Creating a Formally Verified Neural Network for Autonomous Navigation

Maynooth University
National University of Ireland Maynooth

Syed Ali Asadullah Bukhari, Barak A. Pearlmutter and Rosemary Monahan
Department of Computer Science
National University of Ireland, Maynooth

1. Overview

Motivation:

- The increased reliance of self-driving vehicles on neural networks

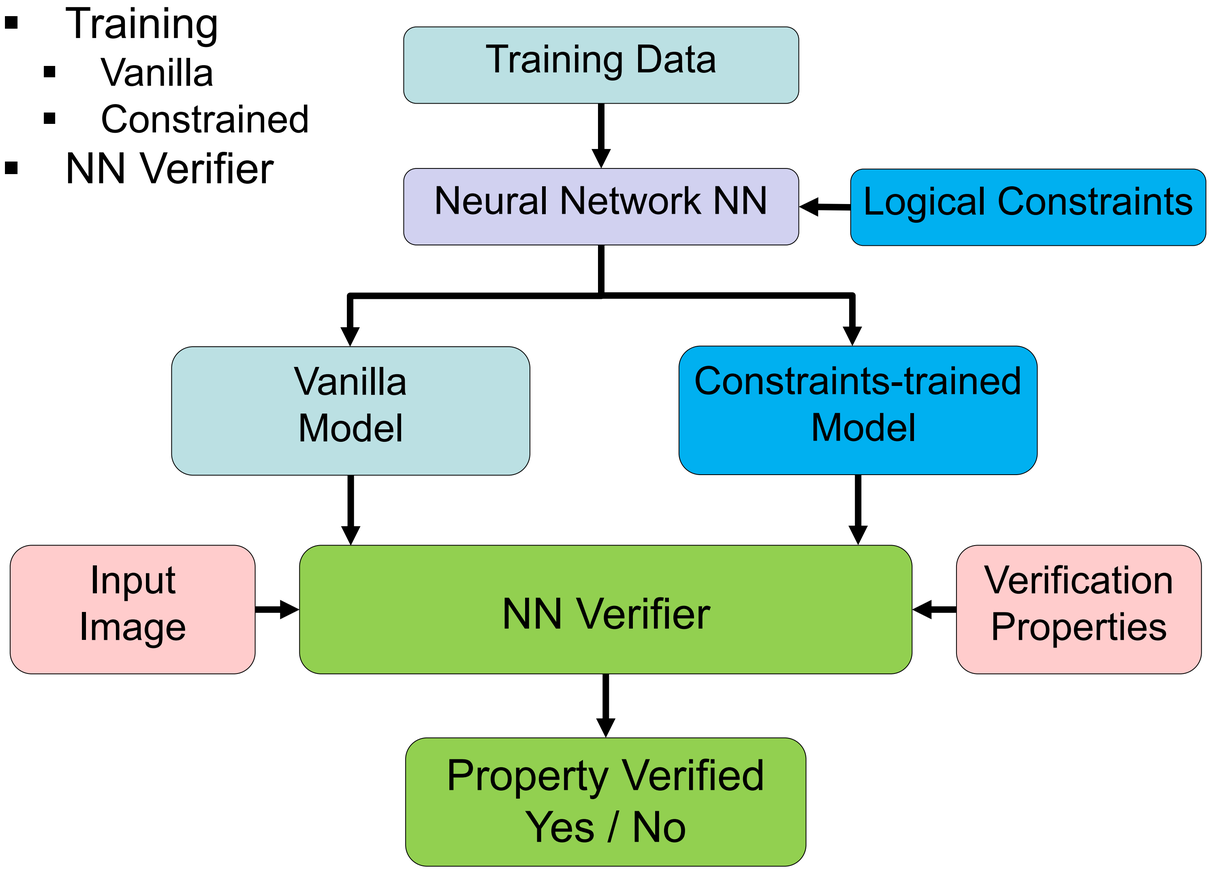
Must be formally verified before put to actual use !

Challenge:

- Design and training of a neural network on a custom dataset
- Use of machine learning with differentiable logics
- Obtaining formal guarantees of the neural network after training.

Suggested Approach:

- Training
  - Vanilla
  - Constrained
- NN Verifier



2. Case Study

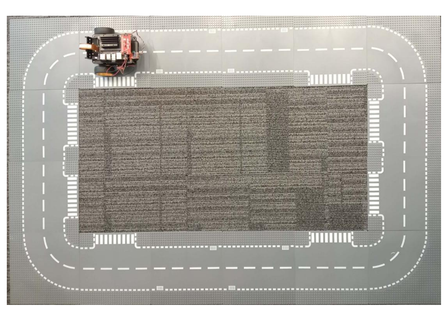
- Build a formally verified regression neural network that detects the centre of the track on which the vehicle is travelling.

Experimental Setup

- JetBot: an open-source AI robotic vehicle based on NVIDIA Jetson Nano.
- Reference track: built using readily available Lego road plates.

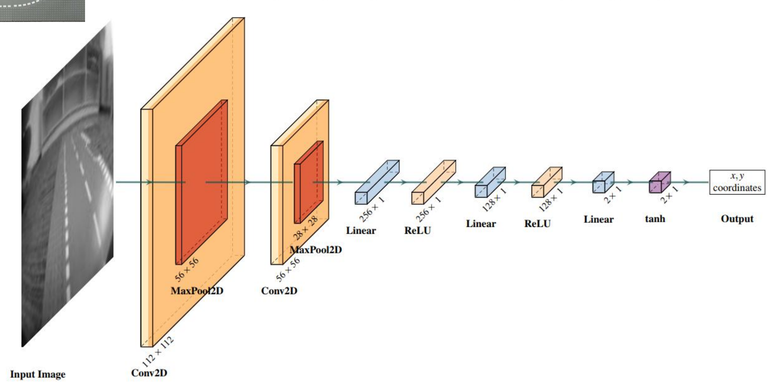
Data Collection

Data: Images from the onboard camera facing down on the track.
Labels: *x*, *y* coordinates indicating the centre of the track.
Total Samples : 385 images of size 224×224 pixels



Neural Network

Approximate the function
$\mathbb{N} : \mathbb{R}^{112\times112} \rightarrow \mathbb{R}^2$
Input : 112×112 image
Output: *x, y* coordinates



Funding

Science Foundation Ireland sfi For what's next

Contact

ali.bukhari@mu.ie
barak.pearlmutter@mu.ie
rosemary.monahan@mu.ie

3. Verification Property

Local Robustness:

$(x_0, \varepsilon, \delta) := \forall x. ||x_0 - x||_\infty \leq \varepsilon \Rightarrow ||\mathcal{N}(x_0) - \mathcal{N}(x)||_\infty \leq \delta$

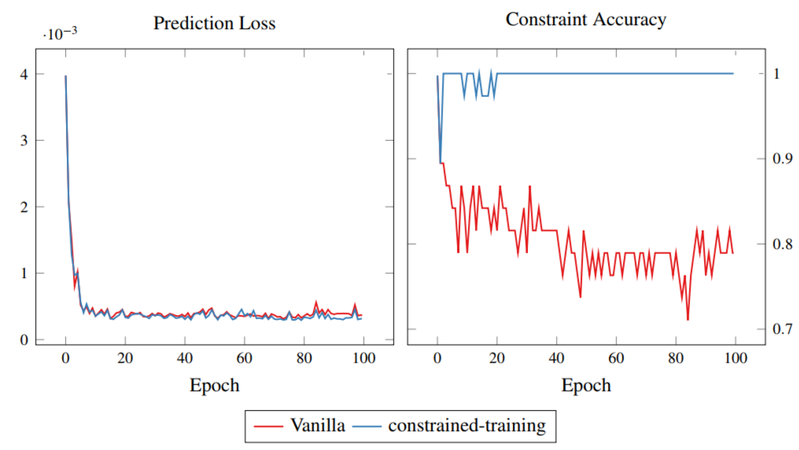
- This property checks that for slight perturbations *x* within some bound ε of a given input image x_0, the neural network $\mathcal{N}$ should give roughly the same output, i.e., difference between $\mathcal{N}(x_0)$ and $\mathcal{N}(x)$ should be within an acceptable threshold δ.

Property specification using DNNV:

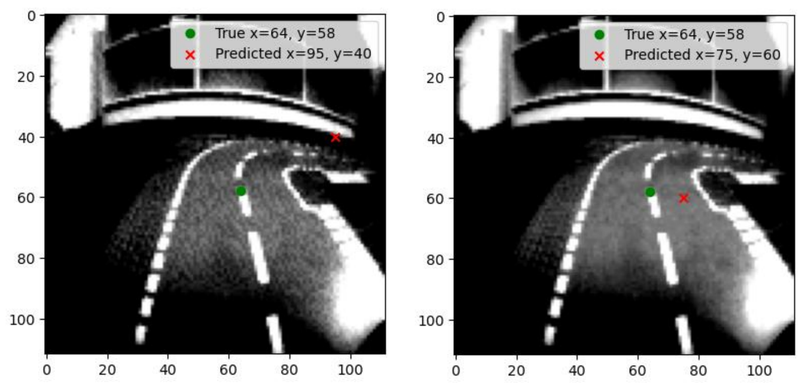
```
from dnnv . properties import *
N = Network ("N")
x = Image (" data / image_0 .npy ")
epsilon = Parameter (" epsilon ", float , default =(48. / 255))
delta = Parameter (" delta ", float , default =0.1)
Forall ( x_ ,
Implies (
((x - epsilon ) <= denormalise (x_) <= (x + epsilon )),
(abs(N(x_ )[0][0] - N(x )[0][0]) <= delta ) &
(abs(N(x_ )[0][1] - N(x )[0][1]) <= delta )
), )
```

4. Experimental results

Performance:



Prediction loss and Constraint accuracy of Vanilla vs Constrained-trained.



Vanilla (left) vs. Constrained-training predictions on an adversarial example.

Verification of Robustness Property:

- DNNV: A unified interface for NN verifiers
- Verifiers used: BAB, ERAN, Reluplex, nnenum, and MATLAB NNV

Output from DNNV:

```
dnnv . verifiers .bab
     result : BabTranslatorError ( Unsupported computation graph detected )
     time : 0.3495
dnnv . verifiers . eran
     result : unknown
     time : 4.9732
dnnv . verifiers . nnenum
     result : NnenumError ( Return code : 1)
     time : 0.7476
dnnv . verifiers . reluplex
     result : ReluplexTranslatorError ( Unsupported computation graph detected )
     time : 0.3622
```

5. Conclusion

Several insights were obtained including (but not limited to):

- Difference b/w traditional formal verification and verification of NN
  - NN verification limited to local properties in the available data.
- Effect of the neural network architecture on verification effort
  - Max-pooling vs. Average layers
- Efforts required in use of NN verification tools
- Tools focus more on classification rather than regression

Ref: **S. A. A. Bukhari**, T. Flinkow, M. Inkarbekov, B. A. Pearlmutter & R. Monahan, *"Creating a Formally Verified Neural Network for Autonomous Navigation: An Experience Report"*, Sixth International Workshop on Formal Methods for Autonomous System FMAS 2024, Accepted.