UNIVERSITATEA DE VEST DIN TIMIŞOARA
FACULTATEA DE MATEMATICĂ ŞI INFORMATICĂ
PROGRAMUL DE STUDII DE MASTER:
Securitate cibernetică

Traffic Sign Recognition

Supervizor:
Lect. Dr. Mădălina Erașcu

Studenti:
Andreea Bîra-Negrut
Bogdan Pinghireac
Eduard Marian Neguriță
Larisa Drăgănescu

2023

Abstract

Acest studiu explorează performanța și robustețea rețelelor neuronale în recunoașterea semnelor de circulație. În detalierea rezultatelor experimentale, raportul abordează, de asemenea, instrumentele utilizate și provocările întâmpinate în procesul lor de instalare.

Contents

1 Introducere

Rețelele neuronale profunde au adus o transformare semnificativă în modul în care interacționăm cu tehnologia în viața de zi cu zi. Acestea au devenit un factor determinant în domenii precum mașinile autonome, smartphone-uri, jocuri, drone etc., marcând o schimbare fundamentală în modul în care sistemele complexe sunt create și utilizate. Capacitatea lor de a învăța modele complexe din seturi de date și de a face prognoze sau decizii în funcție de acestea le face o unealtă valoroasă în domeniile menționate și în multe altele.

Algoritmii Alpha-Beta-Crown și Marabou reprezintă instrumente și structuri de verificare folosite în contextul evaluării rețelelor neuronale. Acești algoritmi au scopul de a furniza analize detaliate ale modului în care rețelele neuronale se comportă și de a aborda provocările asociate testării acestora.

2 Dataset

Traffic Signs Recognition[2][3][5] este folosit pentru a automatiza recunoașterea semnelor de circulație și este deosebit de important în sistemele avansate de asistență. Aceasta reprezintă o problemă complexă în domeniul viziunii artificiale și recunoașterii modelelor.

Setul de date conține peste mii de imagini care reprezintă diferite semne de circulație. Aceste imagini reflectă variații semnificative în aspectul vizual al semnelor, din cauza factorilor precum distanța, iluminarea, condițiile meteorologice, obstrucțiii parțiale și unghiurile de rotație. Fiecare imagine este însoțită de mai multe seturi de caracteristici pre-calculate, ceea ce facilitează utilizarea algoritmilor de învățare automată fără a fi nevoie de cunoștințe avansate în prelucrarea imaginilor.

Setul de date cuprinde 43 de clase, a căror frecvențe nu sunt echilibrate. Participanții la benchmark au de clasificat două seturi de testare, fiecare conținând peste 12.000 de imagini.

3 Instalare Tool-uri

3.1 Alpha Beta Crown

GitHub: https://github.com/Verified-Intelligence/alpha-beta-CROWN

3.1.1 Introducere

Alpha Beta Crown sau α,β-Crown[6] este un tool care verifica o retea neuronala construit pe un cadru care utilizeaza propagarea liniara legata si tehnici de ramificare legate.

Tool-ul este foarte eficient daca este folosit cu un GPU deoarece are mai multe resurse decat un CPU si astfel executia este mult mai rapida. De asemenea se poate scala in mod eficient la retele convoluționale de dimensiuni semnificative, inclusiv cele cu milioane de parametri.

Este compatibil cu o varietate extinsă de arhitecturi de rețele neuronale, cum ar fi CNN, ResNet și diferite funcții de activare, datorita bibliotecii versatile auto_LiRPA[7]

dezvoltate de acceasi echipa.

auto_LiRPA reprezinta o biblioteca pentru derivarea și calcularea automata a limitelor folosind analiza perturbațiilor bazata pe relaxare liniară (LiRPA) pentru rețelele neurale, fiind o unealta utila pentru verificarea formala a robustetii.

Algoritmul LiRPA foloseste de asemenea un algoritm de graf pe grafuri computationale generale definite prin intermediu PyTorch.

α,β-Crown este castigatorul in competitiile VNN-COMP (International Verification of Neural Networks Competition), din anul 2021, 2022 si 2023, obtinand cel mai bun scor total si depasind semnificativ alte tool-uri de verificare a retelelor neuronale pe o gama extinsa de peste 2 ani.

3.1.2 Instalare

Pentru instalare avem nevoie de un sistem de operare Linux(noi am ales WLS prezent in Windows) si urmatoarele programe instalate, altfel se pot genera erori:

- git
- unzip
- nvidia driver
- cuda tool-kit
- anaconda sau miniconda conform documentatiei alpha-beta-CROWN

Primul pas este clonarea repository folosind urmatoarele comenzi:

git clone git@github.com:Verified-Intelligence/alpha-beta-CROWN.git
cd alpha-beta-CROWN
git clone git@github.com:Verified-Intelligence/auto_LiRPA.git

Dupa clonare trebuie creat enviroment-ul general folosind urmatoarele comenzi

conda env create -f complete_verifier/environment.yaml –name alpha-beta-crown
conda activate alpha-beta-crown

Dupa acest pas se ruleaza urmatoarele comenzi:

cd auto_LiRPA
python setup.py install

Dupa terminarea instalarii am incercat un exemplu prezent in repository si anume cifar_resnet_2b.yaml.

cd complete_verifier
python abcrown.py –config exp_configs/tutorial_examples/cifar_resnet_2b.yaml

Dupa testarea unui exemplu am trecut la instalarea benckmark-ului[1] folosind urmatoarele comenzi:

git ../..
git clone git@github.com:ChristopherBrix/vnncomp2023_benchmarks.git
cd vnncomp2023_benchmarks
sh setup.sh

Dupa acest pas se poate trece la rularea benchmark-ului.

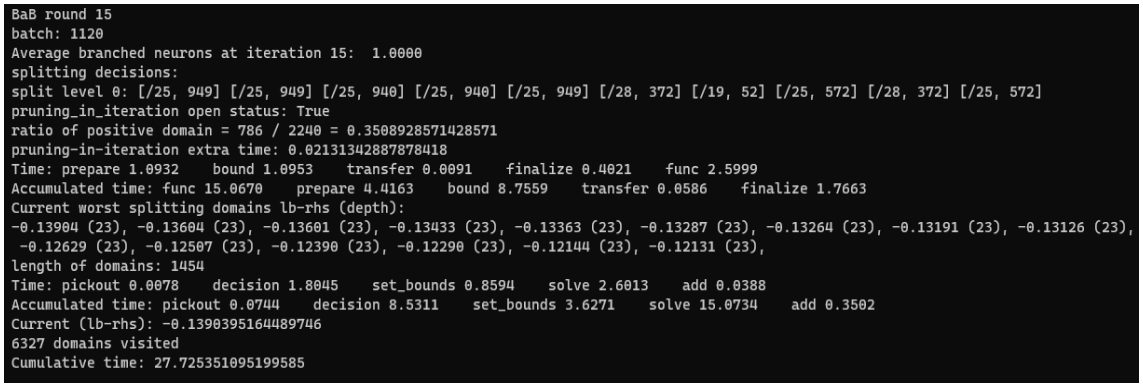

Figure 1:

3.2 Marabou

3.2.1 Introducere

Marabou este o unealtă bazată pe teoria satisfacției constrângerilor (SMT) care poate răspunde la întrebări legate de proprietățile unei rețele, transformând aceste întrebări în probleme de satisfacție a constrângerilor.Designul Marabou este conceput pentru a furniza o platformă robustă și eficientă pentru verificarea și analiza rețelelor neuronale profunde.

Marabou tratează fiecare neuron ca o variabilă și caută o atribuire a acestor variabile care să respecte atât constrângerile liniare, cât și cele non-liniare ale interogării.Se concentrează pe o reprezentare eficientă a acestor variabile și constrângeri.

Pentru a garanta că procesul de verificare este rapid și eficient, Marabou folosește un algoritm de căutare eficient, capabil să găsească soluții care să satisfacă toate constrângerile date.

Marabou este proiectat pentru a avea o performanță optimă în timpul procesului de verificare, concentrându-se pe minimizarea timpului de execuție și pe gestionarea eficientă a resurselor de calcul disponibile. Marabou reprezinta o resursa valoroasa pentru cei implicati in verificarea si analiza retelelor neuronale profunde.

3.2.2 Instalare

In aceasta sectiune vom detalia procesul de instalare al tool-ului Marabou[4]. Pentru a pregatii instalarea rulam urmatoarele comenzi:
sudo apt install cmake
sudo apt-get update && sudo apt-get install build-essential
git clone git@github.com:NeuralNetworkVerification/Marabou.git
cd Marabou
mkdir build
cd build
cmake ..
In cazul in care aveti urmatoarea problema:

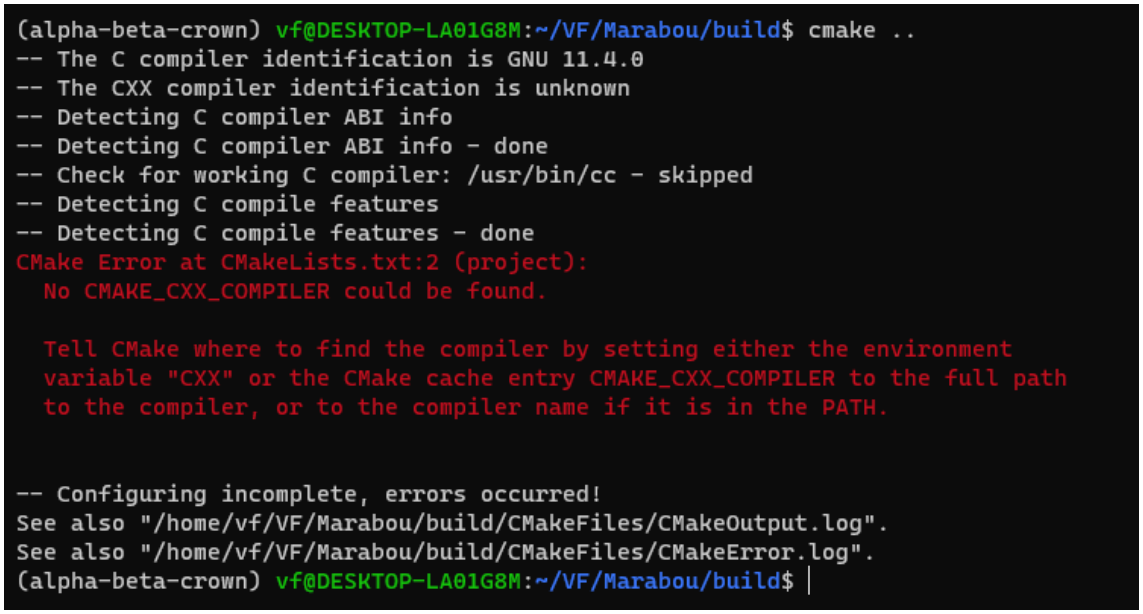

Figure 2: Eroare

Rulati urmatoarea comanda:
sudo apt-get update && sudo apt-get install build-essential
Dupa se executa urmatoarele comenzi pentru a da build la tool-ul Marabou[4]:
cd build
cmake –build .
Daca toate comenzile au fost executate corect trebuie sa ajungeti la pasul din poza urmatoare:

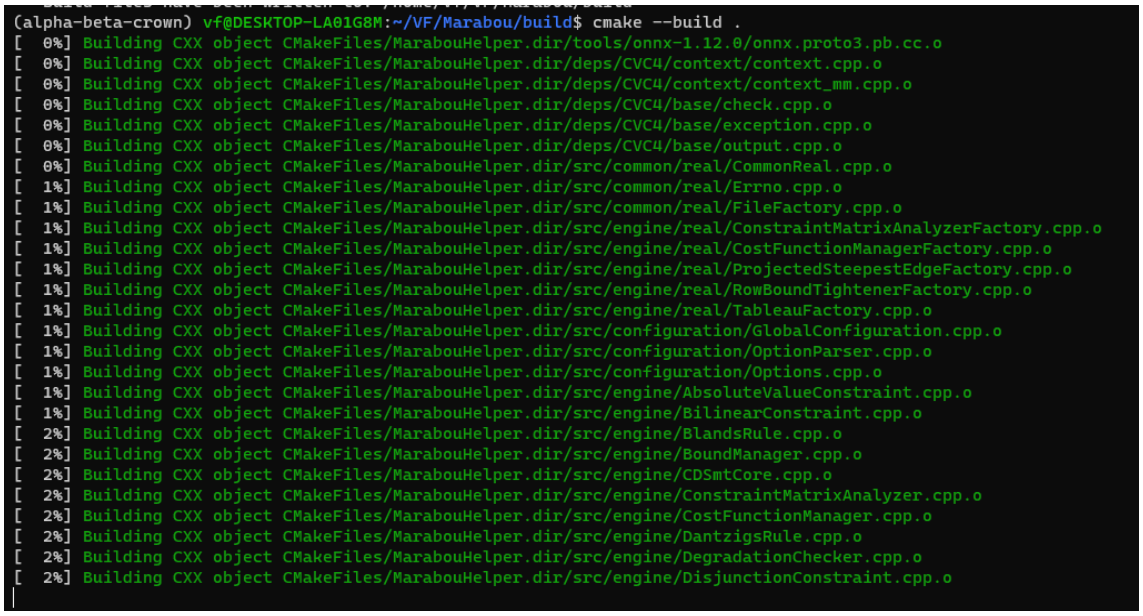

Figure 3: Build

4 Rulare benchmark

4.1 Alpha Beta Crown

Dupa finalizarea instalarii tool-urilor avem instalat tot mediul de lucru pentru a putea rula tool-ul alpha_beta_crown.

Pentru rulare putem opta pentru rularea prin GPU sau prin CPU, noi am ales GPU pentru ca are putere de calcul mai mare. (pentru rularea prin CPU se adauga urmatorul argument **-device CPU** la comanda de executie)

Comenzile pentru rulare sunt urmatoarele:

cd alpha-beta-CROWN/complete_verifier

python abcrown.py –config exp configs/vnncomp23/gtrsb.yam

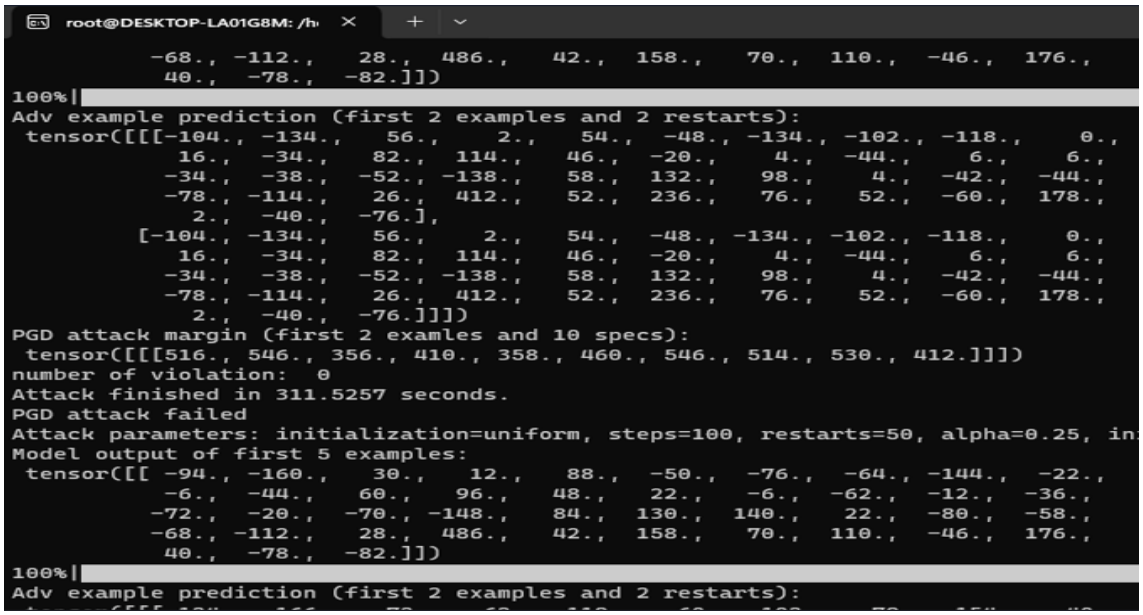

Figure 4:

La finalul executiei se va genera un fisier out.txt care contine rezultatul executiei in format binar dupa cum se poate observa in figura urmatoare:

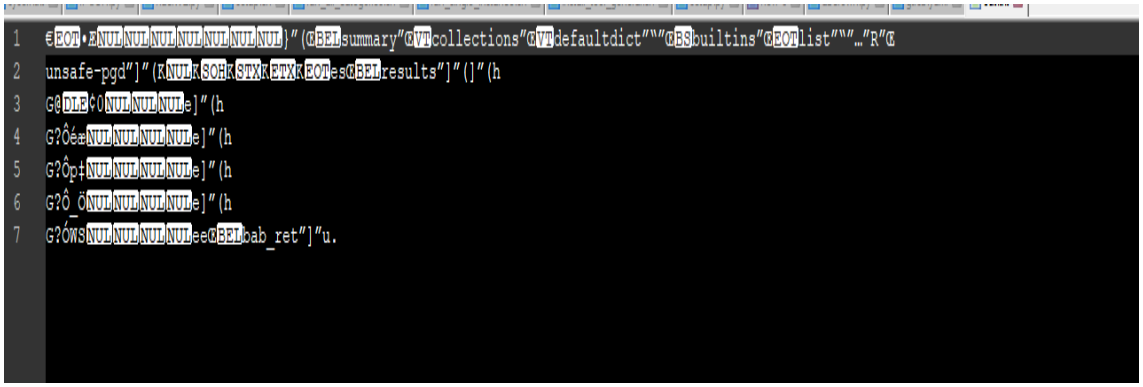

Figure 5:

4.2 Marabou

Dupa finalizarea build-ului, trebuie sa aveti urmatorul status daca build-ul a fost cu succes:

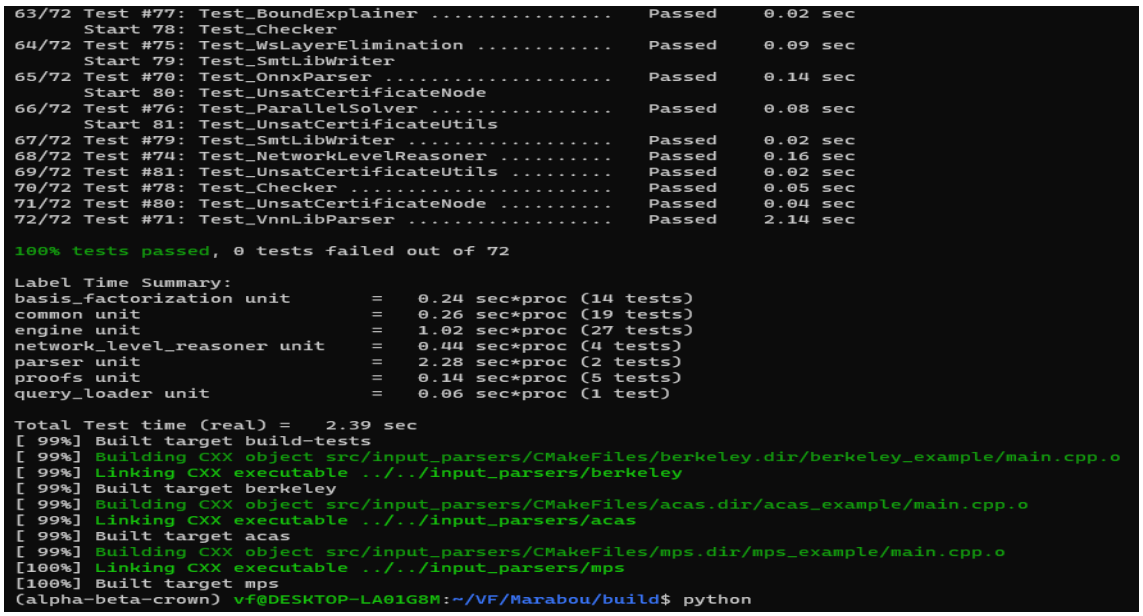

Figure 6: Final build

Apoi putem trece la rularea Marabou folosind urmatoarele comenzi:
./build/Marabou resources/nnet/acasxu/ACASXU_experimental_v2a_2_7.nnet resources/properties/acas_property_3.txt

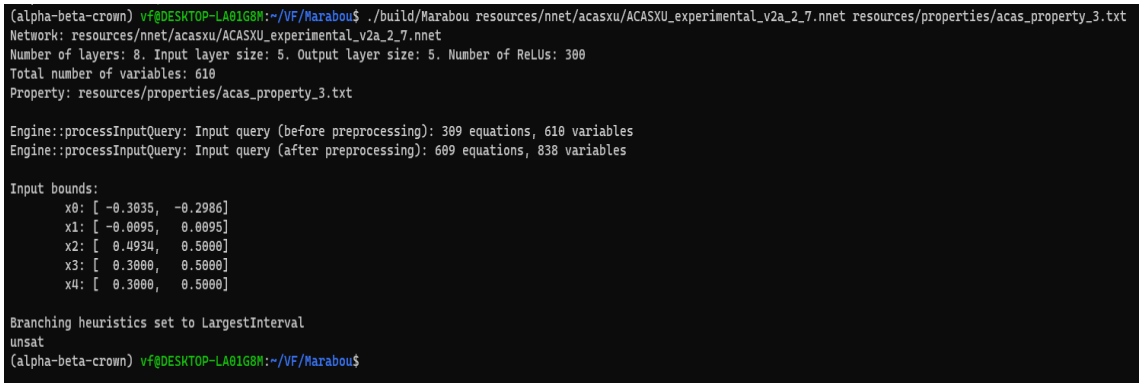

Figure 7: Run Marabou

5 Rezultate

Alpha Beta Crown

Rezultatele obținute arată că din cele 46 de instanțe, 42 au fost clasificate ca sigure, iar doar instanța 45 a fost clasificată ca nesigură. Pentru instanțele 6 și 35, rezultatul a fost "run_instance_timeout", iar instanța 5 a obținut "no_result_in_file". Cele 42 de instanțe au valoarea "sat", fiind satisfiabile, astfel încât instanțele au reușit să satisfacă expresiile logice pentru ca problema să fie considerată rezolvată si pentru a găsi o soluție validă.

idx	Rezultat	Timpi(s)
0	sat	7.157972967
1	sat	7.286744442
2	sat	6.850318482
3	sat	7.472602460
4	sat	6.911085371
5	no_result_in_file	460.654262260
6	run_instance_timeout	540.033374469
7	sat	19.430611468
8	sat	7.768919104
9	sat	7.433775311
10	sat	7.918002813
11	sat	7.321949306
12	sat	7.205714470
13	sat	7.728228881
14	sat	7.777620317
15	sat	8.973640245
16	sat	7.840924381
17	sat	7.467596722
18	sat	8.919894638
19	sat	7.409615798
20	sat	7.674657312
21	sat	7.536734055
22	sat	7.530943546
23	sat	7.643930597
24	sat	7.925060305
25	sat	7.649686196
26	sat	7.522822988
27	sat	7.749032502
28	sat	7.893192003
29	sat	7.275387435
30	sat	8.904811028
31	sat	8.852287157
32	sat	8.927887925
33	sat	16.933491927
34	sat	16.378010803
35	run_instance_timeout	540.002000284
36	sat	11.136481450
37	sat	9.845497913
38	sat	9.480329464
39	sat	9.572339116
40	sat	8.855822936
41	sat	9.365495261
42	sat	8.925496347
43	sat	17.016346806
44	sat	8.997252323
45	unsat	6.135465521

Marabou

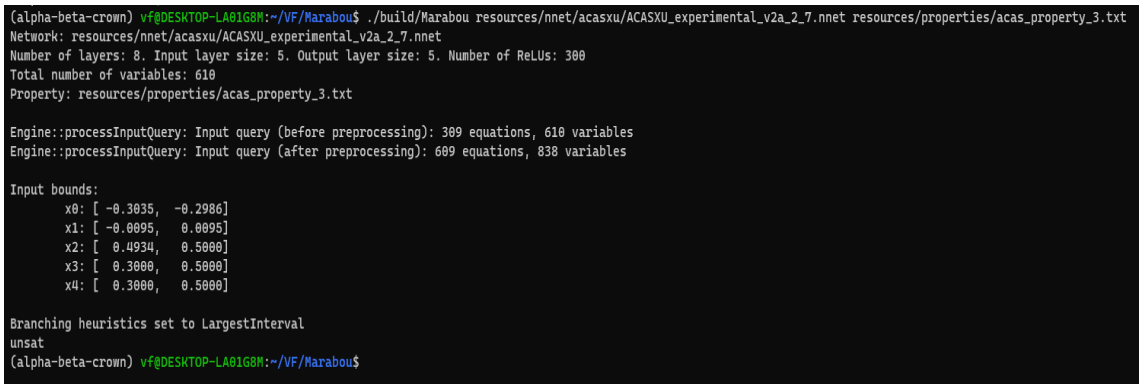

Figure 8: Marabou

6 Concluzie

In concluzie, este esențial să continuăm să dezvoltăm și să îmbunătățim instrumentele de verificare pentru a face față noilor provocări generate de evoluția tehnologiei. Această abordare continuă este crucială pentru a ne asigura că sistemele de recunoaștere a semnelor de circulație devin tot mai precise, mai sigure și mai eficiente.

Contribuția constantă la îmbunătățirea acestor instrumente este esențială pentru a răspunde dinamicii în schimbare a mediului rutier și pentru a promova în mod semnificativ siguranța și eficiența în gestionarea traficului.

7 Bibliografie

References

[1] Christopher Brix. *ChristopherBrix: vnncomp2023_benchmarks*. URL: `https://github.com/ChristopherBrix/vnncomp2023_benchmarks`.

[2] Kaoutar Sefrioui Boujemaa, Ismail Berrada, Afaf Bouhoute, Karim Boubouh. "Traffic sign recognition using convolutional neural networks". In: *IEEEXplore* (2017). URL: `https://ieeexplore.ieee.org/abstract/document/8238205`.

[3] Lai Jia Shyan, T H Lim, Dk Norhafizah Pg Hj Muhammad. "Real time road traffic sign detection and recognition systems using Convolution Neural Network on a GPU platform". In: *IEEEXplore* (2023). URL: `https://ieeexplore.ieee.org/document/10007277`.

[4] NeuralNetworkVerification. *NeuralNetworkVerification: Marabou*. URL: `https://github.com/NeuralNetworkVerification/Marabou`. (accessed: 23.01.2024).

[5] Twine AI. "The Best Road Sign Detection Datasets of 2022". In: *twine* (2022). URL: `https://www.twine.net/blog/road-sign-detection-datasets/`.

[6] Verified-Intelligence. *Verified-Intelligence: alpha-beta-CROWN*. URL: `https://github.com/Verified-Intelligence/alpha-beta-CROWN`. (accessed: 15.01.2024).

[7] Verified-Intelligence. *Verified-Intelligence: auto_LiRPA*. URL: https://github.com/Verified-Intelligence/auto_LiRPA. (accessed: 15.01.2024).