Verificarea rețelelor neuronale folosind alpha_beta_crown și NeuralSat pentru benchmark-ul cGan al competiției VNN-Comp 2023

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

Abstract

In this paper, we attempted to reproduce the results from the VNNCOMP-2023 competition, specifically focusing on the alpha-beta-CROWN and Marabou 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 data obtained from the runs and compared them with those obtained in the competition. **TO DO: Maybe add a brief ideea of the conlusions made after analysis**

Contents

1. Introducere - Funcționarea rețelei neuronale

Pentru a putea înțelege mai bine cum funcționează rețeaua neuronală din cadrul **Benchmark-ul cGAN al competiției VNN-Comp 2023** 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-Comp 2023 ar arăta în felul următor:

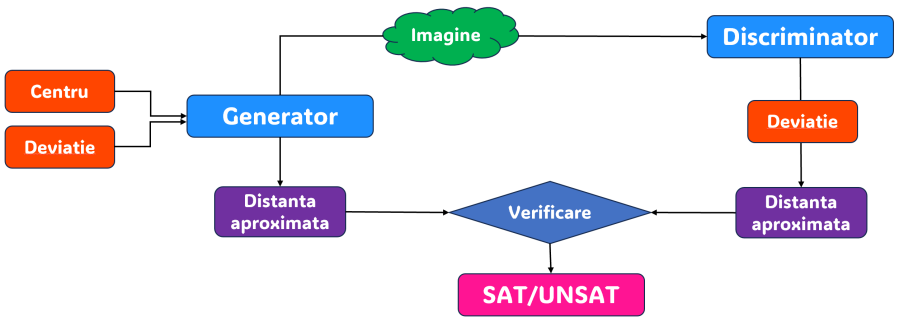


Figure 1: Reprezentare grafică a rețelei neuronale din cadrul Benchmark-ul cGAN al competiției VNN-Comp 2023

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:

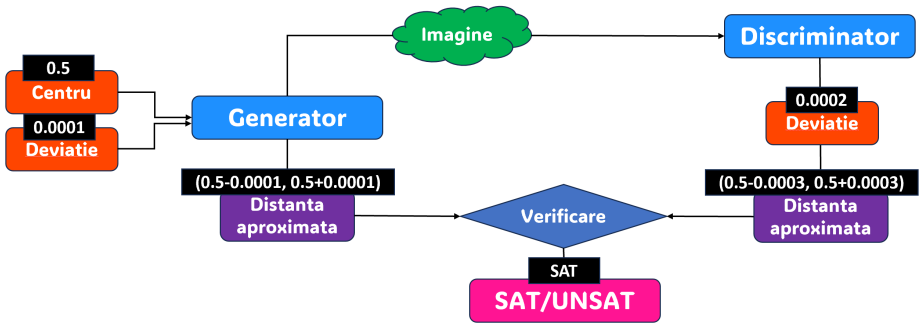


Figure 2: Reprezentare grafică a rețelei neuronale din cadrul Benchmark-ul cGAN al competiției VNN-Comp 2023 - exemplu de date de intrare

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).

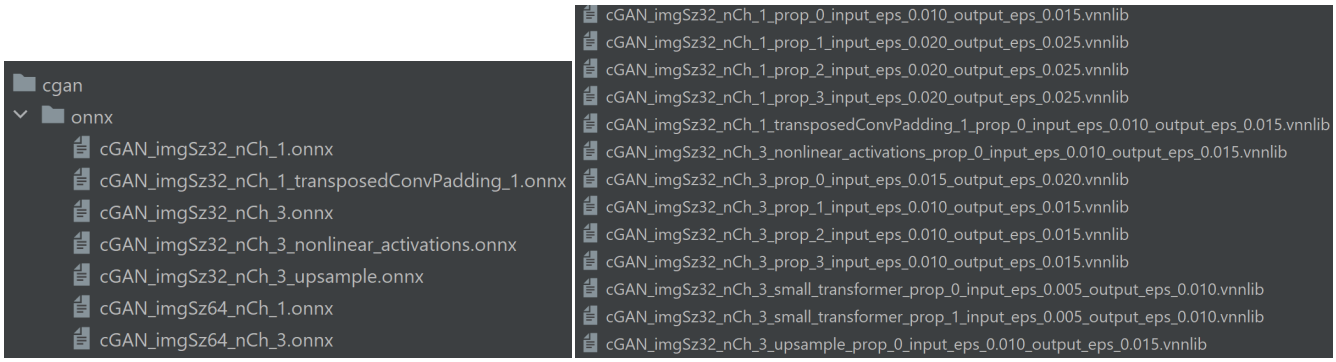
Dacă obținem din partea discriminatorului o distanță din acel interval, acest lucru indică faptul că **imaginile generate respectă condiția de distanță de intrare**.

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

2. Caracterizarea setului de date

Benchmark-ul cGAN aparține mulțimii de benchmark-uri din cadrul competiției VNN-COMP 2023 [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. 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 , 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 [4]. 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 [5] în interiorul acestuia. Ulterior, am modificat **root path-ului** 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 [6].

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 [7]. 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

În tabelul de mai jos se prezintă o analiză comparativă între rezultatele obținute în urma rulării proprii și cele obținute în cadrul competiției. Fișierele cu rezultatele rulărilor pot fi accesate la următoarele linkuri: rezultate competiție rezultate echipă

Din tabelul 4, este de notat faptul că pentru fiecare intrare, rezultatele(sat/unsat) au fost aceleași, atât pentru rezultatele rulării noastre, cât și pentru cele din competiție. Diferenta dintre competitie si echipa o constituie timpul de verificare. Timpul de verificare înregistrat al echipei este mai mic cu aproximativ 3 secunde pentru intrările unde rezultatul este satisfiabil. În schimb, pentru intrările cu rezultat nesatisfiabil timpul de verificare este considerabil mai mare.

Conform tabelului prezentat în 4, este de remarcat faptul că pentru fiecare intrare, rezultatele satisfiabil/nesatisfiabil) au fost aceleași, atât pentru rezultatele rulării proprii, cât și pentru cele din competiție. Diferența majoră între competiție și echipa noastră o constituie timpul de verificare. Timpul de verificare înregistrat al echipei noastre este mai mic cu aproximativ 3 secunde pentru intrările cu rezultat satisfiabil. În schimb, pentru intrările cu rezultat nesatisfiabil, timpul de verificare este considerabil mai mare.

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	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: Rezultate alpha-beta-CROWN

4.2 NeuralSAT

4.3 Compararea rezultatelor obținute pe cele două tool-uri

5. Rețelele neuronale în simularea provocărilor reale

Verificarea rețelelor neuronale este o etapă crucială în dezvoltarea și aplicarea acestor modele avansate. 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. Aceste tehnologii permit generarea de imagini sintetice pentru antrenarea și testarea vehiculelor autonome, simularea

diverselor scenarii de conducere și optimizarea sistemelor de vizualizare și senzorilor. 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

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

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

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

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