UNIVERSITATEA DE VEST DIN TIMIŞOARA
FACULTATEA DE MATEMATICĂ ŞI INFORMATICĂ

Verificarea Reţelelor Neuronale

Bordea Liviu Valentin
Ciurdea Roberta Carla
Burcea Adina Mădălina
Todoca Ioana Raluca
Szopuch Denis Emilian

Dr. Mădălina Eraşcu

Abstract

Neural networks are being applied to a growing variety of applications, including safetycritical domains like autonomous vehicles and aircraft, due to their ability to approximate complex functions from limited datasets and adapt by continuing to learn from real-world data after deployment.

This scientific paper it's focused on the alpha-beta-CROWN tool, that uses the ACAS XU benchmark by displaying various results for neural network verification in the context of an open-loop approach. Alpha-beta-CROWN is a neural network verifier based on an efficient linear bound propagation framework and branch and bound. ACAS XU (Automated Collision Avoidance System) is an air-to-air collision avoidance system designed for unmanned aircraft. The first chapter presents the description of the dataset, while the following chapters present the installation of the tool and its operation on the chosen dataset.

Contents

0.1 Descrierea Dataset-ului

ACAS XU (Automated Collision Avoidance System) este un sistem de evitare a coliziunilor aer-la-aer proiectat pentru aeronave fără pilot, care emite sfaturi privind virajele orizontale pentru a evita o aeronavă intruză. Datorită utilizării unei tabele de căutare mari în proiectare, s-a propus o comprimare a politicii cu ajutorul unei rețele neurale.

Analiza acestui sistem a generat o cantitate semnificativă de cercetare în comunitatea metodelor formale privind verificarea rețelelor neurale. Deși au fost dezvoltate multe metode puternice, majoritatea cercetărilor se concentrează pe proprietățile în buclă deschisă ale rețelelor, mai degrabă decât pe aspectul principal al sistemului - evitarea coliziunilor - care necesită analiză în buclă închisă.

Funcționalitatea Rețelei. Sistemul ACAS Xu mapează variabilele de intrare la recomandările de acțiune. Fiecare recomandare este asignată un scor, cu cel mai mic scor corespunzător celei mai bune acțiuni. Starea de intrare este compusă din șapte dimensiuni (prezentate în Figure 1) care reprezintă informațiile determinate din măsurătorile senzorilor:

(i) ρ: Distanța de la propria navă la intrus.

(ii) θ : Unghiul față de intrus în raport cu direcția de îndreptare a propriei nave.

(iii) ψ: Unghiul de îndreptare al intrusului în raport cu direcția de îndreptare a propriei nave.

(iv) vown: Viteza propriei nave.

(v) vint: Viteza intrusului.

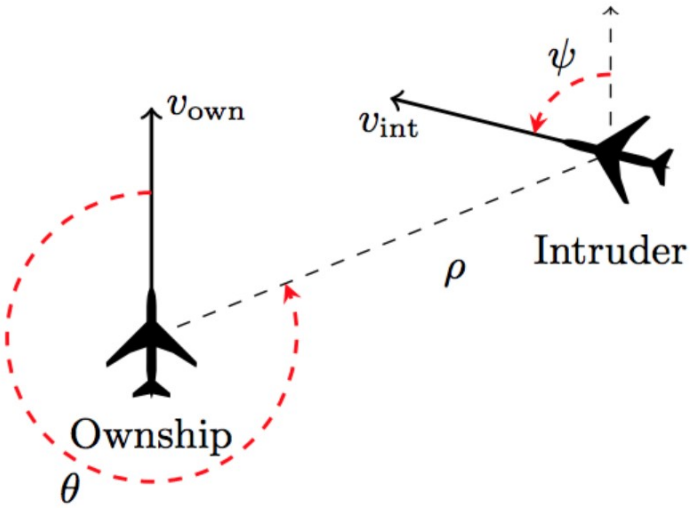

Figure 1: Sistemul ACAS Xu (adaptat din (Katz et al. 2017))

Există cinci ieșiri care reprezintă diferitele recomandări orizontale pe care le poate primi propria navă: Fără Conflict (COC), dreapta slabă, dreapta puternică, stânga slabă sau stânga puternică. Slab și puternic înseamnă rate de îndreptare de 1.5 ° /s și 3.0 ° /s, respectiv.

Într-un sistem critic pentru securitate precum ACAS Xu, un atacator poate exploata cu ușurință o situație limită gestionată incorect pentru a cauza daune semnificative, punând în pericol mii de vieți. Metodele existente pentru testarea rețelelor neurale în fața situațiilor limită se concentrează pe identificarea exemplelor adversare,

dar nu oferă garanții formale cu privire la inexistența intrărilor adversare chiar și în cadrul unor intervale de intrare foarte mici.

Sistemul de evitare a coliziunilor cu aeronave fără pilot X (ACAS Xu) utilizează rețele neurale pentru a prezice cele mai bune acțiuni în funcție de locația și viteza avioanelor atacatoare/intruziune din apropiere. A fost testat cu succes de NASA și FAA și este în program pentru a fi instalat la peste 30.000 de aeronave de pasageri și cargo la nivel mondial, precum și în flotele Marinei SUA.

Utilitate. Dosarul unde se poate gasi benchmark-ul ACAS Xu conține fișiere .onnx și .vnnlib utilizate pentru această categorie, si poate fi descarcat de aici.

Fișierul `acasxu_instances.csv` conține lista completă a instanțelor de benchmark, câte una pe linie: `onnx_file`, `vnn_lib_file`, `timeout_secs`. Fișierele .vnnlib și .csv au fost create cu ajutorul scriptului generate.py inclus.

Acest benchmark este același ca în 2021/2022 și este utilizat pentru a compara îmbunătățirile de la un an la altul.

0.2 Instalarea tool-ului

0.2.1 Alpha Beta-CROWN

Alpha-beta-CROWN este un verificator de rețele neurale bazat pe un cadru eficient de propagare a limitelor liniare și pe tehnica de branch and bound. Poate fi accelerat eficient pe unități de procesare grafică (GPU) și poate scala către rețele convoluționale relativ mari.

Pentru a instala Alpha Beta-CROWN, am folosit miniconda3 si sistemul de operare Linux Kali.

Dupa instalarea miniconda si restartarea terminalului, a fost nevoie de clonarea Alpha Beta-CROWN (Figure 2) si Auto Lirpa (Figure 3) de pe github.

```
(base) ┌──(kali㉿kali)-[~]
└─$ git clone --recursive https://github.com/Verified-Intelligence/alpha-beta-CROWN.git
Cloning into 'alpha-beta-CROWN'...
remote: Enumerating objects: 860, done.
remote: Counting objects: 100% (127/127), done.
remote: Compressing objects: 100% (60/60), done.
remote: Total 860 (delta 84), reused 67 (delta 67), pack-reused 733
Receiving objects: 100% (860/860), 72.09 MiB | 14.49 MiB/s, done.
Resolving deltas: 100% (393/393), done.
Submodule 'auto_LiRPA' (git@github.com:Verified-Intelligence/auto_LiRPA.git) registered for path 'auto_LiRPA'
Cloning into '/home/kali/alpha-beta-CROWN/auto_LiRPA'...
git@github.com: Permission denied (publickey).
fatal: Could not read from remote repository.
```

Figure 2: Alpha-beta CROWN

```
(base) ┌──(kali㉿kali)-[~]
└─$ git clone https://github.com/Verified-Intelligence/auto_LiRPA
Cloning into 'auto_LiRPA'...
remote: Enumerating objects: 811, done.
remote: Counting objects: 100% (440/440), done.
remote: Compressing objects: 100% (236/236), done.
^[[B^[[B^[[Bremote: Total 811 (delta 247), reused 355 (delta 197), pack-reused 371
Receiving objects: 100% (811/811), 31.82 MiB | 17.89 MiB/s, done.
Resolving deltas: 100% (406/406), done.
```

Figure 3: Auto Lirpa

Am creat environmentul folosind instructiunea:

"conda env create -f complete_verifier/environment.yaml –name alpha beta-crown" (Figure 4), si am activat alpha-beta-crown (Figure 5)

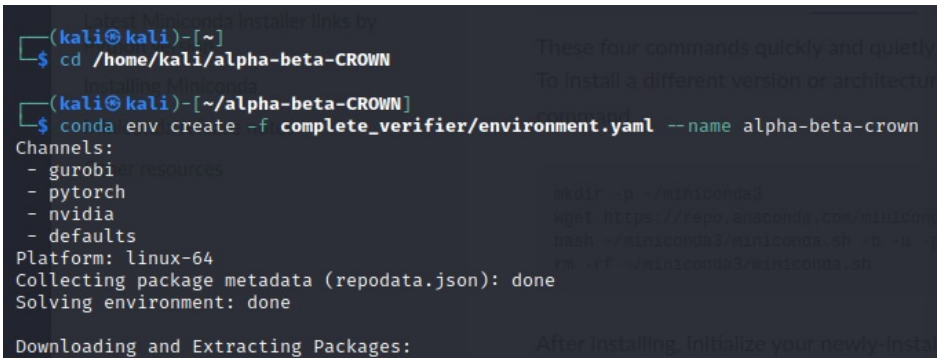

Figure 4: Crearea environmentului

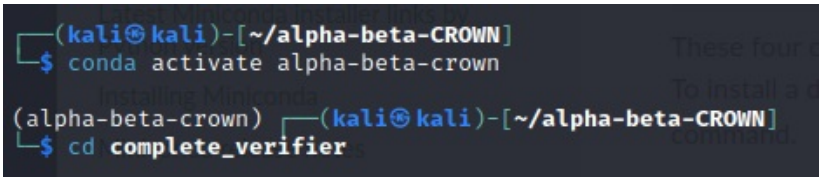

Figure 5: Activarea tool-ului

Dupa activarea "alpha-beta-CROWN", a fost nevoie de instalarea unor pachete pentru a putea rula benchmark-ul. Aceste pachete sunt: onnx2pytorch, onnxruntime, onnxoptimizer, skl2onnx, torch, torchvision.

0.3 Aplicarea tool-ului pentru dataset-ul ales

Pentru a rula AcasXu, am descarcat arhiva care contine toate benchmark-urile din 2023. Aceasta se poate descarca de la adresa "https://github.com/ChristopherBrix/vnncomp2023_benchmarks/tree/main". Dupa descarcare, am dezarhivat folderul "acasxu" si componentele acestuia. Am creat un nou folder cu denumirea "vnncomp2023_benchmarks" in care am creat folderul "benchmarks", unde am adaugat folderul "acasxu" dezarhivat si descarcat de pe github.

Din cauza unor erori, a fost nevoie de adaugarea unei instructiuni in fisierul de configurare al benchmark-ului. Instructiunea este "device: cpu" pentru ca pytorch sa utilizeze CPU in loc de GPU.

Pentru a aplica benchmark-ul si a primi rezultatele am folosit urmatoarea instructiune pe care am rulat-o in folderul "complete_verifier" aflat in "alpha-beta-CROWN": "python abcrown.py –config exp_configs/vnncomp23/acasxu.yaml".

Rezultatele pot fi gasite in fisierul "rezultat_acasxu.txt" de pe github "https://github.com/IoanaTodoca/VerificareFormalaProiect".

Bibliography

[1] Mike Marston and Gabe Baca. "ACAS-Xu initial self-separation flight tests". In: NASA Technical Reports Server (2015), "https://ntrs.nasa.gov/api/citations/20150008347/downloads/20150008347.pdf"

[2] An Introduction to ACAS Xu and the Challenges Ahead, https://enac.hal.science/hal-01638049/file/acasxu.pdf

[3] Beta-CROWN: Efficient Bound Propagation with Per-neuron Split Constraints for Neural Network Robustness Verification, "https://arxiv.org/pdf/2103.06624.pdf"

[4] Reluplex: An Efficient SMT Solver for Verifying Deep Neural Networks https://www.researchgate.net/figure/Geometry-for-ACAS-Xu-Horizontal-Logic-Table_fig2_313394663

[5] ART: Abstraction Refinement-Guided Training for Provably Correct Neural Networks "https://www.researchgate.net/publication/334695061_ART_Abstraction_Refinement-Guided_Training_for_Provably_Correct_Neural_Networks#pf2"

[6] Evaluation of Neural Network Verification Methods for Air-to-Air Collision Avoidance https://arc.aiaa.org/doi/10.2514/1.D0255