UNIVERSITATEA TEHNICĂ
DIN CLUJ-NAPOCA

Identificarea si monitorizarea factorilor de risc interni si externi pentru siguranta traficului auto - INOVSAFE PN-III-P1-1.1-PD-2021-0247

Echipa de cercetare: Răzvan Itu, Radu Dănescu

ETAPA 2 (01/01/2023 - 31/12/2023) - Implementare sistem de percepție și analiză

Introducere - rezumatul etapei 2
1 - Proiectare, antrenare si evaluare retele neuronale si algoritmi pentru a extrage date relevante din traficul rutier 2
2 - Proiectare, antrenare si evaluare retele neuronale si algoritmi pentru a monitoriza starea soferului si a vehiculului propriu 4
3 - Implementarea sistemului de procesare pe sisteme desktop 6
4 - Integrarea retelelor neuronale si algoritmilor de viziune artificiala intr-un singur sistem cadru de procesare 6
5 - Migrarea sistemului cadru de procesare pe sisteme mobile de procesare 8
Diseminare rezultate 9
Rezumat obiective 10
Indicatori de rezultat: 10
Rezumat activități: 10
Concluzii 11
Bibliografie 11

Introducere - rezumatul etapei

Etapa 2 a avut cinci activități majore:
1 - Proiectare, antrenare si evaluare retele neuronale si algoritmi pentru a extrage date relevante din traficul rutier (O2) - partea 2
2 - Proiectare, antrenare si evaluare retele neuronale si algoritmi pentru a monitoriza starea soferului si a vehiculului propriu (O3) - partea 2
3 - Implementarea sistemului de procesare pe sisteme desktop (O2, O3) - partea 2
4 - Integrarea retelelor neuronale si algoritmilor de viziune artificiala intr-un singur sistem cadru de procesare (O2, O3)
5 - Migrarea sistemului cadru de procesare pe sisteme mobile de procesare (O1, O2) - partea 1

1 - Proiectare, antrenare si evaluare retele neuronale si algoritmi pentru a extrage date relevante din traficul rutier

În contextul vehiculelor autonome, imaginile de trafic rutier conțin adesea informații vizuale diverse, incluzând șoseaua, vehiculele, pietonii și alte obiecte relevante din scena de trafic.

Arhitectura U-Net [1], prezentată și în primul raport (etapa 1), reprezintă un model de rețea neurală convoluțională (CNN) proiectat pentru sarcini de segmentare a imaginilor, iar structura sa distinctivă îl face un model deosebit de util în contextul vehiculelor autonome pentru extragerea caracteristicilor relevante din imagini de trafic rutier. Capacitatea modelului de a captura informații de context îi permite să distingă între diferite elemente ale drumului, oferind o înțelegere detaliată a mediului înconjurător, esențială pentru navigarea autonomă în siguranță și eficientă.

În plus, arhitectura U-Net este foarte adaptabilă, permițând integrarea fără probleme a diferitelor tehnici de preprocesare și încorporarea de informații suplimentare, cum ar fi datele de adâncime sau informațiile temporale din cadre consecutive. Această adaptabilitate îl face potrivit pentru natura dinamică și diversificată a scenariilor de trafic rutier. Versatilitatea U-Net a condus la adoptarea sa largă în dezvoltarea modulelor de percepție pentru vehiculele autonome, contribuind semnificativ la extragerea caracteristicilor relevante din intrarea vizuală și, în final, îmbunătățind capacitatea vehiculului de a interpreta și răspunde la mediul său.

Modelul SS-LSTM (Spatiotemporal Self-Learning Neural Network) [2] reprezintă o arhitectură sofisticată proiectată pentru provocările unice create de datele spațio-temporale, făcându-l deosebit de valoros în contextul vehiculelor autonome. Spre deosebire de rețelele

neurale convoluționale (CNN-uri) tradiționale, care se concentrează în principal pe informațiile spațiale, modelul SS-LSTM integrează unități Long Short-Term Memory (LSTM), componente de rețele neurale recurente (RNN) create în mod specific pentru a captura dependențele temporale în date secvențiale. Această combinație permite modelului SS-LSTM să proceseze eficient atât caracteristici spațiale, cât și temporale, oferind o înțelegere cuprinzătoare a scenelor dinamice întâlnite în timpul conducere autonomă.

În prima etapă am prezentat o implementare a U-Net pentru a extrage zona de drum, precum și obiectele (vehiculele) din scena de trafic. În această etapă, am realizat o implementare proprie a rețelei SS-LSTM pentru a extrage informațiile din scena de trafic. Arhitectura SS-LSTM este concepută pentru manipularea datelor spațiotemporale, fiind potrivită pentru sarcini precum vehicule autonome, unde înțelegerea atât a aspectelor spațiale, cât și a celor temporale ale mediului este esențială. Structura rețelei modificată și implementată de mine este următoarea:

1. Partea de procesare spațială (Straturi Convoluționale):
Modelul începe cu o serie de straturi convoluționale pentru procesarea caracteristicilor spațiale ale datelor de intrare (imaginilor din scena de trafic rutier). Aceste straturi extrag caracteristici spațiale prin operațiuni de convoluție cu filtre de diferite dimensiuni. Intrarea rețelei este definită ca: "*(nr_imagini, înălțime_imagine, lățime_imagine, număr_canale)*", reprezentând o secvență de cadre (imagini) în timp.

2. Partea de procesare temporală (Straturi de tip Long Short Term Memory):
După procesarea spațială, modelul include straturi de Long Short-Term Memory (LSTM). Straturile LSTM sunt componente ale rețelei neurale recurente (RNN) proiectate pentru a captura dependențele temporale în date secvențiale. Straturile LSTM procesează caracteristicile temporale extrase de straturile convoluționale, permițând modelului să învețe și să memoreze modele în timp.

3. Straturi complet conectate (Fully Connected/Dense):
După straturile LSTM, există straturi de tip "Fully Connected/Dense" (complet conectate). Aceste straturi procesează în continuare caracteristicile spațiale și temporale combinate pentru a face predicții. Stratul final utilizează funcția de activare "softmax".

Compilarea modelului:
Modelul este compilat folosind optimizatorul Adam [3] și funcția de pierdere "categorical cross-entropy", potrivite pentru sarcinile de clasificare. Metrica de precizie este aleasă pentru evaluarea performanței modelului în timpul antrenamentului.

Astfel, modelul implementat de mine a fost folosit cu baza de date descrisă în secțiunea 4 a acestui raport. Modelul are ca intrare un set de 20 de imagini consecutive dintr-o scenă de trafic rutier, împreună cu un set de 20 de valori de viteză ale vehiculului propriu. Ieșirea modelului reprezintă valoarea vitezei curente a vehiculului propriu. Intrările din model reprezintă starea vehiculului pentru cele 20 de cadre de timp anterioare (T_{-21}:T_{-1}), iar predicția se face pentru cadrul de timp curent (T_0). Un exemplu de predicții pe un set de date este ilustrat în figura 1.

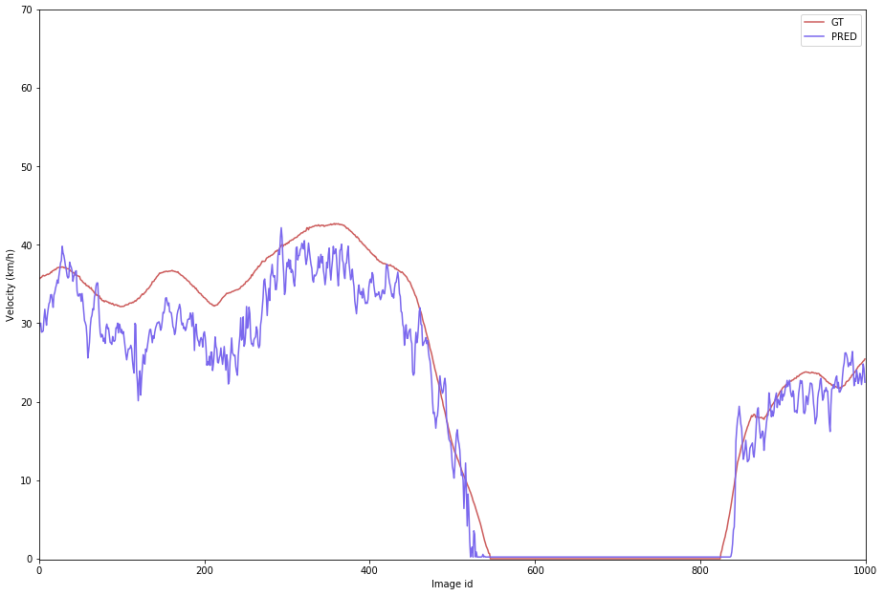

Fig. 1. Exemplu de predicție a stării curente a vehiculului (viteza), având ca intrare în model setul de 20 de imagini și viteze asociate din stările precedente. Viteza prezisă este ilustrată cu albastru, iar valoarea actuală din baza de date cu roșu. Datele sunt exprimate în km/h.

Acest model are un total de ~26 milioane de parametri antrenabili și necesită un timp mediul de ~18 milisecunde pentru o predicție.

2 - Proiectare, antrenare si evaluare retele neuronale si algoritmi pentru a monitoriza starea soferului si a vehiculului propriu

Modelul implementat de mine în secțiunea anterioară este bazat pe rețele neuronale recurente, care sunt folosite și pentru analiză stării curente a vehiculelor din traficul rutier, bazat pe date din cadrele din trecut. Pentru a implementa un sistem de monitorizare a stării șoferului și a vehiculului propriu am ales folosirea unei soluții bazată pe o rețea convoluțională neuronală, și nu prin folosirea unei rețele neuronale recurente, deoarece acestea din urmă au o dimensiune mai mare (număr de parametri mai mare) și implicit vor fi mai greu de exportat și utilizat în sisteme cu constrângeri legate de hardware (în special dimensiunea memoriei).

Astfel, modelul propus de mine pentru a monitoriza starea vehiculului propriu este următorul:

Modelul primește două intrări, similar cu modelul bazat pe recurență implementat și prezentat în secțiunea 1. Intrările sunt reprezentate de o secvență de 20 de cadre consecutive din trecut ale scenei de trafic rutier, unde fiecare cadru are:
1. imaginea scenei de trafic (T_n)
2. valoarea vitezei vehiculului propriu pentru acea imagine (T_n)

Rezultatul modelului constă într-o singură ieșire, care reprezintă viteza vehiculului propriu din cadrul de timp curent.

Prima intrare a rețelei neuronale convoluționale utilizează un set de 20 de imagini cu dimensiuna 300x300 de pixeli. Acestea sunt imagini de color, formatul RGB pe 3 canale, și captează informațiile vizuale.

Procesarea celor 20 de imagini de intrare este realizată prin mai multe straturi/niveluri convoluționale:
- C1: Strat convoluțional cu 24 de filtre de dimensiune 5x5. Apoi se aplică funcția de activare ReLU pentru a introduce non-linearitate.
- C2: Strat convoluțional cu 36 de filtre de dimensiune 5x5, urmat de activare ReLU.
- C3: Strat convoluțional cu 48 de filtre de dimensiune 5x5, urmat de activare ReLU.
- C4: Strat convoluțional cu 64 de filtre de dimensiune 3x3, folosind activarea ReLU.
- C5: Strat convoluțional cu 64 de filtre de dimensiune 3x3, folosind activarea ReLU.

Apoi se aplică un strat de tip “dropout” după ultimul strat convoluțional.

Cea de-a doua intrare a modelului propus de mine, reprezintă viteza vehiculului propriu, care este structurată sub forma unui semnal unidimensional de dimensiune 20. Ieșirea din nivelul de tip “dropout” (rezultatul procesării primei intrări) și a doua intrare (vectorul de viteză) sunt apoi concatenate.

După concatenare, rețeaua continuă cu o serie de straturi complet conectate (en. “Fully Connected”, denumite și dense):
- S1: Strat dens cu 100 de unități și funcție de activare ReLU.
- S2: Strat dens cu 50 de unități și funcție de activare ReLU.
- S3: Strat dens cu 10 unități și funcție de activare ReLU.

Stratul/nivelul final al rețelei va fi un nivel de tip dens singură unitate. Ieșirea modelului va fi viteza vehiculului pentru cadrul de timp curent, fiind prezisă pe baza informațiilor concatenate și procesate din imagini și viteza vehiculului din cadrele de timp anterioare.

Un exemplu de predicții pe un set de date este ilustrat în figura 2.

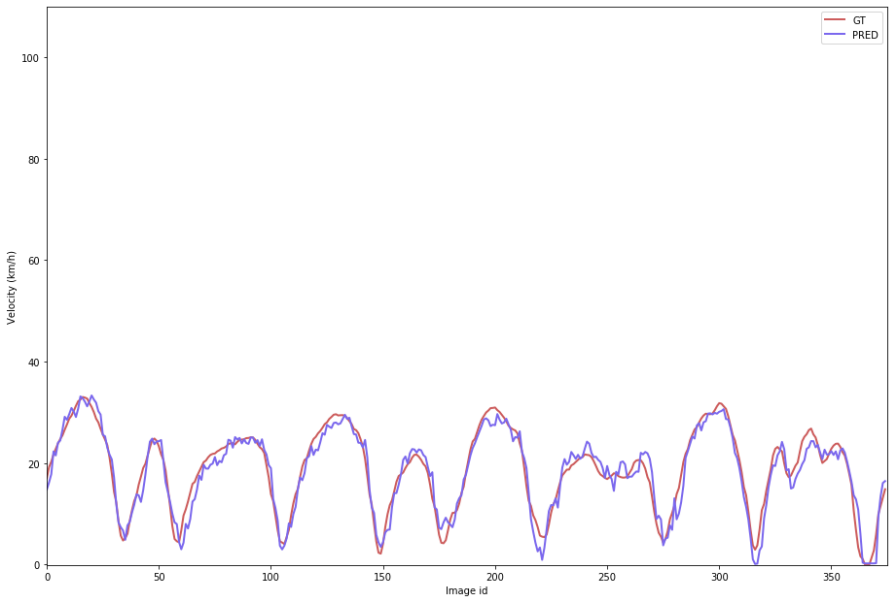

Fig. 2. Exemplu de predicții a vitezei curente (albastru) și viteza reală din baza de date (roșu). Valorile sunt exprimate în km/h.

Modelul propus de mine are un total de ~4.9 milioane de parametri antrenabili și un timp mediul predicție de ~3 milisecunde.

3 - Implementarea sistemului de procesare pe sisteme desktop

Configurația experimentală folosește un computer desktop echipat cu un procesor Intel Core i7, împreună cu două plăci video Nvidia 1080 Ti, având o capacitate totală de memorie video (VRAM) de 22 GB. Aceste două plăci video sunt utilizate în faza de antrenare a rețelei neurale. Pentru a gestiona eficient constrângerile de memorie ale stației de lucru de tip desktop, întreg setul de date de la Honda Research Institute este accesat direct de pe hard disk-uri în timpul antrenării. Dezvoltarea software pentru configurația experimentală este realizată în limbajul de programare Python și cu ajutorul framework-urilor TensorFlow [4] și Keras [5], care servesc drept baze pentru implementarea rețelei neurale. În plus, bibliotecile OpenCV [6] și Matplotlib sunt folosite pentru vizualizarea și generarea de videoclipuri care prezintă rezultatele obținute din rețeaua neurală.

4 - Integrarea retelelor neuronale si algoritmilor de viziune artificiala intr-un singur sistem cadru de procesare

Pentru a integra modelele dezvoltate într-o singură stație de lucru, trebuie să mă asigur mai întâi că toate datele sunt disponibile pe același sistem/desktop. Acest lucru asigură că antrenarea este realizată bine și datele de antrenare nu sunt modificate. Există multe baze de date disponibile public pentru dezvoltarea vehiculelor autonome bazate pe viziune, dar

puține dintre ele includ secvențe continue de imagini (sau video) împreună cu date de măsurare suplimentare (cum ar fi orientarea, viteza vehiculului, datele senzorului de frână etc.). Acesta este motivul principal pentru care am decis să folosesc baza de date Honda Deep Drive (HDD) [7], disponibilă în scopuri de cercetare. Baza de date de la Honda Research Institute a fost obținută cu permisiunea lor scrisă și a fost descărcată local pe desktop (pe stocarea HDD și SSD).

Setul de date HDD conține aproximativ 104 ore de înregistrări continue video și date de conducere din Statele Unite (zona San Francisco), în scene și situații de trafic rutier. Informațiile suplimentare din baza de date includ metadate referitoare la comportamentul șoferului, care depășesc datele limitate din alte baze de date (care de obicei se bazează pe indicații simple precum viraj la stânga, la dreapta, pentru estimarea comportamentului sau starea vehiculului). Baza de date HDD include informații senzoriale obținute de la rețeaua Controller Area Network (CAN bus), cum ar fi viteza vehiculului propriu, presiunea pe pedala de accelerație și de frână, fiind o bază de date potrivită pentru evaluarea modului în care șoferii se comportă atunci când interacționează cu ceilalți participanți din traficul rutier.

Am folosit și simulatorul open-source CARLA [8] pentru crearea unor noi baze de date cu scopul antrenării unor rețele neuronale pentru detecția de obstacole din traficul rutier.

Instalarea simulatorului CARLA folosind comenzi din consolă:
docker pull carlasim/carla:0.9.9
docker images
sudo docker run -d -p 2000-2002:2000-2002 --runtime=nvidia -e NVIDIA_VISIBLE_DEVICES=0 carlasim/carla:0.9.9 /bin/bash CarlaUE4.sh

Comanda pentru generarea traficului rutier dintr-un scenariu:
python2 generate_traffic.py -w 15 -v 30

Comenzi pentru a schimba scenariul (diferite orașe):
python2 ./config.py -m Town01

Comanda pentru a porni simulatorul cu comenzi manuale pentru ego-vehicul:
python2 manual_control.py

Exemplu de imagine din scenarii de trafic rutier folosind simulatorul CARLA:

Fig. 3. Scenă de trafic rutier din simulatorul CARLA.

Am adoptat un format standardizat propriu pentru toate imaginile și informațiile din bazele de date publice, precum și pentru datele generate din simulator. Acestea sunt accesibile printr-un sistem de tip desktop.

5 - Migrarea sistemului cadru de procesare pe sisteme mobile de procesare

Pentru a migra sistemul pe dispozitive portabile sau mobile, a trebuit să proiectez și să propun o rețea neuronală ușoară care să se potrivească cu constrângerile de memorie și alte restricții hardware ale acestor dispozitive. Abordarea utilizată în general constă în exportarea modelului propus folosind instrumentele disponibile.

Implementarea actuală a modelului CNN este scrisă în limbajul de programare Python și folosește TensorFlow, în mod specific biblioteca Keras. Modelul antrenat poate fi salvat folosind următoarea comandă:
model.save("CNN_model.h5")

Salvarea ponderilor modelului poate fi realizată cu următoarea comandă:
model.save_weights(filepath, overwrite=True)

Utilizarea funcției "save" este mai bună, datorită faptului că va salva automat arhitectura modelului, ponderile acestuia și starea optimizatorului. Încărcarea modelului salvat folosind Keras/TensorFlow se realizează cu comanda:
model.load_model(filepath)

Dacă e nevoie de optimizare a modelului pentru implementarea pe dispozitive cu hardware și mai redus, se poate converti în format TensorFlow Lite. Acest pas este opțional și depinde de hardware-ul ales. Codul este:

```
converter = tf.lite.TFLiteConverter.from_keras_model(model)
tflite_model = converter.convert()
with open("CNN_model.tflite", "wb") as f:
    f.write(tflite_model)
```

Fișierul modelului salvat (de exemplu, *CNN_model.h5* sau *CNN_model.tflite*) se poate transfera pe dispozitivul NVIDIA Jetson. Se poate folosi comanda "*scp*" sau copiere directă folosind un stick USB.

Încărcarea modelul salvat pe Jetson se face folosind liniile următoare:
```
loaded_model = load_model("CNN_model.h5")
```

Trebuie instalate toate dependențele necesare pe Jetson, inclusiv orice biblioteci legate de GPU pentru a beneficia de accelerare grafică (GPU).

Utilizarea modelul transferat pe platforme de tipul Nvidia Jetson se face folosind următoarea linie de cod:
```
output = loaded_model.predict(input_data)
```

Modelul propus de mine va fi testat pe Nvidia Jetson Nano sau pe Jetson Xavier AGX, dispozitive disponibile în laboratorul de cercetare.

Diseminare rezultate

Conferințe:
1. Razvan Itu, Radu Danescu, “On-Board Estimation of Vehicle Speed and The Need of Braking Using Convolutional Neural Networks”, In Proceedings of the 20th International Conference on Informatics in Control, Automation and Robotics (ICINCO 2023), in print.

2. Razvan Itu, Radu Danescu, “Predicting Emergency Braking in Vehicles Using a CNN with Sequential Image and Velocity Data”, 2023 IEEE 19th International Conference on Intelligent Computer Communication and Processing (ICCP 2023), in print.

Jurnal ISI:
1. Razvan Itu, Radu Danescu, “Fully Convolutional Neural Network for Vehicle Speed and Emergency Brake Prediction”, Intelligent Vehicle Sensing and Monitoring, Sensors, under review.

Rezumat obiective

Obiectivele prezentate în propunerea de proiect sunt următoarele:

O1. Crearea unei platforme hardware și software pentru achiziția, vizualizarea, organizarea și procesarea datelor de imagine și senzoriale (en. “Creating a hardware and software platform for the image and sensorial data acquisition, visualization, organization and processing”).

O2. Extragerea caracteristicilor relevante ale scenei de trafic din imagini monoculare (en. “Extracting the relevant traffic scene features from monocular images”).

O3. Analiza stării ego-vehiculului și a comportamentului șoferului din imagini de trafic și datele vehiculului (en. “Analyzing ego-vehicle state and driver behaviour from traffic imagery and vehicle data”).

Gradul de realizare al obiectivelor:
Obiectivul 1 este realizat în proporție de aproximativ 90%. Pentru îndeplinirea completă a acestui obiectiv este nevoie să termin de exportat și implementat soluția propusă pe sisteme mobile (Nvidia Jetson sau Xavier).
Obiectivul 2 este realizat complet.
Obiectivul 3 este realizat complet.

Indicatori de rezultat:

Articole ISI: 1 articol de jurnal în evaluare (ISI Q1 - zona roșie)
Articole conferințe ISI: 2 articole

Rezumat activități:

1 - Proiectare, antrenare si evaluare retele neuronale si algoritmi pentru a extrage date relevante din traficul rutier (O2) - partea 2
În această etapă, am implementat o versiune proprie și modificată a modelului SS-LSTM, pentru a extrage informații din scena de trafic rutier. Acest model utilizează 20 de imagini consecutive și viteza vehiculului propriu pentru a prezice viteza curentă. Modelul a fost testat pe o bază de date publică și date sintetice din simulator.

2 - Proiectare, antrenare si evaluare retele neuronale si algoritmi pentru a monitoriza starea soferului si a vehiculului propriu (O3) - partea 2
Pentru proiectarea unui sistem de monitorizare a șoferului și a vehiculului, am folosit o soluție bazată pe o rețea convoluțională neuronală, evitând utilizarea unei rețele neuronale recurente din considerente legate de dimensiunea mai mică a parametrilor. Modelul primește două intrări și are o singură ieșire, reprezentând starea vehiculului în cadrul temporal curent.

3 - Implementarea sistemului de procesare pe sisteme desktop (O2, O3) - partea 2
Software-ul pentru configurarea experimentală este dezvoltat în limbajul de programare Python, utilizând framework-urile TensorFlow și Keras pentru implementarea rețelelor neuronale. Pentru vizualizare, sunt folosite bibliotecile OpenCV și Matplotlib.

4 - Integrarea retelelor neuronale si algoritmilor de viziune artificiala intr-un singur sistem cadru de procesare (O2, O3)
Am folosit un format comun pentru toate imaginile și informațiile din bazele de date publice, dar și pentru datele generate din simulator. Toate sunt disponibile pe un sistem de tip desktop.

5 - Migrarea sistemului cadru de procesare pe sisteme mobile de procesare (O1, O2) - partea 1
Am început configurarea plăcii Nvidia Jetson Nano pe care va fi testată soluția propusă de mine. Am început și partea de documentare și testare a diferitelor posibilități de exportare a modelului de pe sisteme de tip desktop, pentru cele portabile (Nvidia Jetson sau Xavier).

Concluzii

În această etapă am reușit implementarea cu succes a unor soluții de procesare și percepție a scenei de trafic rutier direct din imagini, dar și prin folosirea unor semnale adiționale (de exemplu: viteza vehiculului), cu scopul de a analiza scena de trafic, dar și starea vehiculului propriu. Aceste soluții s-au bazat inițial pe modele recurente, iar apoi pe un model propriu care folosește doar straturi convoluționale, având o dimensiune mult mai redusă (ca număr de parametri antrenabili), și implicit un timp de procesare/predicție mai mic, ceea ce o va face o soluție mai viabilă pentru implementarea pe sisteme hardware cu resurse reduse. În aceasta etapă am reușit validarea rezultatelor obținute prin publicarea și prezentarea a două lucrări științifice în conferințe, și am trimis o lucrare științitică la o revistă (care la momentul publicării acestui raport încă nu a fost evaluată).

Bibliografie

[1] - O. Ronneberger, P. Fischer, T. Brox, "U-Net: Convolutional Networks for Biomedical Image Segmentation", Medical Image Computing and Computer-Assisted Intervention (MICCAI), Springer, Vol. 9351, pp. 234-241, 2015.

[2] - H. Xue, D. Q. Huynh, M. Reynolds, "SS-LSTM: A Hierarchical LSTM Model for Pedestrian Trajectory Prediction," 2018 IEEE Winter Conference on Applications of Computer Vision (WACV), 2018, pp. 1186-1194.

[3] - Kingma, D. P.; Ba, J. Adam: A method for stochastic optimization. arXiv 2014, arXiv:1412.6980.

[4] - TensorFlow. Online: https://www.tensorflow.org/

[5] - F. Chollet et al., "Keras," GitHub. Online: https://github.com/fchollet/keras

[6] - OpenCV. Online: https://opencv.org/

[7] - V. Ramanishka, Y. Chen, T. Misu and K. Saenko, "Toward Driving Scene Understanding: A Dataset for Learning Driver Behavior and Causal Reasoning," in 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Salt Lake City, UT, USA, 2018 pp. 7699-7707.

[8] - CARLA. Online: https://carla.org/

Director proiect,

Sl.Dr.Ing. Razvan Itu