Verificarea rețelelor neuronale folosind alpha-beta-CROWN și NeuralSAT pentru benchmark-ul cGan al competiției VNN-Comp2023

Diaconu Laura
Domșa Emanuel
Laptedulce Anastasia
Morariu Ioana-Alexandra
Romaneț Rareș

Abstract

In this paper, we attempted to reproduce the results from the VNN-Comp-2023 competition, specifically focusing on the alpha-beta-CROWN and NeuralSAT tools applied to the cGAN benchmark within the same competition. The paper begins with a brief description of how cGAN neural networks operate, followed by a characterization of the dataset and the steps taken to install the tools and run them. We analyzed the results obtained from the runs and compared them with those obtained in the competition.

Contents

1. Introducere - Funcționarea rețelei neuronale 2
2. Caracterizarea setului de date 4
3. Instalarea și rularea tool-urilor 6
   - 3.1 alpha-beta-CROWN 6
   - 3.2 NeuralSAT 7
4. Interpretarea rezultatelor 7
   - 4.1 alpha-beta-CROWN 7
   - 4.2 NeuralSAT 9
   - 4.3 Compararea rezultatelor obținute pe cele două tool-uri . . . 11
5. Rețelele neuronale în simularea provocărilor reale 14
6. Concluzii 15

1. Introducere - Funcționarea rețelei neuronale

Pentru a putea înțelege mai bine cum funcționează rețeaua neuronală din cadrul **benchmark-ului cGAN al competiției VNN-Comp2023** este necesar să înțelegem cum funcționează în general o rețea de tipul GAN, iar apoi una de tip cGAN.

- **GAN**(Generative Adversarial Networks) este un model neuronal mai special [1] a cărui concept poate fi reprezentat ca un joc între două rețele neuronale distincte adversare.
- **cGAN**(Conditional GAN) [2] ghidează procesul de creare a datelor prin încorporarea unor etichete specificice în GAN ca cele două rețele neuronale adversare să se poată orienta după acestea.

O reprezentare grafică a rețelei neuronale din cadrul Benchmark-ul cGAN al competiției VNN-Comp2023 ar arăta în felul următor:

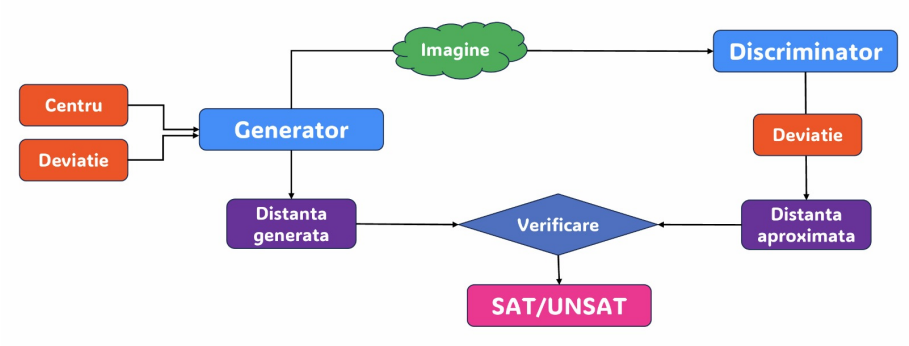


Figure 1: Reprezentare grafică a rețelei neuronale din cadrul Benchmark-ul cGAN al competiției VNN-Comp 2023

Conform schemei, Fig.1, în cazul acestei rețele neuronale cele două rețele distincte adversare se împart în:

- *Generatorul* - funcționează pe baza datelor de intrare(adică etichetele); are rolul de a genera o imagine cu un obstacol aflat la distanța data ca input.
- *Discriminatorul* - funcționează pe baza imaginii returnate de generator, are rolul de a aproxima distața până la obstacol.

Iar datele de intrare sunt:

- *Etichetele* - sunt reprezentate de condiția de distanță și un vector de zgomot, care practic funcțiomnează ca un fel de centru și o deviație

Evaluarea performanței acestei rețele se concentrează pe capacitatea ei de a genera conținut condiționat. În particular, verificarea se bazează pe alinierea distanței prezise de discriminator cu condiția distanței de intrare oferită de generator.

De exemplu, pentru o înțelegere mai clară, am construit următorul exemplu:

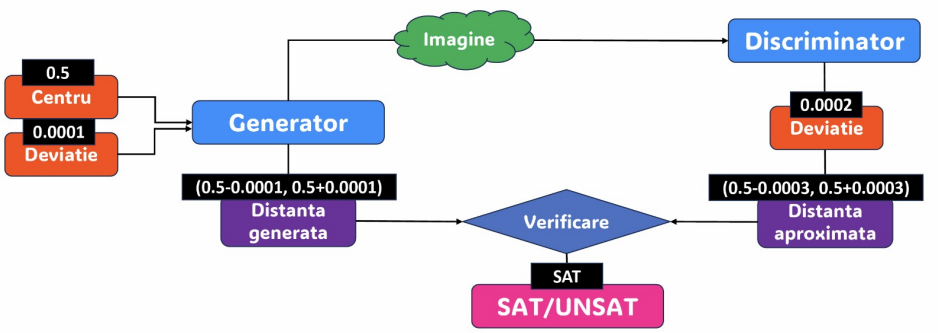


Figure 2: Reprezentare grafică a rețelei neuronale din cadrul Benchmark-ul cGAN al competiției VNN-Comp 2023 - exemplu de date de intrare

În exemplul de mai sus, Fig.2, să presupunem că pentru datele de intrare avem un centru de 0,5 și o deviație de 0,0001. Prin urmare, intervalul distanței la care se poate afla obstacolul este de (0,5-0,0001, 0,5+0,0001).

Obiectivul nostru este să ne asigurăm că distanța prezisă de discriminator se potrivește cu acest interval. Mai exact, distanța prezisă de discriminator trebuie să se situeze în intervalul de intrare adăugând din nou o deviație, de data asta să zicem 0,0002, adică putem obține o distanță aproximativă în intervalul (0,5-0,0003, 0,5+0,0003).

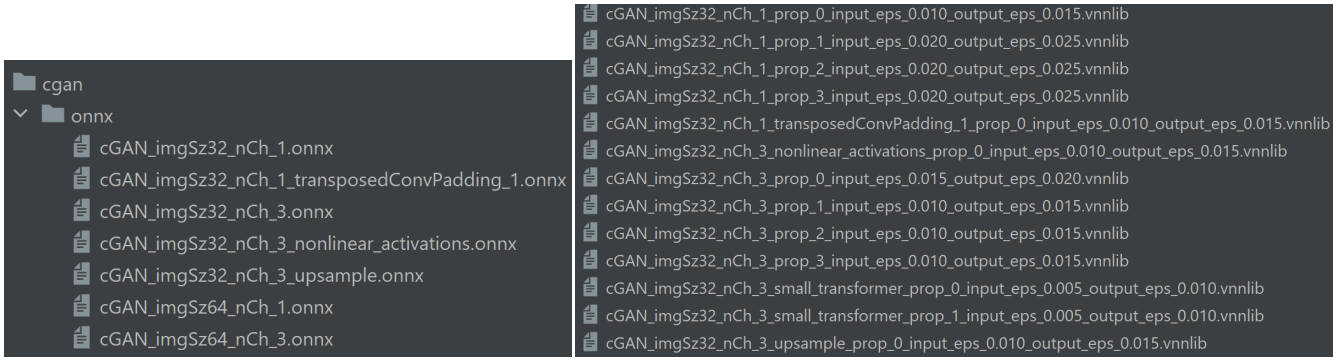
Dacă obținem din partea discriminatorului o distanță din acel interval, acest lucru indică faptul că **imaginile generate respectă condiția de distanță de intrare**.

Rezultatul acestei verificări este **nesatisfiabil** dacă se indică o **aproximare corectă a distanței de către discriminator**, adică nu se poate găsi un contra exemplu, în caz contrar se obține un rezultat **satisfiabil**.

2. Caracterizarea setului de date

Benchmark-ul cGAN aparține mulțimii de benchmark-uri din cadrul competiției VNN-Comp2023 [3]. Acesta conține o rețea de tip generativă adversarială condiționată și un set de specificații. Benchmark-ul este folosit pentru a verifica corectitudinea și robustețea rețelei pe care o conține. În cadrul set-ului de date, există două tipuri de fișiere: .onnx (Open Neural Network Exchange) și .vnnlib(Verification of Neural Networks Library.). În fișierele .onnx sunt reprezentate rețele neuronale, acestea conțin informații necesare pentru a executa modelul neuronal. Fișierele .vnnlib conțin specificațiile ce trebuie respectate de rețeaua neuronală, astfel încât aceasta să fie corectă și robustă.

Atât fișierele .onnx cât și .vnnlib au o denumire sugestivă care să indice informații despre conținutul acestora.



(a) Fișiere .onnx (b) Fișiere .vnnlib

Figure 3: Fișierele benchmark-ului cGAN

- *CGan* - tipul de rețea neurală, în acest caz, o rețea generatoare adversarială condiționată.
- *imgSz32* - dimensiunea imaginei generate, în acest caz, o dimensiune de 32x32 pixeli.
- *nCh_1* - numărul de canale de culoare utilizate în imaginile de intrare sau de ieșire ale rețelei.
- *prop_0* - parametru/proprietate specifică a imaginei
- *input_eps_0* - valoare epsilon utilizată peste datele de intrare ale rețelei.
- *output_eps_0.015* - valoare epsilon utilizată peste datle de ieșire a rețelei.

- *transportedConvPadding_ 1* - tip specific de convoluție a rețelei.
- *nonlinear_ activations* - rețeaua conține funcții de activare non-liniare între straturi.
- *upsample* - modelul neuronal efectuează operații de upsampling, care sunt utilizate pentru a mări dimensiunea spațială a imaginilor sau a datelor. Aceasta poate fi utilă în cazul modelelor generatoare pentru a genera imagini de rezoluție mai mare sau în alte scenarii în care este necesară mărirea dimensiunii datelor.

Toate fisierele .vnnlib au același conținut, diferențiindu-se prin valorile datelor.

Listing 1: Exemplu de cod SMT-LIB din fisierele vnnlib

```
(declare-const X_0 Real)
(declare-const X_1 Real)
(declare-const X_2 Real)
(declare-const X_3 Real)
(declare-const X_4 Real)

(declare-const Y_0 Real)

; Input constraints:
(assert (<= X_0 val_1_X0))
(assert (>= X_0 val_2_X0))

(assert (<= X_1 val_1_X1))
(assert (>= X_1 val_2_X1))

(assert (<= X_2 val_1_X2))
(assert (>= X_2 val_2_X2))

(assert (<= X_3 val_1_X3))
(assert (>= X_3 val_2_X3))

(assert (<= X_4 val_1_X4))
(assert (>= X_4 val_2_X4))

; Output constraints:
(assert (or
    (and (>= Y_0 val_1_Y0))
    (and (<= Y_0 val_2_Y0))
))
```

Luând în considerare succinta descriere a autorului benchmark-ului [4], din fișiere .vnnlib, am dedus următoarele:

- *X_ 0* - reprezintă condiția de distanță ce este o variabilă cuprinsă între 0 și 1 și reprezintă normalizarea de la 0m la 30 m (de exemplu dacă condiția de distanță este 0,5, înseamnă că va fi generată o imagine ce are obstacolul la 0,5*30m=15m)
- *X_ 1 și până la X_ 4* - reprezintă valorile vectorului de zgomot care controlează mediul

- Y_0 - reprezintă distanța dată de către generator

3. Instalarea și rularea tool-urilor

Pentru a efectua rularea benchmark-ului cGan, am selectat cu atenție tool-urile alpha-beta-CROWN și NeuralSAT, inițial verificând dacă acestea pot procesa setul de date specificat. Am făcut această selecție pentru a realiza o comparație între primul tool care a obținut un timp de verificare eficient în timpul competiției și al doilea tool care a înregistrat cel mai ineficient timp de verificare. Alegerea a fost orientată de dorința de a evalua performanțele și eficacitatea fiecărui tool în contextul benchmark-ului cGan.

Ambele tool-uri au fost rulate folosind Python 3.11, pe sistemul de operare Ubuntu 20.04 prin Windows Subsystem for Linux (WSL) pe Windows 10. Am folosit un GPU de laptop NVIDIA GeForce RTX 3060.

3.1 alpha-beta-CROWN

Pentru a instala alpha_beta_CROWN am urmărit pașii de la capitolul *Installation and Setup* din fișierul de READ.ME gasit pe link-ul de github al tool-ului [5]. Suplimentar pașilor din ghid, am clonat submodulul *auto_LIRPA* în interiorul directorului alpha-beta-CROWN.

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

Următorul pas efectuat ce nu se regăsește în ghid, a fost crearea directorului `vnncomp2023_benchmarks` și clonarea repository-ului [6] în interiorul acestuia. Ulterior, am modificat **root path-ul** în fișierul `cgan.yaml` conform locației reale a directorului `cgan`.

La rularea tool-ului ne-am confruntat cu o eroarea legată de lipsa librării `libcudnn_cnn_infer.so.8`. Adăugarea următoarei linii în fișierul `.bashrc` a rezolvat problema.

```
export LD_LIBRARY_PATH=/usr/lib/wsl/lib:$LD_LIBRARY_PATH
```

Comanda de rularea a tool-ului este redată mai jos.

```
python abcrown.py --config exp_configs/vnncomp23/cgan.yaml
```

In dependență de conținutul fișierului *cgan.yaml*, respectiva comandă are output diferit.

Cel mai dificil pas a fost remedierea erorii legate de libraria libcudnn_cnn_infer.so.8., fiind si pasul care a durat cel mai mult timp. Am rezolvat aceasta eroare folosind multiple sugestii gasite pe diferite platforme online [7].

3.2 NeuralSAT

Pentru instalare tool-ului NeuralSAT am urmărit cu strictețe pașii enumerați in ghidul gasit pe site-ul de github al tool-ului [8]. Efectuarea unor pași suplimentari nu a fost necesară.

Rularea tool-ului am facut-o prin executarea următoarei comenzi:

```
python3 main.py --net "cGAN_imgSz32_nCh_3_upsample.onnx" --spec
    "cGAN_imgSz32_nCh_3_upsample_prop_0_input_eps_0.010
    _output_eps_0.015.vnnlib" --result_file result19.txt
    --export_cex--timeout1200
```

Ca si output, comnda de mai sus returneaza un rezultat in fisierul dat ca argument, care va conține sat/unsat și timpul de verificare. Este important de menționat că rularea tool-ului NeuralSAT a implicat un consum mai mare de timp, deoarece a fost necesar să rulăm tool-ul pentru fiecare instanță în parte. În schimb, în cazul alpha-beta-CROWN, am putut procesa toate instanțele simultan.

4. Interpretarea rezultatelor

4.1 alpha-beta-CROWN

Pentru a putea interpreta datele rezultate le-am clasificat manual într-un tabel, precum cel din cadrul competiției pentru a putea face o comparație între ele. Câmpurile de tabel sunt următoarele:

- **Benchmark**: Benchmark-ul pentru care avem înregistrate rezultatele.
- **neural network (ONNX)**: Fișierul ONNX pentru care a rulat tool-ul.
- **specifications (VNNLIB)**: Fișierul VNNLIB pentru care a rulat tool-ul.
- **results (vnncomp2023/us)**: Poate avea valori sat/unsat; sat înseamnă că modelul îndeplinește condițiile din fișierul VNNLIB, adică distanța aproximată de discriminator este aceeași cu distanța pe care a primit-o generatorul ca și dată de intrare; unsat înseamnă că discriminatorul nu a reușit să facă o aproximare corectă. Prima coloană cu rezultate reprezintă rezultatele obținute în competiție, iar a două coloană cu rezultate obținute de noi.

- **time to verify (vnncomp2023/us)**: Timpul de rulare a tool-ului pentru o instanță. Prima coloană reprezintă timpii obținuți în competiție, iar cea de-a doua coloană reprezintă timpii obținuți de noi.

În tabelul, Fig.4, se prezintă o analiză comparativă între rezultatele obținute în urma rulării proprii și cele obținute în cadrul competiției.

Benchmark	neural network (ONNX)	specification (VNNLIB)	results (vn	results (us	time to ve	time to ve
cgan	cGAN_imgSz32_nCh_1.c	cGAN_imgSz32_nCh_1_	sat	sat	7.45	4.71
cgan	cGAN_imgSz32_nCh_1.c	cGAN_imgSz32_nCh_1_	sat	sat	7.43	4.23
cgan	cGAN_imgSz32_nCh_1.c	cGAN_imgSz32_nCh_1_	sat	sat	7.44	4.27
cgan	cGAN_imgSz32_nCh_1.c	cGAN_imgSz32_nCh_1_	unsat	unsat	11.41	7.24
cgan	cGAN_imgSz32_nCh_3.c	cGAN_imgSz32_nCh_3_	sat	sat	7.45	4.29
cgan	cGAN_imgSz32_nCh_3.c	cGAN_imgSz32_nCh_3_	unsat	unsat	13.74	12.26
cgan	cGAN_imgSz32_nCh_3.c	cGAN_imgSz32_nCh_3_	sat	sat	7.43	4.19
cgan	cGAN_imgSz32_nCh_3.c	cGAN_imgSz32_nCh_3_	sat	sat	7.47	4.31
cgan	cGAN_imgSz64_nCh_1.c	cGAN_imgSz64_nCh_1_	unsat	unsat	19.36	26.06
cgan	cGAN_imgSz64_nCh_1.c	cGAN_imgSz64_nCh_1_	unsat	unsat	14.53	14.93
cgan	cGAN_imgSz64_nCh_1.c	cGAN_imgSz64_nCh_1_	sat	sat	7.42	3.12
cgan	cGAN_imgSz64_nCh_1.c	cGAN_imgSz64_nCh_1_	sat	sat	7.46	3.15
cgan	cGAN_imgSz64_nCh_3.c	cGAN_imgSz64_nCh_3_	unsat	unsat	15.68	16.66
cgan	cGAN_imgSz64_nCh_3.c	cGAN_imgSz64_nCh_3_	unsat	unsat	15.29	15.79
cgan	cGAN_imgSz64_nCh_3.c	cGAN_imgSz64_nCh_3_	sat	sat	7.44	3.15
cgan	cGAN_imgSz64_nCh_3.c	cGAN_imgSz64_nCh_3_	sat	sat	7.47	3.14
cgan	cGAN_imgSz32_nCh_3_	cGAN_imgSz32_nCh_3_	unsat	unsat	11.61	6.43
cgan	cGAN_imgSz32_nCh_1_	cGAN_imgSz32_nCh_1_	sat	sat	7.45	3.52
cgan	cGAN_imgSz32_nCh_3_	cGAN_imgSz32_nCh_3_	unsat	unsat	10.79	5.07

Figure 4: Comparare rezultate alpha-beta-CROWN

Conform acestuia este de remarcat faptul că pentru fiecare intrare, rezultatele (sat/unsat) au fost aceleași, atât pentru rezultatele rulării noastre, cât și pentru cele din competiție. Diferența dintre rezultatele din competiție și cele extrase de echipa noastră o constituie timpul de verificare. În tabel sunt evidențiați cu culoarea verde timpii mai reduși de verificare pentru aceleași instanțe.

Timpul de verificare înregistrat al echipei este în medie mai mic cu aproximativ 3 secunde pentru intrările unde rezultatul este satisfiabil. În schimb, pentru intrările cu rezultat nesatisfiabil am obținut un timp de verificare mai mare comparativ cu cel din competiție.

Graficul, Fig.5, evidențiază și mai mult distribuirea timpilor de execuție pentru fiecare instanță în parte.

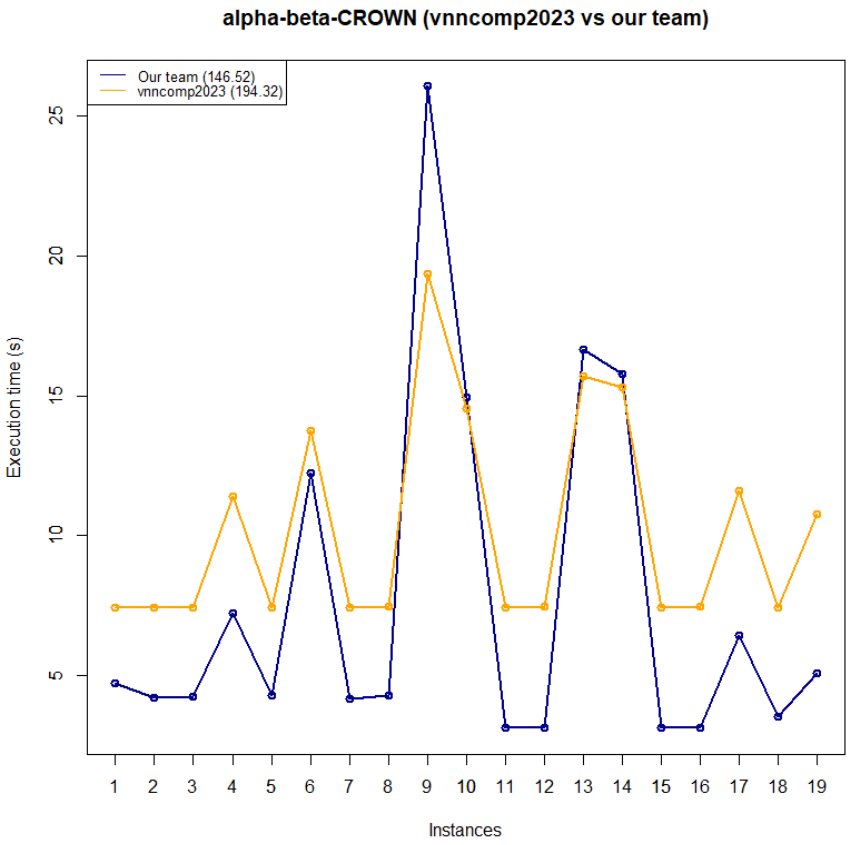


Figure 5: Comparare rezultate alpha-beta-CROWN

4.2 NeuralSAT

Pentru a putea interpreta rezultatele le-am clasificat manual în tabel la fel ca și cele obținute folosind alpha-beta-CROWN. Câmpurile de tabel sunt identice cu cele de la alpha-beta-CROWN, singura excepție fiind ocurența a două tipuri noi de rezultate, și anume:

- **results (vnncomp2023/us)**: Poate avea valori sat/unsat/unknown/error; pentru **sat** și **unsat** interpretările sunt la fel ca și în cazul alpha-beta-CROWN, pentru **unknown** înseamnă că proprietatea nu a putut fi demonstrată, iar **error** rezultă atunci când a avut loc o eroare în timpul rulării tool-ului. Prima coloană cu rezultate reprezintă rezultatele obținute în competiție, iar a două coloană contine rezultatele obținute de noi.

În tabelul, Fig.6, se prezintă o analiză comparativă între rezultatele obținute în urma rulării proprii și cele obținute în cadrul competiției.

Benchmark	neural network (ONNX)	specification (VNNLIB)	results (vr	results (us	time to ve	time to ve
cgan	cGAN_imgSz32_nCh_1.c	cGAN_imgSz32_nCh_1_	sat	sat	4.15	2.79
cgan	cGAN_imgSz32_nCh_1.c	cGAN_imgSz32_nCh_1_	sat	sat	4.11	2.68
cgan	cGAN_imgSz32_nCh_1.c	cGAN_imgSz32_nCh_1_	sat	sat	4.1	2.7
cgan	cGAN_imgSz32_nCh_1.c	cGAN_imgSz32_nCh_1_	unsat	unsat	20.07	13.74
cgan	cGAN_imgSz32_nCh_3.c	cGAN_imgSz32_nCh_3_	sat	sat	4.09	2.92
cgan	cGAN_imgSz32_nCh_3.c	cGAN_imgSz32_nCh_3_	unsat	unsat	813.17	59.65
cgan	cGAN_imgSz32_nCh_3.c	cGAN_imgSz32_nCh_3_	sat	sat	4.06	2.77
cgan	cGAN_imgSz32_nCh_3.c	cGAN_imgSz32_nCh_3_	sat	sat	4.09	2.88
cgan	cGAN_imgSz64_nCh_1.c	cGAN_imgSz64_nCh_1_	unknown	unsat	13.63	159.81
cgan	cGAN_imgSz64_nCh_1.c	cGAN_imgSz64_nCh_1_	unknown	unsat	12.82	150.88
cgan	cGAN_imgSz64_nCh_1.c	cGAN_imgSz64_nCh_1_	sat	sat	4.1	3.02
cgan	cGAN_imgSz64_nCh_1.c	cGAN_imgSz64_nCh_1_	sat	sat	4.07	3.01
cgan	cGAN_imgSz64_nCh_3.c	cGAN_imgSz64_nCh_3_	unknown	unsat	13.49	157.76
cgan	cGAN_imgSz64_nCh_3.c	cGAN_imgSz64_nCh_3_	unknown	unsat	14.1	169.16
cgan	cGAN_imgSz64_nCh_3.c	cGAN_imgSz64_nCh_3_	sat	sat	4.11	3.14
cgan	cGAN_imgSz64_nCh_3.c	cGAN_imgSz64_nCh_3_	sat	sat	4.12	3.26
cgan	cGAN_imgSz32_nCh_3_(	cGAN_imgSz32_nCh_3_	unsat	unsat	25.97	24.09
cgan	cGAN_imgSz32_nCh_1_1	cGAN_imgSz32_nCh_1_1	sat	sat	4.07	3.11
cgan	cGAN_imgSz32_nCh_3_(	cGAN_imgSz32_nCh_3_	error	unsat	5.87	25.37

Figure 6: Comparare rezultate NeuralSAT

Comparând coloanele de *results*, am observat faptul că pentru toate instanțele am obținut valori satisfiabile sau nesatisfiabile. În schimb, competiția a obținut în cazul unor instanțe valori de *unknown* și *error*.

Graficul prezentat mai jos, Fig.7, evidențiază distribuirea timpilor de execuție pentru fiecare instanță în parte. În cadrul graficului de mai jos, se pot observa diferențele dintre timpii de execuție.

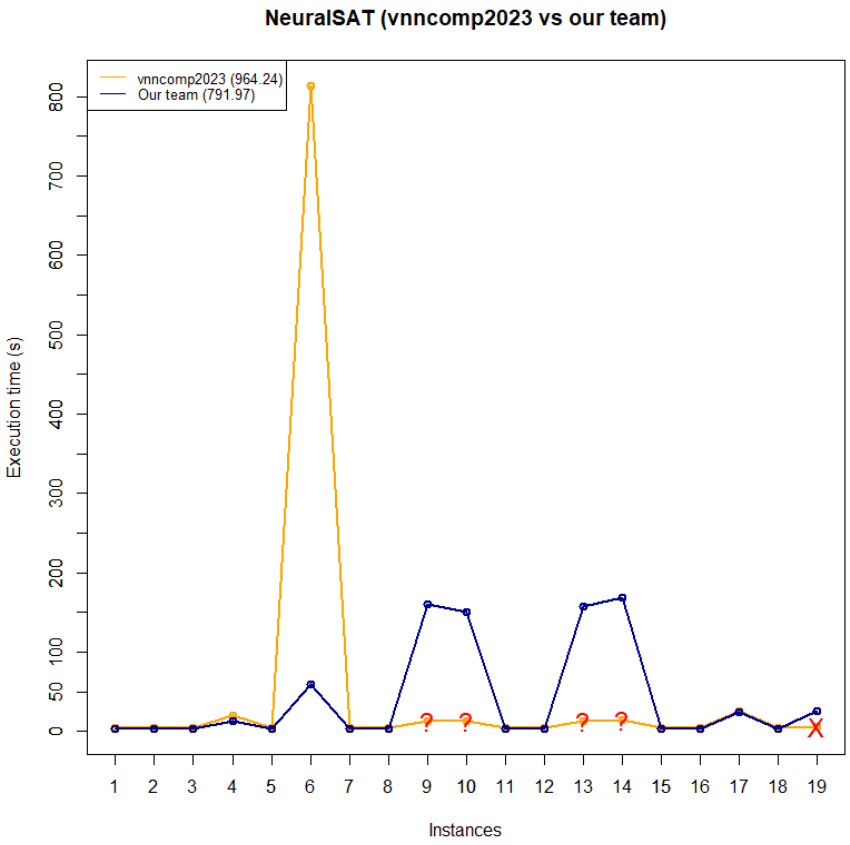


Figure 7: Comparare rezultate NeuralSAT

4.3 Compararea rezultatelor obținute pe cele două tool-uri

În urma rulării a tuturor fișierelor atât pentru alpha_beta_CROWN, cât și pentru NeuralSAT am obținut un număr de instanțe, pentru care timpul alocat verificării a fost depășit, egal cu 0. Prin urmare nu avem penalități pentru niciunul dintre tool-uri.

Totodată am obținut un Total verified egal cu 8, și un Total falsified egal cu 11 pentru ambele tool-uri. Pentru alpha_beta_CROWN rezultatele sunt identice cu cele din competiție. În cazul NeuralSAT, se poate observa că solver-ul a suferit îmbunătățiri în urma livrărilor constante făcute de dezvoltatori deoarece rezultatele obținute de noi au strâns un scor perfect de 100%, mult peste cel obținut de solver în cadrul competiției.

Din tabelul de mai jos putem sa ne dam seama(mai mult sau mai puțin) că formula de calculare a scorului este:

#	Tool	Verified	Falsified	Fastest	Penalty	Score	Percent
1	α-β CROWN	8	11	0	0	190	100%
2	NeuralSAT	8	11	0	0	190	100%

$$\rightarrow \textbf{Verified} \times 10 + \textbf{Falsified} \times 10 - \textbf{Penalty} \times 150$$

Procentajul reprezintă scorul obținut exprimat ca procent din scorul maxim posibil, oferind o imagine de ansamblu asupra performanței, iar formula de calcul este:

$$\frac{\textbf{Score} \times 100}{\textbf{max}(\textbf{Score})}$$

Benchmark	neural network (ONNX)	specification (VNNLIB)	alpha-bet	NeuralSA	alpha-bet	NeuralSA
cgan	cGAN_imgSz32_nCh_1.c	cGAN_imgSz32_nCh_1_	sat	sat	4.71	2.79
cgan	cGAN_imgSz32_nCh_1.c	cGAN_imgSz32_nCh_1_	sat	sat	4.23	2.68
cgan	cGAN_imgSz32_nCh_1.c	cGAN_imgSz32_nCh_1_	sat	sat	4.27	2.7
cgan	cGAN_imgSz32_nCh_1.c	cGAN_imgSz32_nCh_1_	unsat	unsat	7.24	13.74
cgan	cGAN_imgSz32_nCh_3.c	cGAN_imgSz32_nCh_3_	sat	sat	4.29	2.92
cgan	cGAN_imgSz32_nCh_3.c	cGAN_imgSz32_nCh_3_	unsat	unsat	12.26	59.65
cgan	cGAN_imgSz32_nCh_3.c	cGAN_imgSz32_nCh_3_	sat	sat	4.19	2.77
cgan	cGAN_imgSz32_nCh_3.c	cGAN_imgSz32_nCh_3_	sat	sat	4.31	2.88
cgan	cGAN_imgSz64_nCh_1.c	cGAN_imgSz64_nCh_1_	unsat	unsat	26.06	159.81
cgan	cGAN_imgSz64_nCh_1.c	cGAN_imgSz64_nCh_1_	unsat	unsat	14.93	150.88
cgan	cGAN_imgSz64_nCh_1.c	cGAN_imgSz64_nCh_1_	sat	sat	3.12	3.02
cgan	cGAN_imgSz64_nCh_1.c	cGAN_imgSz64_nCh_1_	sat	sat	3.15	3.01
cgan	cGAN_imgSz64_nCh_3.c	cGAN_imgSz64_nCh_3_	unsat	unsat	16.66	157.76
cgan	cGAN_imgSz64_nCh_3.c	cGAN_imgSz64_nCh_3_	unsat	unsat	15.79	169.16
cgan	cGAN_imgSz64_nCh_3.c	cGAN_imgSz64_nCh_3_	sat	sat	7.44	3.14
cgan	cGAN_imgSz64_nCh_3.c	cGAN_imgSz64_nCh_3_	sat	sat	7.47	3.26
cgan	cGAN_imgSz32_nCh_3_	cGAN_imgSz32_nCh_3_	unsat	unsat	11.61	24.09
cgan	cGAN_imgSz32_nCh_1_	cGAN_imgSz32_nCh_1_	sat	sat	7.45	3.11
cgan	cGAN_imgSz32_nCh_3_	cGAN_imgSz32_nCh_3_	unsat	unsat	10.79	25.37

Figure 8: Comparare rezultate NeuralSAT vs alpha-beta-CROWN

Analizând datele din tabelul, Fig.8, putem observa că în cazul instanțelor cu rezultat satisfiabil (sat) verificatorul NeuralSAT a obținut timpi de execuție mai reduși decât alpha-beta-CROWN. Pe de altă parte, în cazul instanțelor cu rezultat nesatisfiabil (unsat) timpii de execuție obținuți de alpha-beta-CROWN sunt semnificativ mai reduși decât cei obținuți de NeuralSAT. Acest lucru este evidențiat și in graficul Fig.9. Graficul respectiv arată distribuirea timpilor de execuție pentru ambele tool-uri, comparând instanța cu instanța.

Al doilea grafic, Fig.10, prezintă suma timpilor de execuție pe parcursul rulării tuturor celor 19 instanțe pe ambele verificatoare. Din acest grafic se poate trage concluzia că alpha-beta-CROWN este un tool mult mai eficient din punct de vedere a timpului de execuție.

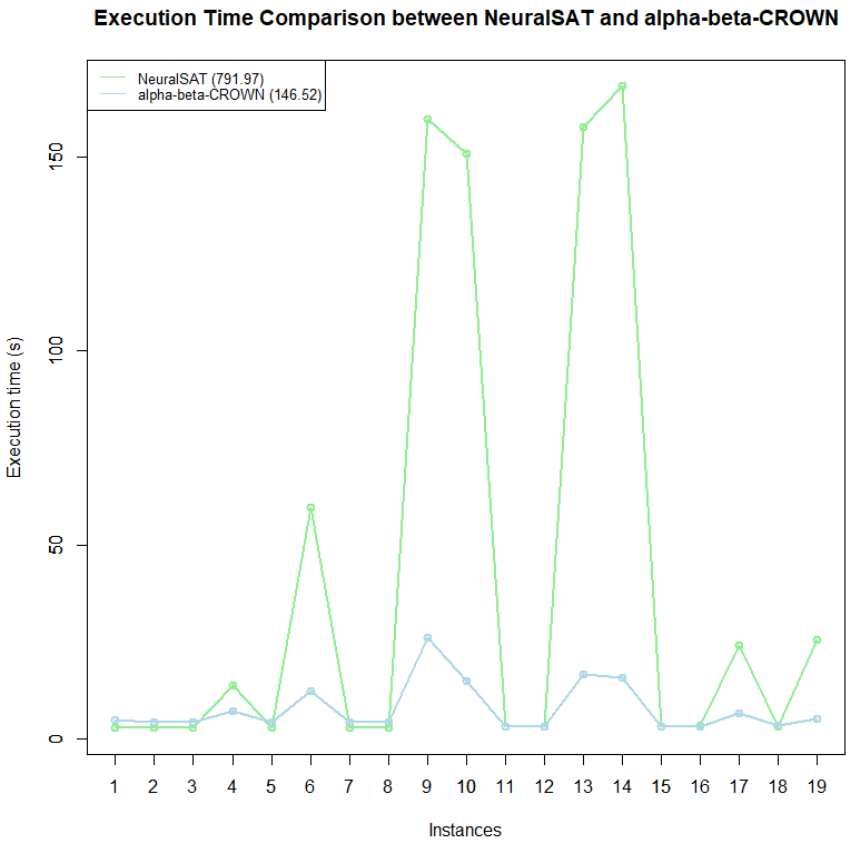


Figure 9: Comparare rezultate NeuralSAT vs alpha-beta-CROWN

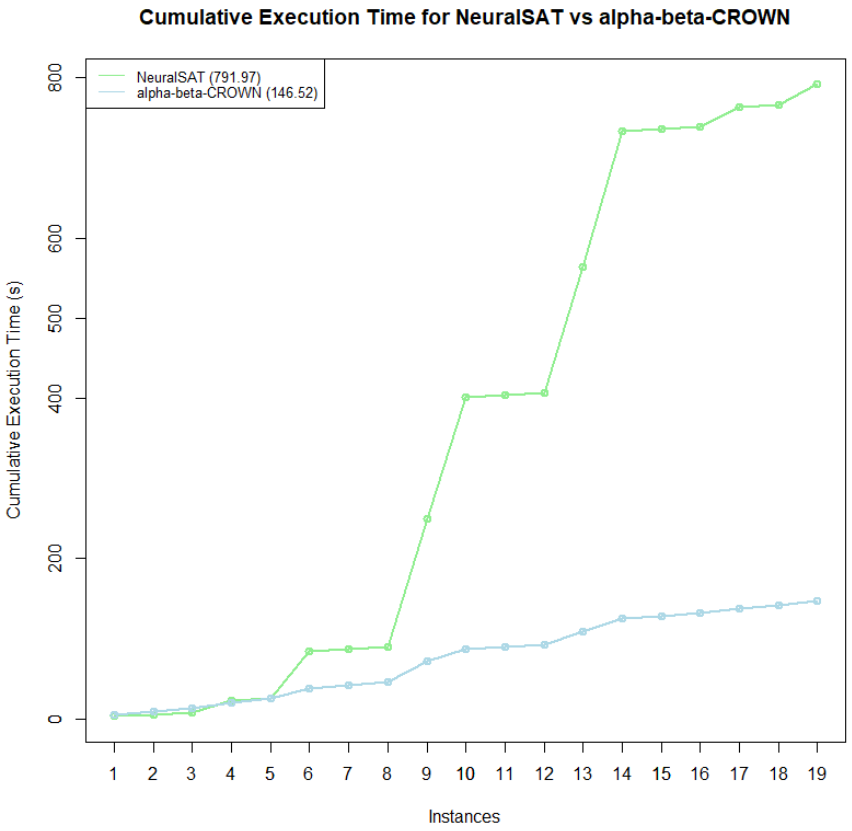


Figure 10: Cumulative Execution Time for NeuralSAT vs alpha-beta-CROWN

5. Rețelele neuronale în simularea provocărilor reale

Verificarea rețelelor neuronale este o etapă crucială în dezvoltarea și aplicarea modelelor avansate. Aceasta este esențială din mai multe motive importante.

În primul rând, corectitudinea și fiabilitatea rețelelor neuronale sunt imperative. Ele sunt utilizate într-o varietate de domenii, de la medicină și tehnologie până la securitate cibernetică și vehicule autonome. Verificarea ne asigură că aceste rețele funcționează conform așteptărilor, oferind rezultate precise și fiabile într-o gamă largă de situații.

Siguranța reprezintă un alt aspect esențial. În domenii critice precum medicina sau industria automotive, erorile în funcționarea rețelelor neuronale pot avea consecințe grave. Verificarea este necesară pentru a identifica și remedia eventualele vulnerabilități care ar putea pune în pericol sistemul.

În industria auto, utilizarea învățării profunde și a viziunii artificiale pentru generarea de imagini noi joacă un rol esențial. Rețelele de tip CGAN permit

generarea de imagini sintetice pentru antrenarea și testarea vehiculelor autonome, simularea diverselor scenarii de conducere și optimizarea sistemelor de vizualizare și senzorilor [9]. Ele contribuie la îmbunătățirea siguranței și eficienței în transporturi și la dezvoltarea vehiculelor autonome mai avansate. Utilizarea rețelelor neuronale în generarea de imagini pentru industria auto contribuie prin antrenarea eficientă a algoritmilor, simularea sigură a situațiilor de trafic și optimizarea senzorilor, accelerând dezvoltarea și îmbunătățirea vehiculelor autonome.

De asemenea, verificarea rețelelor neuronale contribuie la prevenirea bias-ului și discriminării. Aceste rețele pot fi influențate de prejudecăți încorporate în datele de antrenament. Prin teste și evaluări riguroase, putem identifica și corecta aceste bias-uri pentru a asigura obiectivitate și corectitudine în rezultatele obținute.

6. Concluzii

În această lucrare, am analizat în detaliu benchmark-ul cGAN din cadrul competiției VNN-Comp2023. Am descris pașii de instalare și rulare a instrumentelor alpha-beta-CROWN și NeuralSAT. Ulterior, am interpretat rezultatele obținute în urma rulării. Aceste acțiuni au avut ca scop analizarea performanței și funcționării rețelelor neuronale.

Instrumentele folosite au fost dezvoltate special pentru verificarea formală a corectitudinii rețelelor neuronale. Prin intermediul rulării tool-urilor pe setul cGAN, s-a evaluat capacitatea rețelelor, din cadrul benchmark-ului, de a se alinia cu condițiile de intrare specificate în fișierele .vnnlib.

Rezultatele în urmă rulării, au fost extrase în tabele pentru a facilita compararea cu rezultatele din cadrul competiției. În compararea rezultatelor noastre cu cele din competiție, s-a observat o aliniere în ceea ce privește rezultatele satisfiabile și nesatisfiabile. Diferențe au existat, în schimb, la timpii necesari pentru verificare. Spre final, am menționat despre eficiența utilizării cGAN în contextul recunoașterii imaginilor.

Bibliography

[1] A. Limarc, "What Is a Conditional Generative Adversarial Network? - DZone — dzone.com," https://dzone.com/articles/what-is-a-conditional-generative-adversarial-netwo, 2023.

[2] M. Mirza and S. Osindero, "Conditional generative adversarial nets," *CoRR*, vol. abs/1411.1784, 2014. [Online]. Available: http://arxiv.org/abs/1411.1784

[3] "vnncomp2023 benchmarks," https://github.com/ChristopherBrix/vnncomp2023_benchmarks, [Accessed 15-02-2024].

[4] "Benchmark discussion · Issue 2 · stanleybak/vnncomp2023 — github.com," https://github.com/stanleybak/vnncomp2023/issues/2, [Accessed 25-01-2024].

[5] "Read.me alpha-beta-crown," https://github.com/feiyang-cai/alpha-beta-CROWN_vnncomp23/blob/master/README_abcrown.md, [Accessed 15-02-2024].

[6] "vnncomp2023_benchmarks/benchmarks/cgan at main Christopher-Brix/vnncomp2023_benchmarks — github.com," https://github.com/ChristopherBrix/vnncomp2023_benchmarks/tree/main/benchmarks/cgan, [Accessed 25-01-2024].

[7] "Bashrc fix," https://discuss.pytorch.org/t/libcudnn-cnn-infer-so-8-library-can-not-found/164661, [Accessed: 19.12.2023].

[8] "Neuralsat install.md," https://github.com/dynaroars/neuralsat/blob/develop/INSTALL.md, [Accessed: 15.02.2024].

[9] S. Ranjan and D. S. Senthamilarasu, "Applied Deep Learning and Computer Vision for Self-Driving Cars — books.google.ro," https://books.google.ro/books?hl=en&lr=&id=nIX4DwAAQBAJ&oi=fnd&pg=PP1&dq=generate+object+with+a+neural+network+to+help+automotive+industry&ots=DGTqyWd5jC&sig=eRE37mHEAJMuohJsdyj4W5odKbg&redir_esc=y#v=onepage&q&f=false, 2020, [Accessed 10-02-2024].