UNIVERSITATEA DE VEST DIN TIMIȘOARA
FACULTATEA DE MATEMATICĂ ȘI INFORMATICĂ
PROGRAMUL DE STUDII DE MASTER: Securitate Cibernetică

Verificare Formală

Concursul de verificare a rețelelor neuronale VNN 2023

Versiunea Draft

Github Repository

Alin-Daniel Popescu, Adriana-Andreea Copil, Mircea Zeiconi
Simona Tuns, Marian-Milovan Arsul

Cuprins

Introducere

Verificarea rețelelor neuronale (VNN - Verification of Neural Networks) este un domeniu de cercetare, important și actual, pentru comunitatea științifică. Acest lucru se datorează faptului că rețelele neuronale sunt din ce în ce mai utilizate în sistemele autonome, recunoașterea imaginilor și procesarea limbajului natural. Verificarea rețelelor neuronale poate contribui la creșterea încrederii, transparenței și siguranței rețelelor și a aplicațiilor lor. Acest domeniu implică utilizarea unor metode formale, cum ar fi logica, analiza statică, testarea și optimizarea, pentru a demonstra corectitudinea, robustețea și eficiența rețelelor neurale.

Odată cu crearea rețelelor neuronale, au apărut și provocările testării acestora. Motivele principale sunt datorate complexității, lipsei de interpretare, a diversității modelelor și a specificațiilor acestora. Tot datorită acestui fapt au apărut și multe instrumente și cadre de verificare, ce folosesc diverse tehnici și algoritmi, cum ar fi Alpha-Beta-Crown, Marabou, neuralSAT și multe altele. Există și o competiție internațională, numită VNN-COMP [1], care are loc anual și care adună cercetători interesați de metode și instrumente ce oferă testări minuțioase ale comportamentelor rețelelor neuronale.

Arhitectura VGGNet-16

VGGNet-16, numit și VGG16, este un model de rețea neuronală convoluțională (CNN), folosit în recunoașterea imaginilor și clasificarea acestora. Această arhitectură este cel mai des folosită alături de ImageNet, un set de date ce conține peste 14 milioane de imagini de antrenare, alcătuit din 1000 de clase de obiecte. [2]

Arhitectura VGGNet-16 integrează cele mai importante caracteristici ale rețelelor neuronale convoluționale. Aceasta se remarcă prin 16 straturi adâncime, 13 fiind straturi de convoluție și 3 fiind complet conectate. [3] Pe lângă straturile propriu-zise, se efectuează și operații de reducere a informațiilor suplimentare, pentru a maximiza calitatea rezultatului. Ultimul start din arhitectură, cel de softmax, îndeplinește rolul de a clasifica imaginea conform celor 1000 clase. Schema arhitecturii, reprezentată vizual, poate fi observată în Figura 2.1.

Arhitectura VGGNet-16 este compusă după cum urmează [4]:

- **Input** VGGNet primește ca intrare, o imagine de 224x224 pixeli în format RGB
- **Straturi de convoluție** acestea folosesc filtre de 3x3 pixeli pentru a reține caracteristicile importante din imagini. Filtrele de 3x3 duc la o arhitectură mai simplă și mai ușor de utilizat dar, în același timp, foarte eficientă
- **ReLU** Funcția de activare liniară rectificată (ReLU), este o funcție liniară ce ajută la prevenirea creșterii exponențiale a calculului necesar pentru operarea rețelei neuronale
- **Hidden layers** Toate straturile ascunse ale rețelei VGG16 folosesc ReLU ca și funcție de activare, scăzând timpul de antrenare și consumul de memorie

- **Pooling layers** un strat de pooling urmează după mai multe straturi de convoluție. Aceste straturi ajută la reducerea dimensiunii și a numărului de parametri, creați în cadrul fiecărui pas de convoluție
- **Fully connected layers** VGGNet include trei straturi complet conectate. Primele doua straturi au fiecare 4096 de canale, iar al treilea strat are 1000 de canale, unul pentru fiecare clasă.

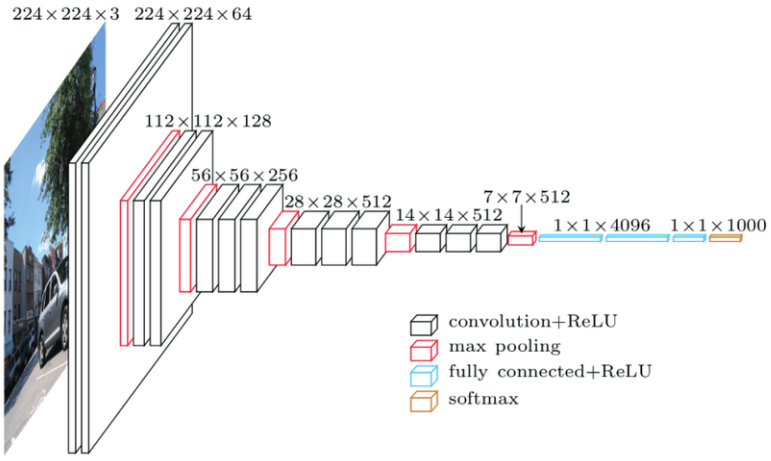

Figura 2.1: Arhitectura rețelei VGGNet-16 [3]

Instrumente de verificare

3.1 Instalare

Acest capitol prezintă procesul de instalare al tool-ului utilizat: **alpha-beta-crown**. Sunt prezentate librăriile și instrucțiunile necesare pentru a rula tool-ul cu succes.

Primul pas constă în dezinstalarea librăriilor deja existente în mediul de programare: `PyTorch`, `TorchVision`, `TorchAudio` și `TorchText`, folosind următoarea comandă:

```
!pip uninstall --yes torch torchvision torchaudio torchtext
```

În al doilea pas, instalăm `PyTorch 1.12.1`, cu ajutorul `CUDA 11.3`.
Flag-ul `-extra-index-url` specifică un URL adițional pentru `PyTorch Wheel`:

```
!pip install torch==1.12.1+cu113 torchvision==0.13.1+cu113 --extra-
   index-url https://download.pytorch.org/whl/cu113
```

Instalarea `PyTorch` este urmată de instalarea unor dependințe adiționale, precum `onnxruntime, onnxsim, skl2onnx, termcolor, onnxoptimizer, gurobipy psutil, pyyaml` și `sortedcontainers`, folosind următoarele comenzi:

```
!pip install onnxruntime==1.16.3
!pip install onnxsim==0.4.35
!pip install skl2onnx==1.15.0
!pip install termcolor==2.3.0
!pip install onnxoptimizer==0.3.13
!pip install gurobipy==11.0.0
!pip install psutil==5.9.5
!pip install pyyaml==6.0.1
!pip install sortedcontainers==2.4.0
```

Instalăm `onnx2pytorch`:

```
!pip install git+https://github.com/KaidiXu/onnx2pytorch.git
```

3.2 Rulare

Google Colab este mediul de dezvoltare ales datorită limitărilor impuse de hardware-ul personal. Prin utilizarea capacităților de calcul superioare ale Google Colab, ne asigurăm că tool-ul ales rulează în condiții optime și returnează rezultate calitative.

Rularea și output-ul tool-ului pot fi consultate la următorul link: **Google Drive**.

În continuare, este prezentat procesul de rulare al tool-ului utilizat.

Deoarece mediul de dezvoltare este Google Colab, următoarea comandă este necesară pentru montarea Google Drive:

```
from google.colab
import drive drive.mount("/content/drive/", force_remount=True)
```

Următoarea comandă schimbă directorul de lucru curent în calea specificată, care este un director din Google Drive:

```
%cd '/content/drive/MyDrive/VF/alpha-beta-CROWN-main/
    complete_verifier'
```

Următoarele rânduri execută scriptul `abcrown.py` de mai multe ori, de fiecare dată cu un fișier de configurare diferit, specificat folosind parametrul `--config`:

```
1 !python abcrown.py --config exp_configs/vnncomp23/
    vggnet16_spec0_pretzel.yaml
2 !python abcrown.py --config exp_configs/vnncomp23/
    vggnet16_spec1_football_helmet.yaml
3 !python abcrown.py --config exp_configs/vnncomp23/
    vggnet16_spec2_grey_whale.yaml
4 !python abcrown.py --config exp_configs/vnncomp23/
    vggnet16_spec3_African_crocodile.yaml
5 !python abcrown.py --config exp_configs/vnncomp23/
    vggnet16_spec4_Norfolk_terrier.yaml
6 !python abcrown.py --config exp_configs/vnncomp23/
    vggnet16_spec5_shopping_cart.yaml
7 !python abcrown.py --config exp_configs/vnncomp23/
    vggnet16_spec6_groom.yaml
8 !python abcrown.py --config exp_configs/vnncomp23/
    vggnet16_spec7_howler_monkey.yaml
9 !python abcrown.py --config exp_configs/vnncomp23/
    vggnet16_spec8_sunscreen.yaml
```

```
!python abcrown.py --config exp_configs/vnncomp23/
   vggnet16_spec9_Border_terrier.yaml
!python abcrown.py --config exp_configs/vnncomp23/
   vggnet16_spec10_beacon.yaml
!python abcrown.py --config exp_configs/vnncomp23/
   vggnet16_spec11_EntleBucher.yaml
!python abcrown.py --config exp_configs/vnncomp23/
   vggnet16_spec12_Dutch_oven.yaml
!python abcrown.py --config exp_configs/vnncomp23/
   vggnet16_spec13_steel_drum.yaml
!python abcrown.py --config exp_configs/vnncomp23/
   vggnet16_spec14_Afghan_hound.yaml
!python abcrown.py --config exp_configs/vnncomp23/
   vggnet16_spec15_carpenters_kit.yaml
!python abcrown.py --config exp_configs/vnncomp23/
   vggnet16_spec16_restaurant.yaml
!python abcrown.py --config exp_configs/vnncomp23/
   vggnet16_spec17_golf_ball.yaml
```

Rezultate

<table>
<tr><th></th><th>Class</th><th>Avg. perturbation</th><th>Perturbed dimension</th><th>Worst class (+rhs)</th><th>Result</th><th>Time (sec)</th></tr>
<tr><td>0</td><td>Pretzel</td><td>2.00E-05</td><td>1</td><td>6.6468</td><td>safe/unsat</td><td>34.0446</td></tr>
<tr><td>1</td><td>Football Helmet</td><td>0.0002</td><td>1</td><td>8.1515</td><td>safe/unsat</td><td>21.726</td></tr>
<tr><td>2</td><td>Grey Whale</td><td>0.002</td><td>1</td><td>7.9019</td><td>safe/unsat</td><td>24.5389</td></tr>
<tr><td>3</td><td>African Crocodile</td><td>2.00E-05</td><td>5</td><td>11.5987</td><td>safe/unsat</td><td>26.9463</td></tr>
<tr><td>4</td><td>Norfolk Terrier</td><td>0.0002</td><td>5</td><td>2.6645</td><td>safe/unsat</td><td>33.2313</td></tr>
<tr><td>5</td><td>Shopping Cart</td><td>0.002</td><td>5</td><td>5.1363</td><td>safe/unsat</td><td>46.5878</td></tr>
<tr><td>6</td><td>Groom</td><td>2.00E-05</td><td>10</td><td>6.4349</td><td>safe/unsat</td><td>37.7166</td></tr>
<tr><td>7</td><td>Howler Monkey</td><td>0.0002</td><td>10</td><td>2.1527</td><td>safe/unsat</td><td>37.7551</td></tr>
<tr><td>8</td><td>Sunscreen</td><td>0.002</td><td>10</td><td>0.5677</td><td>safe/unsat</td><td>493.7566</td></tr>
<tr><td>9</td><td>Border Terrier</td><td>2.00E-05</td><td>20</td><td>8.4281</td><td>safe/unsat</td><td>43.4245</td></tr>
<tr><td>10</td><td>Beacon</td><td>0.0002</td><td>20</td><td>12.6673</td><td>safe/unsat</td><td>47.6443</td></tr>
<tr><td>11</td><td>Entlebucher</td><td>0.0019</td><td>20</td><td>1.5755</td><td>safe/unsat</td><td>58.1348</td></tr>
<tr><td>12</td><td>Dutch Oven</td><td>2.00E-05</td><td>100</td><td colspan="3" rowspan="3">No results</td></tr>
<tr><td>13</td><td>Steel Drum</td><td>0.0001</td><td>100</td></tr>
<tr><td>14</td><td>Afghan Hound</td><td>0.002</td><td>100</td></tr>
<tr><td>15</td><td>Carpenter's Kit</td><td>N/A</td><td>N/A</td><td>N/A</td><td>unsafe/sat</td><td>12.9192</td></tr>
<tr><td>16</td><td>Restaurant</td><td colspan="5" rowspan="2">Timeout</td></tr>
<tr><td>17</td><td>Golf Ball</td></tr>
</table>

Tabela 4.1: Rezultate VGGNet-16 - CPU

Tabela 4.1 prezintă rezultatele rețelei neuronale VGGNet-16, rulat cu ajutorul resurselor CPU. Clasificările reușite, etichetate ca "safe/unsat", domină rezultatele, indicând eficacitatea modelului. Cu toate acestea, apar provocări, în special în cazurile cu dimensiuni mai mari ale perturbației, care conduc la constrângeri computaționale, cum ar fi depășiri ale timpului de execuție și cazuri unde nu s-a reușit obținerea unor rezultate.

	Class	Avg. perturbation	Perturbed dimension	Worst class (+rhs)	Result	Time (sec)
0	Pretzel	2.00E-05	1	6.6467	safe/unsat	33.2506
1	Football Helmet	0.0002	1	8.1514	safe/unsat	21.0254
2	Grey Whale	0.002	1	7.9019	safe/unsat	21.9771
3	African Crocodile	2.00E-05	5	11.5987	safe/unsat	21.4638
4	Norfolk Terrier	0.0002	5	2.6645	safe/unsat	23.1208
5	Shopping Cart	0.002	5	5.1362	safe/unsat	21.3505
6	Groom	2.00E-05	10	6.4349	safe/unsat	21.3549
7	Howler Monkey	0.0002	10	2.1527	safe/unsat	21.3719
8	Sunscreen	0.002	10	0.5677	safe/unsat	26.5134
9	Border Terrier	2.00E-05	20	8.4281	safe/unsat	21.1552
10	Beacon	0.0002	20	12.6673	safe/unsat	22.2805
11	Entlebucher	0.0019	20	1.5755	safe/unsat	21.6416
12	Dutch Oven	CUDA out of memory				
13	Steel Drum					
14	Afghan Hound	0.002	100	7.9489	safe/unsat	110.3424
15	Carpenter's Kit	N/A	N/A	N/A	unsafe/sat	15.9534
16	Restaurant	No results				
17	Golf Ball					

Tabela 4.2: Rezultate VGGNet-16 - GPU

Tabela 4.2 prezintă rezultatele rețelei neuronale VGGNet-16, rulat cu ajutorul resurselor GPU. Clasificările reușite, etichetate ca "safe/unsat", domină rezultatele, indicând eficacitatea modelului. Însă, apar provocări în ceea ce privește clasificările "unsafe/sat" pentru o anumită categorie de imagini ("Carpenter's Kit") și cazurile în care resursele computaționale au fost insuficiente ("CUDA out of memory").

Bibliografie

[1] 4th International Verification of Neural Networks Competition (VNN-COMP'23). URL https://sites.google.com/view/vnn2023.

[2] Muneeb ul Hassan. VGG16 - convolutional network for classification and detection, May 2023. URL https://neurohive.io/en/popular-networks/vgg16/.

[3] Manolis Loukadakis, José Cano, and Michael O'Boyle. Accelerating Deep Neural Networks on Low Power Heterogeneous Architectures. 01 2018.

[4] VGG16. URL https://datagen.tech/guides/computer-vision/vgg16/.