## Proiect Verificare Formala - DRAFT

Cristian Ciocoi Vladimir Britchi Costisanu Vlad Tarta Roberto Iacobescu Alexandru

December 2023



#### Introduction

For our **project**, the **benchmark** we have chosen is **Traffic Sign Recognition**. The data-set it contains represents a set of multiple images (transformed to VNN-LIB standard which uses the SMT-LIB format) of **traffic signs** taken under **various scenarios** such as different angles, light conditions and damaged structure of the sign (paint marks, stickers etc.). The **goal** we have to achieve is **testing** the **efficiency** of 2 select DNN verifiers to **recognize them correctly** by comparing the end results of the execution.

The tools we have chosen are:

- NeuralSat
- Alpha-Beta-Crown.

## Installation of NeuralSat

NeuralSat has a **step by step installation guide** in its **Github**. We had **no issues** with installation, except a few libraries to update (pytorch, cuda-toolkit). You can find it in the figure below:

30-30000002/ant/clience/obschribektop/lentificare formala/Project/neuralisat-develop/neuralisat-develop\$ conds now create =f new yawl. Gelletting scalesy studies (repudsta_ison): done Selving environment: done			
=== WARMING: A newer version of conda exists. <== current westion: 23.7.4 latest version: 23.11.0			
Please update conda by running			
\$ conda update -n base -c defaults conda			
Or to minimize the number of packages updated during conda update use			
conda install conda=23.11.8			
Domeleading and Extracting Packages			
	323 NB		
python-3.18.9	26 8 MB	10% brotligg-0.7.0	
		1887 VS	
		THE PROPERTY OF THE PROPERTY O	
enterent and a second a			
######################################			
librofft-18 7 2 124	93.6 MB	10141111111111111111111111111111111111	
cuda-nyete-11.7.99	17.3 MB	398	
cuda-cupti-11.7.101	22.9 #8	######################################	
####   188% openh264-2		731 HB   ##################################	
ests=2.28.1   93 MB			
177 88			
######################################			
torchaudio-0.13.1	6.5 MB	11000	
pytorch-mutex-1.8	3 MB	31	
libwebp-base-1.2.4	347 KB	411141111111111111111111111111111111111	
libcufile=1.6.1.9	764 KB	313131111111111111111111111111111111111	
pysocks-1.7.1	28 NB	41	
	2.8 MB	313141111111111111111111111111111111111	
tqdn-4.64.1	126 KB	ALITERENTE DE LE CONTROL DE LA	

## **Execution of NeuralSat**

Executing NeuralSat was **quite simple**, since it accepts onnx and vnnlib files as input. The issue was that it runs the test for a single onnx-vnnlib value pair which inclined us to create a **python script** that executes **all pair instances** listed in the instances.csv file. Below you have an example of execution of a single pair.

```
> python3 main.py --net traffic_signs_recognition/onnx/3_30_38_0Conv_16_3_0Conv_32_2_Dense_43_ep_38.onnx --spec traffic_signs_recognition/vnmlib/model_38_i
dx_7840_eps_3.00000.vmlib
Restricted license - for non-production use only - expires 2024-10-28
Automatic inference of operator: sign
Automatic inference of operator: sign
  utomatic inference of operator: sign
  utomatic inference of operator: sign
  utematic inference of operator: sign
    omatic inference of operator: sign
  utomatic inference of operator: sign
  stomatic inference of operator: sign
      18:69:41 [!] VNWLIB: 2788 inputs, 43 output
                                                                                                                                                    | 42/42 [00:00<00:00. 126.52it/s]
  (Transpose_sequential_10/quant_conv2d_20/quantConv2D_105:0): Transpose()
(Conv_sequential_10/quant_conv2d_20/quantConv2D:0): Conv2d(3, 16, kernel_size=(3, 3), stride=(1, 1), bias=False)
  (Transpose_sequential_10/quant_conv2d_20/QuantConv20_107:0): Transpose()
  (Add_sequential_10/quant_conv2d_21/ste_sign_52/add:0): Add()
  (Transpose_sequential_10/quant_cosv2d_21/quantCosv2b_2109:0): Transpose()
(Coss_sequential_10/quant_cosv2d_21/quantCosv2b:0): Cosv2d(6,9,2,4 kernel_size=(2, 2), stride=(1, 1), bias=False)
(Transpose_sequential_10/quant_cosv2d_21/quantCosv2b=10/10:0): Transpose()
  (Reshape_sequential_10/flatten_10/Reshape:0): Flatten()
  (Add_sequential_10/quant_dense_10/ste_sign_54/add:0): Add()
(MatMul_sequential_10/quant_dense_10/MatMul.0): Linear(in features=23328, out_features=83, bias=False)
  (Softmax_activation_10): Identity()
 JNFO 18:69:41 [1] Input shape: (1, 3, 38, 30)
 INFO 18:09:41 [1] Output shape: (1, 43)
 [!] Current settings:
         - max hidden branches
          - use restart
         18:09:41 [Failed] RandomAttack(seed=837, device=cuda)
         18:09:43 [Success] PGDAttack(seed=568. device=cuda)
        18:49:48 [1] Eteration: 0
18:49:48 adv (first 5): tersor((294.8914, 254.5998, 254.5661, 251.6367, 252.35461)
18:49:48 output: tensor((594., 928., 1138., 892., 1161., 624., 485., 918., 896., 586.,
          732., 1414., 718., 466., 1176., 1174., 68., 1552., 936., 898.
          1246., 646., 1408., 838., 558., 1170., 1342., 530., 1210., 752
         1858., 776., 538., 176., -298., -88., 524., -274., 258., 184.
248., 588., 338.])
```

# **Execution of Script**

Below you can see our script in action. It executes all pairs of .onnx and .vnnlib files and writes the results in a .txt file.

```
42/42 [80:86480:60
                                                                                                                                                                                                 42/42 [00:00
                                                                                                                                                                                                Smeout 400.0
                                                                                                                                                                                                Ineast 488.6
                                                                                                                                                                                                 42/42 [00:02:0
                                                                                                                                                                                                 imeaut 480.0
                                                                                                                                                                                                 42/42 [66:61<66:66
                                                                                                                                                                                                timeout 400.0
                                                                                                                                                                                                 42/42 [88:88<88:68
                                                                                                                                                                                                 | 42/42 | 60:60-60:60, 119.00it/s
                                                                           42/82 [88:82:88:80, 00.02it/s]# Executing: python3 main.py --net 'traffic_sign/sens/1_00_00,00,00,12_5_89_2_80_Cons_64_5_89_2_80_Cons_64_5_89_2_80_Cons_64_5_89_2_80_Cons_64_5_89_2_80_Cons_64_5_89_2_80_Cons_64_5_89_2_80_Cons_64_5_89_2
                                                                           model 48 ids TMER cos 5.00000 vonlin' --timenut 488.6
ent/c/esl/seuralsat-develop/neuralsat/onnezpyterch/convert/layer.py:95: UserWarning: The given NamPy array is not writable, and PyTorch does not support non-writable tensors. This means writing to this tensor
will result in undefined behavior. You may want to copy the array to protect its data or make it writable before converting it to a tensor. This type of warning will be suppressed for the rest of this program
```

## Alpha-Beta-Crown

We were **able** to install Alpha-Beta-Crown but unfortunately, executing it was **filled with** issues and trying to solve them led to various other compatibility problems. They were **mostly related** with **cuda-toolkit**, **pytorch versions** and some other libraries we had difficulties setting up on the **conda environment** itself.

```
raceback (most recent call last)
File "/home/alex/Documents/repos/alpha-beta-CROWN/complete_verifier/abcrown.py", line 612, in <module>
and and a stanty of the stanty
      attack_ret, attack_images, attack_margins, all_adv_candidates = attack_function(
 File "/home/alex/Documents/repos/alpha-beta-CROWN/complete_verifier/custom/custom_attacker.pv", line 27, in use_LiRPANet
     res, attack_image, attack_margin, all_adv_candidates = attack_with_general_specs(wrapped_model.net, x, data_min, data_max, ile "/home/alex/Documents/repos/alpha-bota-CROWN/complete_verifier/attack/attack_pgd.py", line 1259, in attack_with_general_specs
 best_deltas__ last_deltas, best_loss__ early_stopped = ppd_attack_with_general_specs(
File "/home/alex/Documents/repos/alpha-beta-CROWN/complete_verifier/attack/attack_ppd.py", line 677, in ppd_attack_with_general_specs
      output = model(inputs.view(-1, *input_shape[1:])).view(
             "/home/alex/Documents/repos/alpha-beta-CROWN/complete verifier/auto LiRPA/bound general.pv", line 263, in _call_
     return self.forward(*input. **kwargs)
 File "/home/alex/Documents/repos/alpha-beta-CROWN/complete_verifier/auto_LiRPA/bound_general.py", line 421, in forward
      deque([self.get_forward_value(self[n])
             "/home/alex/Documents/repos/alpha-beta-CROWN/complete_verifier/auto_LiRPA/bound_general.py", line 421, in 421,
            "/home/alex/Documents/repos/alpha-beta-CRONN/complete verifier/auto LiRPA/bound general.pv", line 372, in get forward value
 inputs = [self.get.forward_value(inp) for inp in node.inputs]

File "/home/alex/Documents/repos/alpha-beta-CROWN/complete_verifier/auto_LIRPA/bound_general.py", line 372, in <listcomp>
       inputs = [self.get_forward_value(inp) for inp in node.inputs]
 File "/home/alex/Documents/repos/alpha-beta-CROMM/complete_verifier/auto_LiRPA/bound_general.py", line 372, in get_forward_value
          puts = [self.get_forward_value(inp) for inp in node.inputs]
             "/home/alex/Documents/repos/alpha-beta-CROWN/complete_verifier/auto_LiRPA/bound_general.py", line 372, in listcomp>
       inputs = [self.get_forward_value(inp) for inp in node.inputs]
 File "/home/alex/Documents/repos/alpha-beta-CROMM/complete_verifier/auto_LiRPA/bound_general.py", line 372, in get_forward_value
       inputs = [self.get_forward_value(inp) for inp in node.inputs]
 File "/home/alex/Documents/repos/alpha-beta-CROMN/complete_verifier/auto_LiRPA/bound_general.py", line 372, in istcomp>
 inputs = [self.get_forward_value(inp) for inp in node.inputs]

File "/hone/alex/Documents/repos/alpha-beta-CROWN/complete_verifier/auto_LiRPA/bound_general.py", line 372, in get_forward_value
 File "/home/alex/Documents/repos/alpha-beta-CROWN/complete verifier/auto LiRPA/bound general.pv". line 372. in in intromp>
          nouts = [self.get_forward_value(inp) for inp in node.inputs]
 File "/home/alex/Documents/repos/alpha-beta-CROWN/complete_verifier/auto_LiRPA/bound_general_py". line 372, in get_forward_value
       inputs = [self.get_forward_value(inp) for inp in node.inputs]
 inputs = [set-get_rormanc_value(inp) +or inp in node.inputs]
File "/home/alex/Bocuments/repos/alpha-beta-CROMM/complete_verifier/auto_LiRPA/bound_general.py", line 372, in stcomp>inputs = [setf.get_forward_value(inp) for inp in node.inputs]
 File "/home/alex/Documents/repos/alpha-beta-CROWN/complete_verifier/auto_LiRPA/bound_general.py", line 376, in get_forward_value
 File "/home/alex/Documents/repos/alpha-beta-CROWN/complete verifier/auto LiRPA/operators/shape.py", line 18, in forward
   ntimeError: CUDA error: the launch timed out and was terminated
CUDA kernel errors might be asynchronously reported at some other API call, so the stacktrace below might be incorrect
or debugging consider passing CUDA_LAUNCH_BLOCKING=1.
```