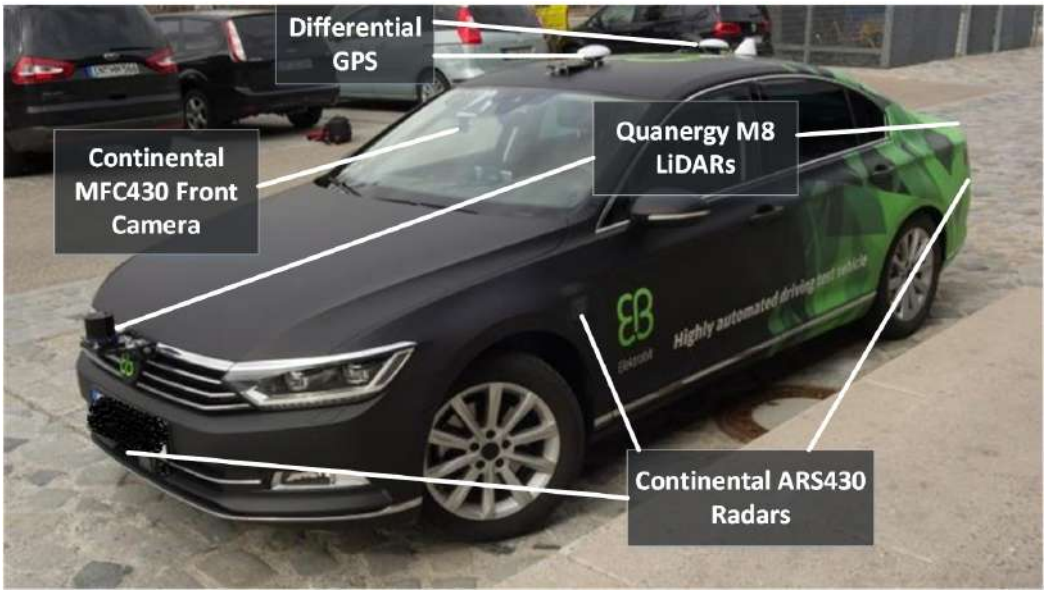



Figura 4.6: Autovehicul autonom [68] utilizat pentru achiziția de date.

### 4.5.1 EB Robinos

*EB Robinos* este o arhitectură software funcțională dezvoltată de Elektrobit Automotive, cu interfețe publice și module software care gestionează complexitatea sarcinilor de conducere autonomă. Arhitectura platformei EB Robinos, care este prezentată în figura 4.5, integrează componente ce urmăresc paradigma de detecție, planificare și acționare.

Prin dezvoltarea EB Robinos se dorește furnizarea unor module software care să aducă funcțiile de conducere autonomă în producția de serie. Serviciile furnizate prin această platformă activează capacitatea unui autovehicul de a avea o bună înțelegere a mediului în care se deplasează.

Algoritmul care stă la baza detecției contextului în care un vehicul se deplasează, DGN, este utilizat în EB Robinos pentru a furniza informațiile necesare selectării unei anumite strategii autonome de condus, în funcție de cerințele mediului respectiv.

### 4.5.2 Strategia de antrenare a algoritmului DGN

Setul de date utilizat pentru antrenare, testare și validare conține griduri de ocupanță, ce au fost generate din datele achiziționate de un vehicul care s-a deplasat atât în interiorul unui oraș, cât și în afara acestuia, pentru o varietate cât mai mare a scenariilor de trafic. Datele au fost colectate utilizându-se un vehicul de test (Volkswagen Passat), prezentat în figura 4.6. Acest autovehicul este echipat cu o cameră video frontală (Continental MFC430), cu doi senzori de tip LIDAR (Quanergy M8), unul amplasat în partea frontală și unul în partea din spate a vehicului, și senzori de tip RADAR (Continental ARS430) amplasați în partea din spate și pe părțile laterale.

Fluxul de date provenit de la senzori este fuzionat în griduri de ocupanță având o dimensiune de 128×128 și o rezoluție de 0.25 metri. Înregistrările au fost realizate în timpul zilei, pe o vreme cu precădere însorită, fără ploi, și includ condiții de trafic aglomerat cât și fluent.

Eșantioanele de date sunt salvate în timpul deplasării, la un timp de ciclu ce variază între 50 și 90 de milisecunde. S-au obținut astfel aproximativ 60 000 de eșantioane, conținând diferite tipuri de scenarii de trafic: deplasarea pe autostradă, deplasarea în interiorul orașului, deplasarea pe drumurile județene, deplasarea în parcare și deplasarea în arii cu ambuteiaje.

Structura rețelei neurale a fost determinată prin utilizarea algoritmului de neuro-evoluție descris în secțiunea 4.3.1. Hiperparametri care au fost furnizați ca intrare a algoritmului genetic, pentru a fi utilizați în cadrul procesului de antrenare, se regăsesc în lista de mai jos:

- **funcții de optimizare**: rmsprop, adam, Stochastic Gradient Descent (SGD), adagrad, adadelta, adamax, nadam;
- **funcții de cost**: categorical crossentropy, mean squared error;
- **număr neuroni**: 16, 32, 64, 128, 258;
- **număr filtre Kernel**: 16, 32, 64, 96;
- **dimensiune filtre Kernel**: 3, 5, 7, 9.

Evoluția funcției de potrivire este descrisă prin intermediul figurii 4.7. Algoritmul genetic utilizat pentru evoluția parametrilor modelului DGN a folosit un număr de 10 generații, fiecare generație având un număr de 20 de indivizi. Un individ reprezintă o instanță a rețelei neurale DGN, cu arhitectura definită în secțiunea 4.4, având ca hiperparametri setul selectat prin evoluția algoritmului genetic.

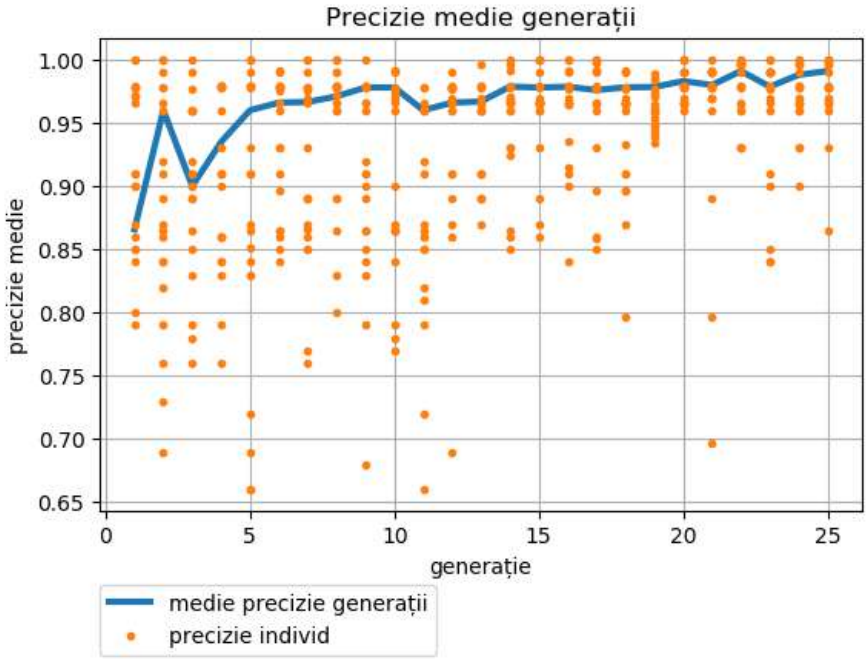



Figura 4.7: Evoluția funcției de potrivire în timpul procesului de antrenare [59].

O medie a acurateței clasificării este măsurată la finalul unui ciclu ce cuprinde o generație. În momentul în care antrenarea pentru ultima generație are loc, individul cu cel mai bun scor este selectat ca $h^*_{DGN}$. Prin setul extins de hiperparametri utilizat în acest studiu, modelul a atins acurateță foarte ridicată, de 99.1%, acurateță calculată prin funcția de potrivire. Arhitectura selectată pentru modelul final al algoritmului DGN conține 96, respectiv 64 de neuroni pentru primul și al doilea strat complet conectat, *nadam* ca optimizator pentru procesul de propagare înapoi a erorii, și *categorical crossentropy* ca funcție de cost. Numărul optim al filtrelor Kernel și dimensiunea acestora au fost determinate ca fiind 7×7×64 filtre pentru primul strat de convoluție și 3×3×32 filtre pentru cel de-al doilea strat de convoluție. Sunt utilizate și straturi de DropOut care au rolul de a reduce numărul de parametri de ieșire ale straturilor total complet conectate.

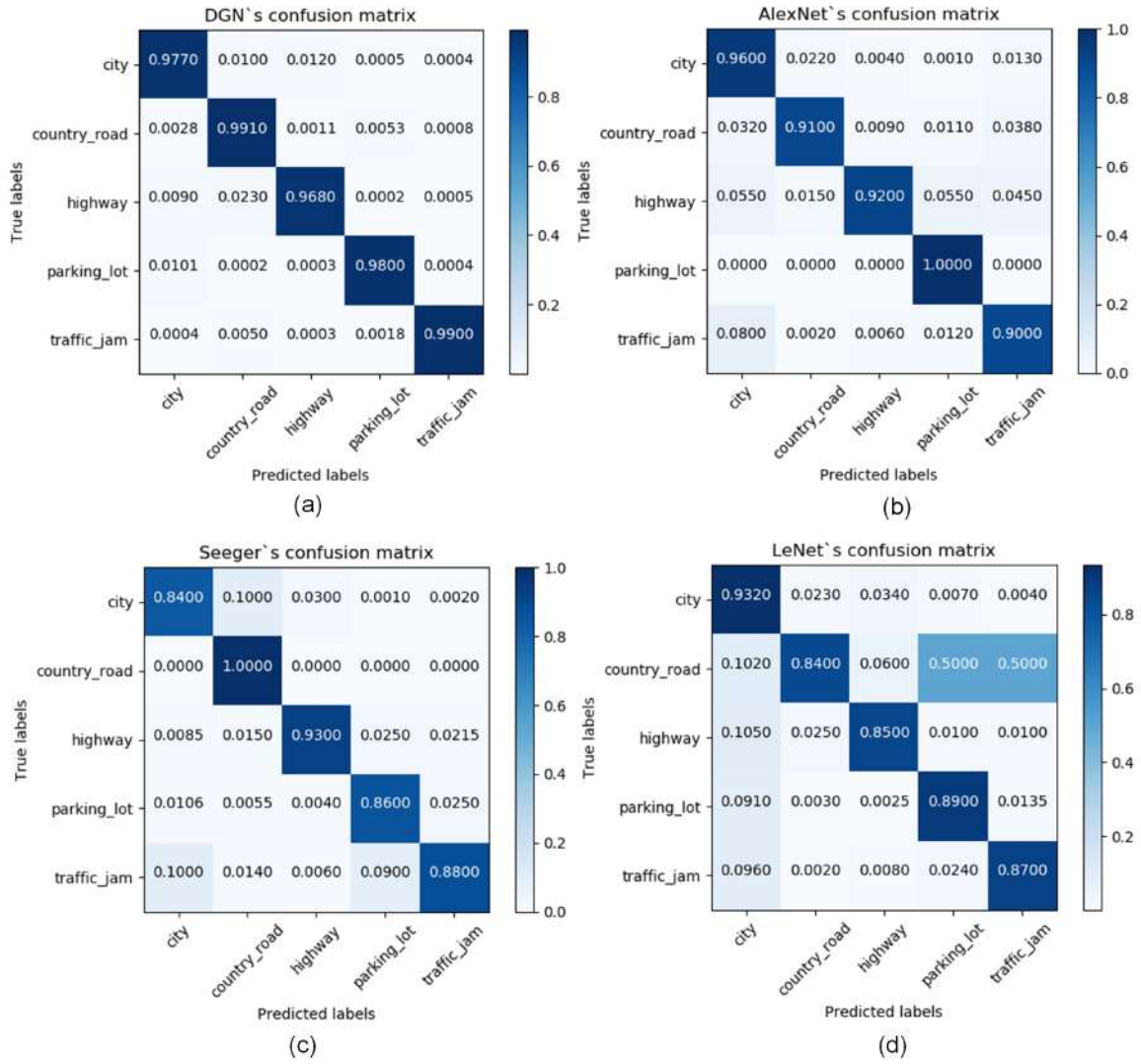



Figura 4.8: Matricele de confuzie generate pentru evaluarea performanței. În această figură sunt utilizate patru matrice de confuzie pentru a compara acuratețea obținută utilizându-se algoritmul DGN (a) cu acuratețea obținută prin evaluarea algoritmilor: (b) AlexNet [21], (c) Seeger [31] și (d) LeNet [69].

### 4.5.3 Evaluarea performanței

Performanța algoritmului de clasificare DGN este prezentată utilizându-se matricea de confuzie din Fig. 4.8 (a), unde se poate observa că acuratețea, calculată folosindu-se setul de date pentru testare, este foarte ridicată pentru fiecare dintre clasele învățate.

O comparație între rezultatele obținute cu algoritmul DGN și metodele state of the art este prezentată în tabelul 4.1. Competitorii principali sunt reprezentați de topologii de rețele neurale ca AlexNet [21], *LeNET* [69], sau GoogLeNet [70], și, de asemenea, algoritmul dezvoltat prin lucrarea [31], care este cel mai apropiat, ca și abordare, de metoda dezvoltată în acest studiu din perioada doctorală. Se poate observal că topologia DGN, chiar dacă are o complexitate mai scăzută, performează mai bine față de arhitecturi cu o complexitate mai ridicată.

Pentru o mai bună vizualizare a comparației performanțelor obținute pe setul de testare pentru algoritmii selectați ca fiind competitorii DGN, au fost generate matrice de confuzie. În figura 4.8 (b),

Tabela 4.1: Rezultatele comparației dintre performanța obținută pentru clasificarea contextului în care un autovehicul se deplasează.

| **Metodă** | **Acuratețe** | **Rata de regăsire** | **Precizie** | **F-measure** | **Specificitate** |
|---|---|---|---|---|---|
| LeNET | 0.88 | 0.91 | 0.93 | 0.92 | 0.86 |
| GoogLeNet | 0.94 | 0.96 | 0.97 | 0.97 | 0.97 |
| ResNet | 0.9 | 0.92 | 0.94 | 0.928 | 0.88 |
| AlexNet | 0.95 | 0.95 | 0.96 | 0.96 | 0.95 |
| Seeger | 0.91 | 0.9 | 0.92 | 0.92 | 0.93 |
| **DGN** | **0.98** | **0.988** | **0.983** | **0.984** | **0.95** |

Tabela 4.2: Comparație privind timpul de procesare al unei reprezentări a gridului de ocupanță sub formă de imagine.

| | LeNet | GoogLeNet | ResNet | AlexNet | Seeger | **DGN** |
|---|---|---|---|---|---|---|
| Timp (*milisecunde*) | 175 | 930 | 820 | 710 | 620 | **145** |

(c) și (d) se regăsesc matricele de confuzie pentru modelele AlexNet, LeNet și metoda Seeger [31] și se poate realiza o analiză a acestora. O observație interesantă este faptul că modelele AlexNet și Seeger au obținut un scor perfect pentru una dintre clase, ceea ce poate sugera că setul de testare creat nu a fost suficient de variat și competitiv pentru aceste modele care prezintă o complexitate mai mare în comparație cu DGN. Adițional, AlexNet furnizează o acuratețe ridicată pentru clasa *oraș*, acuratețe foarte apropiată de cea obținută prin DGN.

Viteza de execuție a algoritmului DGN a fost, de asemenea, comparată cu competitorii menționați anterior. În tabelul 4.2 este afișată o comparație a vitezei de procesare. Poate fi observat că timpul necesar procesării unei imagini în cazul DGN este de 145 milisecunde..

## 4.6 Algoritmul DGN: variantă de complexitate redusă

Scopul algoritmului DGN a fost de a se obține un model de complexitate redusă, care să conducă la performanțe ridicate pentru o problemă de clasificare multiplă. În acest scop s-a propus o metodă prin care numărul de parametrii ai rețelei să fie reduși prin utilizarea k-NN (eng. *k-Nearest Neighbor*) [71] o metodă non-parametrică de ML, ce nu necesită antrenare.

Utilizându-se algoritmii genetici pentru reglajul fin al ponderilor rețelei neurale, este exclusă procedura clasică de propagare înapoi a erorii. În cazul de față, k-NN este aplicat după straturile de convoluție, așa cum se poate observa în figura 4.9. Ponderile din straturile de convoluție sunt calculate prin neuro-evoluție într-un mod care respectă principiile de funcționare a k-NN.

Algoritmul genetic va crea automat caracteristici optimizate pentru k-NN. Principalii parametrii propuși pentru optimizare sunt valorile lui $k$ și nivelul, definit ca o poziție din stratul de pooling unde clasificatorul k-NN va fi introdus.

Rezultatele obținute utilizându-se această versiune de DGN au fost încurajatoare, fiind o opțiune de luat în considerare pentru industrializarea modelului. Un dezavantaj al acestei abordări este timpul mare necesar antrenării. Utilizarea algoritmilor genetici conduce la un timp mult mai mare de antrenare și la necesitatea unei capacități extinse de memorie.

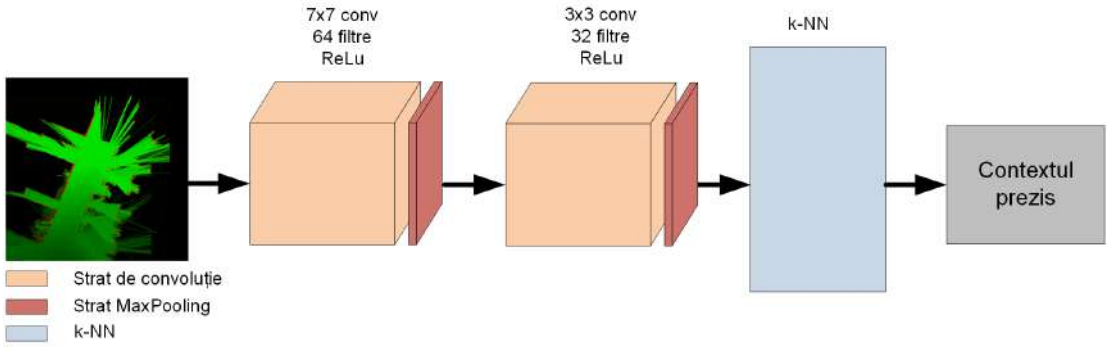



Figura 4.9: Arhitectură optimizată DGN. Conține două straturi de convoluție filtrate de funcții de activare ReLu. Operațiile de pooling și normalizare urmează fiecare strat de convoluție, urmate de un algoritm non-parametric k-NN.

## 4.7 Concluzii

În acest capitol s-a prezentat algoritmul *Deep Grid Net* (DGN), o soluție pentru detectarea contextului în care se deplasează un autovehicul, informație necesară componentelor de arbitrare a comportamentului din sistemele HAD. Acest algoritm a fost proiectat pentru a clasifica tipul scenei de trafic direct dintr-un grid de ocupanță obținut prin date provenite de la senzorii de tip LIDAR, în contrast cu majoritatea metodelor deja existente care sunt bazate pe procesarea de imagini provenite de la camerele video montate pe vehicule. Prin implementarea acestui algoritm s-a demonstrat că o topologie simplificată a unei rețele neurale de convoluție este suficientă pentru a clasifica, în timp-real, diferite tipuri de griduri de ocupanță.

Pentru validarea metodei dezvoltate, algoritmul DGN a fost integrat în platforma software dedicată conducerii autonome a autovehiculelor EB Robinos [61].

Au fost considerate mai multe posibile îmbunătățiri care pot fi aduse algoritmului DGN pentru a crește performanța, stabilitatea și viteza de execuție. Una dintre îmbunătățirile care poate fi adusă fără a modifica structura rețelei este dezvoltarea setului de date utilizat pentru antrenare, atât prin creșterea numărului de eșantioane cât și prin creșterea numărului de clase ce reprezintă scenele de trafic. Noi eșantioane pot fi generate utilizându-se un proces automat de generare a datelor sintetice, prin utilizarea algoritmului GOL, introdus prin lucrarea [72]. O altă strategie de îmbunătățire poate fi reprezentată de fuzionarea informațiilor provenite de la camerele video cu gridurile de ocupanță deja existente.

Diseminarea rezultatelor obținute a fost realizată prin participarea în cadrul conferinței internaționale IRC 2019 în lucrarea [59] și prin publicarea în jurnalul *International Journal of Robotic Computing* (IJRC) a lucrării [62].

De asemenea, rezultatele și inovația acestui studiu au condus la două propuneri de brevete internaționale [73, 74, 75].

# 5. Realizarea unui framework pentru generarea și industrializarea instrumentelor destinate conducerii autonome

Devine din ce în ce mai evident faptul că inteligența artificială și știința datelor (eng. *data science*) sunt tehnologii cheie pentru viitorul industriei constructoare de autovehicule. Deși succesul tehnicilor de inteligență artificială este semnificativ, prezintă totuși limitări în ceea ce privește acuratețea și industrializarea codului pentru producția în masă și pentru instalarea algoritmilor în hardware-ul existent în autovehicule. Deoarece industria constructoare de autovehicule este un domeniu în care sunt respectate o multitudine de standarde de calitate și siguranță, aplicarea algoritmilor de inteligență artificială devine dificilă deoarece în multe cazuri rețelele neurale funcționează ca o cutie neagră, fiind disponibile doar informațiile de intrare și cele de ieșire.

În acest capitol se propune o platformă unică pentru a se genera o serie de soluții bazate pe algoritmi de IA, utilizându-se un cadru de dezvoltare standardizat și folosit în momentul actual de producători de autovehicule pentru a se crea și valida funcții de conducere autonomă. Utilizându-se librării standard pentru implementarea algoritmilor de tip ML, împreună cu o platformă software independentă de sistemul de operare, se dorește soluționarea problemei integrării sistemului de detectare a contextului în care un autovehicul se deplasează, într-un mediu embedded și validarea funcționării conforme a acestuia [76]. În acest scop se va folosi algoritmul DGN prezentat pe larg în capitolul 4.

## 5.1 Noțiuni introductive

Chiar dacă succesul IA în industria constructoare de autovehicule este semnificativ, merită evidențiate și o serie de limitări . Una dintre criticile majore ale sistemelor de IA este aceea că de cele mai multe ori acestea sunt percepute ca o cutie neagră [77], care învață relația dintre variabilele de intrare și cele de ieșire, bazându-se pe un set de date de antrenare. O platformă robustă care permite vizualizarea caracteristicilor în diferite etape ale învățării modelelor deep learning poate să îmbunătățească aceste aspecte.

În lucrarea [76] se prezintă o analiză a rolului IA în industria constructoare de autovehicule și se subliniază necesitatea unei platforme software suficient de robuste pentru a permite dezvoltarea unor astfel de sisteme.

În acest capitol s-au avut în vedere:

- integrarea unei variante simplificate a algoritmului DGN [59] într-o platformă hardware care permite validarea acestuia;
- prezentarea conceptului de învățare prin utilizarea unui singur eșantion ( eng. *generative one-shot learning*) [72];

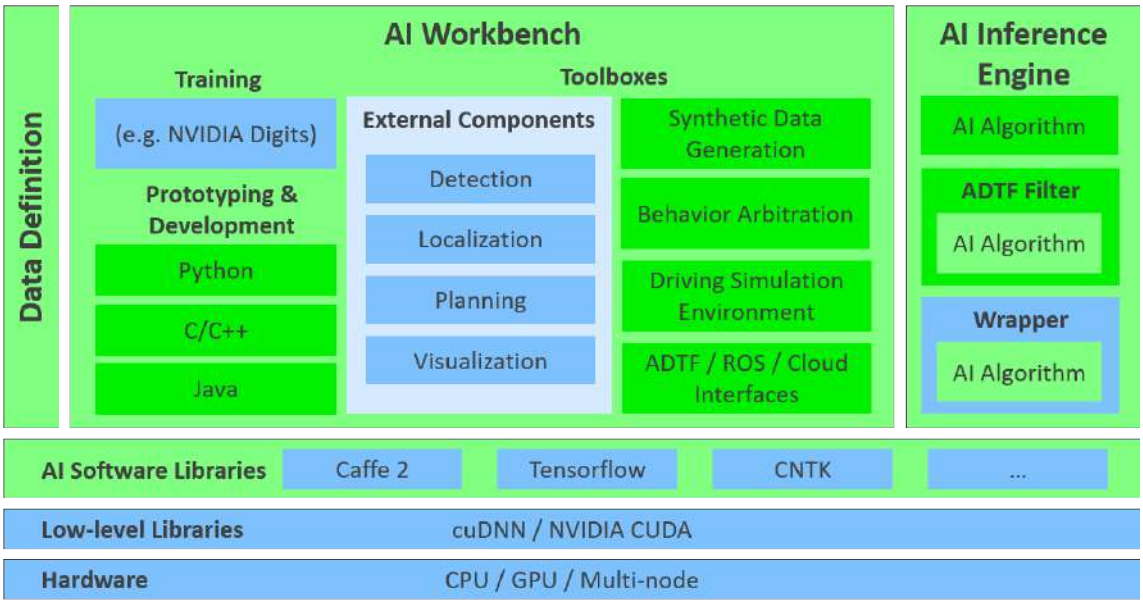



Figura 5.1: Platformă *AI Engines*, care extinde cele două operații majore ale procesului necesar algoritmilor deep learning: antrenarea și inferența, cu straturi adiționale pentru integrarea hardware și optimizarea performanței [76].

- evidențierea avantajelor oferite de utilizarea unor module de IA independente de sistemele de operare sau modulele software, în momentul proiectării, aplicării și integrării în medii diferite;
- promovarea importanței respectării anumitor standarde ca ISO26262 [78] sau ASPICE (eng. *Automotive Software Process Improvement and Capability Determination*) pentru a se asigura calitatea modelelor deep learning dezvoltate.

## 5.2 Platformă software pentru dezvoltarea modulelor de inteligență artificială

### 5.2.1 Arhitectură generală

Arhitectura generală a platformei software propuse, descrisă prin figura 5.1, urmărește cele două operații majore din fluxul de lucru al metodologiei DL: antrenarea și inferența. Alături de componentele care asigură antrenarea și inferența regăsim straturi și module pentru integrarea hardware și îmbunătățirea performanței.

Componenta centrală a acestei arhitecturi este reprezentată de *AI Workbench*, modulul care este responsabil cu antrenarea, prototiparea și dezvoltarea sistemelor, utilizându-se metode și instrumente deja existente, și considerând algoritmii de IA ca toolbox-uri în cadrul întregului framework. În cadrul acestei componente se poate observa că se asigură legătura cu o serie de module externe, care pot furniza datele necesare antrenării. Prin această legătură se asigură accesul către informațiile privind localizarea în plan local sau global al unui autovehicul și către algoritmii responsabili pentru detecția participanților la trafic, participanți care pot fi alte autovehicule sau pietoni. Prin interfațarea cu module ca cel de planificare a traiectoriei, *AI Workbench* poate să influențeze deciziile privind strategiile de deplasare pentru autovehiculele autonome.

Primul pas pentru dezvoltarea unei funcționalități constă în definirea cerințelor și colectarea datelor necesare pentru procesul de antrenare. Modulul *Data Definition* are rolul de a preprocesa datele,

astfel încât să poată fi utilizate pentru antrenare sau validare.

Odată ce un algoritm de IA este validat și testat, poate fi utilizat pentru inferență, prin componenta *AI Inference Engine*. Inferența constă în utilizarea unui model deja antrenat pentru fiecare caz particular. Se evaluează în acest sens datele provenite din lumea reală, așteptându-se ca acuratețea predicției să fie apropiată de cea calculată în timpul procesului de validare a modelului. Prin această componentă se pot genera inferențe pentru diverse platforme, dintre care se poate aminti ADTF, unde modelele sunt generate sub forma unor filtre, respectiv ROS.

### 5.2.2 Module de inteligență artificială independente de platformă

Instalarea unui algoritm de inteligență artificială într-un autovehicul autonom nu este o sarcină trivială datorită limitărilor introduse de dependențele de anumite platforme hardware, cât și de limitările în ceea ce privește capacitatea de calcul. În lucrarea [76] se propune o soluție pentru a se depăși aceste inconveniente prin construirea unor module de IA independente de platforma de dezvoltare.

Platforma dezvoltată este capabilă, utilizându-se interfețe către o serie de biblioteci specifice dezvoltării algoritmilor de deep learning, să integreze topologii variate de rețele neurale și să utilizeze date de la numeroase tipuri de senzori pentru evaluare. În această direcție a fost dezvoltat un framework care posedă capacitatea de a genera motoare de inferență pentru fiecare bibliotecă selectată: TensorFlow [79], Caffe2 [80] și CNTK [81].

Pentru a implementa algoritmul propus într-un mediu robust și stabil, platforma creată pentru dezvoltarea modulelor de IA a fost integrată în sistemul de dezvoltare de funcționalități specifice autovehiculelor autonome *Automotive Data and Time Triggered Framework* (ADTF), sistem care este descris în detaliu în secțiunea 5.3.3.

### 5.2.3 Conformitatea cu procesele standard din industria constructoare de autovehicule

În industria constructoare de autovehicule, unul dintre cele mai importante aspecte care trebuie respectate în procesul de dezvoltare software este conformitatea cu standardele acceptate în acest domeniu, standarde stricte mai ales când se implementează funcționalități care pot influența siguranța pasagerilor sau a pietonilor.

Este important ca abordarea selectată pentru dezvoltarea algoritmilor de inteligență artificială să fie ghidată din perspectiva unui model controlat și utilizat cu succes, denumit *V-model*. În mod tradițional acest model de creare a modulelor software împarte procesul de dezvoltare în două faze principale.

Partea din stânga este reprezentată de analiza și descompunerea cerințelor, crearea cerințelor de sistem, și proiectarea sistemului software sau a funcționalităților. Partea din dreapta a acestui model corespunde testării și validării.

În cazul aplicațiilor bazate pe algoritmi de tip DL acest model suferă modificări. Astfel, așa cum se poate observa în figura 5.2, partea stângă cuprinde definirea și achiziția datelor necesare antrenării, alături de normalizarea și augmentarea acestora. Odată finalizată procesarea datelor, se poate dezvolta arhitectura algoritmului de IA. Partea din dreapta a modelului conține procesul de integrare a rețelei neurale într-o platformă software, în cazul de față ADTF, și metodele de testare și de evaluare statistică. Cele două părți ale modelului V sunt conectate prin etapa de dezvoltare propriu-zisă a codului necesar pentru crearea algoritmului.

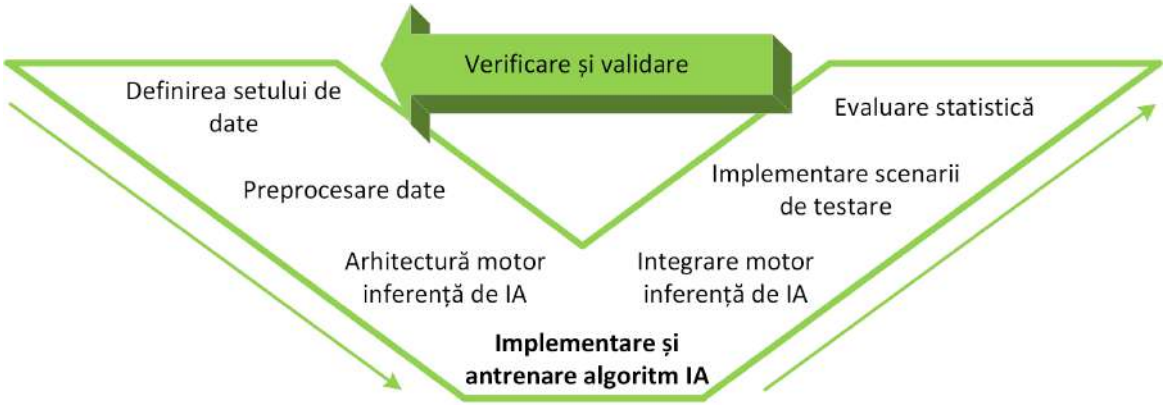



Figura 5.2: Reprezentarea schematică a modelului *V-model* [76], model care este conform cu procesul standard de dezvoltare software accceptat în industria constructoare de autovehicule.

## 5.3 Evaluarea performanței

### 5.3.1 Setul de date și procesul de antrenare

Pentru a se demonstra aplicabilitatea platformei software propusă pentru generarea modelelor de inferență ce pot realiza sarcini specifice conducerii autonome a autovehiculelor, algoritmul DGN a fost reantrenat, validat și integrat într-un mediu embedded pentru a clasifica medii de deplasare din interiorul unei clădiri de birouri.

Pentru reantrenarea și validarea algoritmului DGN, a fost creat un set particular de date folosindu-se fluxul de date provenit de la senzori de proximitate montați pe un prototip de autovehicul, un model în miniatură, echipat complet pentru a se deplasa autonom, prezentat în figura 5.3. Prototipul a fost deplasat prin interiorul unei cladiri de birouri, pe holuri și în birouri individuale, pentru achiziția și construirea setului de date. Exemple de eșantioane pot fi vizualizate în figura 5.5.

Gridurile de ocupanță au fost reprezentate sub forma unor matrice bidimensionale, fiecare acoperind o arie de 8 metri pătrați la o rezoluție de 25 de centimetri.

O parte din setul de date, un număr de 4 000 de eșantioane, au fost achiziționate cu ajutorul prototipului de mașină și etichetate în două clase: *hol* și *birou*. Pentru completarea setului de date, a fost generat un număr egal de eșantioane sintetice, utilizându-se metoda GOL [72].

Așa cum a fost specificat, pentru clasificarea gridurilor de ocupanță ce reprezintă medii de deplasare în interiorul unei clădiri, se utilizează arhitectura rețelei neurale DGN. Prin procesul de neuro-evoluție a rezultat o serie de modele cu performanțe ridicate. Pentru această clasificare simplă, între două clase, se va utiliza o arhitectură mai puțin complexă, pentru o rulare mai rapidă. Algoritmul *Adam* [82] a fost utilizat ca metodă stochastică de optimizare, iar *categorical cross-entropy* a fost selectată ca funcție de cost.

### 5.3.2 Prototipul vehiculului utilizat pentru integrarea algoritmului de clasificare

În această secțiune se va furniza o scurtă descriere a prototipului de vehicul utilizat ca mediu embedded pentru integrarea modelelor de inferență dezvoltate prin platforma *AI Engines*.

Așa cum poate fi observat în figura 5.3, acest prototip are o lungime de 60 de centimetri, o lățime de 30 de centimetri, reprezentând un model al unui vehicul la o scară de 1:8. Dispune de o gamă largă de senzori și actuatoare de calitate apropiată de producția de serie. Caracteristicile suplimentare includ

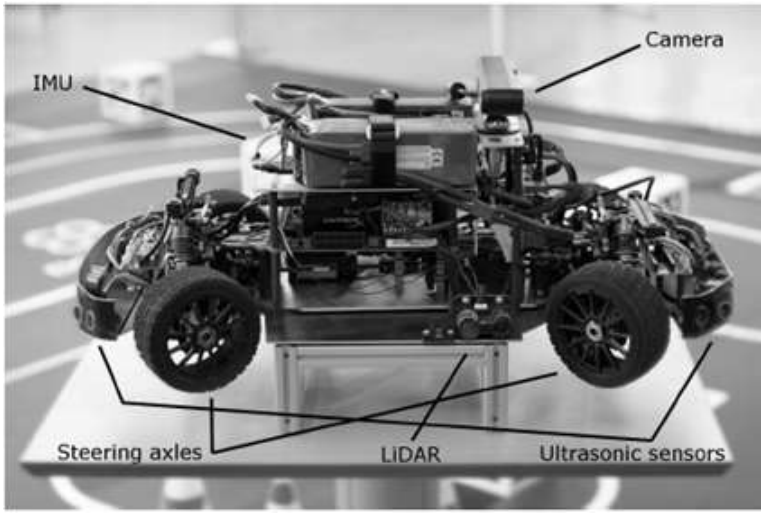



Figura 5.3: Prototipul vehiculului utilizat pentru integrarea și testarea modelelor de inferență într-un mediu embedded [76].

o unitate de procesare puternică, tracțiune integrală și două osii de direcție.

Acest vehicul este echipat cu senzori de proximitate ultrasonici, o cameră ce poate furniza informații bidimensionale dar și tridimensionale, și un senzor LIDAR simplu, cu un singur fascicul, ce poate fi utilizat pentru a calcula distanțe. Cinci senzori ultrasonici model HC-SR04 sunt montați pe bara din față a vehiculului și trei pe bara din spate.

### 5.3.3 Platforma software destinată dezvoltării algoritmilor pentru conducerea autonomă ADTF

Utilizată pentru integrarea și testarea modelelor de inferență *AI engines*, ADTF (eng. *Automotive Data and Time-Triggered Framework*) [83] este o platformă software destinată conceperii algoritmilor pentru industria constructoare de autovehicule, respectând toate standardele de dezvoltare software. Este o platformă populară în industrie, fiind utilizată de o multitudine de furnizori de componente software.

ADTF este un instrument utilizat pentru dezvoltarea, validarea, vizualizarea și testarea funcțiilor de asistență a șoferului și de conducere autonomă. De asemenea, este unul dintre cele mai utilizate medii de dezvoltare și testare pentru sistemele HAD din lume. Este compatibil cu o multitudine de cazuri de utilizare și aplicații. Acestea variază de la funcționalități de confort și siguranță, la funcții de conducere extrem de automatizate. Poate fi utilizat din faza de pre-dezvoltare, până la faza de producție de serie. ADTF este un sistem modular cu componente standard și interfețe deschise. Prin urmare, poate fi utilizat ca bază pentru dezvoltarea rapidă a unei mari varietăți de aplicații.

O captură din timpul rulării algoritmului DGN în cadrul ADTF pentru clasificarea mediilor în care un model în miniatură al unui vehicul se deplasează, este prezentată în figura 5.4.

### 5.3.4 Clasificarea gridurilor de ocupanță

Așa cum a fost menționat și în capitolele anterioare, una dintre cele mai importante sarcini ale unui vehicul autonom este localizarea într-un mediu necunoscut. În mod tradițional, localizarea unui autovehicul se realizează printr-un sistem GPS, sistem care nu este suficient de stabil pentru a acționa ca un factor de decizie în conducerea autonomă.

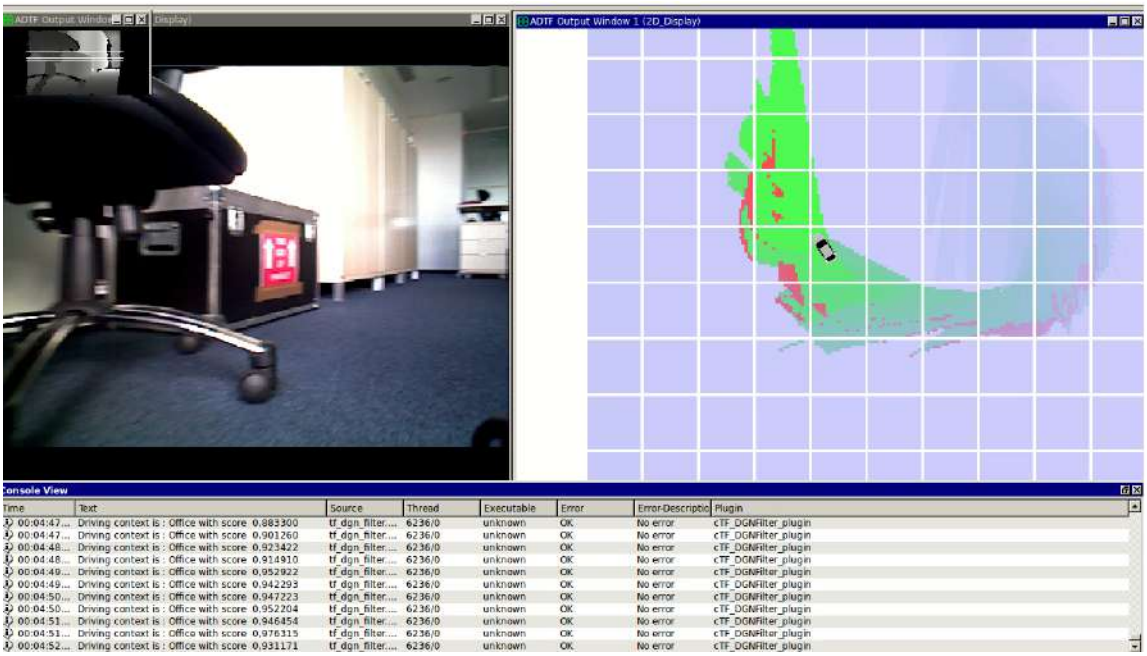



Figura 5.4: Captură din timpul rulării algoritmului DGN în cadrul ADTF pentru testarea și validarea modelelor de inferență generate prin platforma *AI Engines*.

În această direcție, cazul de utilizare propus pentru demonstrarea funcționării modulelor de inferență generate prin *AI Engines* este clasificarea contextului de deplasare utilizându-se griduri de ocupanță. Algoritmul DGN introdus în capitolul 4 realizează clasificarea scenariilor de trafic în mediul exterior, în situații reale de deplasare ale unui autovehicul. Pentru a se demonstra capacitatea de a integra un astfel de algoritm într-un mediu embedded, în acest capitol se realizează clasificarea unor medii de deplasare interioare.

Pentru a se demonstra versatilitatea algoritmului de clasificare s-au utilizat trei medii de implementare: un calculator de tip desktop, având Windows ca sistem de operare, un prototip de mașină bazat pe Unix, și tehnologia Azure Cloud .

Exemple de griduri de ocupanță sunt vizibile în figura 5.5. Celulele roșii reprezintă spațiul ocupat, în timp ce spațiul liber este marcat cu verde. Intensitatea culorii reprezintă gradul de ocupare. Dacă intensitatea culorii roșie este mai mare, crește și probabilitatea ca o celulă să fie ocupată.

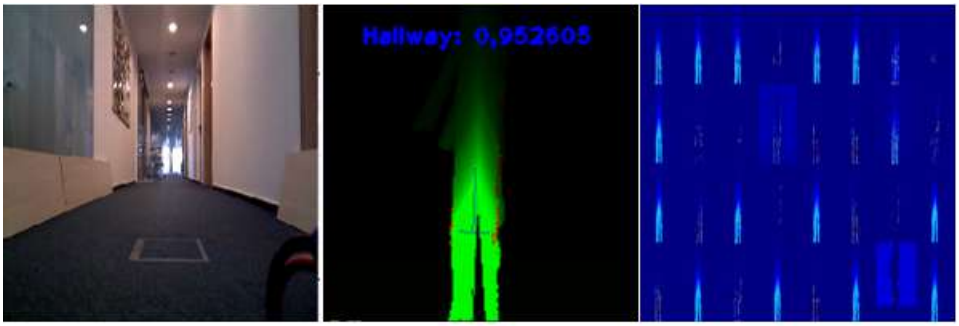
Gridurile au fost construite folosindu-se camera de adâncime montată pe partea superioară a prototipului mașinii și senzorii ultrasonici, fiind exportate sub formă de imagini. Ulterior eșantioanele au fost folosite pentru a se antrena și valida algoritmul de învățare profundă.

### 5.3.5 Rezultate experimentale

Portabilitatea modelelor utilizate pentru inferență a fost demonstrată prin integrarea algoritmului de DL într-un prototip de vehicul controlat de un sistem de operare Unix.

Rezultatele obținute în urma rulării algoritmului pe vehiculul prototip au fost comparate cu cele obținute prin rularea inferenței pe un calculator de tip desktop și cu cele obținute prin rularea pe un mediu Cloud dezvoltat de Microsoft, Azure.

Există câteva diferențe majore între gridurile de ocupanță construite pentru *hol* și *birou*, diferențe care ajută algoritmul să clasifice cu ușurință eșatioanele de intrare în clasele corespunzătoare.




a)


b)

Figura 5.5: Exemple de griduri de ocupanță (în centru) și activări ale primului strat de convoluție al modelului (în dreapta), împreună cu instanțele vizuale ale mediului interior (stânga): (a) hol; (b) birou.

Pentru a se demonstra acuratețea algoritmului, a fost construită o matrice de confuzie, descrisă prin tabelul 5.1. Pentru calcularea ratei de confuzie au fost necesare 200 de eșantioane, care au fost testate în raport cu valorile de referință. Rezultatul arată că topologia rețelei poate distinge între *birou* și *hol*, cu o precizie ridicată.

Tabela 5.1: Matrice de confuzie pentru prezicerea mediului de deplasare interior.

| | | **Clasă actuală** | |
|---|---|---|---|
| | | Birou | Hol |
| **Clasă prezisă** | Birou | 0.97 | 0.03 |
| | Hol | 0.05 | 0 .95 |

Dezvoltarea modulelor software în conformitate cu standardele cerute în industria constructoare de autovehicule conduce la avantajul de a avea independență față de platforma de inferență. Astfel, modelul de IA antrenat a putut fi evaluat în diferite medii, cum ar fi medii embedded sau bazate pe tehnologii web.

Așa cum a fost prezentat mai sus, accentul a fost pus pe testarea fiabilității modelelor pentru inferență pe o platformă embedded. În ceea ce privește posibilitatea de a se utiliza și alte medii pentru rulare, au fost utilizate un computer desktop și o configurație Cloud. A fost folosită o

configurație bazată pe CPU pentru rularea algoritmului DL, pentru a avea posibilitatea de a compara performanțele ratei de detecție în aceleași condiții. Folosirea unui computer desktop este facilă, integrarea modulelor de inferență fiind similară cu integrarea pe o platformă embedded.

Codul este independent de platforma de dezvoltare, așadar nu sunt necesare adaptări ale acestuia. Datorită faptului că modulele software au fost scrise în cadrul ADTF, funcționalitatea este ușor activată prin încărcarea respectivelor module în diverse configurații. În ceea ce privește rata de detecție, nu există nicio diferență majoră între rularea pe un calculator de tip desktop și pe prototipul de vehicul, așa cum se poate observa în tabelul 5.2.

### 5.3.6 Integrarea și rularea algoritmului în Cloud

Pentru a se demonstra portabilitatea modelelor generate pentru inferență s-a utilizat Microsoft Azure Cloud [84] pentru integrarea și rularea unui model antrenat.

Pentru a putea testa motoarele de inferență AI în Cloud, modulele software au fost salvate ca și componente de sine stătătoare în ADTF și rulate ca parte a unei imagini de docker. Fiecare docker procesează o listă de înregistrări și salvează rezultatele în format XML. Folosindu-se tehnologia Cloud performanța algoritmului dezvoltat a fost semnificativ îmbunătățită, în ceea ce privește viteza de procesare, așa cum se poate observa și în tabelul 5.2.

Tabela 5.2: Compararea performanțelor de clasificare într-un mediu interior.

| **Platformă** | **Rata cadrelor** |
|---|---|
| Prototip autovehicul | 10 cadre pe secundă |
| Calculator tip desktop | 12 cadre pe secundă |
| Cloud | 25 cadre pe secundă |

## 5.4 Concluzii

În acest capitol a fost prezentat un sistem conceput pentru a testa module software de inteligență artificială care sunt independente de platforma de dezvoltare și respectă standardele de calitate impuse de industria producătoare de autovehicule. Contribuția principală este realizarea unui framework ce integrează biblioteci și tehnici de IA pentru dezvoltarea și implementarea nu doar a unor prototipuri, ci și a unor module ce pot intra în producția de serie.

S-a demonstrat capacitatea proiectării și implementării unei platforme software complete, capabilă să îndeplinească toate fazele necesare lansării unui modul software, de la antrenarea algoritmului până la validare și inferență. Posibilitatea de a se utiliza diverse biblioteci de DL într-un singur framework este foarte utilă pentru eficiența și versatilitatea producției.

În comparație cu platformele deja existente, ce integrează tehnologii limitate și topologii standard de rețele neurale, propunerea descrisă aici este capabilă să rezolve sarcini diferite, pe diverse platforme hardware. S-a demonstrat posibilitatea de a se rula modelele pentru inferență în trei medii diferite, pe un calculator de tip desktop, pe o platformă embedded și în Microsoft Azure Cloud, cu modificări minime de configurație.

# 6. Prelucrarea datelor senzoriale pentru generarea traiectoriei și controlul unui vehicul autonom

Învățarea unui comportament asemănător cu cel uman în conducerea autonomă este o problemă încă nerezolvată, multe dintre sistemele actuale fiind proiectate pentru învățarea de tip *end2end*, sau sub forma unei secvențe de operații decuplate, constituită din modulele pentru percepție-planificare-acțiune.

Soluțiile actuale de IA aplicate în domeniul construcțiilor de autovehicule utilizează cu precădere funcții de cost cu un singur obiectiv și algoritmul de propagare înapoi a erorii pentru a învăța o mapare directă între informațiile de intrare, ce pot fi sub forma unui flux video, și comenzile pentru virare. Maparea este urmată de o planificare a traiectoriei și de controlul acțiunilor. Aceste soluții au multiple limitări, funcționând în mod optim doar în anumite scenarii, cum ar fi deplasarea pe autostradă. Nu sunt capabile să simuleze deocamdată un comportament apropiat celui uman.

Pentru depășirea acestor dificultăți, se propune o abordare de tip multi-obiectiv, bazată pe neuro-evoluție, pentru estimarea traiectoriei locale a unui vehicul autonom într-un orizont finit, utilizându-se o rețea neurală destinată percepției și planificării [1]. Acest comportament este antrenat printr-un model monolitic reprezentat de o rețea neurală, formată din straturi de convoluție și straturi recurente. Acest model este antrenat utilizându-se griduri de ocupanță din lumea reală și griduri generate sintetic. Spre deosebire de metodele tradiționale de învățare, în lucrarea [68] se propune o abordare bazată pe neuro-evoluție pentru învățarea unui front Pareto de rețele neurale, unde fiecare element din front reprezintă o rețea ce furnizează un răspuns pentru minimizarea unei funcții de fitness multi-obiectiv. Un front Pareto reprezintă o metodă de optimizare multi-obiectiv, care implică una sau mai multe funcții de cost care să fie optimizate simultan. Comportamentul dorit este codificat într-o funcție de fitness cu trei elemente, ce descrie energia minimă pentru deplasare, o viteză laterală minimă și o viteză maximă de deplasare înainte, limitate la un anumit interval.

Sistemul a fost comparat cu algoritmii de învățare *end2end* și de reinforcement learning de ultimă generație, demonstrându-se că învățarea prin folosirea unui set mixt de date sintetice și de date din lumea reală poate duce la o planificare eficientă a traiectoriei și un control robust al vitezei unui vehicul autonom.

## 6.1 Noțiuni introductive

Abilitatea unui autovehicul autonom de a vira singur, imitând un comportament specific uman, a devenit unul dintre obiectivele principale ale cercetării în cazul conducerii autonome. Un vehicul autonom este un agent inteligent care observă vecinătățile scenei, poate lua decizii și efectuează acțiuni bazate pe aceste decizii. Aceste funcționalități mapează datele de la senzori cu ieșirile de comandă, fiind de cele mai multe ori implementate ca module separate pentru percepție, planificare și acțiune [85], așa cum este prezentat în figura 6.1(a). Prin această abordare modulară, fiecare

[1] Acest capitol se bazează pe eforturile colective depuse pentru realizarea lucrării [68]

problemă este împărțită în mai multe subprobleme, în care fiecare modul este construit pentru a rezolva o sarcină specifică.

Învățarea de tip *end2end* [86], mapează datele de intrare în formă brută la semnalele de control. Datele utilizate în procesul de antrenare, sunt adesea sub formă de imagini, provenite de la o cameră frontală, fiind colectate împreună cu unghiurile de virare, sincronizate în timp. Achiziția datelor se realizează, de cele mai multe ori, prin conducerea de către un operator uman al unui autovehicul. O rețea neurală este mai apoi antrenată prin furnizarea comenzilor de virare și a imaginilor ce conțin calea de rulare [87].

Învățarea prin întărire (eng. *Deep Reinforcement Learning* (DRL)) [88] este un tip de algoritm de ML în care agenții învață acțiuni prin interacțiunea cu mediul în care se află. Într-un astfel de sistem, o politică de învățare este reprezentată de maparea unei stări la o distribuție de acțiuni. Algoritmul nu se bazează pe datele de antrenament etichetate, dar maximizează o recompensă cumulativă, care este pozitivă dacă vehiculul este capabil să-și mențină direcția fără coliziuni, și negativă în caz contrar.

Considerându-se cele expuse anterior, este din ce în ce mai evident că în domeniul conducerii autonome algoritmii tradiționali vor migra către sisteme bazate pe rețele neurale, așa cum este reprezentat în figura 6.1(b), unde senzorii sunt mapați la o strategie comportamentală de conducere utilizându-se sisteme bazate pe rețele neurale. Tehnicile *end2end* și DLR tind să generalizeze pentru scenarii de condus specifice (de exemplu, conducerea unui autovehicul pe autostradă), sau necesită sesiuni de învățare prelungite în medii de simulare care, în cele mai multe cazuri, nu furnizează informații suficient de precise pentru a antrena un algoritm pentru scenarii ce apar în lumea reală.

Așadar, în acest capitol se propune o abordare bazată pe neuro-evoluție pentru conducerea autonomă, care are ca punct de start reformularea conducerii autonome ca o problemă de estimare a stării traiectoriei locale pentru un agent artificial, unde o rețea neurală adâncă are rolul de a prezice traiectoria locală a unui vehicul. Acest concept este ilustrat în figura 6.2, unde algoritmii genetici sunt utilizați pentru a antrena o populație de rețele neurale, ce conțin atât straturi recurente cât și de convoluție, atât cu date pentru conducere reale, cât și simulate. Datele de antrenare sunt reprezentate de o combinație dintre o secvență de griduri de ocupanță, descrise în secțiunea 4.2.2, și informații ce descriu un comportament de conducere, conținând traiectorii de deplasare, unghiuri de virare și viteza de deplasare. Datele sintetice sunt obținute printr-un proces generativ care modelează comportamentul vehiculului controlat și, de asemenea, comportamentul altor participanți la trafic, în cadrul unui simulator denumit *GridSim* [89]. Acest sistem generativ este bazat pe un model cinematic al unui robot non-holonomic.

Mixul de date este utilizat pentru a se evolua o populație de rețele neurale $\mathbf{\Phi}$, algoritmii genetici fiind aplicați pentru învățarea parametrilor sau ponderilor $\Theta_1, ..., \Theta_K$. Prin antrenare se dorește minimizarea funcției de fitness multi-obiectiv $L = [l_1(\varphi), ..., l_W(\varphi)]$ în spațiul obiectiv multidimensional $L$, unde fiecare axă de coordonate reprezintă o valoare de fitness. Pentru

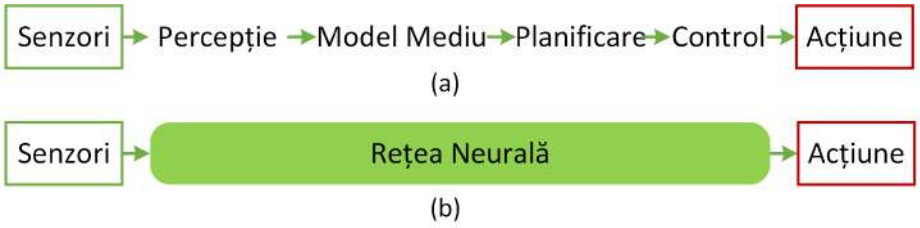



Figura 6.1: Tranziția de la un sistem software modular la o rețea neurală monolitică pentru a se determina strategia de control în conducerea autonomă: (a) maparea senzorilor la actuatori utilizându-se un lanț clasic de procesare; (b) rețea neurală de adâncime pentru maparea datelor provenite de la senzori la acțiunile de control.

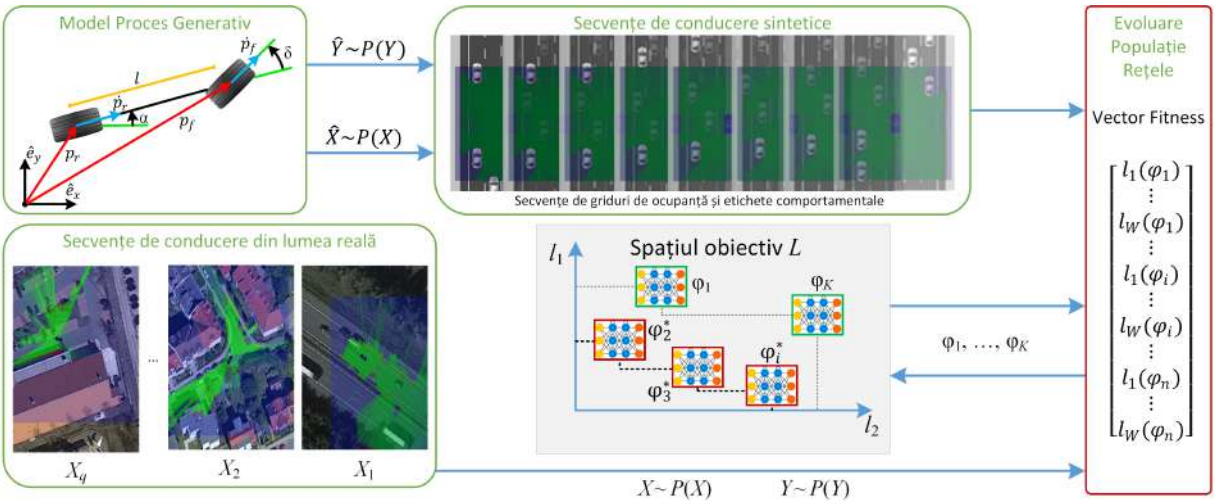



Figura 6.2: Sistem de estimare a traiectorie utilizându-se procesul de neuro-evoluție.

selectarea celor mai performante modele din cadrul populației $\Phi$ se utilizează un front Pareto. Procesul de antrenare utilizează un vector care conține mai multe obiective: calea parcursă de agent, viteza laterală a acestuia și viteza longitudinală.

Cele mai importante contribuții aduse în lucrarea [68] și prezentate în acest capitol sunt:

- prelucrarea statistică a datelor senzoriale pentru estimarea traiectoriei unui vehicul autonom;
- definirea unei rețele convoluționale-LSTM profunde pentru prezicerea comportamentelor optime de conducere;
- introducerea unei abordări neuro-evolutive pentru antrenarea sistemelor de conducere autonome, bazată pe optimizarea Pareto multi-obiectiv și pe algoritmii genetici.

## 6.2 Descrierea metodologiei

### 6.2.1 Controlul unui vehicul autonom utilizându-se griduri de ocupanță

Pentru controlul și generarea traiectoriei unui vehicul autonom se combină predicțiile temporale robuste ale unei rețele LSTM cu un model generativ. O prezentare a spațiului problemei poate fi observată în figura 6.3, unde dându-se poziția curentă a vehiculului $p_{ego}^{<t>}$, o destinație dorită $p_{dest}^{<t+\tau_o>}$ și o secvență de intrare ce conține griduri de ocupanță $X^{<t-\tau_i,t>} = [\hat{x}^{<t-\tau_i>}, ..., \hat{x}^{<t>}]$, se dorește estimarea unei strategii de control $Y^{<t,t+\tau_o>} = [\hat{y}^{<t+1>}, ..., \hat{y}^{<t+\tau_o>}]$, unde fiecare element $y$ este secvența de ieșire ce reprezintă poziția dorită, înclinarea și viteza vehiculului la un moment specific de timp.

Furnizându-se o secvență de griduri de ocupanță 2D $\mathbf{I} : \mathbb{R}^2 \times \tau_i \rightarrow \mathbb{R}^2 \times \tau_o$, poziția vehiculului $p_{ego}^{<t>} \in \mathbb{R}^2$ în $I^{<t>}$, și coordonatele destinației $p_{dest}^{<t>} \in \mathbb{R}^2$ în spațiul gridului la momentul de timp $t$, sarcina este de a învăța strategia de control optimă pentru ca vehiculul să se deplaseze la coordonatele destinației $p_{dest}^{<t+\tau_o>}$, unde $\tau_o$ este numărul de pași în timp pentru care este planificată mișcarea sau deplasarea vehiculului. Cu alte cuvinte, cu $p_0^{<t>}$ fiind o coordonată în actuala observație a hărții de ocupanță, se caută o strategie de navigare a vehiculului, dintr-un punct de start ales aleatoriu $p_0^{<t>}$ către o destinație $p_{dest}^{<t+\tau_o>}$, cu următoarele proprietăți:

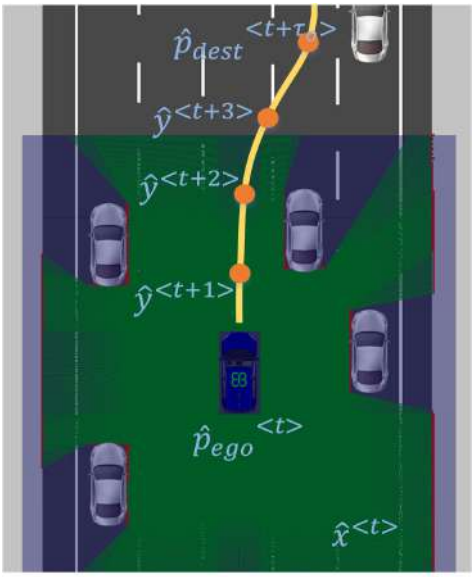



Figura 6.3: Generarea traiectoriei unui vehicul utilizându-se date sintetice [68].

- calea de deplasare $||p_0^{<t>} - p_{dest}^{<t+\tau_o>}||$ este minimă;
- viteza laterală, dată de rata de modificare a unghiului de virare $v_\delta \in \left[\dot{\delta}_{min}, \dot{\delta}_{max}\right]$, descrisă prin $v_\delta^{<t,t+\tau_o>}$ este minimă;
- viteza de deplasare, sau viteza longitudinală $v_f^{<t,t+\tau_o>}$, este limitată la un interval $[v_{min}, v_{max}]$.

Pentru optimizarea problemei de mai sus se consideră o funcție de fitness multi-obiectiv care cuantifică comportamentul vehiculului:

$$L^{s^{<t+\tau_o>}}_{s^{<t>},a^{<t+1,t+\tau_o>}} = \left[I_1^{<t+\tau_o>} \quad I_2^{<t+\tau_o>} \quad I_3^{<t+\tau_o>}\right]. \tag{6.1}$$

În mod intuitiv, $I_1^{<t+\tau_o>}$ reprezintă un feedback bazat pe distanță, care are valoare scăzută dacă mașina urmează o traiectorie definită de o energie minimă la $p_{dest}^{<t+\ tau_o>}$ și o valoare ridicată în caz contrar. $I_2^{<t+\tau_o>}$ cuantifică mișcările periculoase și disconfortul pasagerilor prin însumarea vitezei laterale a vehiculului. Funcția de feedback $I_3^{<t+\tau_o>}$ reprezintă viteza de deplasare longitudinală a vehiculului, limitată la viteze adecvate pentru diferite sectoare de drum ca, de exemplu, $v_f^{<t,t+\tau_o>} \in [80 \text{ km/h}, 130 \text{ km/h}]$ pentru conducerea pe autostradă.

Pentru modelarea comportamentului descris mai sus se antrenează un model de aproximare optim, în cazul de față definit ca o rețea neurală $\varphi$, care poate să prezică strategia comportamentală optimă a unui vehicul $Y^{<t,t+\tau_o>}$ , furnizându-se ca date de intrare o secvență de griduri de ocupanță $X^{<t-\tau_i,t>}$ și un vectorul de fitness multi-obiectiv definit prin ecuația ( 6.1).

Rețeaua neurală proiectată pentru îndeplinirea acestei sarcini este prezentată în figura 6.4, unde secvențele de griduri de ocupanță sunt procesate de un set de straturi convoluționale, înainte de a fi procesate de ramuri diferite compuse din straturi LSTM. Fiecare ramură este responsabilă cu estimarea setului de puncte ale traiectoriei de-a lungul intervalului de timp$[t+1, t+\tau_o]$. Fluxul de date este procesat inițial de o rețea neurală de convoluție, fiind apoi furnizat ca intrare într-o stivă de straturi LSTM prin două straturi complet conectate cu 1024, respectiv 512 unități. Această topologie poate fi utilizată atât în cazul utilizării datelor sintetice, cât și în cazul utilizării datelor reale. Atât ponderile straturilor de convoluție cât și cele din straturile LSTM sunt învățate în timpul antrenării.

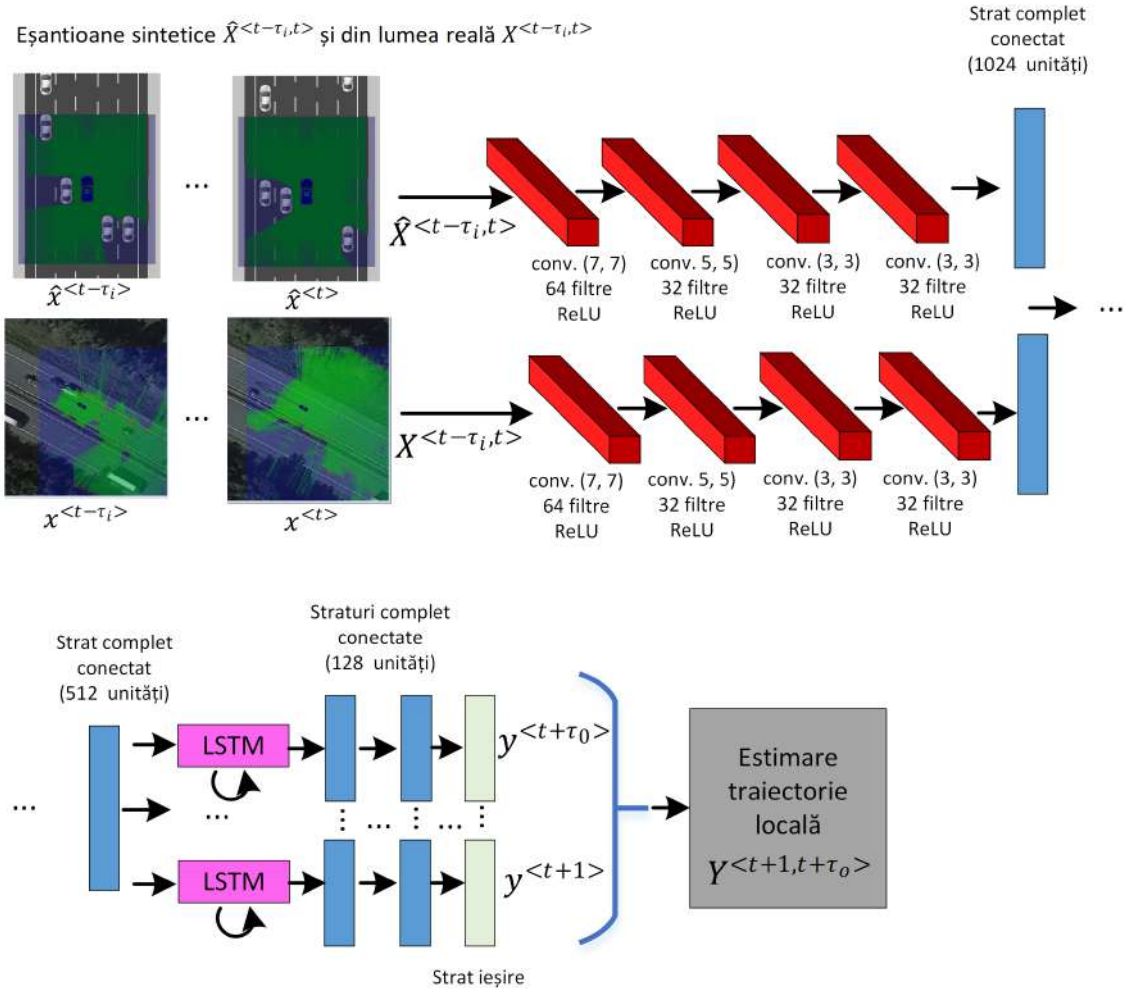



Figura 6.4: Arhitectura rețelei neurale concepută pentru estimarea traiectoriilor locale.

Atât parametri rețelei de convoluție, cât și cei ai rețelei LSTM sunt notați cu $\Theta$. Așa cum se poate observa în figura 6.4, cele două fluxuri de date, cele sintetice și cele achiziționate din lumea reală, sunt procesate în paralel de două rețele neurale înainte de a fi concatenate și furnizate ca date de intrare într-o rețea LSTM, prin intermediul unui strat complet conectat cu 512 de unități.

### 6.2.2 Antrenarea rețelelor neurale utilizându-se metoda de neuro-evoluție

Scopul procesului de antrenare este de a se calcula ponderile optime pentru o colecție de rețele neurale $\Phi$ prin optimizarea simultană a vectorului fitness $L$ din ecuația (6.1), propusă inițial în lucrarea [90].

Pentru antrenarea modelului de rețea neurală din figura 6.4, setul de date de antrenare $\mathscr{D}$ a fost compus din secvențe de griduri de ocupanță, atât simulate cât și achiziționate din lumea reală:

$$\mathscr{D} = \left\{ \begin{array}{l} (\hat{X}^{<t-\tau_i,t>}, \hat{Y}^{<t+1,t+\tau_o>}) \sim P(X^{<t-\tau_i,t>}, Y^{<t+1,t+\tau_o>}) \\ (X^{<t-\tau_i,t>}, Y^{<t+1,t+\tau_o>}) \sim P(X^{<t-\tau_i,t>}, Y^{<t+1,t+\tau_o>}) \end{array} \right\}, \tag{6.2}$$

unde $P(X^{<t-\tau_i,t>}, Y^{<t+1,t-\tau_i>})$ reprezintă gridul de ocupanță, respectiv datele ce descriu comportamentul vehiculului.

În cazul unei proceduri de antrenare neuro-evolutive, $\mathbf{\Phi}$ reprezintă o colecție de rețele neurale $K$, fiecare având propriul set de ponderi $\Theta_i$:

$$\mathbf{\Phi} = \left[\varphi_1(\Theta_1), \varphi_2(\Theta_2), ..., \varphi_K(\Theta_K),\right]^T. \tag{6.3}$$

Ponderile unei singure rețele sunt memorate într-un așa-numit vector soluție $\Theta = \left[\theta_1, \theta_2, ..., \theta_n,\right]^T$, compus din $n$ variabile de decizie $\theta_i$, cu $i = 1, ..., n$ și $\theta \in \mathbb{R}^n$. $\theta_i$ reprezintă ponderile a unei rețele.

În cazul unei învățări multi-obiectiv scalarizată, rezultatele diferitelor funcții de loss sunt agregate într-o singură funcție de cost scalară $F$, care poate fi minimizată sau maximizată după cum urmează:

$$\begin{aligned} &\text{minimize / maximize } F(I_w(\Theta), \lambda), \quad w = 1, 2, ..., W, \\ &\text{such that } \Theta \in \Theta_\lambda, \end{aligned} \tag{6.4}$$

unde $F : \mathbb{R}^{W+1} \mapsto \mathbb{R}$ este o funcție care agregă toate funcțiile de cost $W$ bazându-se pe valorile de scalarizare date de vectorul $\lambda$, astfel încât $\Theta \in \Theta_\lambda$:

$$F(I_w(\Theta), \lambda) = \min / \max \sum_{i=1}^{W} \lambda_i J_i(\Theta). \tag{6.5}$$

O soluție este un vector variabil în spațiul decizional, cu o coordonată $I$ ca și vector obiectiv corespondent. În optimizarea Pareto există un set de soluții optime $\mathbf{\Theta}^*$, niciuna dintre ele neavând capacitatea de a minimiza sau maximiza simultan toate funcțiile obiectiv. Soluțiile optime Pareto nu pot fi îmbunătățite pentru niciun obiectiv, fără a degrada cel puțin un alt obiectiv. O soluție fezabilă $\Theta_1$ se spune că domină o altă soluție $\Theta_2$ dacă:

1. $I_i(\Theta_1) \leq I_i(\Theta_2)$ pentru toate $i \in \{1, 2, ..., W\}$, și
2. $I_j(\Theta_1) < I_j(\Theta_2)$ pentru cel puțin un index $j \in \{1, 2, ..., W\}$.

O soluție $\Theta^*$ se consideră a fi o soluție optimă Pareto dacă nu există altă soluție care să o domine. Setul de soluții optime Pareto este denumit *front Pareto*. Căutarea soluțiilor optime se efectuează în spațiul de decizie, în timp ce evaluarea lor are loc în spațiul obiectiv. Pentru calcularea limitelor Pareto s-a utilizat o abordare evolutivă [91].

Utilizându-se algoritmii genetici, au fost evaluate ponderile $\mathbf{\Theta}$ ale unei populații de rețele neurale $\mathbf{\Phi}$, unde un individ $\Theta$ este un vector soluție, care conține ponderile unei rețele $\varphi(\Theta)$. Primul pas al antrenării constă în propagarea înainte prin populația de rețele, obținându-se astfel valori pentru vectorul de fitness multi-obiectiv $L(\cdot)$, descris de ecuația (6.1). După finalizarea propagării înainte prin populația de rețele, se selectează cei mai buni indivizi. Acești indivizi sunt cei aflați pe frontul Pareto în spațiul obiectiv $L$. La fiecare iterație de antrenament, se calculează un nou front Pareto. Algoritmul de selecție asigură faptul că un număr de indivizi, care prezintă cea mai bună acuratețe, sunt transferați către următoarea generație, fără să sufere transformări. Pentru explorarea spațiului de decizie $S$ se utilizează mutația și recombinarea uniformă între doi indivizi. Operațiile de mutație și recombinare sunt aplicate aleatoriu, cu o probabilitate de 50%. Următoarea generație de gene individuale este supusă unui zgomot Gaussian aditiv $\sigma$:

$$\Theta' = \Theta + \sigma; \sigma \in [-3, 3]. \tag{6.6}$$

Noua populație este evaluată și procesul se repetă pentru un număr de generații $G$. La final se obține frontul Pareto $\mathbf{\Theta}^*$, procesul detaliat fiind prezentat în [92].

## 6.3 Experimente

Pentru realizarea experimentelor se utilizează atât griduri de ocupanță colectate prin conducerea în situații de trafic reale, cât și griduri sintetice generate în cadrul mediului de simulare GridSim [68].

### 6.3.1 Setul de date achiziționat prin intermediul unui vehicul complet echipat pentru conducerea autonomă

Așa cum a fost descris și în capitolul 4, datele de antrenare din lumea reală au fost colectate conducând un vehicul de test, prezentat în figura 4.6, pe mai multe tipuri de drumuri din Germania. Datele de antrenare finale achiziționate din lumea reală, definite prin ecuația (6.2), sunt compuse din griduri temporale mapate la rata de schimbare a unghiului de virare $v_\delta$ și la viteza longitudinală $v_f$.

### 6.3.2 Setul de date sintetice

Pentru generarea datelor sintetice, parte a ecuației (6.2), s-a folosit mediul de simulare GridSim [89], care utilizează un model de senzor pentru a genera gridurile de ocupanță sintetice, împreună cu etichetele ce definesc comportamentul. Un total de 20 km au fost parcurși în interiorul mediului de simulare pentru a se achiziționa datele necesare antrenării. Datele sintetice sunt generate prin calcularea gridurilor la intervale de timp fixe. La fiecare ciclu de stocare a datelor, gridurile sintetice sunt salvate împreună cu rata de modificare a unghiului de virare și cu viteza longitudinală.

### 6.3.3 Antrenarea sistemului

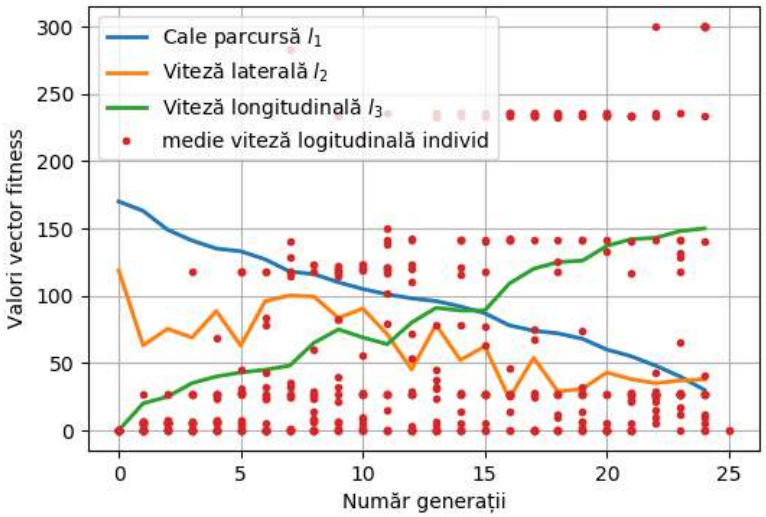



Figura 6.5: Evoluția vectorului de fitness pe parcursul procesului de antrenare.

Setul de date de antrenare, compus din date din lumea reală și date sintetice, conform ecuației (6.2), este furnizat ca intrare în rețeaua neurală din figura 6.4. Ambele tipuri de date sunt alcătuite din $\tau_i$ griduri de ocupanță înregistrate la un interval de timp $[t - \tau_i, t]$, și $\tau_o$ etichete care definesc comportamentul, înregistrate la o rată de $[t + 1, t + \tau_o]$.

Un algoritm genetic evoluează o populație de indivizi sau genotipuri. O populație $\Phi$ reprezintă o colecție de rețele neurale, unde fiecare rețea $\varphi(\Theta)$ este definită conform arhitecturii prezentate

în figura 6.4. Un individ este un vector soluție $\Theta$, care codifică ponderile unei rețele. La începutul procesului de antrenare, ponderile sunt inițializate utilizându-se o distribuție Gaussiană deterministă.

În cadrul unei generații, fiecare individ $\Theta$ este evaluat în raport cu vectorul multi-obiectiv de fitness din ecuația (6.1). După fiecare iterație a procesului de antrenare, frontul Pareto își schimbă structura prin minimizarea traseului parcurs și a vitezei laterale, și maximizarea vitezei longitudinale, așa cum se poate observa în figura 6.5, unde punctele roșii specifică valorile medii de fitness pentru viteza longitudinală, la fiecare generație. În cadrul ciclului de antrenare, se utilizează media valorilor minimale ale vectorului de fitness pentru a se extrage top 5 indivizi. Indivizii selectați devin *părinți* în următoarea generație. Fiecare individ selectat suferă un proces de mutație, aplicându-se un zgomot Gaussian, conform ecuației (6.6).

### 6.3.4 Evaluarea performanței

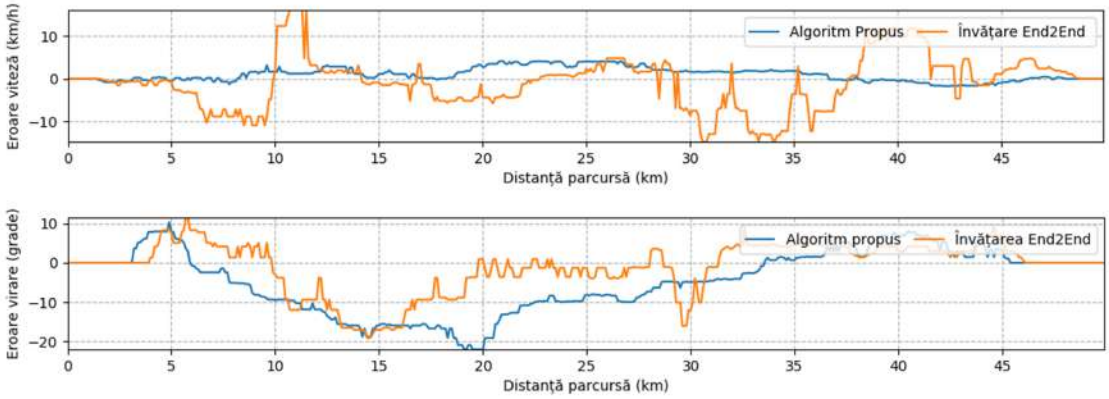



Figura 6.6: Predicția pentru conducerea pe autostradă.

Performanța sistemului propus a fost evaluată în comparație cu metode de top pentru învățarea procesului de conducere autonomă prin tehnici de machine learning, cum ar fi învățarea *end2end* și DRL.

În cazul învățării *end2end* [86], secvențele de griduri de ocupanță au fost mapate direct la acțiunile viitoare de conducere fără prezicerea comportamentului optim de-a lungul unei traiectorii de conducere estimate. Antrenarea rețelei neurale se bazează pe algoritmul de propagare înapoi a

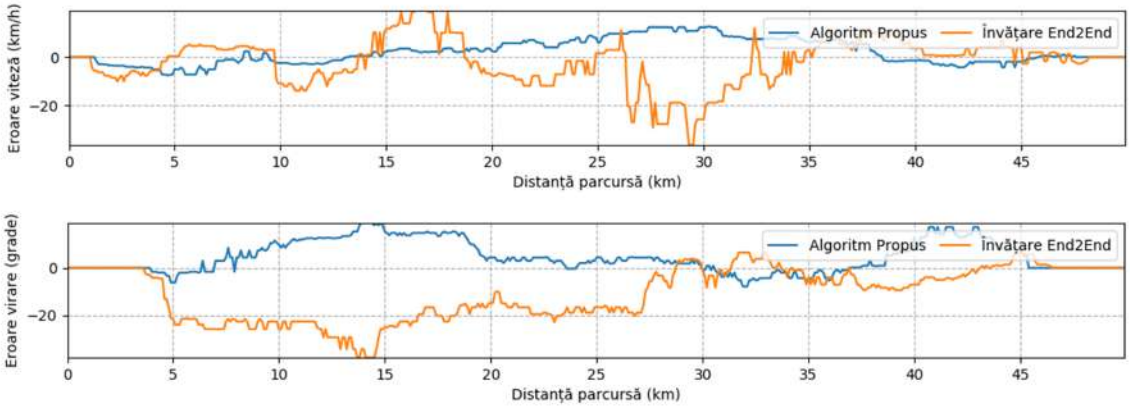
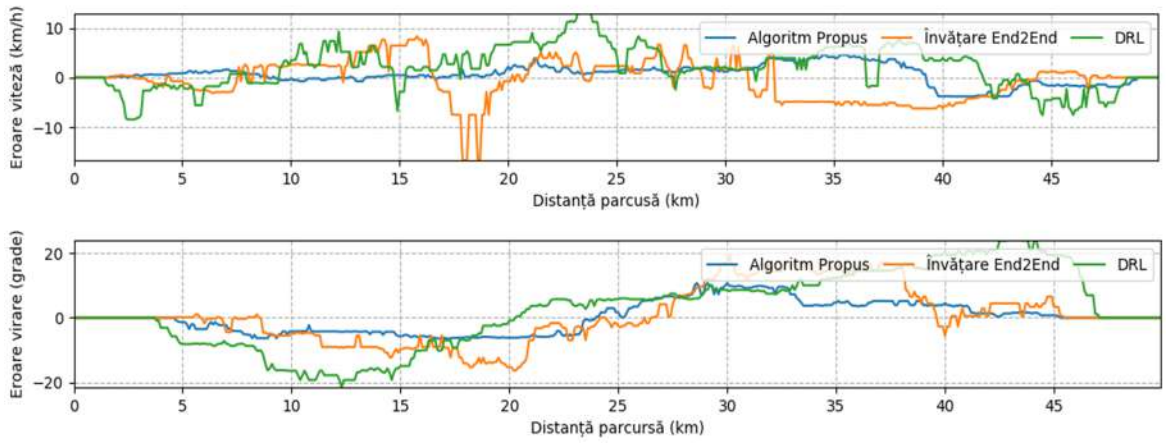
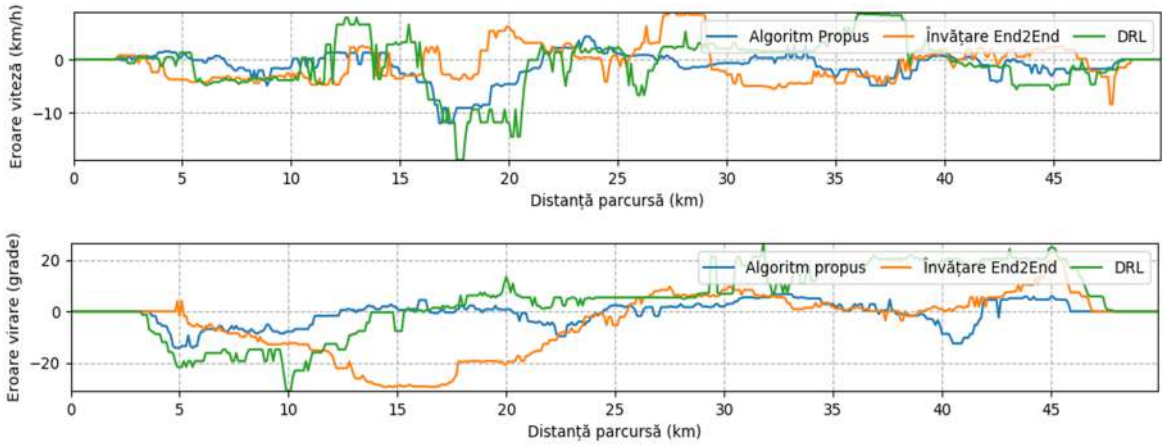



Figura 6.7: Predicția pentru conducerea în interiorul unui oraș.




(a)




(b)

Figura 6.8: Predicțiile de conducere calculate pentru autostradă (a) și oraș (b) utilizând date sintetice.

erorii, urmărind o estimare de tip *maximum likelihood*.

Comparația cu DRL a fost realizată urmărindu-se implementarea așa-numitei rețele *Deep Q-Network (DQN)*, descrisă în lucrarea [88]. Datorită faptului că într-un sistem DRL agentul este instruit prin interacțiunea sa cu mediul, agentul a fost antrenat în cadrul simulatorului GridSim. Agentul DQN a fost antrenat utilizându-se următorii hiperparametri:

- rata de actualizare $\gamma$: 0.95;
- factorul inițial de acaparare $\epsilon_0 = 1.0$ ;
- valoarea de degradare: $\epsilon_{decay} = 0.99$;
- factorul minim de acaparare: $\epsilon_{min} = 0.05$;
- rata de învățare: $25 \times 10^{-7}$;
- factorul de regularizare L2: $\lambda_2 = 10^{-4}$.

Datele de intrare sunt furnizate sub forma unui tensor de dimensiune (1,4), reprezentând o secvență de 4 imagini capturate din GridSim, cu o diferență de eșantionare de 5 cadre. Dimensiunea cadrelor a fost redusă la 80×80 pixeli pentru creșterea vitezei de procesare. Activarea liniară a ultimului strat al rețelei DQN produce un număr întreg mapat la un spațiu de decizie ce conține opt acțiuni: accelerare, stânga, dreapta, fără acțiune, frânare, accelerare + stânga, accelerare + dreapta și decelerare.

A fost concepută o politică de recompensă $\rho$, în care recompensa totală este normalizată în intervalul

[-1, 1]. Funcția de recompensă (eng. *reward*), este o sumă ponderată compusă din distanța parcursă de agent la momentul $t$ și viteza curentă a agentului.

Utilizându-se setul de date achiziționate cu vehiculul de test prezentat în figura 4.6 sau efectuat experimente utilizându-se gridurile de ocupanță generate pentru 50 de kilometrii de condus prin interiorul unui oraș și 50 de kilometrii de condus pe autostradă. Traiectoria parcursă de șoferul uman a fost utilizată ca referință în cadrul procesului de evaluare. Aceste date au fost achiziționate doar pentru testare, nefiind folosite în procesul de antrenare. Erorile vitezei $e_{v_f}$ și ale unghiului de virare $e_\delta$ sunt afișate în figurile 6.6 și 6.7. Astfel, valorile absolute ale erorilor vitezei și unghiurilor de virare obținute pentru conducerea pe autostradă sunt: $mean(e_{v_f})$ =1.454 km/h, $max(e_{v_f})$ =3.822 km/h, $mean(e_\delta)$ =3.696 grade, $max(e_\delta)$ =9.589 grade, în comparație cu valorile absolute obținute pentru conducerea în interiorul unui oraș care sunt : $mean(e_{v_f})$ =2.692 km/h, $max(e_{v_f})$ =9.405 km/h, $mean(e_\delta)$ =3.726 grade, $max(e_\delta)$ =10.619 grade. Predicțiile sistemului propus înregistrează erori mai mici în comparație cu sistemele bazate pe învățarea *end2end* și DRL.

Analizând figurile 6.6 și 6.7, se poate observa că erorile obținute în cazul conducerii în interiorul unui oraș sunt mai mari. Acest rezultat este unul așteptat, deoarece, un mediu tipic din interiorul orașului are o structură ce variază mult mai mult față de structura unei autostrăzi, fapt ce poate fi observat cu ușurință în exemplele de griduri furnizate în figura 4.2.

Datorită naturii sale, algoritmul DQN poate fi antrenat doar într-un mediu de simulare. Rezultatele evaluării algoritmului DQN sunt mai puțin exacte, datorită faptului că metoda utilizează doar datele sintetice furnizate de GridSim. Acest fenomen este vizibil și în figura 6.8. DQN funcționează în parametri atunci când este rulat în GridSim, dar are o scădere bruscă a performanței când este utilizat cu date din lumea reală.

## 6.4 Concluzii

Acest capitol reprezintă o continuare a cercetărilor din capitolele anterioare, unde au fost introduse noțiuni ca gridurile de ocupanță, algoritmii genetici, sau datele sintetice.

Se prezintă astfel o abordare neuro-evolutivă pentru a se genera o traiectorie locală dintr-o secvență de griduri de ocupanță. Prin reprezentarea datelor de intrare sub forma unor hărți de ocupanță bidimensionale, spațiul de căutare al ipotezei optime este redus, facilitându-se astfel procesul de antrenare.

Procedura de învățare se bazează pe o optimizare Pareto multi-obiectiv, care codifică o traiectorie învățată prin ponderile unei rețele neurale compusă din straturi de convoluție și LSTM.

Așa cum a fost demonstrat în secțiunea de evaluare a performanței, s-a reușit controlul unei vehicul autonom în scenarii specifice de conducere pe autostradă sau prin interiorul unui oraș. Utilizându-se ca referință informațiile rezultate din traiectoria parcursă de un șofer uman, s-au calculat erorile de viteză și a unghiului de virare.

În concluzie, se poate afirma că, folosindu-se avantajele metodei de învățare neuro-evolutivă alături de cele oferite de optimizarea Pareto multi-obiectiv, algoritmul de control propus în acest capitol a fost capabil să obțină rezultate încurajatoare, în special în scenariul de autostradă. De asemenea, după cum s-a observat, s-au obținut rezultate mai bune față de metodele utilizate pentru comparare: *end2end* și DRL.

Pentru cercetările viitoare, se dorește extinderea rețelei neurale pentru învățarea altor scenarii de conducere, cum ar fi parcarea vehiculului sau scenarii în care se reduce numărul de benzi de circulație. În această direcție, este necesară îmbunătățirea setului de antrenare. De asemenea, se consideră și o optimizare a arhitecturii rețelei neurale, prin utilizarea unui mecanism de atenție [93, 94].

# 7. Concluzii finale

## 7.1 Concluzii

Utilizarea tehnologiilor bazate pe machine learning în industria producătoare de autovehicule a dus la îmbunătățirea siguranței participanților la trafic prin funcționalități performante de asistență a conducerii sau prin monitorizarea șoferului și avertizarea acestuia în momentele când apar semne de oboseală sau lipsă de concentrare. De asemenea, au fost implementate și funcționalități care sporesc confortul în timpul călătoriilor, cum ar fi recunoașterea comenzilor vocale, furnizarea unor rute care pot să scurteze durata călătoriei, sau recomandarea traseelor care întâlnesc obiective de interes.

Creșterea volumului de date de trafic provenite de la senzori precum camerele video, LIDAR-ul, RADAR-ul sau GPS-ul a permis utilizarea algoritmilor care se bazează pe o cantitate mare de date pentru a învăța să generalizele soluțiile pentru care au fost concepuți. În cercetările realizate în cadrul acestei teze de doctorat majoritatea datelor utilizate au fost achiziționate cu ajutorul unor vehicule echipate cu senzorii necesari percepției sau au fost generate sintetic în cadrul unor medii de simulare. S-au utilizat simulatoare open-source sau medii de simulare, cum ar fi GridSim, construit pentru îndeplinirea unor sarcini specifice.

Așa cum a fost prezentat în detaliu în capitolul 4, o cerință esențială pentru ca un autovehicul să treacă la un nivel superior de automatizare este ca acesta să demonstreze abilitatea de percepție și înțelegere în timp-real a mediului din jurul său. În această direcție, s-a demonstrat că un algoritm de tip DL, optimizat pentru reducerea complexității, poate să utilizeze datele primite de la senzori în timpul rulării pentru a determina contextul în care un autovehicul se deplasează, învățând tipare specifice anumitor medii de rulare. Noutatea adusă este modul de selectare a hiperparametrilor optimi și de procesare a datelor de intrare prin translatarea acestora în griduri de ocupanță calculate *on the fly*.

Pentru a se demonstra că algoritmul DGN poate fi integrat pe un dispozitiv periferic, în capitolul 5 a fost implementată cu succes o platforma software care să repecte standardele de calitate impuse de industria constructoare de autovehicule pentru dezvoltarea componentelor software. Prin integrarea arhitecturii DGN într-o arhitectură utilizată în producția de serie, s-a generat un motor de inferență independent de platforma de dezvoltare și operare, care a fost testat și evaluat conform unor standarde de calitate bine definite.

În ultima parte a lucrării, în capitolul 6, a fost demonstrată capacitatea modelelor de ML să estimeze o traiectorie pentru controlul unui autovehicul, prin utilizarea unor metode de optimizare inovative și având disponibil un set redus de date. S-a demonstrat, de asemenea, că generarea unor date sintetice, prin intermediul unui mediu de simulare care replică dinamica unui vehicul și modelele senzorilor, poate duce la îmbunătățirea performanțelor unui sistem.

Chiar dacă în ultimii ani au fost obținute rezultate încurajatoare în ceea ce privește percepția și înțelegerea mediului în care autovehiculele de deplasează, sunt necesare îmbunătățiri pentru ca sistemele să devină suficient de robuste și să prezinte încredere. Factori externi, cum ar fi condițiile

meteo sau iluminarea, influențează rezultatele algoritmilor. De aceea, combinarea mai multor surse de informații poate elimina acest inconvenient. În ceea ce privește controlul autovehiculului, cercetările sunt departe de a se finaliza. Mediile în care se deplasează un automobil sunt foarte complexe și variate, cu situații neprevăzute care pot să apară în orice moment. De asemenea, marcajele rutiere, sau calitatea lor, diferă de la țară la țară, ceea ce conduce la o problemă foarte dificil de rezolvat. Pentru a se gestiona situațiile neprevăzute apărute în trafic, este necesar ca algoritmii să dețină capacitate de a învăța permanent din interacțiunea cu mediul și de a transmite această informație și altor automobile care pot să ajungă în aceste situații neprevăzute.

## 7.2 Contribuții originale

Contribuțiile aduse prin intermediul aceastei teze sunt grupate în următoarele trei arii de cercetare:

- percepția mediului din jurul unui autovehicul;
- industrializarea modelelor de machine learning;
- prelucrarea statistică a datelor senzoriale pentru generarea traiectorie și controlul unui vehicul autonom.

**Percepția mediului din jurul unui autovehicul**

Una dintre cele mai cercetate arii din domeniul conducerii autonome este percepția și înțelegerea mediului de deplasare. În timp ce majoritatea implementărilor utilizează date achiziționate cu ajutorul camerelor video, eforturile pe parcursul acestei teze au fost concentrate pe utilizarea unor soluții alternative. Inițial, au fost folosite medii de simulare din care datele au putut fi salvate și stocate pentru antrenare. S-a continuat prin reprezentarea mediului de deplasare sub forma unor griduri de ocupanță pentru a se reduce spațiul de căutare prin reprezentarea minimală a informației.

În continuare sunt specificate principalele contribuții aduse:

- dezvoltarea unei platforme software care să ruleze în timp-real, platformă prin intermediul căreia pot fi antrenați, validați și testați algoritmii pentru detecția și recunoașterea obiectelor; acest framework este open-source, fiind făcut public pentru comunitatea științifică cu scopul de a se facilita dezvoltarea de algoritmi specifici percepției mediului înconjurător;
- introducerea și optimizarea unui model deep learning (DGN), care învață o reprezentare a scenelor de trafic sub forma unor griduri de ocupanță. Această arhitectură este una optimizată, fiind posibilă utilizarea sa în aplicații de timp-real;
- ajustarea și selectarea hiperparametrilor utilizându-se conceptul de neuro-evoluție. Prin utilizarea algoritmilor genetici s-a asigurat obținerea unei arhitecturi optime din punct de vedere cost/performanță și, de asemenea, s-au selectat funcțiile de cost și de optimizare ideale pentru problema studiată;
- integrarea algoritmului în platforma software destinată conducerii autonome EB Robinos și evaluarea acestuia utilizându-se înregistrări efectuate pe parcursul deplasărilor de test.

**Industrializarea modelelor de machine learning**

Chiar dacă progresul tehnicilor de inteligență artificială aplicate conducerii autonome este semnificativ, prezintă totuși limitări atunci când este necesară industrializarea codului în scopul

integrării funcționalităților în automobile. Deoarece această industrie are standarde stricte în ceea ce privește procesul de dezvoltare a modulelor software, utilizarea modelor DL devine problematică. Din acest motiv în teza s-a implementat o platformă unică pentru a se genera multiple soluții bazate pe algoritmi de IA, utilizându-se un cadru de dezvoltare și validare acceptat de producătorii de autovehicule.

În continuare sunt specificate principalele contribuții aduse:

- integrarea unei variante simplificate a algoritmului DGN [59] într-o platformă hardware care permite validarea acestuia;
- evidențierea avantajelor oferite de utilizarea unor module de IA independente de sistemele de operare sau module software, în momentul proiectării, aplicării și integrării în medii diferite;
- promovarea importanței respectării anumitor standarde ca ISO26262 [78] sau ASPICE pentru a se asigura calitatea modelelor deep learning dezvoltate.

**Prelucrarea statistică a datelor senzoriale pentru generarea traiectoriei și controlul unui vehicul autonom**

Controlul traiectoriei unui autovehicul este o problema care, odată rezolvată, va duce automatizarea autovehiculelor la un alt nivel. Această sarcină este una complexă și necesită o abordare diferită, prin care să se stabilească obiectivele care trebuie atinse în timpul deplasării. De asemenea, volumul de date necesar pentru antrenarea unui astfel de sistem este foarte mare, și trebuie să acopere foarte multe situații ce pot să apară în trafic. Această teză propune utilizarea datelor senzoriale pentru estimarea unei traiectorii locale, utilizându-se o evaluare multi-obiectiv pentru procesul de optimizare..

În continuare sunt specificate principalele contribuții aduse:

- procesarea datelor senzoriale cu scopul de a antrena un model pentru controlul unui vehicul autonom;
- definirea unei rețele convoluționale-LSTM profunde pentru prezicerea comportamentelor optime de conducere ;
- introducerea unei abordări neuro-evolutive pentru antrenarea sistemelor de conducere autonome, bazată pe optimizarea Pareto multi-obiectiv și pe algoritmii genetici;
- demonstrarea eficacității prin evaluarea în comparație cu metodele *end2end* și DRL.

## 7.3 Diseminarea rezultatelor cercetării

Cercetările realizate pe parcursul studiile doctorale au dus la publicarea unui număr de articole în baze de date recunoscute la nivel internațional și în rapoartele unor conferințe. Un număr de 5 lucrări au fost publicate ca prim autor, iar pentru alte 2 au fost aduse contribuții importante:

- **Marina, L.A.**, Moldoveanu, F.D, Grigorescu, S.M. *Real-time Driving Context Understanding using Deep Grid Net: A Granular Approach*. International Journal of Robotic Computing (IJRC), Vol. 2, No./Issue 2, August 2020, p. 40-58, ISSN: 2641-9521 DOI: 10.35708/RC1869-126262, https://www.kspress.org/ijrc-2-2.

- **Marina, L.A.**, Trăsnea, B., Cociaș, T.T., Vasilcoi, A., Moldoveanu, F., Grigorescu, S.M. *Deep Grid Net (DGN): A Deep Learning System for Real-Time Driving Context Understanding*. 2019 Third IEEE International Conference on Robotic Computing (IRC), Naples, Italy, Feb. 25-27, 2019, p. 399-402, ISBN: 978-1-5386-9245-5, DOI: 10.1109/IRC.2019.00073, Accession Number WOS:000465234300064, https://doi.org/10.1109/IRC.2019.00073.

  Aceste două publicații prezintă cercetările realizate pentru clasificarea contextului în care un autovehicul se deplasează utilizându-se o reprezentare a mediului sub forma unor hărți de ocupanță. Pentru analiza datelor s-a utilizat o arhitectură de rețea neurală de convoluție, al cărei set de hiperparameri a fost determinat prin implementarea unei metode de neuro-evoluție. Pe baza acestor publicații a fost construit capitolul 4 al acestei teze de doctorat.

- Trăsnea, B., **Marina, L.A**, Vasilcoi, A., Pozna, C.R., Grigorescu, S.M. *GridSim: A Vehicle Kinematics Engine for Deep Neuroevolutionary Control in Autonomous Driving*. 2019 Third IEEE International Conference on Robotic Computing (IRC), Naples, Italy, Feb. 25-27, 2019, p. 443-444, ISBN: 978-1-5386-9245-5, DOI: 10.1109/IRC.2019.00091, Accession Number WOS: 000465234300082, https://ieeexplore.ieee.org/document/8675568.

  Lucrarea propune un simulator de conducere autonomă, denumit GridSim, care poate fi utilizat pentru generarea gridurilor de ocupanță prin intermediul unor senzori emulați. Mediul de simulare a fost utilizat pentru a studia performanța a două tipuri de algoritmi utilizați pentru controlul unui vehicul autonom: deep reinforcement learning și algortimii genetici. Datele generate au fost utilizate pentru testare metodelor descrise în capitolele 4 și 6.

- Grigorescu, S.M., Trăsnea, B., **Marina, L.A.**, Vasilcoi, A., Cociaș, T.T. *NeuroTrajectory: A Neuroevolutionary Approach to Local State Trajectory Learning for Autonomous Vehicles*. IEEE Robotics and Automation Letters, Vol. 4, Issue 4, Oct. 2019, p. 3441-3448, ISSN: 2377-3766, DOI: 10.1109/LRA.2019.2926224, Accession Number WOS: 000477983400005 (FI = 3.608), https://doi.org/10.1109/LRA.2019.2926224.

  În această publicație introduce o soluție pentru generarea unei traiectorii locale, prin implementarea unei abordări de tip multi-obiectiv bazată pe neuro-evoluție. Sunt utilizați algoritmii genetici pentru antrenarea unei populații de rețele neurale, unde fiecare individ este evalut prin intermediul unui front Pareto. Rezultatele acestei metode sunt prezentate, în detaliu, în capitolul 6.

- **Marina, L.A.**, Trăsnea, B., Grigorescu, S.M. *A Multi-Platform Framework for Artificial Intelligence Engines in Automotive Systems*. 22nd International Conference on System Theory, Control and Computing conference (ICSTCC), Sinaia, România, Oct. 10-12, 2018, România, p. 559-564, ISSN: 2372-1618, ISBN: 978-1-5386-4444-7, DOI: 10.1109/ICSTCC.2018.8540753, Accession Number WOS: 000465109800092, https://ieeexplore.ieee.org/document/8540753.

  Deoarece industria constructoare de automobile se bazează pe proceduri și standarde de siguranță și calitate, s-a dezvoltat o platforma software care poate să asigure generarea unor soluții bazate pe tehnicile de inteligență artificială care să asigure portabilitatea algoritmilor, independența de platforma de dezvoltare, și urmărirea unui proces de dezvoltare acceptat în industrie. Algoritmii generați în acest mod, au fost testați utilizându-se o machetă a unui vehicul, complet echipată pentru conducerea autonomă. Capitolul 5 prezintă detaliile de implementare și rezultatele obținute.

- **Marina, L.A.**, Sandu, A. *Deep Reinforcement Learning for Autonomous Vehicles - State of the Art*. Bulletin of The Transilvania University of Brașov, Vol. 10 (59) No. 2, 2017 Series I - Engineering Science, p. 195-203, ISSN 2065-2119 (Print), ISSN 2065-2127, http://webbut.unitbv.ro/bulletin/Series%20I/2017/BULETIN%20I/Marina_L.pd.

Această lucrare a analizat stadiul actual privind metodele de reinforcement learning. Prin înțelegerea metodelor și progresului în acest domeniu, s-au putut realiza exprimentele și comparațiile din cadrul capitolului 6.

- **Marina, L.A.**, Moldoveanu, F., Grigorescu, S.M. *Environment Perception in Racing Simulators Using Deep Neural Networks*. 2017 Joint International Conference on Optimization of Electrical and Electronic Equipment & 2017 Aegean Conference on Electrical Machines and Power Electronics – OPTIM – ACEMP, Brașov, România, May 25-27, 2017, p. 1102–1107, ISBN: 978-1-5090-4488-7, DOI: 10.1109/OPTIM.2017.7975119, Accession Number WOS: 000426909600172 https://ieeexplore.ieee.org/document/7975119.

  În această lucrare s-a dezvoltat un algoritm de detecție și recunoaștere a obiectelor, capabil să ruleze în timp-real în cadrul unui mediu de simulare. Această abordare s-a bazat pe rețele neurale de convoluție și a avut ca scop propunerea unui soluții pentru detectarea participanților la trafic. Capitolul 3 extinde rezultatele și metodele teoretice din această publicație.

De asemenea, inovația cercetărilor din cadrul studiilor doctorale a fost recunoscută prin 2 patente internaționale:

- **Marina, L.A.** , Grigorescu, S.M., *Determination of the Driving Context of a Vehicle*, patenteu, Elektrobit Automotive GmbH, 2020.
- Vasilcoi, A., Radu, P., Grigorescu, S.M., **Marina, L.A.** *Convolutional Neural Network with Reduced Complexity*, patenteu, Elektrobit Automotive GmbH, 2020.

## 7.4 Direcții viitoare de cercetare

În cadrul acestei tezei de doctorat s-a demonstrat faptul că utilizându-se algoritmi bazați pe rețele neurale se poate realiza cu succes percepția mediului în care un autovehicul se deplasează. Așa cum a fost menționat în capitolele anterioare, percepția mediului este o etapă crucială pentru implementarea funcționalităților de conducere autonomă. În capitolul 6, utilizându-se informațiile despre mediul din jurul unui autovehicul, mediu reprezentat sub forma unor griduri de ocupanță, alături de date cum ar fi poziția și viteza unui autovehicul, s-a implementat un algoritm pentru controlul și generarea unei traiectorii locale.

Capitolul 4 prezintă detaliile de implementare ale algoritmului *Deep Grid Net* (DGN), o soluție pentru detectarea contextului în care se deplasează un autovehicul. Utilizat pentru clasificarea scenei de trafic prin analiza unui grid de ocupanță, acest algoritm are ca particularitate faptul ca datele sunt fuzionate sub forma unor hărți de ocupanță în timp-real, nefiind necesară o acumulare a priori de informații.

Validarea acestei metode a fost realizată prin integrarea într-o platformă software dedicată conducerii autonome EB Robinos [61], fapt ce a asigurat o testare robustă, în timp-real, cu ajutorul unui autovehicul complet echipat pentru conducerea autonomă.

Chiar dacă au fost depuse eforturi considerabile pentru a construi o soluție robustă, cu performanțe ridicate, au fost detectate și posibilități de îmbunătățire. Una dintre posibilele îmbunătățiri ce pot fi aduse algoritmului DGN este extinderea setului de date de antrenare și testare. Această extindere poate fi făcută atât prin creșterea numărului și calității eșantionelor utilizate pentru clasificarea curentă, cât și prin creșterea numărului de clase. Noile clase ce pot fi considerate sunt: zone în construcție, intrări și ieșiri pe, respectiv de pe, autostradă, accidente rutiere sau situații de drum îngustat. De asemenea, setul de date poate fi extins și prin generarea unor eșantioane sintetice, prin utilizarea algoritmului GOL [95] sau prin utilizarea unor medii de simulare special dezvoltate pentru implementarea funcționalităților de conducere autonomă [89].

O altă direcție de cercetare pentru îmbunătățirea rezultatelor algoritmului DGN este utilizarea unei tehnici de fuziune de date, prin care să se combine gridurile de ocupanță cu fluxul provenit de la camerele video. Prin această abordare multi-modală, se dorește creșterea acurateței și a capacității de generalizare a rețele neurale. Un dezavantaj iminent este faptul că această soluție va dezvolta arhitectura DGN, rezultând într-o complexitate mai ridicată. De asemenea, este nevoie de alinierea și sincronizarea datelor, ceea ce presupune un efort considerabil.

Deoarece fluxul informațiilor provenit de la senzori este reprezentat ca o secvență temporară, se pot utiliza și arhitecturi bazate pe rețele neurale recurente. Această soluție a fost testată prin utilizarea straturilor de tip LSTM, dar pentru această problemă de clasificare multiplă rezultatele au fost inferioare arhitecturii DGN. Pot fi considerate, de asemenea, și alte tipuri de straturi recurente, cum ar fi *Gated Recurrent Units* (GRU). Aceste unități rezolvă problema minimizării excesive a gradientului și prezintă o complexitate redusă în comparație cu unitățile LSTM.

În cadrul capitolului 5 a fost prezentată o platformă pentru generarea și testarea algoritmilor de machine learning ce înglobează funcționalități de conducere autonomă. Algoritmii au fost dezvoltați urmărindu-se procesele specifice industriei constructoare de autovehicule, impuse pentru a se respecta standardele de calitate cerute. Motoarele de inferență rezultate au fost integrate pe diferite platforme, realizându-se astfel un pas important către industrializarea codului sursă.

Pentru a facilita integrarea algoritmilor dezvoltați în acestă teză pe platforme hardware cu limitări în ceea ce privește performanța, se pot utiliza tehnici de ternarizare (reprezentarea ponderilor prin valori din cadrul unui set finit, de exemplu -1, 0, 1) și cuantizare (proces de reducere a numărului de biți pentru a reprezenta un singur parametru scalar), tehnici ce nu au fost implementate până în acest moment.

O direcție viitoare de cercetare pentru a îmbunătății algoritmul de generare a unei traiectorii locale descris în capitolull 6 este optimizarea topologiei rețele neurale utilizate pentru îndeplinirea acestei sarcini. Pentru a crește performanța sistemului propus, se dorește implementarea unui mecanism de atenție [93, 94]. Această tehnică imită atenția cognitivă, acordând o pondere mai mare anumitor părți din datele de intrare, strategia fiind ca rețeaua să se concentreze pe secțiuni mai mici, dar mai importante ale datelor.

În general, direcțiile viitoare de cercetare se concentrează pe extinderea și îmbunătățirea seturilor de date, tehnici noi de augmentare a datelor și actualizarea arhitecturilor rețelelor neurale prin implementarea unor tehnologii noi.

# Bibliografie


[1] Santokh Singh. „Critical Reasons for Crashes Investigated in the National Motor Vehicle Crash Causation Survey". 2015.

[2] Travis J. Crayton, Benjamin Mason Meier. „Autonomous vehicles: Developing a public health research agenda to frame the future of transportation policy". *Journal of Transport and Health* volume 6 (2017), pp. 245–252. doi: https://doi.org/10.1016/j.jth.2017.04.004.

[3] SAE Committee. *Taxonomy and definitions for terms related to on-road motor vehicle automated driving systems*. 2014.

[4] Zhenpei Yang, Yuning Chai, Dragomir Anguelov, Yin Zhou, Pei Sun, Dumitru Erhan, Sean Rafferty, Henrik Kretzschmar. *SurfelGAN: Synthesizing Realistic Sensor Data for Autonomous Driving*. 2020. arXiv: 2005.03844 [cs.CV].

[5] Alex Krizhevsky, Ilya Sutskever, Geoffrey E Hinton. „ImageNet Classification with Deep Convolutional Neural Networks". *Advances in Neural Information Processing Systems 25*. Curran Associates, Inc., 2012, pp. 1097–1105.

[6] Marcin Andrychowicz, Bowen Baker, Maciek Chociej, Rafal Jozefowicz, Bob McGrew, Jakub Pachocki, Arthur Petron, Matthias Plappert, Glenn Powell, Alex Ray. „Learning dexterous in-hand manipulation". *arXiv preprint arXiv:1808.00177* (2018).

[7] Yoav Goldberg, Graeme Hirst. *Neural Network Methods in Natural Language Processing*. Morgan & Claypool Publishers, 2017.

[8] H. Zhu, K. Yuen, L. Mihaylova, H. Leung. „Overview of Environment Perception for Intelligent Vehicles". *IEEE Transactions on Intelligent Transportation Systems* volume 18.10 (2017), pp. 2584–2601. doi: 10.1109/TITS.2017.2658662.

[9] Joel Janai, Fatma Güney, Aseem Behl, Andreas Geiger. *Computer Vision for Autonomous Vehicles: Problems, Datasets and State-of-the-Art*. 2017. arXiv: 1704.05519 [cs.CV].

[10] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein. „Imagenet large scale visual recognition challenge". *International journal of computer vision* volume 115.3 (2015), pp. 211–252.

[11] Zhong-Qiu Zhao, Peng Zheng, Shou-tao Xu, Xindong Wu. „Object detection with deep learning: A review". *IEEE transactions on neural networks and learning systems* (2019).

[12] Shivang Agarwal, Jean Ogier du Terrail, Frédéric Jurie. „Recent Advances in Object Detection in the Age of Deep Convolutional Neural Networks". *CoRR* volume abs/1809.03193 (2018). arXiv: 1809.03193.

[13] Joseph Redmon, Santosh Kumar Divvala, Ross B. Girshick, Ali Farhadi. „You Only Look Once: Unified, Real-Time Object Detection". *CoRR* volume abs/1506.02640 (2015). arXiv: 1506.02640.

[14] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott E. Reed, Cheng-Yang Fu, Alexander C. Berg. „SSD: Single Shot MultiBox Detector". *CoRR* volume abs/1512.02325 (2015). arXiv: 1512.02325.

[15] Joseph Redmon, Ali Farhadi. „YOLO9000: Better, Faster, Stronger". *CoRR* volume abs/1612.08242 (2016). arXiv: 1612.08242.

[16] Joseph Redmon, Ali Farhadi. „YOLOv3: An Incremental Improvement". *CoRR* volume abs/1804.02767 (2018). arXiv: 1804.02767.

[17] Ross B. Girshick. „Fast R-CNN". *CoRR* volume abs/1504.08083 (2015). arXiv: 1504.08083.

[18] Shaoqing Ren, Kaiming He, Ross B. Girshick, Jian Sun. „Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks". *IEEE Transactions on Pattern Analysis and Machine Intelligence* volume 39 (2015), pp. 1137–1149.

[19] Ross Girshick, Jeff Donahue, Trevor Darrell, Jitendra Malik. „Rich feature hierarchies for accurate object detection and semantic segmentation". *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2014, pp. 580–587.

[20] Alberto Garcia-Garcia, Sergio Orts-Escolano, Sergiu Oprea, Victor Villena-Martinez, José Garcı a Rodrı guez. „A Review on Deep Learning Techniques Applied to Semantic Segmentation". *CoRR* volume abs/1704.06857 (2017). arXiv: 1704.06857.

[21] A. Krizhevsky, I. Sutskevert, G. E. Hinton. „ImageNet Classification with Deep Convolutional Neural Networks". *Advances in Neural Information Processing Systems 25 (NIPS)*. 2016.

[22] K. Simonyan, A. Zisserman. „Very deep convolutional networks for large-scale image recognition". *CoRR* volume abs/1409.1556 (2014).

[23] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott E. Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, Andrew Rabinovich. „Going Deeper with Convolutions". *CoRR* volume abs/1409.4842 (2014). arXiv: 1409.4842.

[24] Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun. „Deep Residual Learning for Image Recognition". *CoRR* volume abs/1512.03385 (2015).

[25] Vijay Badrinarayanan, Alex Kendall, Roberto Cipolla. „SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation". *CoRR* volume abs/1511.00561 (2015). arXiv: 1511.00561.

[26] R. Goeddel, E. Olson. „Learning semantic place labels from occupancy grids using CNNs". *Intelligent Robots and Systems (IROS)*. IEEE/RSJ International Conference, 2016.

[27] P. Ondruska, J. Dequaire, D. Zeng Wang, I. Posner. „End-to-End Tracking and Semantic Segmentation Using Recurrent Neural Network". *Robotics: Science and Systems*. 2016.

[28] P. Ondruska, I. Posner. „Deep tracking: Seeing beyond seeing using recurrent neural networks". *The Thirtieth AAAI Conference on Artificial Intelligence (AAAI)*. Phoenix, Arizona USA, 2016.

[29] J. Dezert, J. Moras, B. Pannetier. „Environment perception using grid occupancy estimation with belief functions". *Information Fusion (Fusion)*. International Conference, 2015.

[30] F. Smarandache, J. Dezert. „Modified PCR Rules of Combination with Degrees of Intersections". *Proc. of Fusion*. USA, 2015.

[31] C. Seeger, A. Müller, L. Schwarz. „Towards Road Type Classification with Occupancy Grids". *Intelligent Vehicles Symposium - Workshop: DeepDriving - Learning Representations for Intelligent Vehicles, IEEE*. Gothenburg, Sweden, 2016.

[32] Alex Kendall, Matthew Grimes, Roberto Cipolla. „Convolutional networks for real-time 6-DOF camera relocalization". *CoRR* volume abs/1505.07427 (2015). arXiv: 1505.07427.

[33] Florian Walch, Caner Hazirbas, Laura Leal-Taixé, Torsten Sattler, Sebastian Hilsenbeck, Daniel Cremers. „Image-based Localization with Spatial LSTMs". *CoRR* volume abs/1611.07890 (2016). arXiv: 1611.07890.

[34] Sepp Hochreiter, Jürgen Schmidhuber. „Long Short-Term Memory". *Neural Computation* volume 9.8 (1997), pp. 1735–1780. doi: 10.1162/neco.1997.9.8.1735. eprint: https://doi.org/10.1162/neco.1997.9.8.1735.

[35] W. Lu, Y. Zhou, G. Wan, S. Hou, S. Song. „L3-Net: Towards Learning Based LiDAR Localization for Autonomous Driving". *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019, pp. 6382–6391. doi: 10.1109/CVPR.2019.00655.

[36] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, Martin A. Riedmiller. „Playing Atari with Deep Reinforcement Learning". *CoRR* volume abs/1312.5602 (2013). arXiv: 1312.5602.

[37] Lingli Yu, Xuanya Shao, Yadong Wei, Kaijun Zhou. „Intelligent Land-Vehicle Model Transfer Trajectory Planning Method Based on Deep Reinforcement Learning". *Sensors*. 2018.

[38] Aleksandr I. Panov, Konstantin S. Yakovlev, Roman Suvorov. „Grid Path Planning with Deep Reinforcement Learning: Preliminary Results". 2018.

[39] Liviu Marina, Andreea Sandu. „Deep Reinforcement Learning for Autonomous Vehicles - State of the art". *Bulletin of the Transilvania University of Braşov* volume 10.2 (2017), pp. 195–203.

[40] Alexey Dosovitskiy, German Ros, Felipe Codevilla, Antonio Lopez, Vladlen Koltun. „CARLA: An Open Urban Driving Simulator". *Proceedings of the 1st Annual Conference on Robot Learning*. volume 78. Proceedings of Machine Learning Research. PMLR, 2017, pp. 1–16.

[41] Shital Shah, Debadeepta Dey, Chris Lovett, Ashish Kapoor. „AirSim: High-Fidelity Visual and Physical Simulation for Autonomous Vehicles". *CoRR* volume abs/1705.05065 (2017). arXiv: 1705.05065.

[42] *Deepdrive*. https://deepdrive.io. 2018.

[43] *Udacity Open Sources Its Self-Driving Car Simulator*. https://techcrunch.com/2017/02/08/udacity-open-sources-its-self-driving-car-simulator-for-anyone-touse/?guccounter=1. 2018.

[44] *Virtual-Based Safety Testing for Self-Driving Cars from NVIDIA DRIVE Constellation*. https://www.nvidia.com/en-us/self-driving-cars/drive-constellation. 2018.

[45] *"Carcraft" Is Waymo's Virtual World for Autonomous Vehicle Testing*. https://www.engadget.com/2017/08/23/waymo-virtual-world-carcraft. 2018.

[46] Valeriano Mendez, Heliodoro Catalan, Joan R Rosell, Jaume Arnó, Ricardo Sanz, Ana Tarquis. „SIMLIDAR–Simulation of LIDAR performance in artificially simulated orchards". *Biosystems engineering* volume 111.1 (2012), pp. 72–82.

[47] St. Bechtold, Bernhard Höfle. „Helios: a Multi-Purpose LIDAR Simulation Framework for Research, Planning and Training of Laser Scanning Operations with Airborne, Ground-Based Mobile and Stationary Platforms". 2016.

[48] John O. Woods, John A. Christian. „Glidar: An OpenGL-based, Real-Time, and Open Source 3D Sensor Simulator for Testing Computer Vision Algorithms". *J. Imaging* volume 2 (2016), pp. 5.

[49] *Radar Simulator (RADSim)|Riverside Research*. https://www.riversideresearch.org/. 2018.

[50] Bernhard Wymann, Christos Dimitrakakisy, Andrew Sumnery, Christophe Guionneauz. *TORCS: The open racing car simulator*. 2015.

[51] Nick Jacobi, Phil Husbands, Inman Harvey. „Noise and the Reality Gap: The Use of Simulation in Evolutionary Robotics". *Proceedings of the Third European Conference on Advances in Artificial Life*. Berlin, Heidelberg: Springer-Verlag, 1995, pp. 704–720.

[52] S. Koos, J. Mouret, S. Doncieux. „The Transferability Approach: Crossing the Reality Gap in Evolutionary Robotics". *IEEE Transactions on Evolutionary Computation* volume 17.1 (2013), pp. 122–145. doi: 10.1109/TEVC.2012.2185849.

[53] L. A. Marina, F. D. Moldoveanu, S. M. Grigorescu. „Environment perception in racing simulators using Deep Neural Networks". *2017 International Conference on Optimization of Electrical and Electronic Equipment (OPTIM) 2017 Intl Aegean Conference on Electrical Machines and Power Electronics (ACEMP)*. 2017, pp. 1102–1107. doi: 10.1109/OPTIM.2017.7975119.

[54] Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshick, Sergio Guadarrama, Trevor Darrell. „Caffe: Convolutional Architecture for Fast Feature Embedding". *arXiv preprint arXiv:1408.5093* (2014).

[55] Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006.

[56] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, A. Zisserman. *The PASCAL Visual Object Classes Challenge 2012 (VOC2012) Results*. http://www.pascal-network.org/challenges/VOC/voc2012/workshop/index.html.

[57] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, A. Zisserman. *The PASCAL Visual Object Classes Challenge 2007 (VOC2007) Results*. http://www.pascal-network.org /challenges/VOC/voc2007/workshop/index.html.

[58] G. Bradski. „The OpenCV Library". *Dr. Dobb's Journal of Software Tools* (2000).

[59] Liviu Alexandru Marina, Bogdan Trasnea, Tiberiu T. Cocias, Andrei Vasilcoi, Florin Moldoveanu, Sorin Mihai Grigorescu. „Deep Grid Net (DGN): A Deep Learning System for Real-Time Driving Context Understanding". *2019 Third IEEE International Conference on Robotic Computing (IRC)* (2019), pp. 399–402.

[60] A. Tchamova, J. Dezert. „On the Behavior of Dempster's Rule of Combination and the Foundations of Dempster-Shafer Theory". *Intelligent Systems (IS)*. 6th IEEE International Conference, 2016.

[61] Elektrobit. *Eb Robinos*. https://www.elektrobit.com/products/automated-driving/eb-robinos. 2015.

[62] Liviu A. Marina, Florin D. Moldoveanu, Sorin M. Grigorescu. „Imagenet large scale visual recognition challenge". *International Journal of Robotic Computing* volume 2.2 (2020), pp. 40–58. eprint: 10.35708.

[63] G. Shafer. „A Mathematical Theory of Evidence". Princeton: Princeton University Press, 1976.

[64] F. Smarandache, J. Dezert. „Advances and applications of DSmT for information fusion (Collected works)". *American Research Press, USA* volume 1-4 (2004).

[65] F. Smarandache, M. Alford. „A Class of DSm Conditional Rules". *COGIS 2009 International Conference*. Paris, 2009.

[66] A. E. Eiben, James E. Smith. *Introduction to Evolutionary Computing*. 2nd. Springer Publishing Company, Incorporated, 2015.

[67] Max Jaderberg, Valentin Dalibard, Simon Osindero, Wojciech M. Czarnecki, Jeff Donahue, Ali Razavi, Oriol Vinyals, Tim Green, Iain Dunning, Karen Simonyan, Chrisantha Fernando, Koray Kavukcuoglu. „Population Based Training of Neural Networks". *CoRR* volume abs/1711.09846 (2017).

[68] Sorin Mihai Grigorescu, Bogdan Trasnea, Liviu Marina, Andrei Vasilcoi, Tiberiu T. Cocias. „NeuroTrajectory: A Neuroevolutionary Approach to Local State Trajectory Learning for Autonomous Vehicles". *CoRR* volume abs/1906.10971 (2019). arXiv: 1906.10971.

[69] Yann Lecun, Léon Bottou, Yoshua Bengio, Patrick Haffner. „Gradient-based learning applied to document recognition". *Proceedings of the IEEE*. 1998, pp. 2278–2324.

[70] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott E. Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, Andrew Rabinovich. „Going Deeper with Convolutions". *CoRR* volume abs/1409.4842 (2014). arXiv: 1409.4842.

[71] Dasarathy Belur. *Nearest neighbor (NN) norms : NN pattern classification techniques*. Los Alamitos, Calif.:IEEE Computer Society Press, 1991.

[72] S.M. Grigorescu. „Generative One-Shot Learning (GOL): A Semi-Parametric Approach for One-Shot Learning in Autonomous Vision". *Int. Conf. on Robotics and Automation ICRA 2018*. Brisbane, Australia, 2018.

[73] Andrei Vasilcoi, Petru Radu, Liviu Marina, Bogdan Trasnea, Sorin Mihai Grigorescu. „Convolutional neural network with reduced complexity". patenteu. Elektrobit Automotive GmbH. 2020.

[74] Liviu Marina, Sorin Mihai Grigorescu. „Determination of the driving context of a vehicle". patenteu. Elektrobit Automotive GmbH. 2020.

[75] Liviu Marina, Sorin Mihai Grigorescu. „Determination of the driving context of a vehicle". patentwo. Elektrobit Automotive GmbH. 2020.

[76] L. Marina, Bogdan Trasnea, S. Grigorescu. „A Multi-Platform Framework for Artificial Intelligence Engines in Automotive Systems". *2018 22nd International Conference on System Theory, Control and Computing (ICSTCC)* (2018), pp. 559–564.

[77] D. Gunning. „Explainable artificial intelligence (XAI)". *Technical report, DARPA/I20*. 2016.

[78] John Birch, Roger Rivett, Ibrahim Habli, Ben Bradshaw, John Botham, Dave Higham, Peter Jesty, Helen Monkhouse, Robert Palin. „Safety cases and their role in ISO 26262 functional safety assessment". *International Conference on Computer Safety, Reliability, and Security*. Springer. 2013, pp. 154–165.

[79] Martin Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro. „TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems". *CoRR* volume abs/1603.04467 (2016).

[80] Grigori Fursin, Abdul Memon, Christophe Guillon, Anton Lokhmotov. „Collective Mind, Part II: Towards Performance- and Cost-Aware Software Engineering as a Natural Science." *18th International Workshop on Compilers for Parallel Computing (CPC'15)*. ArXiv, 2015.

[81] Dong Yu, Adam Eversole, Mike Seltzer, Kaisheng Yao, Zhiheng Huang, Brian Guenter, Oleksii Kuchaiev, Yu Zhang, Frank Seide. „An introduction to computational networks and the computational network toolkit". *Microsoft Technical Report MSR-TR-2014–112* (2014).

[82] Diederik P. Kingma, Jimmy Ba. „Adam: A Method for Stochastic Optimization." *CoRR* volume abs/1412.6980 (2014).

[83] Elektrobit. *ADTF*. https://www.elektrobit.com/products/automated-driving/eb-assist/adtf. 2017.

[84] Microsoft. *Microsoft Azure*. https://azure.microsoft.com/en-us/. 2014.

[85] Xinli Geng, Huawei Liang, Biao Yu, Pan Zhao, Liuwei He, Rulin Huang. „A Scenario-Adaptive Driving Behavior Prediction Approach to Urban Autonomous Driving". *Applied Sciences* volume 7.4 (2017).

[86] Mariusz Bojarski, Davide Del Testa, Daniel Dworakowski, Bernhard Firner, Beat Flepp, Prasoon Goyal, Lawrence D. Jackel, Mathew Monfort, Urs Muller, Jiakai Zhang, Xin Zhang, Jake Zhao, Karol Zieba. „End to End Learning for Self-Driving Cars". *CoRR* volume abs/1604.07316 (2016). arXiv: 1604.07316.

[87] Yann LeCun, Urs Muller, Jan Ben, Eric Cosatto, Beat Flepp. „Off-Road Obstacle Avoidance through End-to-End Learning". *Advances in Neural Information Processing Systems NIPS*. 2005.

[88] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, Demis Hassabis. „Human-level control through deep reinforcement learning". *Nature* volume 518.7540 (2015), pp. 529–533.

[89] Bogdan Trasnea, Liviu Alexandru Marina, Andrei Vasilcoi, Claudiu Pozna, Sorin Grigorescu. „GridSim: A Vehicle Kinematics Engine for Deep Neuroevolutionary Control in Autonomous Driving" (2019).

[90] S.M. Grigorescu. „Generative One-Shot Learning (GOL): A Semi-Parametric Approach for One-Shot Learning in Autonomous Vision". *Int. Conf. on Robotics and Automation ICRA 2018*. Brisbane, Australia, 2018.

[91] Felipe Petroski Such, Vashisht Madhavan, Edoardo Conti, Joel Lehman, Kenneth O. Stanley, Jeff Clune. „Deep Neuroevolution: Genetic Algorithms Are a Competitive Alternative for Training Deep Neural Networks for Reinforcement Learning". *CoRR* volume abs/1712.06567 (2018).

[92] Kalyanmoy Deb. *Multi-objective Evolutionary Optimisation for Product Design and Manufacturing*. Springer, 2011, pp. 3–34.

[93] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, Illia Polosukhin. „Attention Is All You Need". *CoRR* volume abs/1706.03762 (2017). arXiv: 1706.03762.

[94] Abenezer Girma, Seifemichael B. Amsalu, Abrham Workineh, Mubbashar Altaf Khan, Abdollah Homaifar. „Deep Learning with Attention Mechanism for Predicting Driver Intention at Intersection". *CoRR* volume abs/2006.05918 (2020). arXiv: 2006.05918.

[95] Sorin Mihai Grigorescu. „Generative One-Shot Learning (GOL): A Semi-Parametric Approach to One-Shot Learning in Autonomous Vision". *CoRR* volume abs/1812.07567 (2018). arXiv: 1812.07567.

# Rezumat

Datorită avansului tehnologic înregistrat în ultimii ani, algoritmii de inteligență artificială sunt tot mai prezenți în automobile, îndeplinind functionalități de monitorizare a șoferului, de percepție a mediului înconjurător, sau de asistență la condus. Datorită capacității de generalizare și învățare utilizând un volum mare de date, metodele de tip deep learning au beneficiat de o atenție sporită.

Această teză se concentrează pe dezvoltarea și evaluarea soluțiilor bazate pe tehnici de tip deep learning pentru procesarea datelor sezoriale recepționate de la un autovehicul, atât din situații reale de trafic, cât și din medii simulate. Obiectivul acestei lucrări este de a procesa informațiile pentru a percepe mediului din jurul unui autovehicul, clasificarea scenariilor de trafic în care se află acesta, și generarea unei traiectorii locale pe baza căreia să se poată efectua deplasarea fără a exista riscul unei coliziuni.

Chiar dacă progresul realizat în industria constructoare de autovehicule, prin aplicarea metodelor de machine learning, este remarcabil, încă există limitări și provocări deschise viitoarelor direcții de cercetare. Una dintre provocări este capacitatea unui automobil de a procesa în timp-real datele provenite de la senzori pentru a percepe mediul din jurul său. Pentru atingerea acestui obiectiv s-au dezvoltat tehnici pentru selectarea unui set optim de hiperparametri, alături de arhitecturi de rețele neurale cu un număr redus de parametrii, menținându-se un raport benefic între timpul de procesare necesar și acuratețe. Algoritmii astfel optimizați au fost evaluați utilizându-se platforme hardware, cu performanțe limitate.

Due to technological advancement in recent years, artificial intelligence algorithms are increasingly present in cars, fulfilling the functions of driver monitoring, environmental perception, or driving assistance. Due to the ability to generalize and learn using a large volume of data, deep learning methods have received increased attention.

This thesis focuses on developing and evaluating solutions based on deep learning techniques to process sensory data received from a vehicle, both from real traffic situations and from simulated environments. The thesis aims to process the information to perceive the environment around the vehicle, to classify the traffic scenarios in which a vehicle is located, and generate a local trajectory on which to move without the risk of a collision.

Even though the progress made in the automotive industry by applying machine learning methods is remarkable, there are still limitations and challenges open to future research directions. One of the challenges is the ability of a car to process real-time data from sensors to perceive the environment around it. To achieve this goal, techniques have been developed for selecting an optimal set of hyper-parameters, together with neural network architectures with a small number of parameters, while maintaining a beneficial ratio between the required processing time and accuracy. The algorithms thus optimized were evaluated using hardware platforms with limited performance.