Maynooth University
National University of Ireland Maynooth


# Creating a Formally Verified Neural Network for Autonomous Navigation: An Experience Report

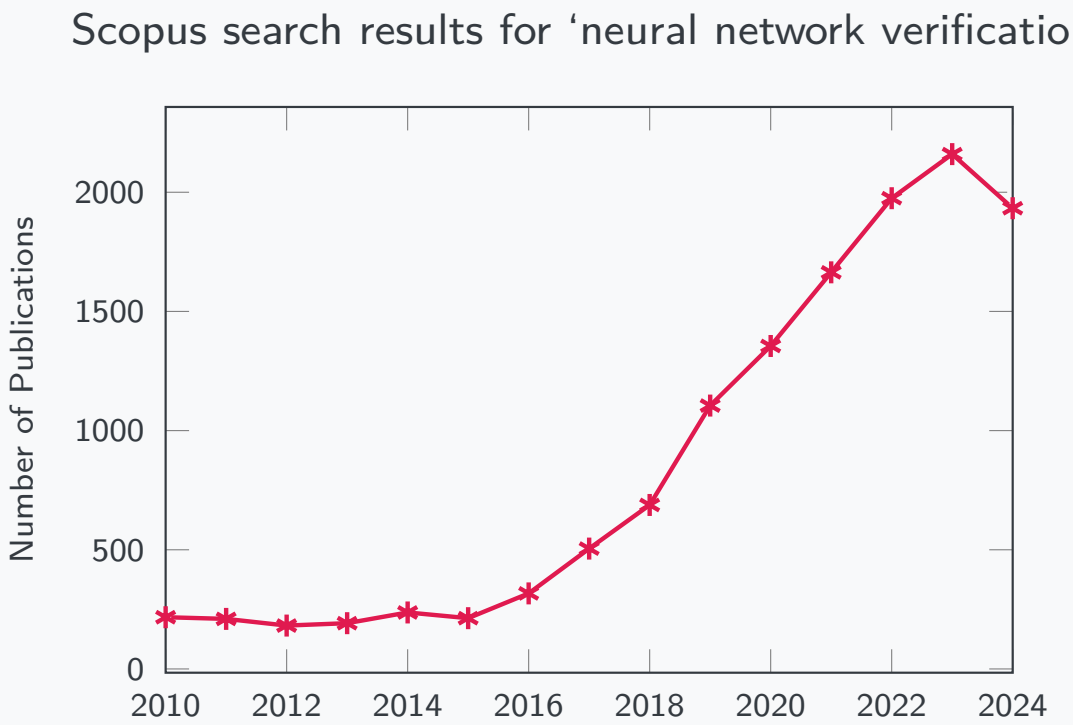
Syed Ali Asadullah Bukhari, Thomas Flinkow, Medet Inkarbekov, Barak A. Pearlmutter, and Rosemary Monahan

*Department of Computer Science*
*Maynooth University*

FMAS 2024
12th November 2024

# Background

# Formal Verification of Neural Networks



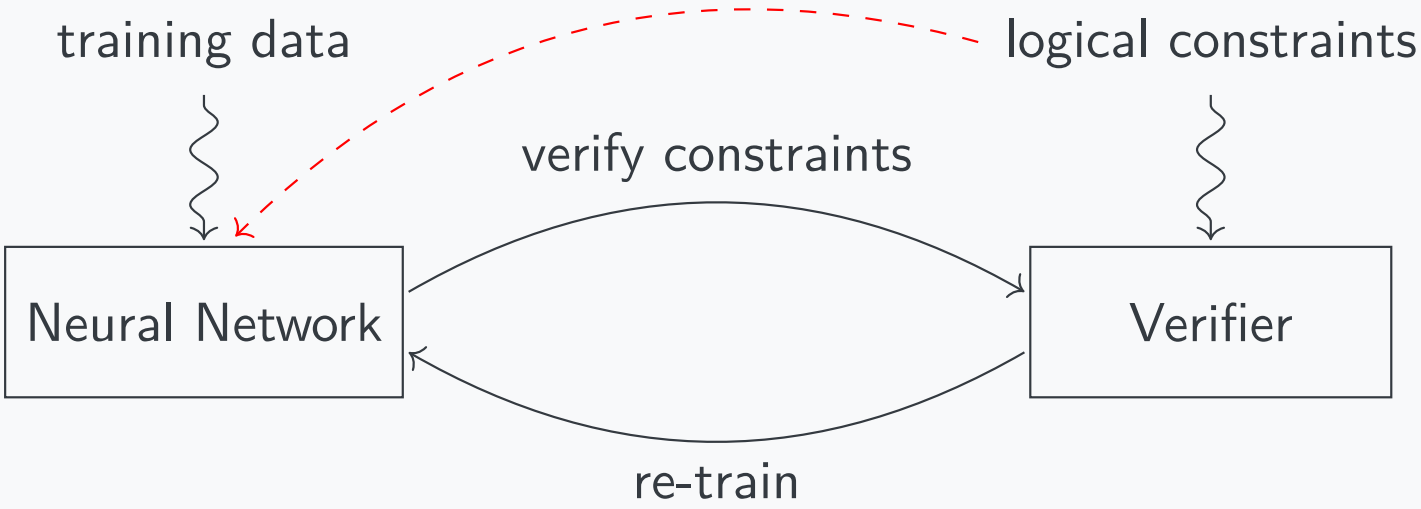

- **SMT Solvers:** Planet (Ehlers 2017), Reluplex (Katz, Barrett et al. 2017), Marabou (Katz, Huang et al. 2019), ...
- **MILP:** Branch-and-Bound (Bunel et al. 2018), MIPVerify (Tjeng et al. 2019), Sherlock (Dutta et al. 2019), ...
- **Abstract Interpretation:** AI2 (Gehr et al. 2018), DeepPoly (Singh et al. 2019), NNV (Tran et al. 2020), ...

# Training with Logical Constraints

**Task:** train a neural network $\mathcal{N}$ to satisfy constraint $\phi$.

**Train:** given data, labels, and loss function, iteratively update network weights.

**Verify:** afterwards, $\alpha, \beta$-CROWN, Marabou, NNV, ERAN, ...




**Note**

Training with constraints does not guarantee their satisfaction!

# Property-driven Training with Differentiable Logics

Given data $\boldsymbol{x}_0$ and label $\boldsymbol{y}$, and constraint $\phi$,
obtain optimal network weights $\boldsymbol{\theta}^+$ by

$$\boldsymbol{\theta}^+ = \arg\min_{\boldsymbol{\theta}} \ \alpha\mathcal{L}_{\mathsf{MSE}}(\boldsymbol{x}_0, \boldsymbol{y}) + \beta\mathcal{L}_{\mathsf{C}}(\boldsymbol{x}_0, \boldsymbol{y}, \phi).$$

Optimisation handled by Gradient Descent:

$$\boldsymbol{\theta}_{i+1} := \boldsymbol{\theta}_i - \eta\nabla\mathcal{L}(\boldsymbol{\theta}_i)$$

**Implications**

Logical constraint $\phi$ must be differentiable!

# DL2[1] and Fuzzy Logics[2,3]

**DL2 (mapping $[\![\cdot]\!]_{\mathbf{DL2}} : \Phi \to [0, \infty)$)**

- $[\![\top]\!]_{\text{DL2}} = 0$.

$$[\![\phi \wedge \psi]\!]_{\text{DL2}} := [\![\phi]\!]_{\text{DL2}} + [\![\psi]\!]_{\text{DL2}}$$
$$[\![\phi \vee \psi]\!]_{\text{DL2}} := [\![\phi]\!]_{\text{DL2}} \cdot [\![\psi]\!]_{\text{DL2}}.$$

**Gödel Fuzzy Logic (mapping $[\![\cdot]\!]_{\mathbf{Gödel}} : \Phi \to [0, 1]$)**

- $[\![\top]\!]_{\text{Gödel}} = 1$ and $[\![\bot]\!]_{\text{Gödel}} = 0$.

$$[\![\phi \wedge \psi]\!]_{\text{Gödel}} := \min(\phi, \psi)$$
$$[\![\phi \vee \psi]\!]_{\text{Gödel}} := \max(\phi, \psi).$$

---

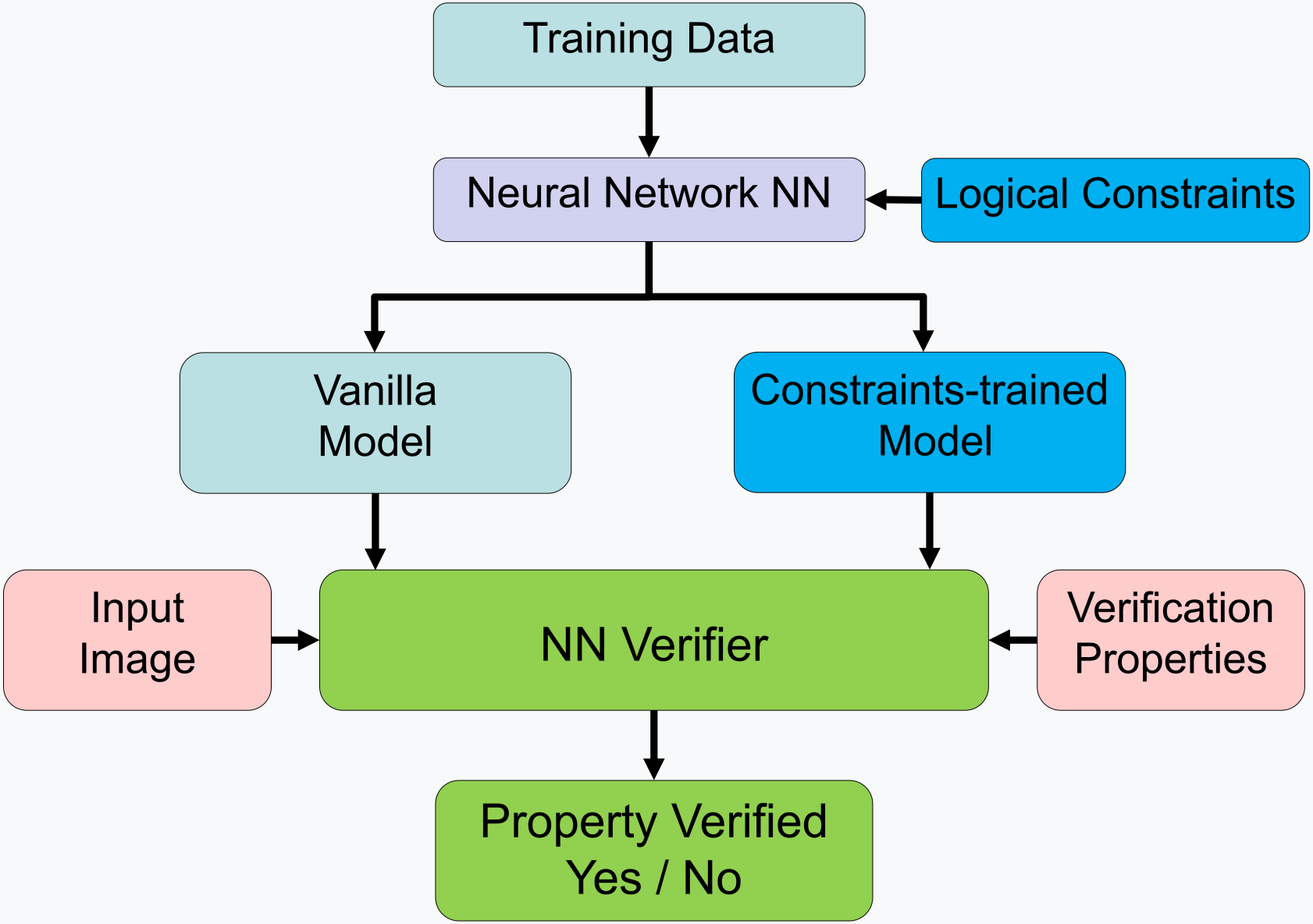
[1] Marc Fischer et al.: 'DL2: Training and Querying Neural Networks with Logic' 2019.

[2] Natalia Ślusarz et al.: 'Logic of Differentiable Logics: Towards a Uniform Semantics of DL' 2023. DOI: 10.29007/c1nt.

[3] Emile van Krieken et al.: 'Analyzing Differentiable Fuzzy Logic Operators' 2022. DOI: 10.1016/j.artint.2021.103602.

# Experimental Setup

# Suggested Approach



# Jetbot & Data Collection

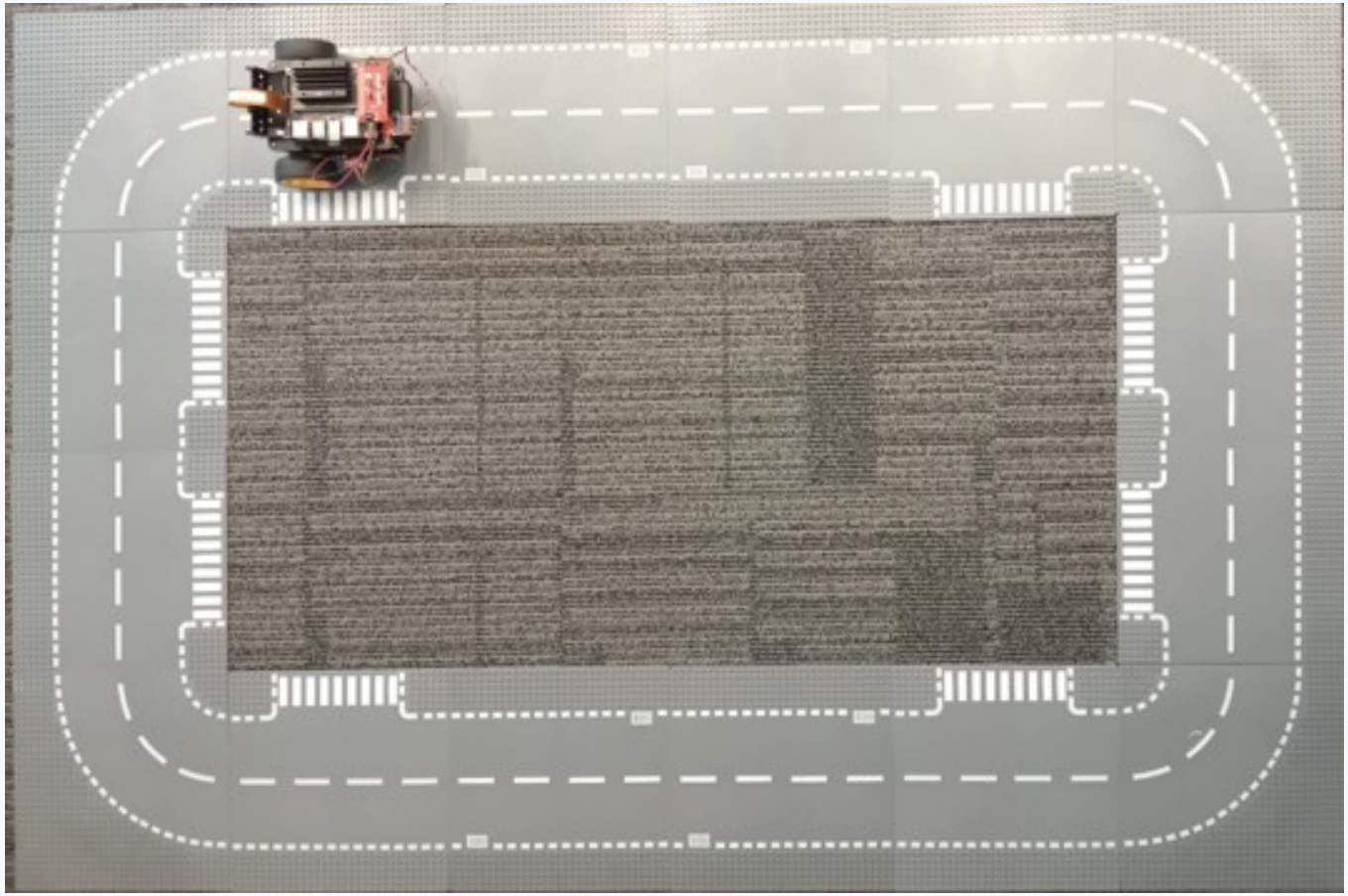
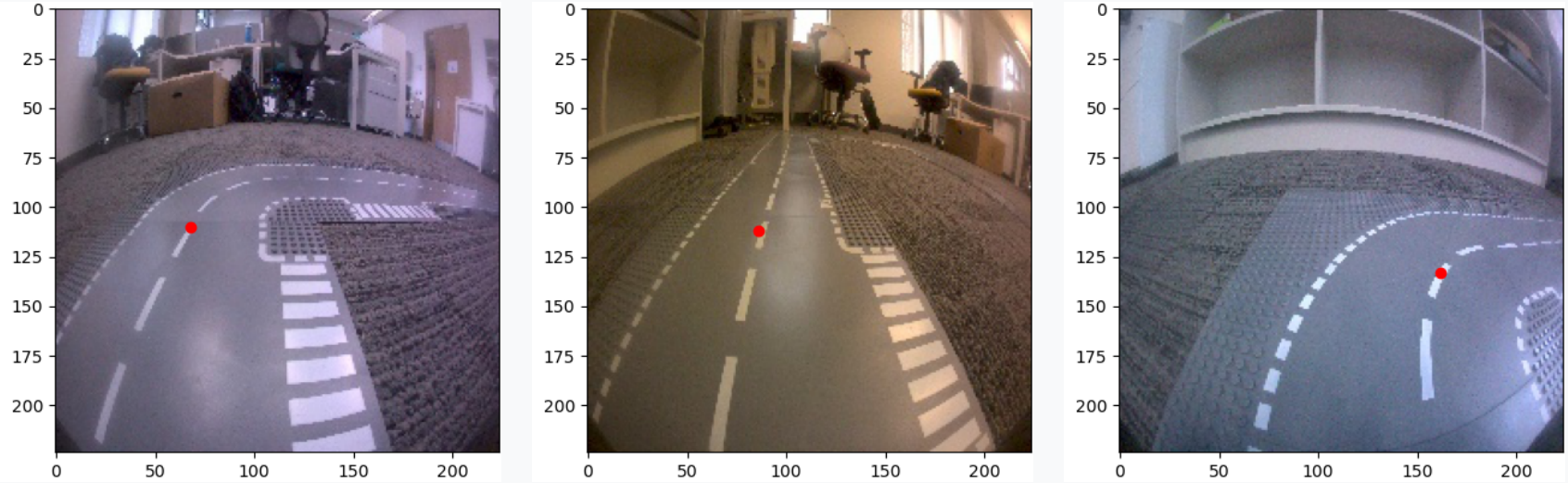


**Figure 1:** For demonstration purposes, we build a Lego reference track and use JetBot, an open-source AI robotic vehicle based on NVIDIA Jetson Nano, for autonomous navigation.

# Dataset

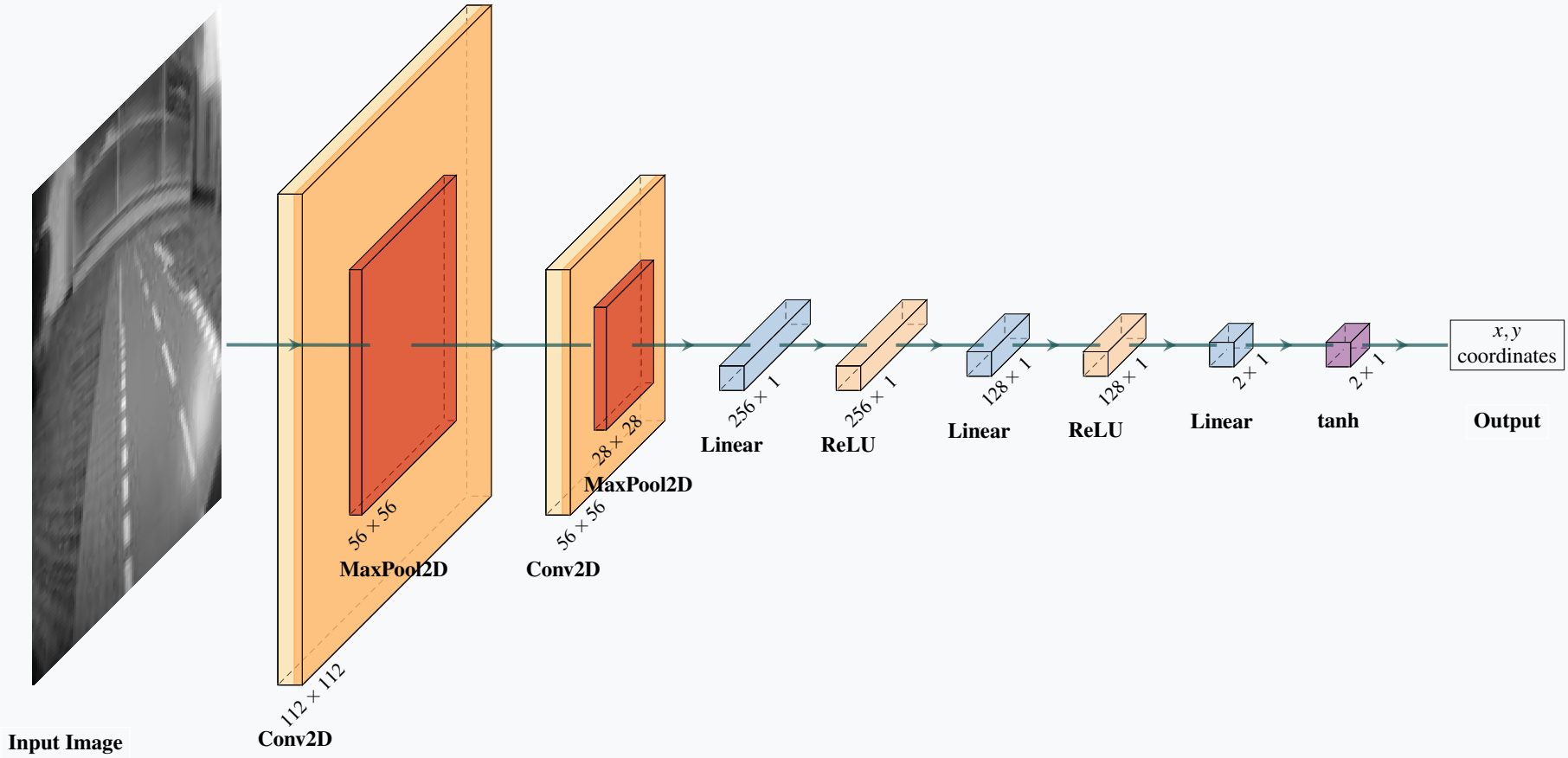



**(a)** label = 68 ,110 **(b)** label = 86, 112 **(c)** label = 162, 133

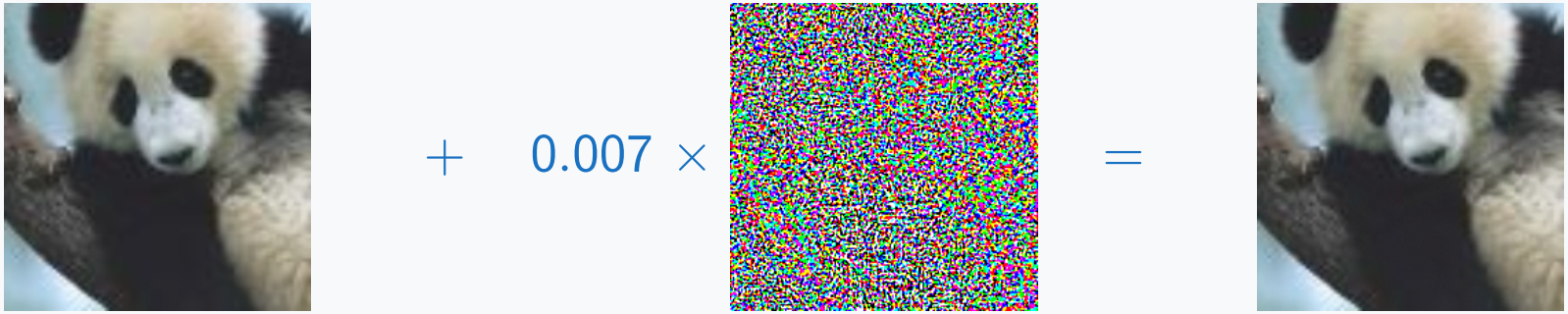
**Figure 2:** Samples from the LEGO road dataset showing various lighting conditions and track configurations. The red dots located at label coordinates show the centre of the path to be followed in the frame.

# Neural Network Architecture

- Network: $\mathcal{N} : \mathbb{R}^{112\times112} \to \mathbb{R}^2$
- Inputs: images of size $112 \times 112$
- Outputs: $x, y$-coordinates (image label)



# Local Robustness Constraint




‘panda’
57.7 % confidence

adversarial
noise

‘gibbon’
99.3 % confidence

**Figure 3:** Adversarial attack (Goodfellow et al. 2015).

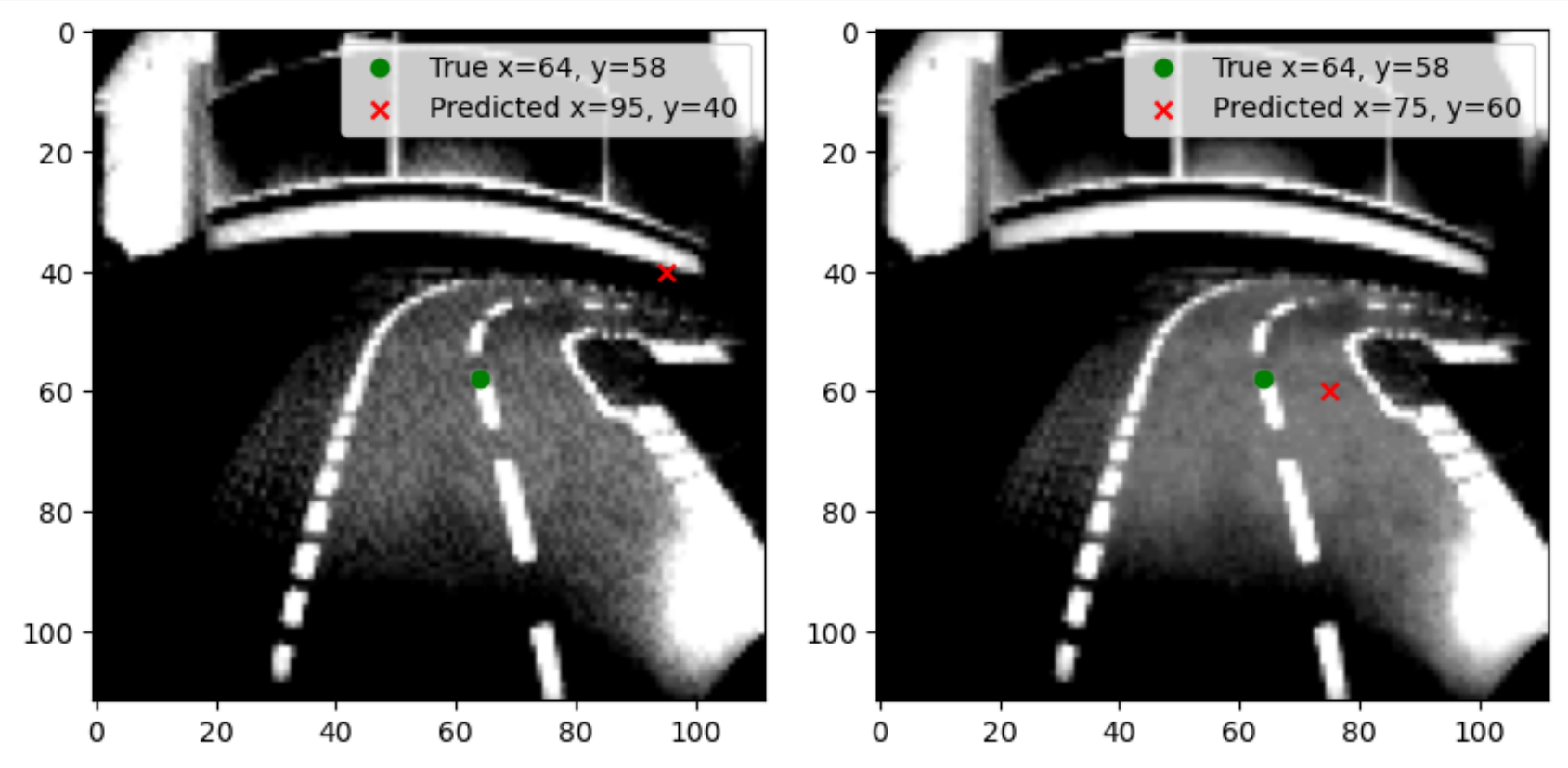
**Definition**

A neural network is locally robust in input $\boldsymbol{x}_0$, if

$$\underbrace{\forall \boldsymbol{x}.\, \|\boldsymbol{x} - \boldsymbol{x}_0\|_\infty \leq \varepsilon}_{\text{all elements in the input space close to } \boldsymbol{x}_0} \quad \text{implies} \quad \underbrace{\|\mathcal{N}(\boldsymbol{x}) - \mathcal{N}(\boldsymbol{x}_0)\|_\infty \leq \delta}_{\text{the classification is roughly the same}}$$

# Results

# Results – Performance




**(a)** Vanilla

**(b)** Constrained-training

**Figure 4:** Comparison of predictions of models trained without (vanilla) and with logical constraints (constrained-training) on an adversarial example in epoch 45 of 100.

# Results – Loss & Accuracy

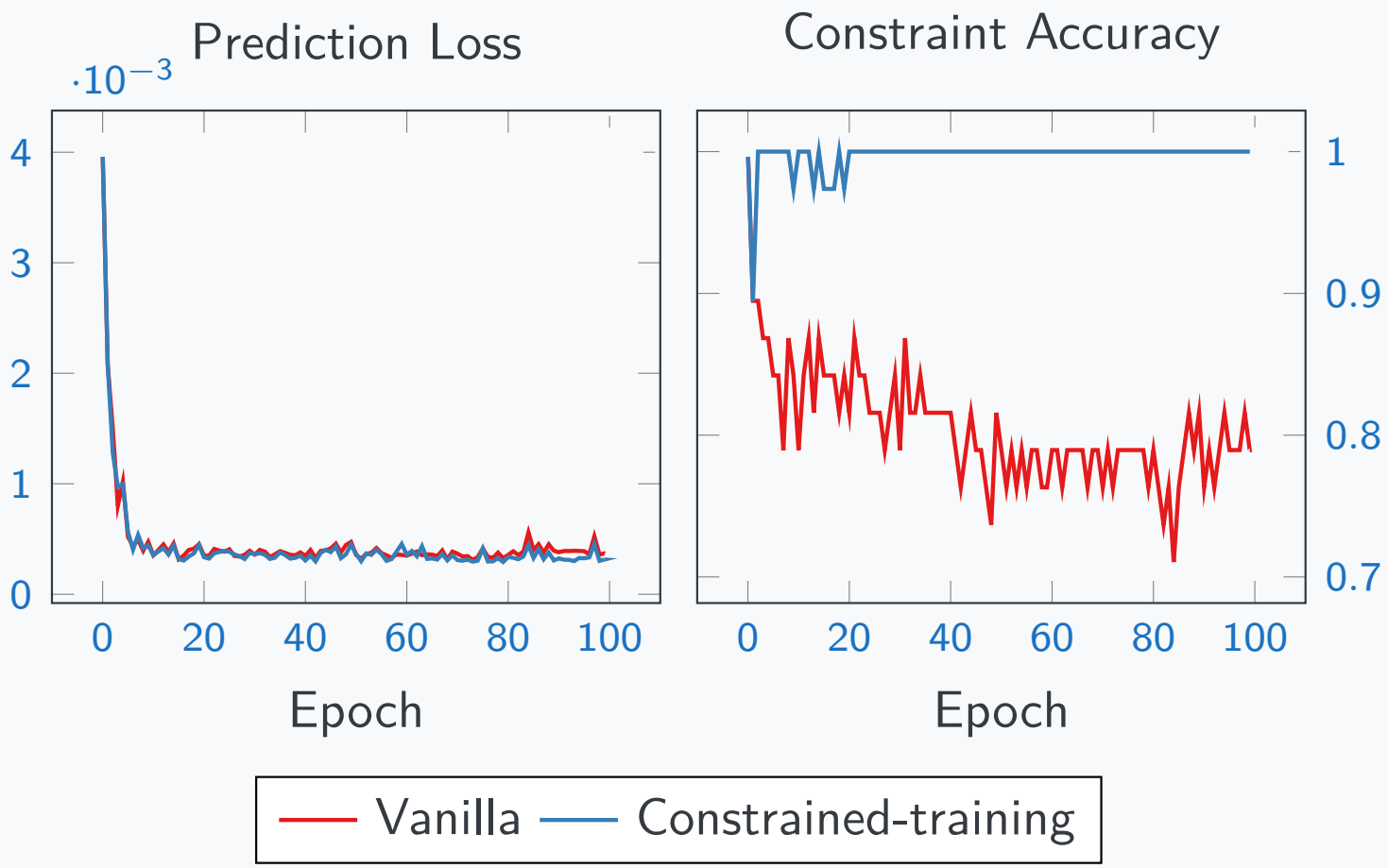



**Figure 5:** The prediction loss (i.e., mean squared error; lower is better) and constraint accuracy (higher is better) for a model trained only on data (vanilla) and a model trained on both data and logical constraints with differentiable logics (constrained-training).

# Verification Property in DNNV (Shriver et al. 2021)

all elements in the input space close to $x_0$ — $\overbrace{\forall x.\, \|x - x_0\|_\infty \leq \varepsilon}$ implies $\overbrace{\|\mathcal{N}(x) - \mathcal{N}(x_0)\|_\infty \leq \delta}$ — the output is roughly the same

**Listing 1:** Verification Property in DNNV's input language.

```
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)
    ),
)
```

# Results – Verification in DNNV

**Listing 2:** Output of DNNV for verification of robustness property.

```
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
```

# Results – Verification in NNV (Tran et al. 2020)

| **Input Perturbations $\epsilon$ (Normalised Image)** | **Network Configuration** | | | |
|---|---|---|---|---|
| | Vanilla | | Constrained-training | |
| | $x$ | $y$ | $x$ | $y$ |
| 0.001 | [9 – 107] | [19 – 97] | [8 – 107] | [15 – 97] |
| 0.01 | [1 – 112] | [1 – 112] | [1 – 112] | [1 – 112] |

# Lessons Learnt

# Conclusion & Lessons Learnt

## Neural Network Architecture

- Unfavourable layer types (max-pooling) and activation functions (tanh): most verification tools have better support for fully-connected and convolutional layers as well as ReLU and other piece-wise linear activation functions.

## Verification Tools

- Installation issues (unable to install Marabou in DNNV) & outdated requirements (DNNV: Python $\geq 3.7, < 3.10$)
- Most verification tools focus on classification, not on regression.

## Verification Properties

- Robustness verification was limited to point-wise verification around known images from the data set (no global guarantees of robustness).

**Thank you!**
**Any questions?**

# References

Bunel, Rudy R et al. (2018). 'A Unified View of Piecewise Linear Neural Network Verification'. In: *Advances in Neural Information Processing Systems*. Vol. 31. Curran Associates, Inc.

Dutta, Souradeep et al. (16th Apr. 2019). 'Sherlock - A Tool for Verification of Neural Network Feedback Systems: Demo Abstract'. In: *Proceedings of the 22nd ACM International Conference on Hybrid Systems: Computation and Control*. HSCC '19. New York, NY, USA: Association for Computing Machinery, pp. 262–263. ISBN: 978-1-4503-6282-5. DOI: 10.1145/3302504.3313351.

Ehlers, Rüdiger (2017). 'Formal Verification of Piece-Wise Linear Feed-Forward Neural Networks'. In: *Automated Technology for Verification and Analysis*. Ed. by Deepak D'Souza and K. Narayan Kumar. Lecture Notes in Computer Science. Cham: Springer International Publishing, pp. 269–286. ISBN: 978-3-319-68167-2. DOI: 10.1007/978-3-319-68167-2_19.

# References (cont.)


Fischer, Marc et al. (24th May 2019). 'DL2: Training and Querying Neural Networks with Logic'. In: *Proceedings of the 36th International Conference on Machine Learning*. International Conference on Machine Learning. PMLR, pp. 1931–1941.

Gehr, Timon et al. (May 2018). 'AI2: Safety and Robustness Certification of Neural Networks with Abstract Interpretation'. In: *2018 IEEE Symposium on Security and Privacy (SP)*. 2018 IEEE Symposium on Security and Privacy (SP), pp. 3–18. DOI: 10.1109/sp.2018.00058.

Goodfellow, Ian J. et al. (Mar. 2015). *Explaining and Harnessing Adversarial Examples*. DOI: 10.48550/arXiv.1412.6572. arXiv: 1412.6572 [cs, stat].

Katz, Guy, Clark Barrett et al. (2017). 'Reluplex: An Efficient SMT Solver for Verifying Deep Neural Networks'. In: *Computer Aided Verification*. Ed. by Rupak Majumdar and Viktor Kunčak. Lecture Notes in Computer Science. Cham: Springer International Publishing, pp. 97–117. ISBN: 978-3-319-63387-9. DOI: 10.1007/978-3-319-63387-9_5.

# References (cont.)


Katz, Guy, Derek A. Huang et al. (2019). 'The Marabou Framework for Verification and Analysis of Deep Neural Networks'. In: *Computer Aided Verification*. Ed. by Isil Dillig and Serdar Tasiran. Lecture Notes in Computer Science. Cham: Springer International Publishing, pp. 443–452. ISBN: 978-3-030-25540-4. DOI: 10.1007/978-3-030-25540-4_26.

Shriver, David et al. (2021). 'DNNV: A Framework for Deep Neural Network Verification'. In: *Computer Aided Verification*. Ed. by Alexandra Silva and K. Rustan M. Leino. Lecture Notes in Computer Science. Cham: Springer International Publishing, pp. 137–150. ISBN: 978-3-030-81685-8. DOI: 10.1007/978-3-030-81685-8_6.

Singh, Gagandeep et al. (2nd Jan. 2019). 'An Abstract Domain for Certifying Neural Networks'. In: *Proceedings of the ACM on Programming Languages* 3 (POPL), pp. 1–30. ISSN: 2475-1421. DOI: 10.1145/3290354.

# References (cont.)


Ślusarz, Natalia et al. (3rd June 2023). 'Logic of Differentiable Logics: Towards a Uniform Semantics of DL'. In: *EPiC Series in Computing*. Proceedings of 24th International Conference on Logic for Programming, Artificial Intelligence and Reasoning. Vol. 94. EasyChair, pp. 473–493. DOI: 10.29007/c1nt.

Tjeng, Vincent et al. (17th Feb. 2019). *Evaluating Robustness of Neural Networks with Mixed Integer Programming*. DOI: 10.48550/arXiv.1711.07356. arXiv: 1711.07356 [cs]. Pre-published.

Tran, Hoang-Dung et al. (2020). 'NNV: The Neural Network Verification Tool for Deep Neural Networks and Learning-Enabled Cyber-Physical Systems'. In: *Computer Aided Verification*. Ed. by Shuvendu K. Lahiri and Chao Wang. Lecture Notes in Computer Science. Cham: Springer International Publishing, pp. 3–17. ISBN: 978-3-030-53288-8. DOI: 10.1007/978-3-030-53288-8_1.

# References (cont.)

Van Krieken, Emile et al. (Jan. 2022). 'Analyzing Differentiable Fuzzy Logic Operators'. In: *Artificial Intelligence* 302, p. 103602. ISSN: 00043702. DOI: 10.1016/j.artint.2021.103602. arXiv: 2002.06100 [cs].