Măsurarea performanţei instrumentelor specializate de verificare a reţelelor neuronale

Chira Andreea-Marina, andreea.chira00@e-uvt.ro
Hasna Diana-Mihaela, diana.hasna00@e-uvt.ro
Hasan Majd, majd.hasan93@e-uvt.ro
Chera Denis-Marian, denis.chera00@e-uvt.ro
Vanciu Catalin-Marian, catalin.vanciu00@e-uvt.ro

February 4, 2024

Abstract

Este bine cunoscut faptul că numărul de vehicule rutiere a crescut enorm datorită tehnologiilor dezvoltate din industria auto şi, mai precis, disponibilităţii tarifelor mici. Astfel, semnele de circulaţie contribuie atât la siguranţa şoferilor cât şi a pietonilor, având un rol esenţial în fluidizarea şi ordonarea traficului rutier.

În acest sens, lucrarea se concentrează pe tehnicile şi instrumentele disponibile pentru verificarea formală, astfel încat VNN-COMP-2023 (Competiţia de verificare a reţelelor neuronale) a fost alesă ca principal punct de plecare al acestui studiu. Această competiţie propune un model de reţea neuronală în formatul ONNX împreună cu o specificaţie în formatul VNNLIB, fişiere care stabilesc dacă există o situaţie în care specificaţia se potriveşte cu modelul. Astfel de descoperiri facilitează verificarea unor proprietăţi diverse în diferite aplicaţii ale reţelelor neuronale.

Aşadar, din cea mai recentă ediţie, au fost selectate două instrumente de verificare a reţelelor neuronale profunde (DNN) care au obţinut rezultate remarcabile în ceea ce priveşte performanţa, respectiv α-β **CROWN & NeuralSAT**, având printre cele mai multe criterii de referinţă.

Prin urmare, obiectivul acestei lucrări este de a determina, interpreta şi compara rezultatele obţinute în urma utilizării celor două tool-uri asupra Benchmark-ului **Traffic Signs Recognition**, considerat set de date, cu datele obţinute în VNN-COMP 2023.

Cuvinte cheie: ONNX, VNNLIB, α-β CROWN, NeuralSAT

1 Introducere

În prezent, unul dintre cele mai utilizate sisteme de transport este reprezentat de transportul rutier, care generează un trafic semnificativ aflat într-o continuă creştere în întreaga lume. Un studiu realizat de către cei de la National Highway Transportation Safety Administration (NHTSA) din SUA în anul 2016 subliniază faptul că, procentul accidentelor rutiere cauzate de erorile umane reprezintă între 94% şi 96% din numărul total al acestora, amintind de neglijenţa conducătorilor auto, ignorarea sau înţelegerea greşită a indicatoarelor rutiere, ceea ce atestă necesitatea şi importanţa existenţei unui sistem automat şi inteligent pentru a asista şoferul în sarcinile sale de conducere.

Astfel, în zilele noastre, numeroşi producători auto încorporează în vehiculele lor sisteme care oferă diferite grade de autonomie şi securitate. Printre aceste sisteme se numără şi sistemele de recunoaştere a semnelor de circulaţie (TSR), care ajută la minimizarea numărului de decizii eronate şi de accidente. [6]

În cazul sistemelor de tip TSR, un aspect crucial este reprezentat de reţelele neuronale care sunt folosite pentru a recunoaşte pattern-uri complexe şi pentru a face predicţii sau a lua decizii pe baza datelor de intrare. Reţelele neuronale sunt adesea cutii negre şi este greu de asigurat că se comportă în condiţii de siguranţă şi predictibilitate în cazul unor date de intrare perturbate sau rău intenţionate (intrări maliţioase). De asemenea, cercetătorul Huan Zhang a declarat că: "Verificarea reţelelor neuronale urmăreşte să ofere garanţii demonstrabile pentru proprietăţile dorite ale reţelelor neuronale, cum ar fi siguranţa şi robusteţea". [2]

Astfel, în cadrul acestei lucrări, atenţia este orientată pe testarea celor două instrumente, α-β CROWN şi NeuralSAT. În urma obţinerii rezultatelor, are loc interpretarea şi compararea acestora cu datele obţiune în cadrul VNN-COMP 2023.

Pentru a atinge acest obiectiv, vor fi detaliate, în secțiunile următoare, atât modul de funcționare al unui instrument de acest tip, cât și importanța benchmark-ului și impactul pe care îl poate avea asupra performanței tool-urilor selectate. Totodată, vor fi prezentate şi eventualele provocări întâmpinate pe parcursul proceselor de instalare şi execuţie.

2 Benchmark

În contextul acestui experiment, s-a utilizat benchmark-ul **Traffic Sign Recognition** [4] constituit dintr-un ansamblu de fișiere ONNX (Open Neural Network Exchange) și VNNLIB, și care este folosit pentru a evalua performanța diferitelor metode și instrumente de verificare a rețelelor neuronale. Fișierele ONNX din benchmark-ul ales au la baza modelul unei rețele neuronale antrenat pe setul de date German Traffic Sign Recognition Benchmark (GTRSB) existent pe site-ul Kaggle, care cuprinde peste 50000 de imagini ce sunt clasificate în 43 de clase, fiecare simbolizând un tip distinct de indicator rutier.

Un astfel de model de fişier ONNX, se regăseşte sub forma

3_30_30_QConv_16_3_QConv_32_2_Dense_43_ep_30.onnx, unde:

- 3: indică numărul de canale de intrare (RGB)
- 30_30: indică dimensiunea datelor de intrare (lățimea și înălțimea unei imagini)

- QConv_16_3: indică un strat convoluțional cuantificat(Quantized Convolutional Layer)cu 16 filtre de dimensiune 3x3. Fiecare filtru este responsabil de învățarea unor caracteristici diferite din datele de intrare, având dimensiunea precizată anterior.

- QConv_32_2: indică un alt strat convoluțional cuantificat cu 32 de filtre de dimensiune 2x2.

- Dense_43: specifică un strat dens (complet conectat) cu 43 de unități. Acest strat este responsabil de combinarea caracteristicilor învățate de straturile anterioare pentru a face predicții sau clasificări.

- ep_30: indică faptul că modelul rețelei neuronale a fost antrenat pentru un total de 30 de epoci în timpul procesului de antrenament. O epocă reprezintă o trecere completă prin întregul set de date de instruire.

Pe de altă parte, fișierele VNNLIB sunt reprezentate de problemele de verificare asociate cu acele modele, care includ diverse condiții ce pot fi legate de clasificare sau anumite proprietăți și caracteristici specifice modelelor respective.

Un exemplu de fişiere VNNLIB se regăseşte sub forma **model_30_idx_7040_eps_1.00000.vnnlib**, unde:

- model_30: reprezintă unul din modelele specifice unui format de reţea onnx

- idx_7040: reprezintă un indice sau un identificator asociat cu configurația specifică rețelei neuronale.

- eps_1.00000: reprezintă valoarea epsilonului, care este un parametru utilizat pentru a verifica robusteţea.

Conţinutul fişierelor VNNLIB constă în:

1. **Declararea variabilelor de intrare**

   Fiecare valoare reprezintă o caracteristică (ex: pixel) a imaginii de intrare şi sunt declarate ca numere reale.

   Exemplu: (declare-const X_0 Real)

2. **Constrângerile de intrare (assert-uri)**

   Constrângerile sunt reprezentate de condiţiile impuse fiecărei variabile de intrare pentru a specifica intervalele valide pentru valorile pixelilor imaginii. Condițiile respective sunt exprimate cu ajutorul unor valori din intervalul [0, 255], care în contextul imaginilor se referă la scala intensităților de lumină sau culorilor.

   Exemplu:

   (assert(<= X_0 255.00000000))

   (assert(>= X_0 244.00000000))

3. **Constrângerile de ieşire**

   Constrângerile de ieşire sunt definite ca o disjuncţie (sau), în cadrul căreia fiecare clauză reprezintă o condiţie pentru variabilele de ieşire.

   Exemplu:

   (assert (or

   (and (>= Y_0 Y_3))

   (and (>= Y_1 Y_3))))

   Condiţiile impun ca valorile de ieşire să fie mai mari sau egale cu valoarea de ieşire a celui de-al treilea neuron.

4. **Valoarea Epsilon**

   Aceasta poate oscila, indicând nivelul permis de perturbație sau zgomot pentru variabilele de intrare în timpul verificării.

   Exemplu: 10.00000

3 Tool

3.1 α-β CROWN

3.1.1 Descriere

α , β -CROWN (α-β CROWN) este un instrument de verificare a rețelelor neuronale profunde (DNN) care se bazează pe un mecanism eficient de propagare a limitelor liniare și pe o metodă de "branch and bound" (Bab). Procesul poate fi accelerat în mod eficient pe GPU și se poate adapta la rețele convoluționale relativ mari (de exemplu, milioane de parametri). De asemenea, prin intermediul bibliotecii versatile auto_LiRPA acesta acceptă o gamă largă de arhitecturi de rețele neuronale, cum ar fi: CNN, ResNet și diverse funcții de activare. Totodată, tool-ul α , β -CROWN poate oferi garanții demonstrabile de robustețe împotriva atacurilor adversarilor și poate verifica, de asemenea, alte proprietăți generale ale rețelelor neuronale. [12] [10], [11], [5], [9]

Alte detalii despre tool:

- CROWN: reprezintă un algoritm eficient de verificare bazat pe propagarea limitelor. Acesta propagă o inegalitate liniară în sens invers prin rețea și utilizează limite liniare pentru a relaxa funcțiile de activare.

- α-CROWN: este un verificator de rețele neuronale optimizat, utilizat în verificatorul Fast-and-Complete, care optimizează în comun limitele stratului intermediar și limitele stratului final în CROWN prin intermediul variabilei α. De obicei, are o putere mai mare decât LP (Linear Programming), deoarece LP nu poate restrânge în mod necostisitor limitele stratului intermediar.

- β-CROWN: introduce un parametru β optimizabil pentru a încorpora constrângerile de divizare în branch and bound (BaB) în procesul de propagare a limitelor în CROWN. Un verificator de rețele neuronale robust și scalabil este creat prin

combinarea branch and bound cu o propagare eficientă și accelerată de GPU a limitelor.

3.1.2 Instalare

Paşii urmaţi pentru configurarea tool-ului α , β -CROWN: [1]

1. **Instalare Miniconda pentru Sistemul de Operare Linux:**

   Pentru realizarea acestui proces s-au folosit comezile ce pot fi regăsite în Figura 1.

```
mkdir -p ~/miniconda3
wget https://repo.anaconda.com/miniconda/Miniconda3-latest-Linux-x86_64.sh -O ~/miniconda3/miniconda.sh
bash ~/miniconda3/miniconda.sh -b -u -p ~/miniconda3
rm -rf ~/miniconda3/miniconda.sh
```

Figure 1: Instalare Miniconda

2. **Clonarea proiectlui α , β -CROWN**

   Pentru realizarea acestui proces s-a folosit comanda ce poate fi regăsită în Figura 2.

```
git clone --recursive https://github.com/Verified-Intelligence/alpha-beta-CROWN.git
```

Figure 2: Clonarea proiectului

3. **Configurarea şi activarea environment-ului α , β -CROWN:**

   Pentru parcurgerea acestei etape a fost necesară utilizarea fișierul environment.yaml aflat în folderul complete_verifier al proiectului și s-au folosit comezile ce pot fi regăsite în Figura 3.

```
# Remove the old environment, if necessary.
conda deactivate; conda env remove --name alpha-beta-crown
# install all dependents into the alpha-beta-crown environment
conda env create -f complete_verifier/environment.yaml --name alpha-beta-crown
# activate the environment
conda activate alpha-beta-crown
```

Figure 3: Environment-ul α , β -CROWN

4. **Testarea**

   După activarea environment-ului a urmat procesul de testare realizat prin intermediul comenzilor ce pot fi regăsite în Figura 4.

```
cd complete_verifier
python abcrown.py --config exp_configs/cifar_resnet_2b.yaml
```

Figure 4: Testarea tool-ului α , β -CROWN

În urma testării tool-ului α , β -CROWN, rezulatate pot fi observate în Figura 5:

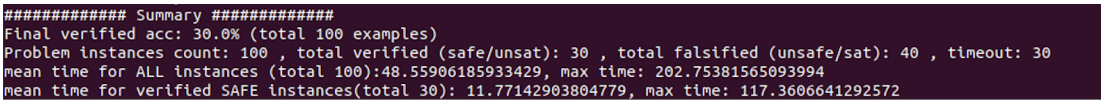

Figure 5: Rezulatele obţinute

3.1.3 Rularea α-β Crown pentru benchmark-ul ales

Acest proces a debutat prin clonarea proiectului VNN-COMP2023 cu scopul de a avea acces la benchmark-ul descris mai sus în cadrul acestei lucrări, și anume Traffic Sign Recognition ce are rolul de a permite evaluarea instrumentului de verificare folosit. După o căutare meticuloasă, s-a descoperit faptul că, configuraţiile aferente acestei ultime ediții ale competiţiei se regăsesc în path-ul `complete_verifier\exp_configs` în folderul numit vnncomp23, făcând parte din proiectul α-β Crown. Astfel, în continuare s-a identificat fișierul gtrsb.yaml, care a fost configurat corespunzător pentru a stabili conexiunea necesară între tool și benchmark pentru a putea iniția, mai apoi, acțiunea de rulare propriu-zisă.

În consecință, după pregătirea configurației, pentru rulare s-a folosit o comanda, cu ajutorul căreia s-a realizat testarea tool-ului, adaptată astfel încat rezultatele din urma rulării să fie salvate în acest caz într-un fişier (output1.txt) pentru a putea fi vizualizate și analizate ulterior.

Aceasta a fost executată după cum urmează:

```
python abcrown.py --config /root/$\alpha$-$\beta$ CROWN/complete_verifier/
exp_configs/test/gtrsb.yaml --device cpu
```

Output-ul obţinut în urma rulării benchmark-ului prin intermediul tool-ului poate fi observat în Figura 6 de mai jos.

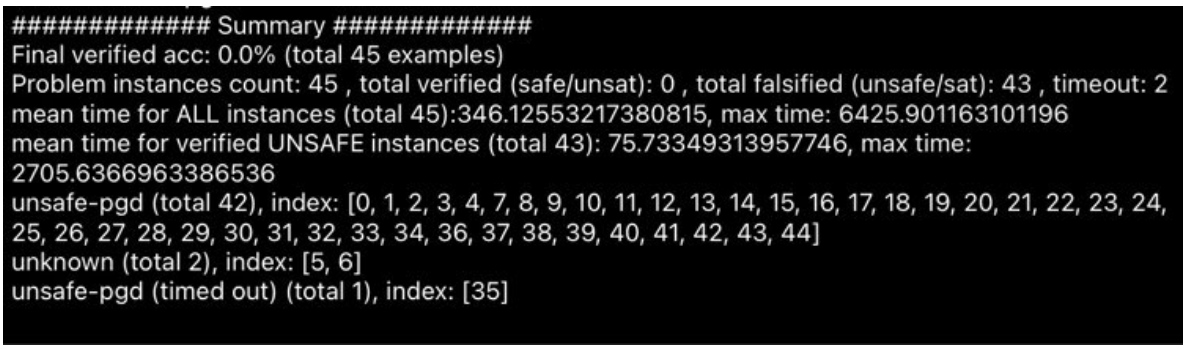

Figure 6: Output-ul afişat în urma rulării tool-ului α-β Crown

Rezultatele furnizate indică faptul că, din cele 45 de instanțe (fiecare reprezentând un fișier vnnlib) niciuna nu a fost verificată ca fiind sigură, majoritatea (42) au fost clasificate ca nesigure, pentru instanţele 5 şi 6 s-a obţinut rezultatul "unknown", iar în cazul instanţei 35, s-a ajuns "unsafe/sat", dar prin depăşirea limitei de timp (timeout). În contextul actual, "unsafe/sat" se referă la instanțele nesigure care sunt satisfiabile,

adică există cel puțin o configurație a rețelei neuronale pentru care nu a fost posibil să se demonstreze că rețeaua răspunde într-un mod dorit sau sigur.

De asemenea, timpul mediu și maxim se calculează atât pentru toate instanțele, cât și în mod particular pentru cele nesigure verificate.

3.2 NeuralSAT

3.2.1 Descriere

NeuralSAT este un instrument de verificare a rețelelor neuronale profunde (DNN). Acesta integrează abordarea DPLL(T) utilizată în mod obișnuit în rezolvarea SMT(Satisfiability Modulo Theories) cu un solver specializat pentru raționamentul DNN. NeuralSAT exploatează multicores și GPU (Graphics Processing Unit) pentru eficiență și se poate dezvolta pentru rețele cu milioane de parametri. De asemenea, acesta suportă o gamă largă de rețele neuronale și funcții de activare.

NeuralSAT oferă o metodă eficientă de verificare a corectitudinii și performanței rețelelor neuronale profunde, ceea ce este esențial pentru asigurarea încrederii în modelele de învățare automată utilizate într-o varietate de aplicații, inclusiv Traffic Sign Recognition. [7], [8]

3.2.2 Instalare

Paşii urmaţi pentru configurarea tool-ului NeuralSAT: [3]

1. **Instalare şi activare Miniconda pentru Sistemul de Operare Linux:** proces realizat odată cu instalarea primului tool (α , β -CROWN)

2. **Instalarea pachetelor necesare**

   Pentru realizarea acestui proces s-a folosit comanda ce poate fi regăsită în Figura 7.

```
conda env create -f env.yaml
```

Figure 7: Instalarea pachetelor necesare

3. **Configurarea şi activarea environment-ului NeuralSAT**

   Pentru realizarea acestui proces s-a folosit comanda ce poate fi regăsită în Figura 8

```
conda activate neuralsat
```

Figure 8: Environment-ul NeuralSAT

4. **Testare**

După activarea environment-ului, a urmat procesul de testare realizat prin intermediul comenzii ce pot fi regăsite în Figura 9.

```
(neuralsat) root@majd-VivoBook:/home/majd/Downloads/neuralsat/neuralsat# python3 main
.py --net /home/majd/Downloads/neuralsat/onnx/3_30_30_QConv_16_3_QConv_32_2_Dense_43_
ep_30.onnx --spec /home/majd/Downloads/neuralsat/vnnlib/model_30_idx_7040_eps_5.00000
.vnnlib --timeout 480 --device cpu
```

Figure 9: Testarea unei instanţe cu NeuralSAT

În urma rulării acestei comenzi, instanţele pot fi testate exclusiv pe rând, iar pentru a simplifica procesul a fost implementat un script Python. Acesta ajută la iterarea prin toate instanţele ce urmează să fie rulate din fişierul instances.csv .

Script-ul Python:

```
import csv
import os
import time

onnx_folder = "/root/neuralsat/onnx/"
vnnlib_folder = "/root/neuralsat/vnnlib/"
output_file = "output.txt"

with open(output_file, "w") as output:
    output.write("Instance,Result\n")
    with open("instances.csv", newline='') as csvfile:
        reader = csv.reader(csvfile)

        for row in reader:
            onnx_filename = os.path.join(onnx_folder, row[0]
                .lstrip('onnx/'))
            vnnlib_filename = os.path.join(vnnlib_folder, row[1]
                .lstrip('vnnlib/'))

            command =
            (f"python3 main.py --net
{onnx_filename} --spec {vnnlib_filename} --timeout 480 --device cpu")
            time.sleep(6)

            result = os.popen(command).read()

            output file
            output.write(f"{row[0]},{result}\n")

print(f"Results saved in {output_file}")
```

3.2.3 Rularea NeuralSAT pentru benchmark-ul ales

Pentru configurarea instrumentului de verificare NeuralSAT, fişierele VNNLIB şi ONNX ale benchmark-ului au fost copiate în acelaşi director cu acesta. În continuare, a fost activat environmentul NeuralSAT, iar după acest pas a avut loc rularea tool-ului prin intermediul comenzii:

```
python3 main.py --net /root/NeuralSATdevelop/onnx/
3_30_30_QConv_16_3_QConv_32_2_Dense_43_ep_30.onnx --spec /root/NeuralSAT-develop/
vnnlib/model_30_idx_7040_eps_1.00000.vnnlib --timeout 480 --device cpu
```

Cu ajutorul acesteia s-a realizat testarea tool-ului pentru o singură instanţă din fişierul **instances.csv**. Ulterior, ea a fost integrată în script-ul Python creat, astfel încât toate instanţele să poată fi rulate pe rând. Rezultatele obţinute în urma rulării au fost salvate într-un fişier (output.txt) pentru a putea fi vizualizate și analizate ulterior.

Output-ul obţinut în urma rulării benchmark-ului prin intermediul tool-ului poate fi observat în Figura 10 de mai jos.

```
onnx/3_30_30_QConv_16_3_QConv_32_2_Dense_43_ep_30.onnx,Restricted license - for non-production use only - expires 2025-11-24
Automatic inference of operator: sign
Automatic inference of operator: sign
Automatic inference of operator: sign
Automatic inference of operator: sign
Automatic inference of operator: sign
Automatic inference of operator: sign
Automatic inference of operator: sign
Automatic inference of operator: sign
INFO     00:37:07     [!] VNNLIB: 2700 inputs, 43 outputs
ConvertModel(
  (Transpose_sequential_10/quant_conv2d_20/QuantConv2D__185:0): Transpose()
  (Conv_sequential_10/quant_conv2d_20/QuantConv2D:0): Conv2d(3, 16, kernel_size=(3, 3), stride=(1, 1), bias=False)
  (Transpose_sequential_10/quant_conv2d_20/QuantConv2D__187:0): Transpose()
  (Add_sequential_10/quant_conv2d_21/ste_sign_52/add:0): Add()
  (Transpose_sequential_10/quant_conv2d_21/QuantConv2D__189:0): Transpose()
  (Conv_sequential_10/quant_conv2d_21/QuantConv2D:0): Conv2d(16, 32, kernel_size=(2, 2), stride=(1, 1), bias=False)
  (Transpose_sequential_10/quant_conv2d_21/QuantConv2D__191:0): Transpose()
  (Reshape_sequential_10/flatten_10/Reshape:0): Flatten()
  (Add_sequential_10/quant_dense_10/ste_sign_54/add:0): Add()
  (MatMul_sequential_10/quant_dense_10/MatMul:0): Linear(in_features=23328, out_features=43, bias=False)
  (Softmax_activation_10): Identity()
)
INFO     00:37:07     [!] Input shape: (1, 3, 30, 30)
INFO     00:37:07     [!] Output shape: (1, 43)

[!] Current settings:
        - max_hidden_branches          : 5000
        - max_hidden_visited_branches  : 20000
        - use_attack                   : True
        - use_restart                  : True
        - use_stabilize                : True


INFO     00:37:07     RandomAttack(seed=637, device=cpu)
INFO     00:37:07     PGDAttack(seed=502, device=cpu)
INFO     00:37:07     [!] Iterations: 0
INFO     00:37:07     adv (first 5): tensor([254.5912, 254.9919, 254.1197, 250.8191, 254.4812])
DEBUG    00:37:07     output: tensor([ 614.,  940., 1166.,  860., 1182.,  680.,  424.,  910.,  840.,  378.,
        768., 1338.,  774.,  490., 1196., 1178.,   60., 1556.,  908.,  818.,
       1214.,  606., 1372.,  854.,  562., 1210., 1378.,  522., 1234.,  736.,
       1078.,  752.,  522.,  380., -334.,  -68.,  568., -254.,  290.,  192.,
        260.,  564.,  366.])
sat,0.6525
```

Figure 10: Output-ul afişat în urma rulării tool-ului NeuralSAT

Rezultatele furnizate indică faptul că, din cele 45 de instanțe, toate au luat valoarea de SAT (sunt satisfiabile). Acest lucru sugerează că toate instanţele au reuşit să satisfacă toate clauzele condiționale ale problemei, respectând limita maximă recomandată pentru timpul de execuţie.

4 Rezultatele obţinute

idx	α-β CROWN		NeuralSAT	
	Rezultat	Timpi (s)	Rezultat	Timpi (s)
0	sat	1.8877	sat	0.6525
1	sat	1.7377	sat	0.6466
2	sat	1.7808	sat	0.6792
3	sat	1.9051	sat	0.6176
4	sat	1.8577	sat	0.6044
5	unknow	6425.9012	sat	0.7532
6	unknow	5893.2110	sat	0.7325
7	sat	38.4874	sat	0.7284
8	sat	9.1305	sat	0.8284
9	sat	6.1772	sat	0.7121
10	sat	3.2396	sat	0.5765
11	sat	3.2536	sat	0.6531
12	sat	3.2427	sat	0.7342
13	sat	1.7774	sat	0.6769
14	sat	1.7746	sat	0.7364
15	sat	40.5247	sat	1.6427
16	sat	12.8479	sat	1.5817
17	sat	7.0552	sat	1.5767
18	sat	7.1875	sat	1.5386
19	sat	70530	sat	1.3434
20	sat	18.4352	sat	1.3438
21	sat	9.9095	sat	1.3408
22	sat	9.9932	sat	1.3404
23	sat	9.8963	sat	1.3495
24	sat	7.1580	sat	1.3509
25	sat	4.1796	sat	1.3449
26	sat	4.2249	sat	1.3440
27	sat	4.1578	sat	1.3501
28	sat	4.2682	sat	1.3481
29	sat	4.1475	sat	1.3497
30	sat	22.6591	sat	3.5344
31	sat	13.5373	sat	2.4835
32	sat	13.1376	sat	3.5024
33	sat	13.1099	sat	3.5833
34	sat	13.2167	sat	3.5172
35	timeout	2705.6367	sat	2.4345
36	sat	68.8048	sat	2.4035
37	sat	51.7306	sat	2.4154
38	sat	35.0477	sat	2.3819
39	sat	24.0989	sat	2.4581
40	sat	13.1931	sat	2.4406
41	sat	13.1376	sat	2.3902
42	sat	14.3406	sat	2.3903
43	sat	14.4335	sat	2.3945
44	sat	13.1659	sat	2.3547

Table 1: Rezultatele tool-urilor α-β Crown & NeuralSAT

5 Interpretarea rezultatelor

În procesul de comparare a rezultatelor noastre cu cele din cadrul concursului VNN-COMP 2023, am constatat că există diferențe semnificative, în special în ce priveşte utilizarea tool-ului NerualSAT. De asemenea, rezultatele variază şi în funţie de instrumentul folosit sau versiunea acestuia.

În cele din urmă, concluziile extrase în cadrul acestei analize subliniază faptul că:

Tool-ul α-β Crown:

- În urma execuţiei acestuia, rezultatele obţinute au fost în mare măsură similare cu cele din competiţia VNN-COMP 2023, diferenţa fiind subliniată de ultima instanţă pentru care s-a obţinut rezultatul SAT.

idx	α-β CROWN		VNN-COMP 2023	
	Rezultat	Timpi (s)	Rezultat	Timpi (s)
0	sat	1.8877	sat	7.6219
1	sat	1.7377	sat	7.638
2	sat	1.7808	sat	7.6334
3	sat	1.9051	sat	7.6266
4	sat	1.8577	sat	7.6384
5	unknow	6425.9012	unknow	421.8103
6	unknow	5893.2110	unknow	540.0023
...	...	...	...	...
35	timeout	2705.6367	timeout	492.7604
...	...	...	...	...
44	sat	13.1659	unsat	5.9352

Table 2: Compararea rezultatelor α-β CROWN & VNN-COMP 2023

Tool-ul NeuralSAT:

- În opoziţie cu performanţele obţinute în cadrul competiţiei desfăşurate în anul 2023, rularea acestui intrument de verificare a condus spre obţinerea unor rezultate exclusiv satisfiabile (SAT).

idx	NeuralSAT		VNN-COMP 2023	
	Rezultat	Timpi (s)	Rezultat	Timpi (s)
0	sat	0.6525	unknow	13.0525
1	sat	0.6466	unknow	12.9815
2	sat	0.6792	unknow	13.0032
3	sat	0.6176	unknow	13.0058
...	...	...	...	...
41	sat	2.3902	sat	5.8161
42	sat	2.3903	sat	5.8391
43	sat	2.3945	sat	5.8242
44	sat	2.3547	unsat	4.9611

Table 3: Compararea rezultatelor NeuralSAT & VNN-COMP 2023

Provocări

Un impediment întâmpinat în încercarea de a rula tool-ul NeuralSAT a fost reprezentat de versiunea acestuia. Acest aspect a condus spre obţinerea unor rezultate exclusiv satisfiabile, spre deosebire de versiunile actuale care implică anumite probleme legată de executie (RuntimeErrors, Killed).

6 Concluzii

În concluzie, lucrarea s-a concentrat pe evaluarea performanței a două instrumente specializate de verificare a rețelelor neuronale profunde (DNN), respectiv α-β CROWN și NeuralSAT, în contextul competiției VNN-COMP 2023. Scopul a fost de a determina, interpreta și compara rezultatele obținute în urma utilizării acestor instrumente asupra Benchmark-ului Traffic Signs Recognition.

Lucrarea subliniază importanța recunoașterii semnelor de circulație în contextul creșterii traficului rutier și a necesității unor sisteme automate și inteligente pentru asistarea șoferilor în conducere. De asemenea, aceasta evidențiază importanța verificării formale a rețelelor neuronale, dat fiind caracterul lor de cutii negre și dificultatea asigurării siguranței și predictibilității în condiții variate.

În final, am subliniat importanța acestui studiu în evaluarea și compararea performanței instrumentelor de verificare a rețelelor neuronale profunde în contextul recunoașterii semnelor de circulație, evidențiind rezultatele obținute.

References

[1] Alpha-beta-crown
https://github.com/verified-intelligence/alpha-beta-crown/blob/main/readme.md, accessed: 02/02/2024.

[2] Neural networks, https://www.cs.cmu.edu/news/2021/zhang_vnn_comp2021, accessed: 29/12/2023.

[3] Neuralsat, https://github.com/dynaroars/neuralsat, accessed: 04/02/2024.

[4] Traffic signs recognition benchmark, https://github.com/christopherbrixvnncomp2023_results/tree/main/alpha_beta_crown/2023_traffic_signs_recognition, accessed: 04/02/2024.

[5] Christopher Brix, Stanley Bak, Changliu Liu, and Taylor T Johnson. The fourth international verification of neural networks competition (vnn-comp 2023): Summary and results. *arXiv preprint arXiv:2312.16760*, 2023.

[6] A. de la Escalera, J.M Armingol, and M. Mata. Traffic sign recognition and analysis for intelligent vehicles. *Image and Vision Computing*, 21(3):247–258, 2003.

[7] Hai Duong, Linhan Li, ThanhVu Nguyen, and Matthew Dwyer. A dpll (t) framework for verifying deep neural networks. *arXiv preprint arXiv:2307.10266*, 2023.

[8] Harald Ganzinger, George Hagen, Robert Nieuwenhuis, Albert Oliveras, and Cesare Tinelli. Dpll (t): Fast decision procedures. In *Computer Aided Verification: 16th International Conference, CAV 2004, Boston, MA, USA, July 13-17, 2004. Proceedings 16*, pages 175–188. Springer, 2004.

[9] Hadi Salman, Greg Yang, Huan Zhang, Cho-Jui Hsieh, and Pengchuan Zhang. A convex relaxation barrier to tight robustness verification of neural networks. *Advances in Neural Information Processing Systems*, 32, 2019.

[10] Kaidi Xu, Zhouxing Shi, Huan Zhang, Yihan Wang, Kai-Wei Chang, Minlie Huang, Bhavya Kailkhura, Xue Lin, and Cho-Jui Hsieh. Automatic perturbation analysis for scalable certified robustness and beyond. *Advances in Neural Information Processing Systems*, 33:1129–1141, 2020.

[11] Huan Zhang, Shiqi Wang, Kaidi Xu, Yihan Wang, Suman Jana, Cho-Jui Hsieh, and Zico Kolter. A branch and bound framework for stronger adversarial attacks of relu networks. In *International Conference on Machine Learning*, pages 26591–26604. PMLR, 2022.

[12] Huan Zhang, Tsui-Wei Weng, Pin-Yu Chen, Cho-Jui Hsieh, and Luca Daniel. Efficient neural network robustness certification with general activation functions. *Advances in neural information processing systems*, 31, 2018.

Link GitHub: https://github.com/MajdHHH/Team-6-SC