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

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

Contents

Abstract

In this paper we explored and tested the use of Conditional Generative Adversarial Networks and it's features in the context of image recognition tasks. For testing matters we use the cGAN benchmark from VNNComp 2023. The paper includes description of the above mentioned benchmark, details of configuration and running alpha_beta_crown tool and a short interpretation of the obtained results.

1. Rețelele neuronale în simularea provocărilor reale

Odată cu avansul rapid al tehnologiei, rețelele neuronale au devenit instrumente important în abordarea unor probleme complexe ale societății din zilele noastre [1]. Un exemplu al acestei evoluții este modelul de rețea neuronală utilizată în Benchmark-ul cGAN al competiției VNN-Comp 2023.

Această rețea neuronală propune o soluție inovatoare pentru problemele din viața reală, contribuind în special la simularea și anticiparea obstacolelor în mediul virtual. Prin intermediul Generatorului și Discriminatorului, modelul poate fi instruit să genereze imagini ale scenelor cu obstacole, având în vedere condiții specifice de intrare.

Un exemplu concret ar putea fi domeniul vehiculelor autonome. Capacitatea de a anticipa și recunoaște corect obstacolele din mediu este esențială pentru siguranța și eficiența acestor vehicule. Modelul de rețea neurală propus poate fi utilizat pentru a genera imagini realiste ale scenelor din trafic, luând în considerare condiții variate, cum ar fi distanța până la obstacole și variabilele de mediu.

Mai mult, abordarea acestui model poate extinde aplicațiile în diverse sectoare, inclusiv în industria jocurilor video, realitatea virtuală sau simularea de medii complexe pentru instruirea roboților. Capacitatea rețelei de a produce conținut condiționat, ghidat de etichete specifice, deschide noi posibilități pentru rezolvarea unor probleme de simulare și anticipare într-un mod mai precis și eficient.

Diversitatea modurilor în care această rețea neuronală poate să fie explorată și folosită către potențiali utilizatori poate avea o influența pozitivă în domenii care necesită simulare și anticipare precisă a unor condiți specifice, deschizând noi direcții pentru inovație.

2. Analiza modului de funcționare a rețelei neuronale

GAN(Generative Adversarial Networks) este un model neuronal mai special [2] a cărui concept poate fi reprezentat ca un joc între două rețele neuronale distincte adversare.

cGAN(Conditional GAN) [3] 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.

În cazul nostru, cele două rețele distincte adversare se împart în:

- *Generatorul* - pe baza pe datele de intrare(adică etichetele), are rolul de a

genera o valoare reprezentată de o distanță și o imagine cu un obstacol aflat la o acea distanță

- *Discriminatorul* - 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, (practic 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, 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 află 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 returnează un rezultat satisfiabil dacă se indică o aproximare corectă a distanței de către discriminator, în caz contrar se obține un rezultat nesatisfiabil.

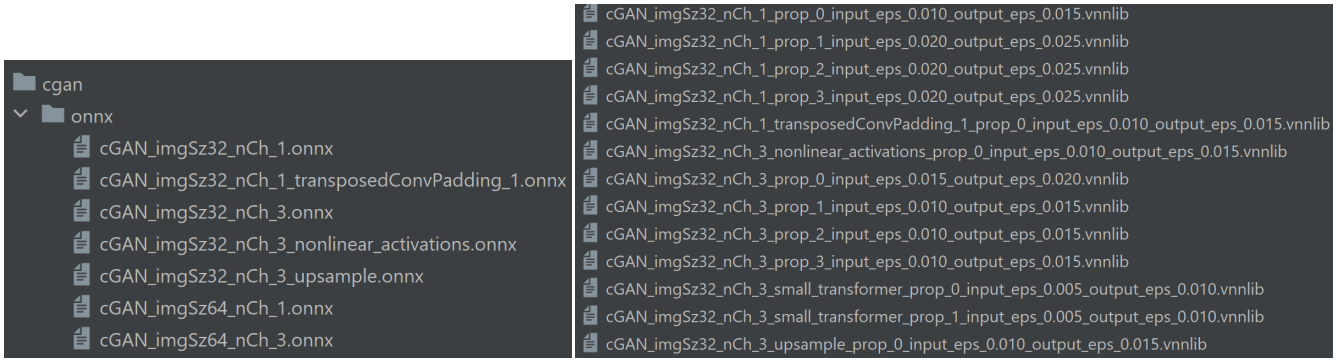
3. Caracterizarea setului de date

Benchmark-ul cGan este un dataset ce se folosește împreună cu un tool de tip SMT solver pentru a verifica corectitudinea rețelelor neuronal. Acesta conține fișiere de tip vnnlib și onnx ce au roluri specifice în evaluarea și testarea rețelelor neuronale:

- Modelul neuronal dat spre testare este reprezentat sub forma fișierelor în format **.onnx**. Aceste fișiere conțin descrierea arhitecturii și parametrii rețelei neuronale.

- Specificările care sunt supuse verificării sunt reprezentate printr-un set de fișiere în format **.vnnlib**. Acestea reprezinta constrangeri ce trebuie sa fie satisfacute de outputul modelului neuronal.

Atât fișierele .onnx cât și .vnnlib au o denumire sugestivă care să indice funcționalitatea fiecăruia.



(a) Fișiere .onnx (b) Fișiere .vnnlib

Figure 1: Fișierele benchmark-ului cGAN

Astfel dacă descompunem numele fișierelor obținem următoarele informații:

- *CGan* - tipul de rețea neurală, în acest caz, o rețea generatoare adversarială condiționată
- *imgSz32* - dimensiunea imaginilor utilizate în antrenare, 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 specific al modelului sau setărilor de antrenare
- *input_eps_0* - valoare specifică a epsilon utilizată în procesul de antrenare
- *output_eps_0.015* - valoare specifică de epsilon pentru procesul de antrenare, dar aplicată pentru datele de ieșiri.
- *transportedConvPadding_1* - tip specific de convoluție care are o anumită configurație pentru gestionarea padding-ului
- *nonlinear_activations* - funcții de activare non-liniare între straturi
- *upsample* - 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.

Pentru partea de constrângeri, toate fișierele arată în felul următor (sunt specificate valorile val_i_Xj și val_i_Y0 în care $i \in \{1, 2\}, j \in \{0, 1, 2, 3, 4\}$):

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))
))
```

Din aceste fișiere, noi 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

4. Configurare tool-uri

Pentru a instala tool-urile am instalat în prealabil WSL și, prin urmare, toate tool-urile au fost instalate utilizând linia de comandă specifică Linux (bash).

4.1 Configurare alpha_beta_crown

Pentru rularea benchmark-ului cGan am ales ca și tool-uri alpha_beta_crown si NeuralSAT, verificând prima dată cu atenție dacă acestea au putut rula setul de date pe tool-urile alese. Am facut această alegere deoarece am dorit o comparație dintre primul tool care a obțiunut un timp de verificare mai eficient în timpul competiției, iar cel de al doilea care a obținut cel mai ineficient timp de verificare.

Alpha_beta_CROWN a fost testat pe Python 3.11, sistemul de operare utilizat a fost Ubuntu 20.04 prin Windows Subsystem for Linux (WSL) pe Windows 10, folosind un GPU de laptop NVIDIA GeForce RTX 3060. A fost instalat într-un mediu miniconda. În urmatorii pasi se va explica cum a fost facută instalarea și rularea tool-ului.

Pai de Instalare:

1. Instalarea Miniconda a fost realizată prin linia de comandă (bash) cu următorii pași [4]:

```
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
~/miniconda3/bin/conda init bash
```

2. Clonarea repository-ului de la adresa [5]:

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

3. Clonarea submodulului `auto_Lirpa` în interiorul directorului `alpha-beta-CROWN`:

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

4. Configurarea mediului Conda:

```
conda deactivate; conda env remove --name alpha-beta-crown
conda env create -f complete_verifier/environment.yaml --name
    alpha-beta-crown
conda activate alpha-beta-crown
```

5. Crearea directorului `vnncomp2023_benchmarks` și clonarea repository-ului [6] în interiorul acestuia. Modificarea **root path-ului** în fișierul `cgan.yaml` conform locației reale a directorului `cgan`.

6. Rularea verificatorului:

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

7. Pentru a rezolva eroarea legată de `libcudnn_cnn_infer.so.8`, adăugarea următoarei linii în `.bashrc`:

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

8. Rularea comenzii și salvarea rezultatelor în `results.txt`:

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

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].

4.2 Configurare Neuralsat

Pentru a instala și rula cu succes neuralsat, a fost necesar să avem instalat Miniconda. Pași de instalare pentru tool au fost descrisi in capitolul anterior.

Pentru a instala și rula NeuralSAT am urmat toți pașii recomandați aici [8]. Am urmat și pașii opționali, dar aceștia nu sunt obligatorii pentru a rula cu succes solver-ul.

5. Interpretare rezultate

5.1 alpha_beta_crown

Rezultatele obținute după rularea toolului alpha_beta_crown pe benchmark-ul cGAN au fost sub forma unui stream în consolă.

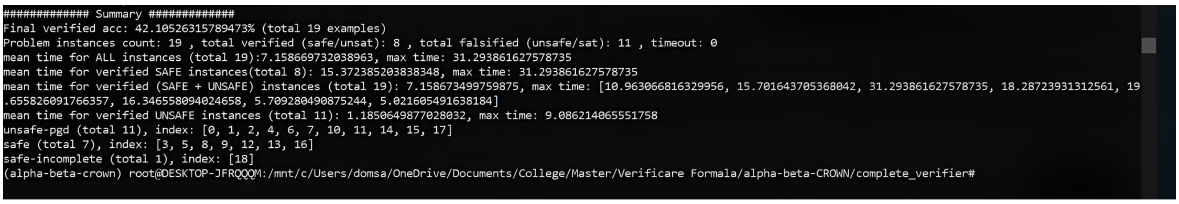


Figure 2: Rezultat stream consola pentru rularea tuturor fisierelor din benchmark-ului cGAN pe alpha_beta_crown

Imaginea de mai sus, figura 2,conține un output de la un solver alpha_beta_crow folosit pentru verificarea formală a unei rețele neuronale. Interpretarea informațiilor afișate este următoarea

- **Final verified acc:** Acuratețea finală a instanțelor verificate ca sigure - 42.10%
- **Problem instances count:** Numărul de instanțe cu probleme detectate în timpul verificării - 19
- **Total verified (safe/unsat):** Numărul total de instanțe verificate ca sigure - 8
- **Total falsified (unsafe/sat):** Numărul de instanțe identificate ca nesigure în timpul verificării - 11
- **timeout:** Instanțele pentru care timpul alocat verificării a fost depășit - 0
- **mean time for ALL instances:** Timpul mediu necesar pentru verificarea tuturor instanțelor - 7.15
- **max time for ALL instances:** Timpul maxim necesar pentru verificarea tuturor instanțelor - 31.29
- **mean time for SAFE instances(total 8):** Timpul mediu necesar pentru verificarea tuturor instanțelor care au dat unsut - 7.15
- **max time for SAFE instances(total 8):** Timpul maxim necesar pentru verificarea tuturor instanțelor care au dat unsut - 31.29

- **mean time for verified SAFE + UNSAFE instances(total 19):** Timpul mediu pentru instanțele verificate ca sigure și nesigure - 7.15
- **max time for verified SAFE + UNSAFE instances(total 19):** Timpul maxim pentru instanțele verificate ca sigure și nesigure - [10.00, 15.70, 31.29...etc.]
- **mean time for verified UNSAFE instances(total 11):** Timpul mediu pentru verificarea instanțelor nesigure. - 1.18
- **max time for verified UNSAFE instances(total 11):** Timpul maxim pentru verificarea instanțelor nesigure. - 9.08
- **unsafe-pgd (total 11):** Lista instanțelor nesigure identificate cu metoda PGD - index: [30, 1, 2, 4, 6, 7, 10, 11, 14, 15, 17]
- **safe (total 7):** Lista instanțelor sigure identificate - index: [3, 5, 8, 9, 12, 13, 16]
- **safe-incomplete (total 1):** Numărul de instanțe sigure dar incomplete, cu indexurile specificate - [18]

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

- benchmark(benchmark-ul pentru care avem înregistrate rezultatele)
- model_neuronal(fișierul ONNX pentru care a rulat tool-ul)
- specificații(fieșierul VNNLIB pentru care a rulat tool-ul)
- rezultat(poate avea valori sat/unsat; sat înseamnă că modelul indeplinește condițiile din fișierul VNNLIB, adică distanța aproximată de discriminator, este aceeași cu distanța pe care a primit-o generator-ul ca și data de intrare; unsat înseamnă că discriminatorul nu a reușit să facă o aproximare corectă)
- timp_de_verificare(timpul de rulare a fișierului ONNX pentru specificatiile din fișierul VNNLIB, exprimat în secunde).

Rezultatele obținute se găsesc în tabelul de mai jos sau (aici):

Benchmark	ONNX	Specification (VNNLIB)	Result	Time to Verify (s)
cgan	cGAN_imgSz32_nCh_1.	cGAN_imgSz32_nCh_1_prop_0_input_eps_0.010_output_eps_0.015.	sat	4.39915
cgan	cGAN_imgSz32_nCh_1.	cGAN_imgSz32_nCh_1_prop_1_input_eps_0.020_output_eps_0.025.	sat	4.45147
cgan	cGAN_imgSz32_nCh_1.	cGAN_imgSz32_nCh_1_prop_2_input_eps_0.020_output_eps_0.025.	sat	4.37929
cgan	cGAN_imgSz32_nCh_1.	cGAN_imgSz32_nCh_1_prop_3_input_eps_0.020_output_eps_0.025.	unsat	11.11230
cgan	cGAN_imgSz32_nCh_3.	cGAN_imgSz32_nCh_3_prop_0_input_eps_0.015_output_eps_0.020.	sat	4.42455
cgan	cGAN_imgSz32_nCh_3.	cGAN_imgSz32_nCh_3_prop_1_input_eps_0.010_output_eps_0.015.	unsat	16.71046
cgan	cGAN_imgSz32_nCh_3.	cGAN_imgSz32_nCh_3_prop_2_input_eps_0.010_output_eps_0.015.	sat	4.31144
cgan	cGAN_imgSz32_nCh_3.	cGAN_imgSz32_nCh_3_prop_3_input_eps_0.010_output_eps_0.015.	sat	4.40774
cgan	cGAN_imgSz64_nCh_1.	cGAN_imgSz64_nCh_1_prop_0_input_eps_0.010_output_eps_0.015.	unsat	30.27740
cgan	cGAN_imgSz64_nCh_1.	cGAN_imgSz64_nCh_1_prop_1_input_eps_0.005_output_eps_0.010.	unsat	19.29929
cgan	cGAN_imgSz64_nCh_1.	cGAN_imgSz64_nCh_1_prop_2_input_eps_0.010_output_eps_0.015.	sat	4.42568
cgan	cGAN_imgSz64_nCh_1.	cGAN_imgSz64_nCh_1_prop_3_input_eps_0.005_output_eps_0.010.	sat	4.42309
cgan	cGAN_imgSz64_nCh_3.	cGAN_imgSz64_nCh_3_prop_0_input_eps_0.010_output_eps_0.015.	unsat	21.70824
cgan	cGAN_imgSz64_nCh_3.	cGAN_imgSz64_nCh_3_prop_1_input_eps_0.010_output_eps_0.015.	unsat	21.05809
cgan	cGAN_imgSz64_nCh_3.	cGAN_imgSz64_nCh_3_prop_2_input_eps_0.005_output_eps_0.010.	sat	4.37459
cgan	cGAN_imgSz64_nCh_3.	cGAN_imgSz64_nCh_3_prop_3_input_eps_0.010_output_eps_0.015.	sat	4.47376
cgan	cGAN_imgSz32_nCh_3_nonlinear_activations.	cGAN_imgSz32_nCh_3_nonlinear_activations_prop_0_input_eps_0.010_output_eps_0.015.	unsat	10.39218
cgan	cGAN_imgSz32_nCh_1_transposedConvPadding_1.	cGAN_imgSz32_nCh_1_transposedConvPadding_1_prop_0_input_eps_0.010_output_eps_0.015.	sat	4.41804
cgan	cGAN_imgSz32_nCh_3_upsample.	cGAN_imgSz32_nCh_3_upsample_prop_0_input_eps_0.010_output_eps_0.015.	unsat	9.82781

Table 1: Rezultatele Benchmark-ului în urma verificării rețelei neuronale rulate de noi

Pentru a ne verifica corectitudinea rezultatelor obținute, am facut o analiză comparativă între fișierul obținut la rularea noastră și cel obținut din competiție pe care puteți să îl vedeți pe linkul (aici) sau în tabelul de mai jos:

Benchmark	Neural Network (ONNX)	Specification (VNNLIB)	Time to Prepare Instance (s)	Result	Time to Verify (s)
cgan	vnncomp2023_benchmarksbenchmarksganGAN_imgSz32_nCh_1.	vnncomp2023_benchmarksbenchmarksganGAN_imgSz32_nCh_1_prop_0_input_eps_0.010_output_eps_0.015.	26.44	sat	7.45
cgan	vnncomp2023_benchmarksbenchmarksganGAN_imgSz32_nCh_1.	vnncomp2023_benchmarksbenchmarksganGAN_imgSz32_nCh_1_prop_1_input_eps_0.020_output_eps_0.025.	24.3	sat	7.43
cgan	vnncomp2023_benchmarksbenchmarksganGAN_imgSz32_nCh_1.	vnncomp2023_benchmarksbenchmarksganGAN_imgSz32_nCh_1_prop_2_input_eps_0.020_output_eps_0.025.	24.33	sat	7.44
cgan	vnncomp2023_benchmarksbenchmarksganGAN_imgSz32_nCh_1.	vnncomp2023_benchmarksbenchmarksganGAN_imgSz32_nCh_1_prop_3_input_eps_0.020_output_eps_0.025.	27.55	unsat	11.41
cgan	vnncomp2023_benchmarksbenchmarksganGAN_imgSz32_nCh_3.	vnncomp2023_benchmarksbenchmarksganGAN_imgSz32_nCh_3_prop_0_input_eps_0.015_output_eps_0.020.	26.35	sat	7.45
cgan	vnncomp2023_benchmarksbenchmarksganGAN_imgSz32_nCh_3.	vnncomp2023_benchmarksbenchmarksganGAN_imgSz32_nCh_3_prop_1_input_eps_0.010_output_eps_0.015.	27.73	unsat	13.74
cgan	vnncomp2023_benchmarksbenchmarksganGAN_imgSz32_nCh_3.	vnncomp2023_benchmarksbenchmarksganGAN_imgSz32_nCh_3_prop_2_input_eps_0.010_output_eps_0.015.	24.47	sat	7.43
cgan	vnncomp2023_benchmarksbenchmarksganGAN_imgSz32_nCh_3.	vnncomp2023_benchmarksbenchmarksganGAN_imgSz32_nCh_3_prop_3_input_eps_0.010_output_eps_0.015.	24.32	sat	7.47
cgan	vnncomp2023_benchmarksbenchmarksganGAN_imgSz64_nCh_1.	vnncomp2023_benchmarksbenchmarksganGAN_imgSz64_nCh_1_prop_0_input_eps_0.010_output_eps_0.015.	33.85	unsat	19.36
cgan	vnncomp2023_benchmarksbenchmarksganGAN_imgSz64_nCh_1.	vnncomp2023_benchmarksbenchmarksganGAN_imgSz64_nCh_1_prop_1_input_eps_0.005_output_eps_0.010.	31.78	unsat	14.53
cgan	vnncomp2023_benchmarksbenchmarksganGAN_imgSz64_nCh_1.	vnncomp2023_benchmarksbenchmarksganGAN_imgSz64_nCh_1_prop_2_input_eps_0.010_output_eps_0.015.	24.51	sat	7.42
cgan	vnncomp2023_benchmarksbenchmarksganGAN_imgSz64_nCh_1.	vnncomp2023_benchmarksbenchmarksganGAN_imgSz64_nCh_1_prop_3_input_eps_0.005_output_eps_0.010.	24.36	sat	7.46
cgan	vnncomp2023_benchmarksbenchmarksganGAN_imgSz64_nCh_3.	vnncomp2023_benchmarksbenchmarksganGAN_imgSz64_nCh_3_prop_0_input_eps_0.010_output_eps_0.015.	34.95	unsat	15.68
cgan	vnncomp2023_benchmarksbenchmarksganGAN_imgSz64_nCh_3.	vnncomp2023_benchmarksbenchmarksganGAN_imgSz64_nCh_3_prop_1_input_eps_0.010_output_eps_0.015.	32.42	unsat	15.29
cgan	vnncomp2023_benchmarksbenchmarksganGAN_imgSz64_nCh_3.	vnncomp2023_benchmarksbenchmarksganGAN_imgSz64_nCh_3_prop_2_input_eps_0.005_output_eps_0.010.	24.68	sat	7.44
cgan	vnncomp2023_benchmarksbenchmarksganGAN_imgSz64_nCh_3.	vnncomp2023_benchmarksbenchmarksganGAN_imgSz64_nCh_3_prop_3_input_eps_0.010_output_eps_0.015.	24.32	sat	7.47
cgan	vnncomp2023_benchmarksbenchmarksganGAN_imgSz32_nCh_3_nonlinear_activations.	vnncomp2023_benchmarksbenchmarksganGAN_imgSz32_nCh_3_nonlinear_activations_prop_0_input_eps_0.010_output_eps_0.015.	30.62	unsat	11.61
cgan	vnncomp2023_benchmarksbenchmarksganGAN_imgSz32_nCh_1_transposedConvPadding_1.	vnncomp2023_benchmarksbenchmarksganGAN_imgSz32_nCh_1_transposedConvPadding_1_prop_0_input_eps_0.010_output_eps_0.015.	26.45	sat	7.45
cgan	vnncomp2023_benchmarksbenchmarksganGAN_imgSz32_nCh_3_upsample.	vnncomp2023_benchmarksbenchmarksganGAN_imgSz32_nCh_3_upsample_prop_0_input_eps_0.010_output_eps_0.015.	29.63	unsat	10.79

Table 2: Rezultatele Benchmark-ului în urma verificării rețelei neuronale din competiție

Am observat că pentru fiecare intrare, în ambele fișiere, s-a obținut același rezultat(sat/unsat). O diferență dintre cele două o constituie coloana ce reprezinta timpul de verificare. Am remarcat că în fișierul obținut de noi, timpul de verificare înregistrat 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.

5.2 NeuralSat

Rezultatele in urma rularii tool-ului NeuralSat le-am obtinut sub forma unor fisiere cu extensie .txt, ce contin rezultatul (sat/unsat), timpul de executie, si valorile variabilelor. Desi fata de alpha_beta_crown folosind NeuralSat a trebuit sa rulam fiecare instanta in parte, intr-un final am obitinut aceleasi rezultate, bineinteles cu timpi de executie diferiti. Rezultate obtinute sunt reprezentate (aici) sau în tabelul de mai jos:

Benchmark	ONNX	Specification (VNNLIB)	Result	Time to Verify (s)
cgan	cGAN_imgSz32_nCh_1.	cGAN_imgSz32_nCh_1_prop_0_input_eps_0.010_output_eps_0.015.	sat	3.175035
cgan	cGAN_imgSz32_nCh_1.	cGAN_imgSz32_nCh_1_prop_1_input_eps_0.020_output_eps_0.025.	sat	3.137382
cgan	cGAN_imgSz32_nCh_1.	cGAN_imgSz32_nCh_1_prop_2_input_eps_0.020_output_eps_0.025.	sat	3.126519
cgan	cGAN_imgSz32_nCh_1.	cGAN_imgSz32_nCh_1_prop_3_input_eps_0.020_output_eps_0.025.	unsat	13.556497
cgan	cGAN_imgSz32_nCh_3.	cGAN_imgSz32_nCh_3_prop_0_input_eps_0.015_output_eps_0.020.	sat	4.432113
cgan	cGAN_imgSz32_nCh_3.	cGAN_imgSz32_nCh_3_prop_1_input_eps_0.010_output_eps_0.015.	unsat	60.970006
cgan	cGAN_imgSz32_nCh_3.	cGAN_imgSz32_nCh_3_prop_2_input_eps_0.010_output_eps_0.015.	sat	3.205494
cgan	cGAN_imgSz32_nCh_3.	cGAN_imgSz32_nCh_3_prop_3_input_eps_0.010_output_eps_0.015.	sat	3.269871
cgan	cGAN_imgSz64_nCh_1.	cGAN_imgSz64_nCh_1_prop_0_input_eps_0.010_output_eps_0.015.	unsat	161.189276
cgan	cGAN_imgSz64_nCh_1.	cGAN_imgSz64_nCh_1_prop_1_input_eps_0.005_output_eps_0.010.	unsat	173.990763
cgan	cGAN_imgSz64_nCh_1.	cGAN_imgSz64_nCh_1_prop_2_input_eps_0.010_output_eps_0.015.	sat	4.552644
cgan	cGAN_imgSz64_nCh_1.	cGAN_imgSz64_nCh_1_prop_3_input_eps_0.005_output_eps_0.010.	sat	3.337892
cgan	cGAN_imgSz64_nCh_3.	cGAN_imgSz64_nCh_3_prop_0_input_eps_0.010_output_eps_0.015.	unsat	159.679505
cgan	cGAN_imgSz64_nCh_3.	cGAN_imgSz64_nCh_3_prop_1_input_eps_0.010_output_eps_0.015.	unsat	175.243838
cgan	cGAN_imgSz64_nCh_3.	cGAN_imgSz64_nCh_3_prop_2_input_eps_0.005_output_eps_0.010.	sat	3.311298
cgan	cGAN_imgSz64_nCh_3.	cGAN_imgSz64_nCh_3_prop_3_input_eps_0.010_output_eps_0.015.	sat	4.474263
cgan	cGAN_imgSz32_nCh_3_nonlinear_activations.	cGAN_imgSz32_nCh_3_nonlinear_activations_prop_0_input_eps_0.010_output_eps_0.015.	unsat	22.691869
cgan	cGAN_imgSz32_nCh_1_transposedConvPadding_1.	cGAN_imgSz32_nCh_1_transposedConvPadding_1_prop_0_input_eps_0.010_output_eps_0.015.	sat	3.235317
cgan	cGAN_imgSz32_nCh_3_upsample.	cGAN_imgSz32_nCh_3_upsample_prop_0_input_eps_0.010_output_eps_0.015.	unsat	24.717013

Table 3: Rezultatele Benchmark-ului în urma verificării rețelei neuronale rulate de noi folosind NeuralSat

Pentru a ne verifica corectitudinea rezultatelor obținute, am facut o analiză comparativă între fișierul obținut la rularea noastră și cel obținut din competiție pe care puteți să îl vedeți pe linkul (aici) sau în tabelul de mai jos:

Benchmark	Neural Network (ONNX)	Specification (VNNLIB)	Result	Time to Verify (s)
cgan	vnncomp2023_benchmarksbenchmarksc gan GAN_imgSz32_nCh_1.	vnncomp2023_benchmarksbenchmarksc gan GAN_imgSz32_nCh_1_prop_0_input_eps_0.010_output_eps_0.015.	sat	4.158000
cgan	vnncomp2023_benchmarksbenchmarksc gan GAN_imgSz32_nCh_1.	vnncomp2023_benchmarksbenchmarksc gan GAN_imgSz32_nCh_1_prop_1_input_eps_0.020_output_eps_0.025.	sat	4.106945
cgan	vnncomp2023_benchmarksbenchmarksc gan GAN_imgSz32_nCh_1.	vnncomp2023_benchmarksbenchmarksc gan GAN_imgSz32_nCh_1_prop_2_input_eps_0.020_output_eps_0.025.	sat	4.100790
cgan	vnncomp2023_benchmarksbenchmarksc gan GAN_imgSz32_nCh_1.	vnncomp2023_benchmarksbenchmarksc gan GAN_imgSz32_nCh_1_prop_3_input_eps_0.020_output_eps_0.025.	unsat	20.074181
cgan	vnncomp2023_benchmarksbenchmarksc gan GAN_imgSz32_nCh_3.	vnncomp2023_benchmarksbenchmarksc gan GAN_imgSz32_nCh_3_prop_0_input_eps_0.015_output_eps_0.020.	sat	4.090721
cgan	vnncomp2023_benchmarksbenchmarksc gan GAN_imgSz32_nCh_3.	vnncomp2023_benchmarksbenchmarksc gan GAN_imgSz32_nCh_3_prop_1_input_eps_0.010_output_eps_0.015.	unsat	813.175559
cgan	vnncomp2023_benchmarksbenchmarksc gan GAN_imgSz32_nCh_3.	vnncomp2023_benchmarksbenchmarksc gan GAN_imgSz32_nCh_3_prop_2_input_eps_0.010_output_eps_0.015.	sat	4.062533
cgan	vnncomp2023_benchmarksbenchmarksc gan GAN_imgSz32_nCh_3.	vnncomp2023_benchmarksbenchmarksc gan GAN_imgSz32_nCh_3_prop_3_input_eps_0.010_output_eps_0.015.	sat	4.093121
cgan	vnncomp2023_benchmarksbenchmarksc gan GAN_imgSz64_nCh_1.	vnncomp2023_benchmarksbenchmarksc gan GAN_imgSz64_nCh_1_prop_0_input_eps_0.010_output_eps_0.015.	unknown	13.634478
cgan	vnncomp2023_benchmarksbenchmarksc gan GAN_imgSz64_nCh_1.	vnncomp2023_benchmarksbenchmarksc gan GAN_imgSz64_nCh_1_prop_1_input_eps_0.005_output_eps_0.010.	unknown	12.826754
cgan	vnncomp2023_benchmarksbenchmarksc gan GAN_imgSz64_nCh_1.	vnncomp2023_benchmarksbenchmarksc gan GAN_imgSz64_nCh_1_prop_2_input_eps_0.010_output_eps_0.015.	sat	4.101576
cgan	vnncomp2023_benchmarksbenchmarksc gan GAN_imgSz64_nCh_1.	vnncomp2023_benchmarksbenchmarksc gan GAN_imgSz64_nCh_1_prop_3_input_eps_0.005_output_eps_0.010.	sat	4.076540
cgan	vnncomp2023_benchmarksbenchmarksc gan GAN_imgSz64_nCh_3.	vnncomp2023_benchmarksbenchmarksc gan GAN_imgSz64_nCh_3_prop_0_input_eps_0.010_output_eps_0.015.	unknown	13.494352
cgan	vnncomp2023_benchmarksbenchmarksc gan GAN_imgSz64_nCh_3.	vnncomp2023_benchmarksbenchmarksc gan GAN_imgSz64_nCh_3_prop_1_input_eps_0.010_output_eps_0.015.	unknown	14.104266
cgan	vnncomp2023_benchmarksbenchmarksc gan GAN_imgSz64_nCh_3.	vnncomp2023_benchmarksbenchmarksc gan GAN_imgSz64_nCh_3_prop_2_input_eps_0.005_output_eps_0.010. 8	sat	4.106288
cgan	vnncomp2023_benchmarksbenchmarksc gan GAN_imgSz64_nCh_3.	vnncomp2023_benchmarksbenchmarksc gan GAN_imgSz64_nCh_3_prop_3_input_eps_0.010_output_eps_0.015.	sat	4.119278
cgan	vnncomp2023_benchmarksbenchmarksc gan GAN_imgSz32_nCh_3_nonlinear_activations.	vnncomp2023_benchmarksbenchmarksc gan GAN_imgSz32_nCh_3_nonlinear_activations_prop_0_input_eps_0.010_output_eps_0.015.	unsat	25.971333
cgan	vnncomp2023_benchmarksbenchmarksc gan GAN_imgSz32_nCh_1_transposedConvPadding_1.	vnncomp2023_benchmarksbenchmarksc gan GAN_imgSz32_nCh_1_transposedConvPadding_1_prop_0_input_eps_0.010_output_eps_0.015.	sat	4.069863
cgan	vnncomp2023_benchmarksbenchmarksc gan GAN_imgSz32_nCh_3_upsample.	vnncomp2023_benchmarksbenchmarksc gan GAN_imgSz32_nCh_3_upsample_prop_0_input_eps_0.010_output_eps_0.015.	error	5.869110

Table 4: Rezultatele Benchmark-ului în urma verificării rețelei neuronale din competiție folosind NeuralSat

Comparand cele doua tabele, am observat ca am obitnute valoare satisfiabila sau nesatisfiabila pentru toate instantele, in timp ce in rezultatele din competitie sau obtinut valori de *unknown* pentru unele instante. Diferentele dintre timpii de executie pot fi vizualizati in graficul de mai jos 3

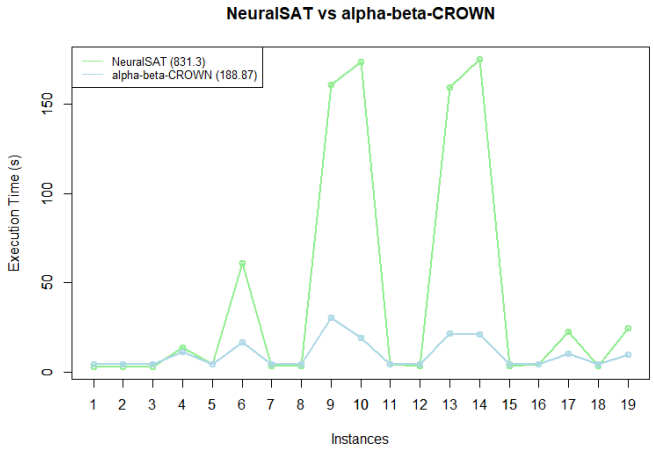


Figure 3: NeuralSAT vs alpha_beta_crown

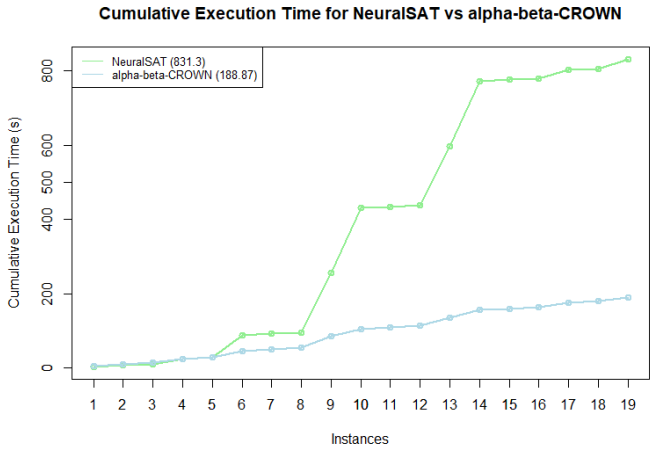


Figure 4: Execuția cumulativa: NeuralSAT vs alpha_beta_crown

5.3 Scorul obținut în cadrul competiției

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.

Table 5: Benchmark 2023-cgan - rezultate competiție VNN-Comp 2023

#	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%

Din tabelul de mai sus putem sa ne dam seama(mai mult sau mai puțin) că formula de calculare a scorului este: Verified×10+Falsified×10-Penalty×150.

Procentajul reprezintă scorul obținut exprimat ca procent din scorul maxim posibil, oferind o imagine de ansamblu asupra performanței, iar formula de calcul este: Score*100/max(Score)

6. Concluzii

Această lucrare a examinat și testat utilizarea Rețelelor Neuronale Generative Adversariale Condiționate (cGAN) în contextul prelucrării de imagini. Focalizându-se pe benchmark-ul cGAN de la Competiția VNN-Comp 2023, lucrarea include o descriere detaliată a benchmark-ului, configurarea și rularea instrumentelor alpha_beta_crown și NeuralSat, urmată de o interpretare succintă a rezultatelor obținute.

S-a analizat funcționarea și performanța rețelelor neuronale prin prisma a două abordări diferite, utilizând alpha_beta_crown și NeuralSat. Aceste instrumente au fost aplicate pentru a verifica formal corectitudinea rețelelor neuronale, cu un accent deosebit pe generarea de conținut condiționat și evaluarea capacitatii rețelelor de a se alinia cu condițiile de intrare specificate.

Rezultatele obținute au fost introduse în tabele pentru a facilita comparația și analiza. Acestea au arătat că ambele instrumente au reușit să valideze instanțele testate, cu unele diferențe în timpul necesar pentru verificare. În plus, compararea rezultatelor noastre cu cele din competiția VNN-Comp 2023 a indicat o aliniere consistentă în ceea ce privește rezultatele satisfăcătoare sau nesatisfăcătoare, cu diferențe notabile în timpii de execuție.

În concluzie, lucrarea a demonstrat eficiența utilizării cGAN în contextul prelucrării imaginilor și a oferit o evaluare valoroasă a instrumentelor de verificare neurală, ceea ce a subliniat potențialul acestora în avansarea tehnologiilor de inteligență bazată pe rețele neuronale.

Bibliography

[1] A. Jabbar, X. Li, and B. Omar, "A survey on generative adversarial networks: Variants, applications, and training," *ACM Comput. Surv.*, vol. 54, no. 8, oct 2021. [Online]. Available: https://doi.org/10.1145/3463475

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

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

[4] "Install miniconda," https://docs.conda.io/projects/miniconda/en/latest/, [Accessed 25-01-2024].

[5] "GitHub - Verified-Intelligence/alpha-beta-CROWN: alpha-beta-CROWN: An Efficient, Scalable and GPU Accelerated Neural Network Verifier (winner of VNN-COMP 2021 and 2022) — github.com," https://github.com/Verified-Intelligence/alpha-beta-CROWN, 2023, [Accessed 19-12-2023].

[6] "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].

[7] .bashrc fix. [Online]. Available: https://discuss.pytorch.org/t/libcudnn-cnn-infer-so-8-library-can-not-found/164661

[8] "Neuralsat installation and usage," https://github.com/dynaroars/neuralsat/blob/develop/INSTALL.md#-usage, [Accessed 25-01-2024].