WEST UNIVERSITY OF TIMIŞOARA
FACULTY OF MATHEMATICS AND COMPUTER SCIENCE
MASTER STUDY PROGRAM: SOFTWARE ENGINEERING

Verification of Neural Networks

Supervisor:
Mădălina Erașcu

Students:
Amalia Duma Diana
Răzvan Maciovan Alexandru
Luca Mitroi
Vlad Petcu
Vlad Țîrcomnicu

TIMIŞOARA
2023

Contents

Chapter 1

Introduction

The assignments for this project, required the usage of one benchmark and two tools for which to analyze upon. For the purpose of our project, we have decided to utilize the ACASXU benchmark[1], along with the tools Alpha Beta Crown[2] and Marabou [3]

1.1 Acasxu

Acas stands for Airborne Collision Avoidance System. There are multiple types of Acas benchmarks, but the one this paper is based on is Xu, which is optimized for unmanned aircraft systems (UAS), issuing turn rate advisories to remote pilots[Katz et al., 2017]. The installation process for the benchmark was easy, only needing to download it from GitHub[1]. The files are divided into 3 components as followed:

Onnx

These files contain the neural network models encoded in the Open Neural Network Exchange (ONNX) format[ONNX Contributors, 2023]. They collectively are the core architecture for the "brain" of the benchmark, which dictates the verification and evaluation steps.

Vnnlib

These files contain specifications and properties that need to be verified or analyzed for the neural network models (ONNX). They contain a similar syntax to that of z3 smt-solvers.

[1] https://github.com/ChristopherBrix/vnncomp2023_benchmarks/tree/main/benchmarks/acasxu

[2] https://github.com/Verified-Intelligence/alpha-beta-CROWN

[3] https://github.com/NeuralNetworkVerification/Marabou

Instances.csv

This CSV file bundles the onnx and vnnlib files into groups, where vnnlib files are associated with onnx ones. There are 10 vnnlib files and 45 onnx files, and they are combined to form a total of 186 combinations. Each combination also has metadata containing how long can the selected tool run one specific combination.

Chapter 2

Tools installation

2.1 Alpha-Beta-CROWN

Alpha Beta Crown serves as a neural network verifier based on an efficient linear bound propagation framework and branch and bound [huanzhang12, 2021]. Setting it up involved a two-step process: first, installing Miniconda and then the tool itself. There were no complications in the first part, as there is an executable file available on their official site that takes care of the entire setup once run. However, the latter part, involving the tool's installation, presented numerous complications.

The setup of Alpha Beta Crown followed their "Installation and Setup" section outlined on GitHub. The first command was supposed to clone their GitHub repository including "auto_LiRPA", an essential component for the tool's functionality. Despite running this command, only the tool's files were retrieved due to permission denials that restricted access to the second GitHub repository. Resolving this required an additional step: manually obtaining the missing necessary files and placing them in their indicated location. The next phase involved creating the conda environment using the specified command, which led to an error that can be observed in figure 2.1.

```
# >>>>>>>>>>>>>>>>>>>> ERROR REPORT <<<<<<<<<<<<<<<<<<<<

  Traceback (most recent call last):
    File "/home/amalia/miniconda3/lib/python3.11/site-packages/conda/exception_handler.py", line 17, in __call__
      return func(*args, **kwargs)
             ^^^^^^^^^^^^^^^^^^^^^
    File "/home/amalia/miniconda3/lib/python3.11/site-packages/conda_env/cli/main.py", line 56, in do_call
      exit_code = getattr(module, func_name)(arguments, parser)
                  ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
    File "/home/amalia/miniconda3/lib/python3.11/site-packages/conda/notices/core.py", line 124, in wrapper
      return func(*args, **kwargs)
             ^^^^^^^^^^^^^^^^^^^^^
    File "/home/amalia/miniconda3/lib/python3.11/site-packages/conda_env/cli/main_create.py", line 107, in execute
      spec = specs.detect(
             ^^^^^^^^^^^^^
    File "/home/amalia/miniconda3/lib/python3.11/site-packages/conda_env/specs/__init__.py", line 72, in detect
      if spec.can_handle():
         ^^^^^^^^^^^^^^^^^
    File "/home/amalia/miniconda3/lib/python3.11/site-packages/conda_env/specs/yaml_file.py", line 19, in can_handle
      self._environment = env.from_file(self.filename)
                          ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
    File "/home/amalia/miniconda3/lib/python3.11/site-packages/conda_env/env.py", line 191, in from_file
      return from_yaml(yamlstr, filename=filename)
             ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
    File "/home/amalia/miniconda3/lib/python3.11/site-packages/conda_env/env.py", line 162, in from_yaml
      data = validate_keys(data, kwargs)
             ^^^^^^^^^^^^^^^^^^^^^^^^^^^
    File "/home/amalia/miniconda3/lib/python3.11/site-packages/conda_env/env.py", line 30, in validate_keys
      new_data = data.copy() if data else {}
                 ^^^^^^^^^
  AttributeError: 'str' object has no attribute 'copy'
```

Figure 2.1: Creating conda environment error

This error consumed a considerable amount of time, as the error message initially suggested an issue with the script. After extensive investigation, the root cause was traced to the configuration file. Instead of containing the necessary configuration

for the conda environment, it referenced another configuration file by name. After resolving the configuration file issue, the installation of the tool seemed successful. However, when attempting to run it, another problem emerged indicating a missing library, "auto_LiRPA". Although the necessary files existed, the error message revealed a discrepancy in the file path. It seemed that the correct location for this library was wrongly specified. With this adjustment, the setup concluded, and the tool run successfully.

2.2 Marabou

Bibliography

[huanzhang12, 2021] huanzhang12 (2021). alpha-beta-crown. `https://github.com/Verified-Intelligence/alpha-beta-CROWN`.

[Katz et al., 2017] Katz, G., Barrett, C., Dill, D. L., Julian, K., and Kochenderfer, M. J. (2017). Reluplex: An efficient smt solver for verifying deep neural networks. In *Computer Aided Verification: 29th International Conference, CAV 2017, Heidelberg, Germany, July 24-28, 2017, Proceedings, Part I 30*, pages 97–117. Springer.

[ONNX Contributors, 2023] ONNX Contributors (Accessed: 2023). Open Neural Network Exchange (ONNX). `https://onnx.ai/`.