Proiect Verificare Formala - DRAFT

Cristian Ciocoi Vladimir Britchi Costisanu Vlad Tarta Roberto Iacobescu Alexandru

December 2023

Introduction

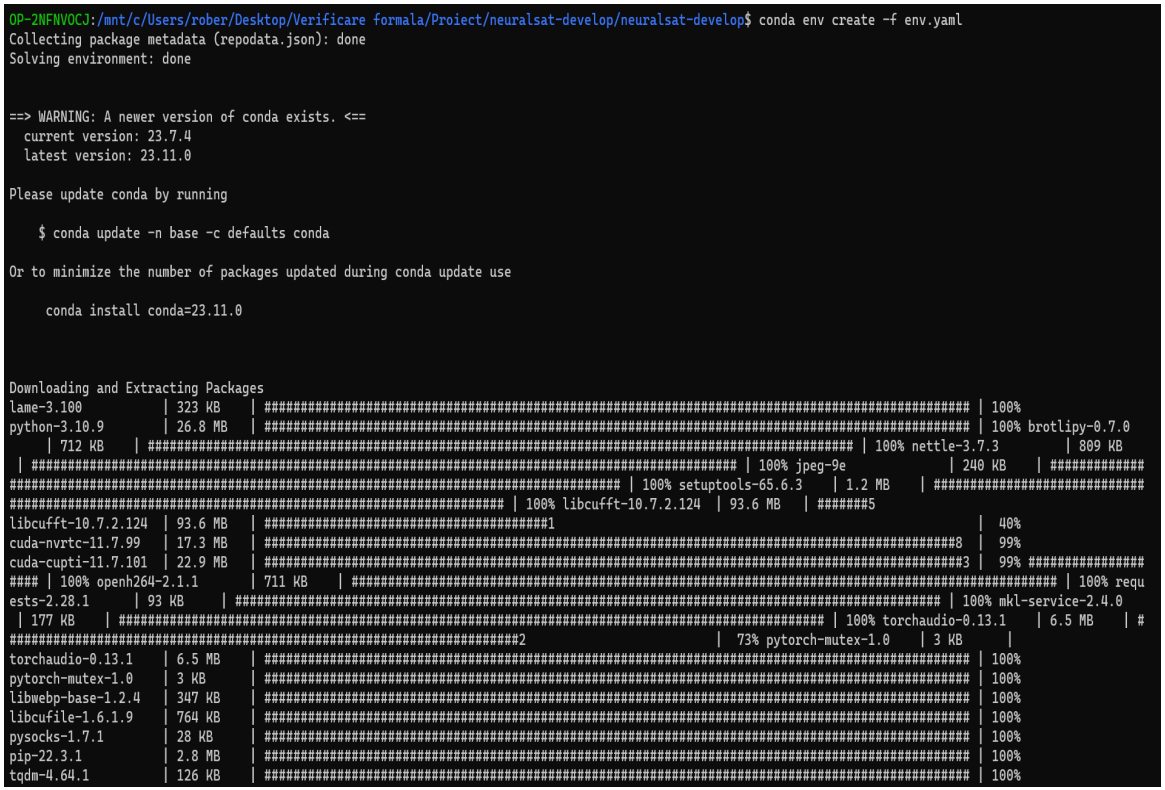
For our **project**, the **benchmark** we have chosen is **Traffic Sign Recognition**. The data-set it contains represents a set of multiple images (transformed to VNN-LIB standard which uses the SMT-LIB format) of **traffic signs** taken under **various scenarios** such as different angles, light conditions 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:



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.

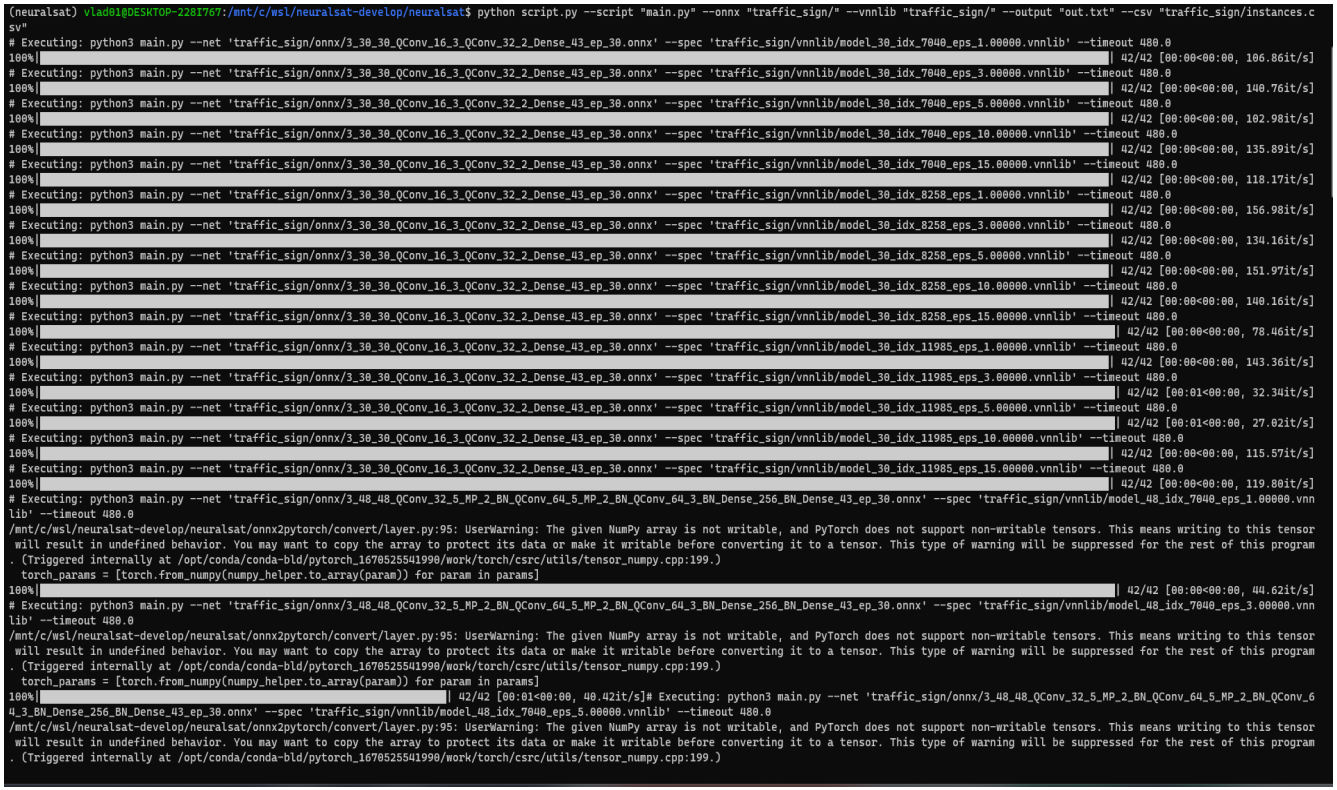
```
❯ python3 main.py --net traffic_signs_recognition/onnx/3_30_30_QConv_16_3_QConv_32_2_Dense_43_ep_30.onnx --spec traffic_signs_recognition/vnnlib/model_30_i
dx_7040_eps_3.00000.vnnlib
Restricted license - for non-production use only - expires 2024-10-28
Automatic inference of operator: sign
Automatic inference of operator: sign
Automatic inference of operator: sign
Automatic inference of operator: sign
Automatic inference of operator: sign
Automatic inference of operator: sign
Automatic inference of operator: sign
Automatic inference of operator: sign
INFO     18:09:41     [!] VNNLIB: 2700 inputs, 43 outputs
100%|████████████████████████████████████████████████████████████████████████████████████████████████████| 42/42 [00:00<00:00, 126.52it/s]
ConvertModel(
  (Transpose_sequential_10/quant_conv2d_20/QuantConv2D__185:0): 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/QuantConv2D__187:0): Transpose()
  (Add_sequential_10/quant_conv2d_21/ste_sign_52/add:0): Add()
  (Transpose_sequential_10/quant_conv2d_21/QuantConv2D__189:0): Transpose()
  (Conv_sequential_10/quant_conv2d_21/QuantConv2D:0): Conv2d(16, 32, kernel_size=(2, 2), stride=(1, 1), bias=False)
  (Transpose_sequential_10/quant_conv2d_21/QuantConv2D__191:0): 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=43, bias=False)
  (Softmax_activation_10): Identity()
)
INFO     18:09:41     [!] Input shape: (1, 3, 30, 30)
INFO     18:09:41     [!] Output shape: (1, 43)

[!] Current settings:
        - max_hidden_branches          : 5000
        - max_hidden_visited_branches  : 20000
        - use_attack                   : True
        - use_restart                  : True
        - use_stabilize                : True


INFO     18:09:41     [Failed] RandomAttack(seed=837, device=cuda)
INFO     18:09:43     [Success] PGDAttack(seed=568, device=cuda)
INFO     18:09:43     [!] Iterations: 0
INFO     18:09:43     adv (first 5): tensor([254.8914, 254.5998, 254.9661, 251.6367, 252.3546])
DEBUG    18:09:43     output: tensor([ 574.,  920., 1130.,  892., 1162.,  624.,  456.,  910.,  896.,  506.,
        732., 1414.,  710.,  466., 1176., 1174.,   60., 1552.,  936.,  890.,
       1246.,  646., 1400.,  838.,  558., 1170., 1342.,  530., 1210.,  752.,
       1058.,  776.,  530.,  376., -290.,  -80.,  524., -274.,  230.,  184.,
        248.,  580.,  338.])
sat,3.6749
```

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.



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.

```
  0%|
Traceback (most recent call last):
  File "/home/alex/Documents/repos/alpha-beta-CROWN/complete_verifier/abcrown.py", line 612, in <module>
    abcrown.main()
  File "/home/alex/Documents/repos/alpha-beta-CROWN/complete_verifier/abcrown.py", line 539, in main
    verified_status, verified_success, _, attack_margins, all_adv_candidates = attack(
  File "/home/alex/Documents/repos/alpha-beta-CROWN/complete_verifier/attack/attack_pgd.py", line 236, in attack
    attack_ret, attack_images, attack_margins, all_adv_candidates = attack_function(
  File "/home/alex/Documents/repos/alpha-beta-CROWN/complete_verifier/custom/custom_attacker.py", 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,
  File "/home/alex/Documents/repos/alpha-beta-CROWN/complete_verifier/attack/attack_pgd.py", line 1259, in attack_with_general_specs
    best_deltas_, last_deltas, best_loss_, early_stopped = pgd_attack_with_general_specs(
  File "/home/alex/Documents/repos/alpha-beta-CROWN/complete_verifier/attack/attack_pgd.py", line 677, in pgd_attack_with_general_specs
    output = model(inputs.view(-1, *input_shape[1:])).view(
  File "/home/alex/Documents/repos/alpha-beta-CROWN/complete_verifier/auto_LiRPA/bound_general.py", 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])
  File "/home/alex/Documents/repos/alpha-beta-CROWN/complete_verifier/auto_LiRPA/bound_general.py", line 421, in <listcomp>
    deque([self.get_forward_value(self[n])
  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]
  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-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]
  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-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]
  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-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]
  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-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]
  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-CROWN/complete_verifier/auto_LiRPA/bound_general.py", line 376, in get_forward_value
    fv = node.forward(*inputs)
  File "/home/alex/Documents/repos/alpha-beta-CROWN/complete_verifier/auto_LiRPA/operators/shape.py", line 18, in forward
    if shape[i] == -1:
RuntimeError: 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.
For debugging consider passing CUDA_LAUNCH_BLOCKING=1.
Compile with `TORCH_USE_CUDA_DSA` to enable device-side assertions.
```