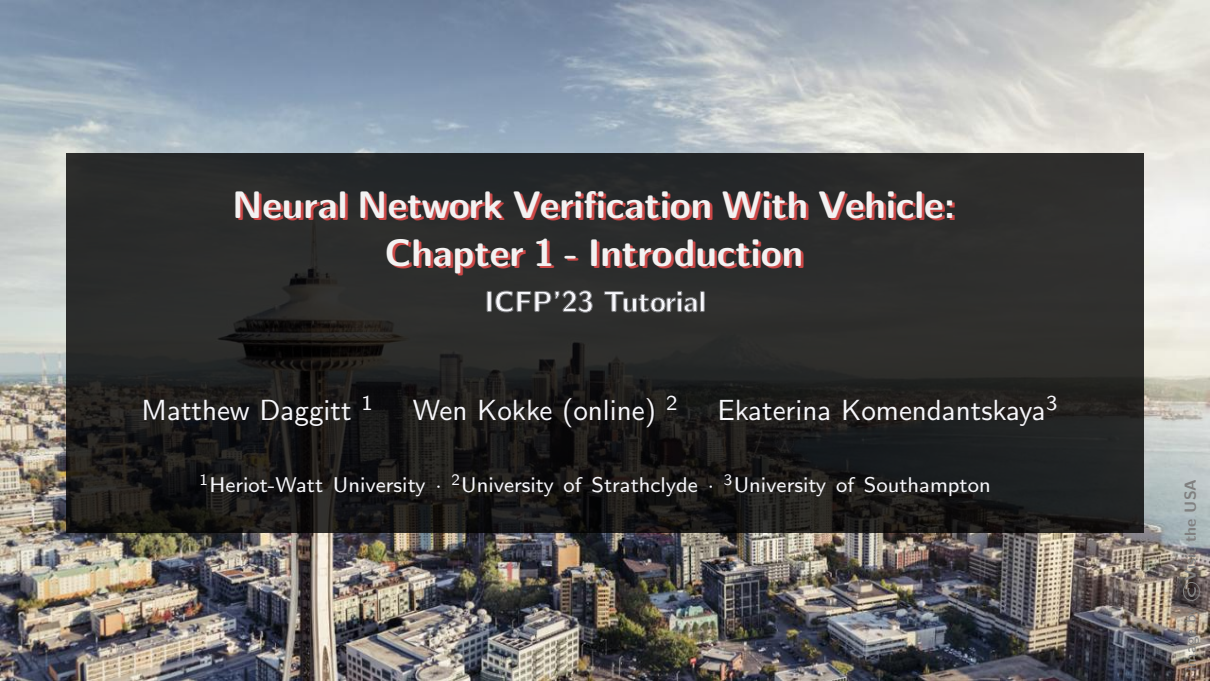


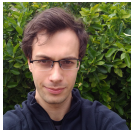
Neural Network Verification With Vehicle: Chapter 1 - Introduction

ICFP'23 Tutorial

Matthew Daggitt [1] Wen Kokke (online) [2] Ekaterina Komendantskaya[3]

[1]Heriot-Watt University · [2]University of Strathclyde · [3]University of Southampton

The Vehicle Team



Matthew Daggitt

Wen Kokke

Bob Atkey

Katya Komendantskaya

Natalia Slusarz

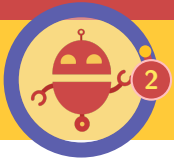
Luca Arnaboldi

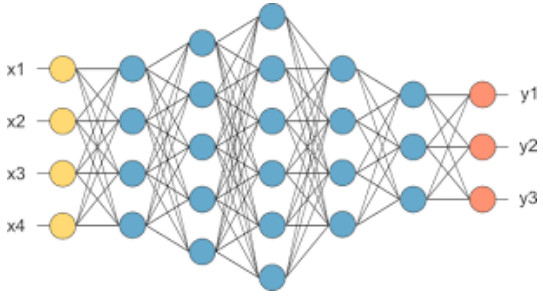
Marco Casadio

Ben Coke

Jeonghyeon Lee

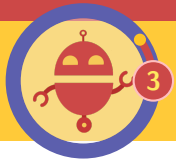
Neural network





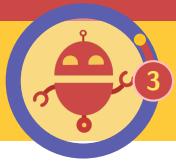
A neural network is a function $N : \mathbb{R}^n \to \mathbb{R}^m$.

Neural networks



They are ideal for tasks where the semantics of the domain are poorly understood:

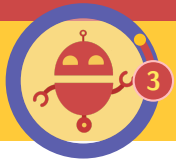
Neural networks



They are ideal for tasks where the semantics of the domain are poorly understood:

- they can learn to approximate complex functions.

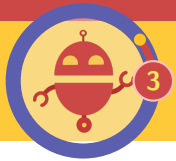
Neural networks



They are ideal for tasks where the semantics of the domain are poorly understood:

- they can learn to approximate complex functions.
- they are tolerant to noisy and incomplete data.

Neural networks



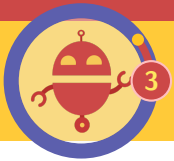
They are ideal for tasks where the semantics of the domain are poorly understood:

- they can learn to approximate complex functions.
- they are tolerant to noisy and incomplete data.

however...

- their solutions are not easily explainable.
- they are prone to a new range of bugs and safety and security problems.

Neural networks



They are ideal for tasks where the semantics of the domain are poorly understood:

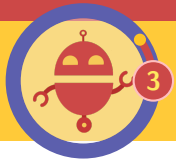
- they can learn to approximate complex functions.
- they are tolerant to noisy and incomplete data.

however...

- their solutions are not easily explainable.
- they are prone to a new range of bugs and safety and security problems.

Question 1: can we verify properties of neural network-enhanced software?

Neural networks



They are ideal for tasks where the semantics of the domain are poorly understood:

- they can learn to approximate complex functions.
- they are tolerant to noisy and incomplete data.

however...

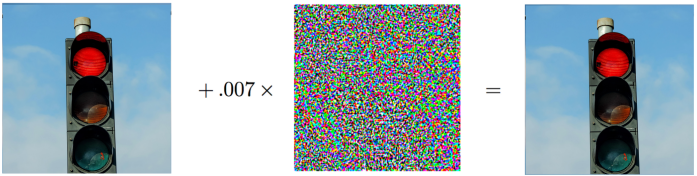
- their solutions are not easily explainable.
- they are prone to a new range of bugs and safety and security problems.

Question 0: can we even write down any properties of neural networks?

Property source 1: Universal properties

Properties that should hold of (almost) any neural network.

Example 1: Adversarial robustness



"red"

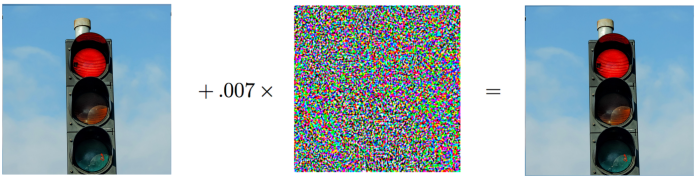
97.6% confidence

noise

"green"

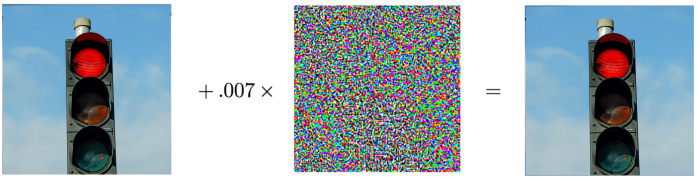
81.2% confidence

Example 1: Adversarial robustness



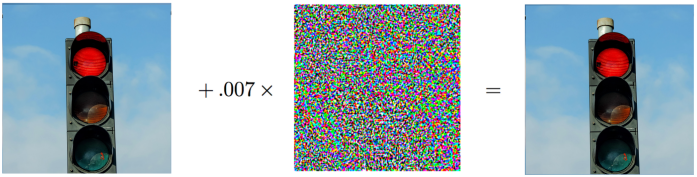
- the perturbations are imperceptible to human eye

Example 1: Adversarial robustness



- the perturbations are imperceptible to human eye
- attacks transfer from one neural network to another

Example 1: Adversarial robustness



"red"

97.6% confidence

noise

"green"

81.2% confidence

- the perturbations are imperceptible to human eye
- attacks transfer from one neural network to another
- affect any domain where neural networks are applied

Example 1: Adversarial robustness

Property: “small” changes in the input should produce “small” changes in the output.

Example 1: Adversarial robustness

Property: “small” changes in the input should produce “small” changes in the output.

Example 1: Adversarial robustness

Property: "small" changes in the input should produce "small" changes in the output.

Definition (Robustness)

Let $f \in \mathbb{R}^m \to \mathbb{R}^n$ be a neural network, $\hat{\mathbf{x}} \in \mathbb{R}^m$ be an input and $\epsilon, \delta \in \mathbb{R}$ then the network is $\epsilon - \delta$-robust around $\hat{\mathbf{x}}$ iff:

$$\forall \mathbf{x} \in \mathbb{R}^m : |\mathbf{x} - \hat{\mathbf{x}}| \leq \epsilon \Rightarrow |f(\mathbf{x}) - f(\hat{\mathbf{x}})| \leq \delta$$

Property source 2: Approximations of standard algorithms

Sometimes neural networks are used to approximate “standard” algorithms.

Property source 2: Approximations of standard algorithms

Sometimes neural networks are used to approximate “standard” algorithms.

Why? Mainly for performance reasons.

Property source 2: Approximations of standard algorithms

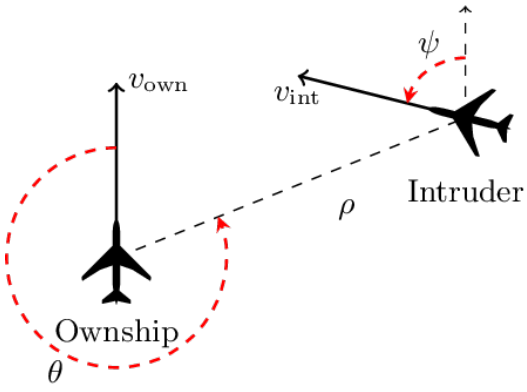
Sometimes neural networks are used to approximate “standard” algorithms.

Why? Mainly for performance reasons.

We can derive desired properties of the network from those of the original algorithm.

Example 2: ACAS Xu

A collision avoidance system for unmanned autonomous aircraft.

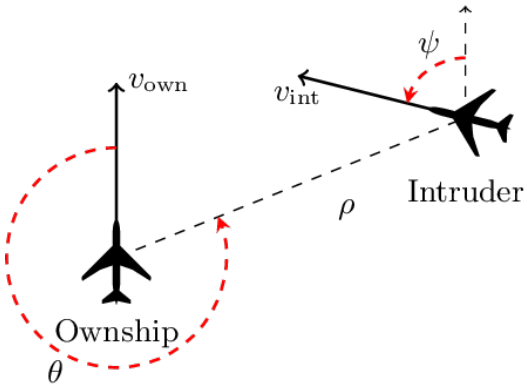


Example 2: ACAS Xu

A collision avoidance system for unmanned autonomous aircraft.

Inputs:

- Distance to intruder, ρ
- Angle to intruder, θ
- Intruder heading, φ
- Speed, v_{own}
- Intruder speed, v_{int}



Example 2: ACAS Xu

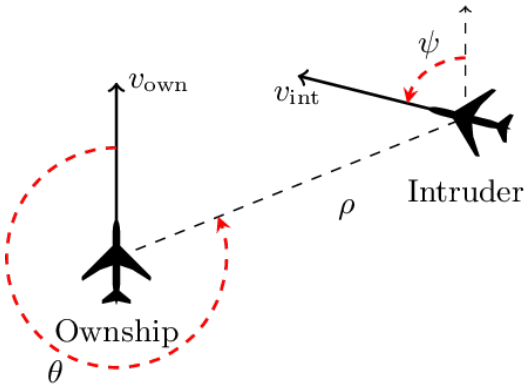
A collision avoidance system for unmanned autonomous aircraft.

Inputs:

- Distance to intruder, ρ
- Angle to intruder, θ
- Intruder heading, φ
- Speed, v_{own}
- Intruder speed, v_{int}

Outputs:

- Clear of conflict
- Strong left
- Weak left
- Weak right
- Strong right



Example 2: ACAS Xu properties

Originally implemented as a 2Gb lookup table but was replaced with a neural network in order to improve size and latency requirements.

Example 2: ACAS Xu properties

Originally implemented as a 2Gb lookup table but was replaced with a neural network in order to improve size and latency requirements.

Properties are derived from those of the original lookup table.

Example 2: ACAS Xu properties

10 different domain-specific properties in total.

Example 2: ACAS Xu properties

10 different domain-specific properties in total.

Definition (ACAS Xu: Property 3)

If the intruder is directly ahead and is moving towards the ownship, the score for COC will not be minimal.

$$1500 \leq \rho \leq 1800 \ \wedge -0.06 \leq \theta \leq 0.06 \ \wedge \psi \geq 3.10 \ \wedge v_{own} \geq 980 \ \wedge v_{int} \geq 960$$
$$\Rightarrow$$
$$\exists a \in \{SL, L, R, SR\}. f(\theta, \rho, \varphi, v_{own}, v_{int})_{COC} < f(\theta, \rho, \varphi, v_{own}, v_{int})_a$$

Property source 3: Domain specific properties

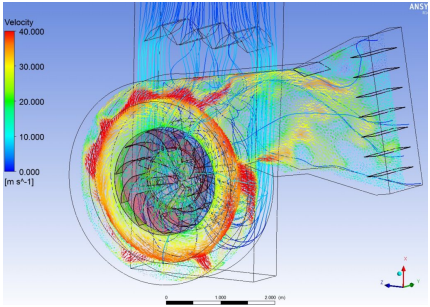
Often there may be properties specific to the domain being modelled.

Example 3: fluid modelling

Consider trying to model a fluid in a tube.

Very difficult to do precisely, but we know:

- energy should be conserved.
- etc.



Current Verifier Landscape

A whole range of domain-specific verifiers exist:

Current Verifier Landscape

A whole range of domain-specific verifiers exist:

- Marabou (SMT technology)

Current Verifier Landscape

A whole range of domain-specific verifiers exist:

- Marabou (SMT technology)
- ERAN (abstract interpretation + MILP)

Current Verifier Landscape

A whole range of domain-specific verifiers exist:

- Marabou (SMT technology)
- ERAN (abstract interpretation + MILP)
- Verisig (interval arithmetic)
- AlphaBetaCROWN (linear bound propagation)
- ...

International Standards and Competitions

https://www.vnnlib.org/

Current Verifier Landscape

A whole range of domain-specific verifiers exist:

- Marabou (SMT technology)
- ERAN (abstract interpretation + MILP)
- Verisig (interval arithmetic)
- AlphaBetaCROWN (linear bound propagation)
- ...

International Standards and Competitions

https://www.vnnlib.org/

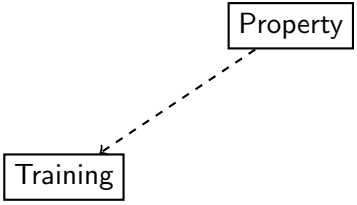
Marabou is our choice as it is complete, and the set of expressible queries is large!

Guy Katz, Clarke Barrett, D. Dill, K. Julian, and M. Kochenderfer. Reluplex: An Efficient SMT Solver for Verifying Deep Neural Networks. In CAV, 2017.

The lifecycle of neural network verification

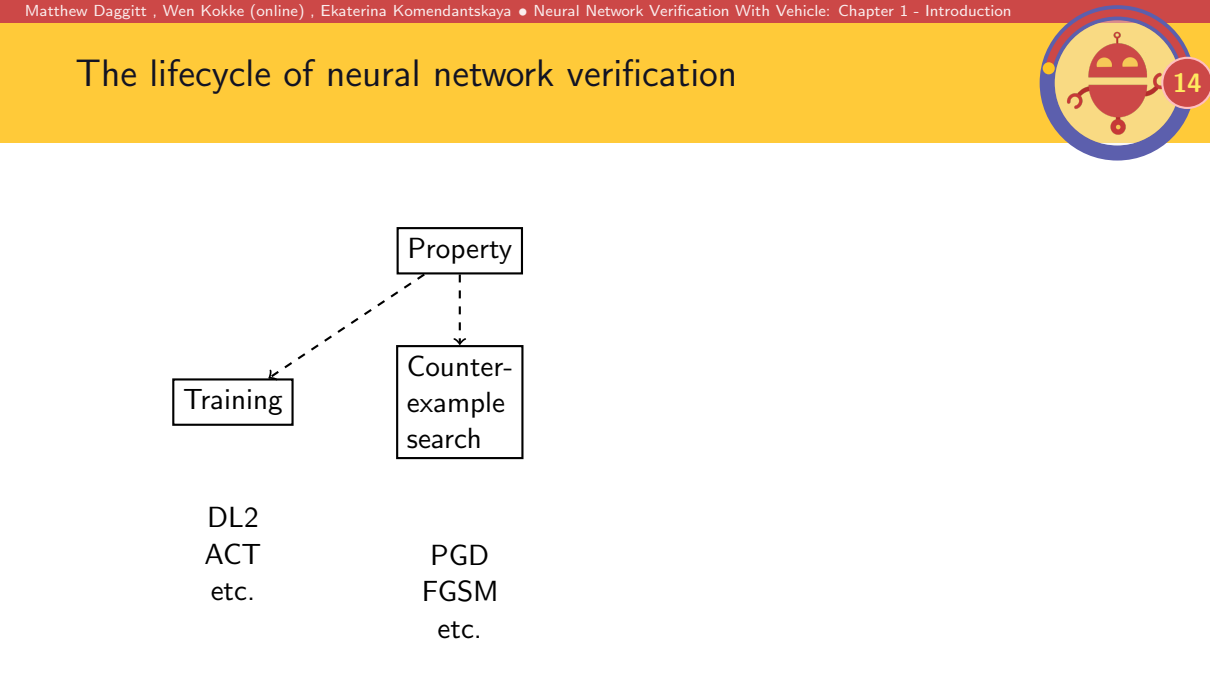
Property

The lifecycle of neural network verification



DL2
ACT
etc.

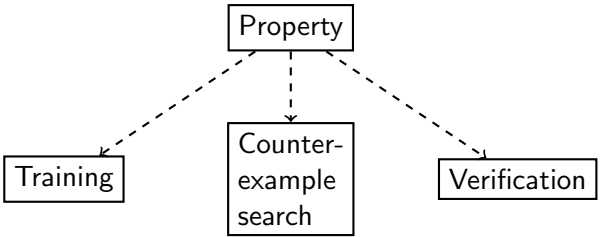
The lifecycle of neural network verification



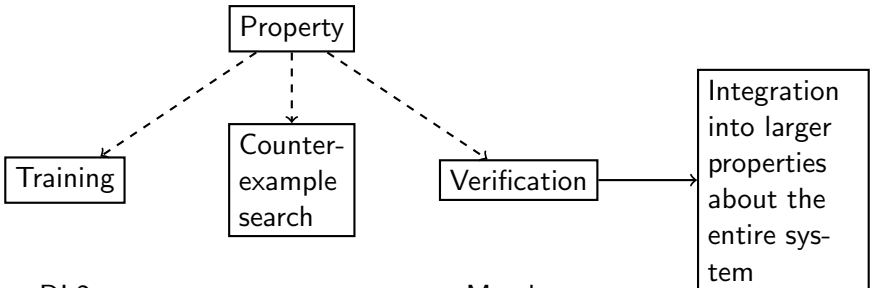
DL2
ACT
etc.

PGD
FGSM
etc.

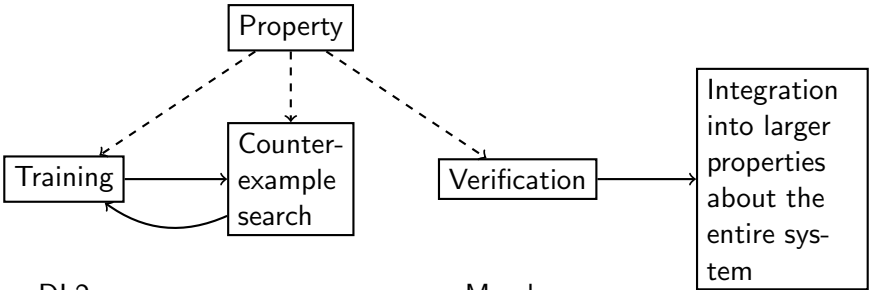
The lifecycle of neural network verification



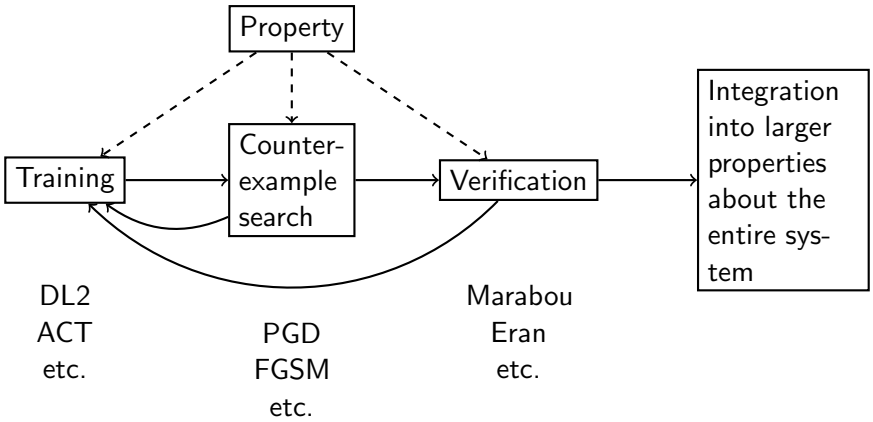
The lifecycle of neural network verification



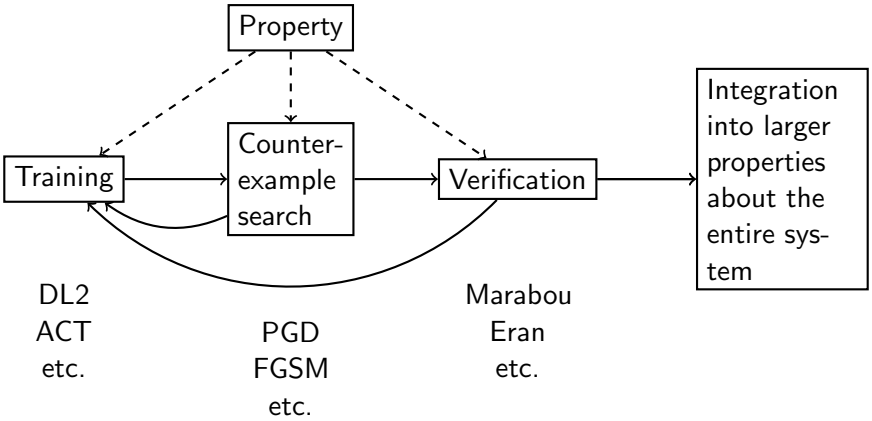
The lifecycle of neural network verification



The lifecycle of neural network verification



The lifecycle of neural network verification



Challenges the area faces

- Theory: finding appropriate verification properties

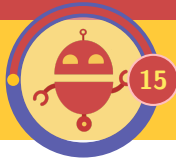
Challenges the area faces

- Theory: finding appropriate verification properties
- Verification: undecidability of non-linear real arithmetic and scalability of neural network verifiers

Challenges the area faces

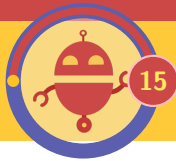
- Theory: finding appropriate verification properties
- Verification: undecidability of non-linear real arithmetic and scalability of neural network verifiers
- Machine-learning: understanding and integrating property-driven training

Challenges the area faces



- Theory: finding appropriate verification properties
- Verification: undecidability of non-linear real arithmetic and scalability of neural network verifiers
- Machine-learning: understanding and integrating property-driven training
- Programming: finding the right languages to support these developments

Challenges the area faces



- Theory: finding appropriate verification properties
- Verification: undecidability of non-linear real arithmetic and scalability of neural network verifiers
- Machine-learning: understanding and integrating property-driven training
- Programming: finding the right languages to support these developments
- Complex systems: integration of neural network verification into complex systems

Some of these problems are aggravated by insufficient programming language or API support.

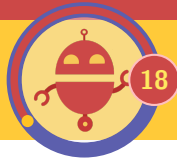
Lets look under the hood...

Training framework: DL2


```
126 class RobustnessConstraint(Constraint):
127
141     def get_domains(self, x_batches, y_batches):
142         assert len(x_batches) == 1
143         n_batch = x_batches[0].size()[0]
144
145         return [[Box(np.clip(x_batches[0][i].cpu().numpy() - self.eps, 0, 1),
146                      np.clip(x_batches[0][i].cpu().numpy() + self.eps, 0, 1))
147                  for i in range(n_batch)]]
148
149     def get_condition(self, z_inp, z_out, x_batches, y_batches):
150         n_batch = x_batches[0].size()[0]
151         z_out = transform_network_output(z_out, self.network_output)[0]
152         #z_logits = F.log_softmax(z_out[0], dim=1)
153
154         pred = z_out[np.arange(n_batch), y_batches[0]]
155
156         limit = torch.FloatTensor([0.3])
157         if self.use_cuda:
158             limit = limit.cuda()
159         return dl2.GEQ(pred, torch.log(limit))
```

Fischer, M., Balunovic, M., Drachsler-Cohen, D., Gehr, T., Zhang, C., and Vechev, M. T. DL2: training and querying neural networks with logic. In Proc. of the 36th Int. Conf. Machine Learning, ICML 2019

Training framework: ART




```
333     @classmethod
334     def property6a(cls, dom: AbsDom):
335         p = AcasProp(name='property6a', dom=dom, safe_fn='cols_is_min', viol_fn='cols_not_min',
336                      fn_args=[AcasOut.CLEAR_OF_CONFLICT])
337         p.set_input_bound(AcasIn.RHO, new_low=12000, new_high=62000)
338         p.set_input_bound(AcasIn.THETA, new_low=0.7, new_high=3.141592)
339         p.set_input_bound(AcasIn.PSI, new_low=-3.141592, new_high=-3.141592 + 0.005)
340         p.set_input_bound(AcasIn.V_OWN, new_low=100, new_high=1200)
341         p.set_input_bound(AcasIn.V_INT, new_low=0, new_high=1200)
342         p.set_all_applicable_as(False)
343         p.set_applicable(1, 1, True)
344         return p
```

Lin, X., Zhu, H., Samanta, R., and Jagannathan, S. (2020). Art: Abstraction refinement-guided training for provably correct neural networks. In FMCAD 2020

Verification framework: Marabou


```
def test_acas_1_1_normalize():
    """
    Test the 1,1 experimental ACAS Xu network.
    By passing "normalize=true" to read_nnet, Marabou adjusts the parameters of the first and last layers of the
      network to incorporate the normalization.
    As a result, properties can be defined in the original input/output spaces without any manual normalization.
    """
    filename =  "acasxu/ACASXU_experimental_v2a_1_1.nnet"
    testInputs = [
        [1000.0, 0.0, -1.5, 100.0, 100.0],
        [10000.0, -3.0, -1.5, 300.0, 300.0],
        [5000.0, -3.0, 0.0, 300.0, 600.0]
    ]
    testOutputs = [
        [177.87553729, 173.75796115, 193.05920806, 153.07876146, 195.00495022],
        [-0.55188079, 0.46863711, 0.44250383, 0.44151988, 0.43959133],
        [29.9190734, 27.2386958, 45.02497222, 14.5610455, 46.86448056]
    ]
    network = evaluateFile(filename, testInputs, testOutputs, normalize = True)
```

Katz, G., Huang, D. A., Ibeling, D., Julian, K., Lazarus, C., Lim, R., Shah, P., Thakoor, S., Wu, H., Zeljic, A., Dill, D. L., Kochenderfer, M. J., and Barrett, C. W. (2019). The Marabou framework for verification and analysis of deep neural networks. In CAV 2019

Verification framework: ERAN


```
[12000, 62000]
[0.7, 3.141592][-3.141592, -0.7]
[-3.141592, -3.136592]
[100, 1200]
[0, 600]
```

```
5
y0 min
```

Singh, G., Gehr, T., Püschel, M., and Vechev, M. T. (2019). An abstract domain for certifying neural networks. PACMPL, 3(POPL):41:1–41:30.

Verification property language: VNNLIB


```
(assert (or
    (and (<= X_0 0.700434925) (>= X_0 -0.129289109)
         (<= X_1 0.499999896) (>= X_1 0.11140846)
         (<= X_2 -0.499204121) (>= X_2 -0.499999896)
         (<= X_3 0.5) (>= X_3 -0.5)
         (<= X_4 0.5) (>= X_4 -0.5))
    (and (<= X_0 0.700434925) (>= X_0 -0.129289109)
         (<= X_1 -0.11140846) (>= X_1 -0.499999896)
         (<= X_2 -0.499204121) (>= X_2 -0.499999896)
         (<= X_3 0.5) (>= X_3 -0.5)
         (<= X_4 0.5) (>= X_4 -0.5))
))

; unsafe if coc is not minimal
(assert (or
    (and (<= Y_1 Y_0))
    (and (<= Y_2 Y_0))
    (and (<= Y_3 Y_0))
    (and (<= Y_4 Y_0))
```

Recap: What are the problems from the PL perspective?

Recap: What are the problems from the PL perspective?

- I^O Interoperability – properties are not portable between training/counter-example search/ verification.

Recap: What are the problems from the PL perspective?

I^O Interoperability – properties are not portable between training/counter-example search/ verification.

I^P Interpretability – code is not easy to understand.

Recap: What are the problems from the PL perspective?

- I^O Interoperability – properties are not portable between training/counter-example search/ verification.
- I^P Interpretability – code is not easy to understand.
- $I^{\int}$ Integration – properties of networks cannot be linked to larger control system properties.

Recap: What are the problems from the PL perspective?

I^O Interoperability – properties are not portable between training/counter-example search/ verification.

I^P Interpretability – code is not easy to understand.

$I^\int$ Integration – properties of networks cannot be linked to larger control system properties.

E^G Embedding gap – little support for translation between problem space (as in original spec) and input space (at neural network level).

Recap: What are the problems from the PL perspective?

I^O Interoperability – properties are not portable between training/counter-example search/ verification.

I^P Interpretability – code is not easy to understand.

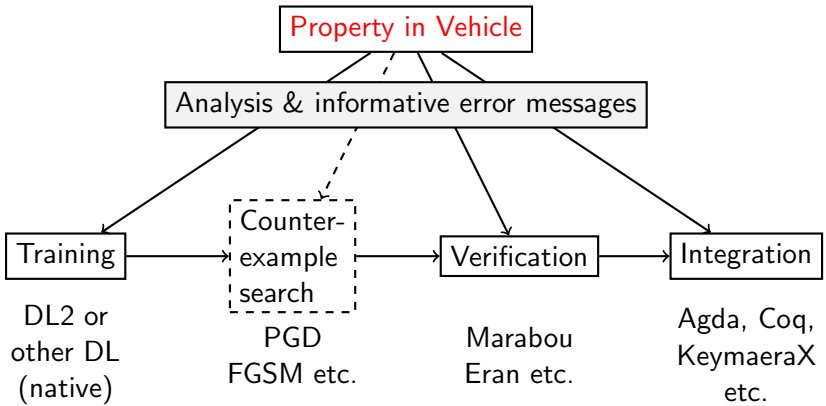
$I^\int$ Integration – properties of networks cannot be linked to larger control system properties.

E^G Embedding gap – little support for translation between problem space (as in original spec) and input space (at neural network level).

Vehicle is designed to address all of these problems

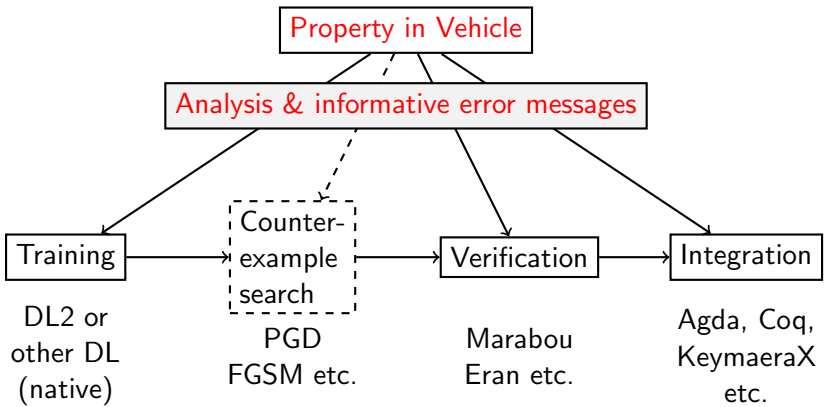
Vehicle ...

Vehicle is a domain-specific functional language for writing high-level property specifications for neural networks



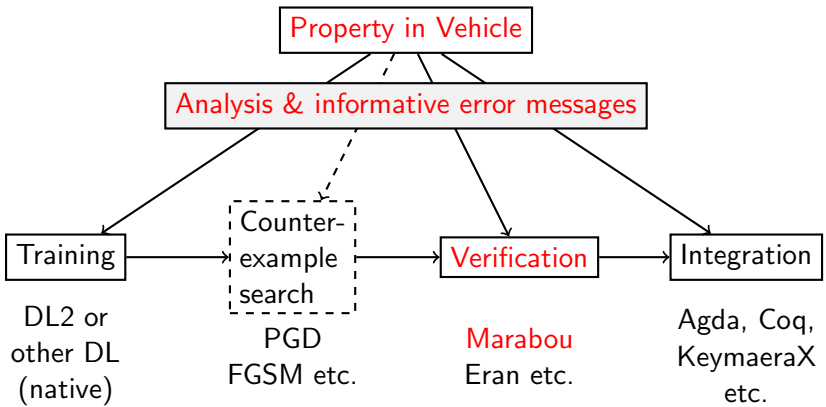
Vehicle ...

Vehicle is a domain-specific functional language for writing high-level property specifications for neural networks



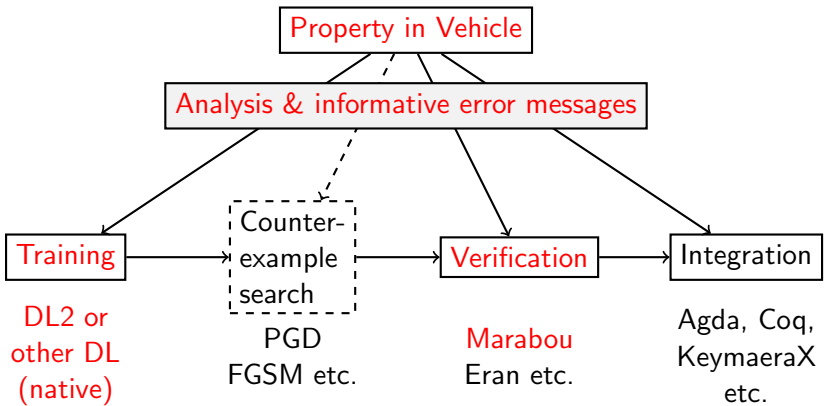
Vehicle ...

Vehicle is a domain-specific functional language for writing high-level property specifications for neural networks



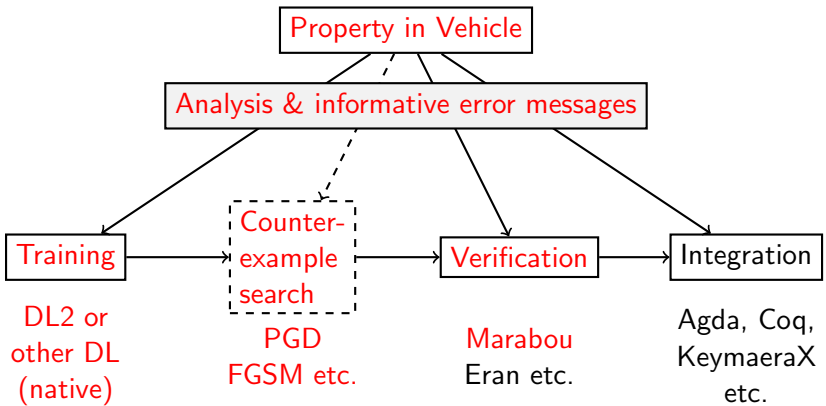
Vehicle ...

Vehicle is a domain-specific functional language for writing high-level property specifications for neural networks



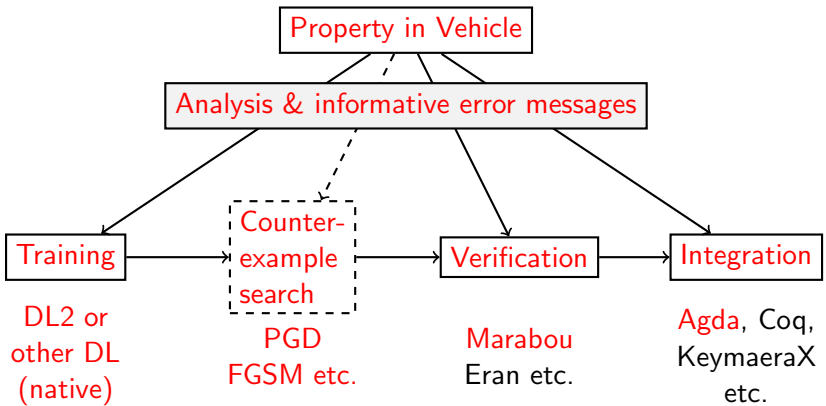
Vehicle ...

Vehicle is a domain-specific functional language for writing high-level property specifications for neural networks



Vehicle ...

Vehicle is a domain-specific functional language for writing high-level property specifications for neural networks



Other Similar APIs

- Socrates [Python]: Given a JSON spec and a network calls a variety of verifiers.

  L. H. Pham, J. Li, and J. Sun. 2020. SOCRATES: Towards a Unified Platform for Neural Network Verification. CoRR abs/2007.11206.

Unresolved: I^O, I^P, $I^{\int}$, E^G

Other Similar APIs

- Socrates [Python]: Given a JSON spec and a network calls a variety of verifiers.

  L. H. Pham, J. Li, and J. Sun. 2020. SOCRATES: Towards a Unified Platform for Neural Network Verification. CoRR abs/2007.11206.

  Unresolved: I^O, I^P, $I^\int$, E^G

- NeVer 2.0 [Python]: added training, pruning and quantisation to this functionality.

  D. Guidotti, L. Pulina, and A. Tacchella. 2020. NeVer 2.0: Learning, Verification and Repair of Deep Neural Networks. CoRR abs/2011.09933.

  Resolved: I^O (partially). Unresolved: I^P, $I^\int$, E^G.

Other Similar APIs

- CoCoNet [Python]: NN format converter with GUI

  D. Guidotti, A. Tacchella, L. Pulina, S. Demarchi 2023.
  http://neuralverification.org/

  Resolved: I^P (partially). Unresolved: I^O, $I^{\int}$, E^G

Other Similar APIs

- CoCoNet [Python]: NN format converter with GUI

  D. Guidotti, A. Tacchella, L. Pulina, S. Demarchi 2023. http://neuralverification.org/

  Resolved: I^P (partially). Unresolved: I^O, $I^\int$, E^G

- Caisar [OCAML] – Why3 specification language, connected to several verifiers.

  J. Girard-Satabin, M. Alberti, F. Bobot, Z. Chihani, and A. Lemesle. 2022. CAISAR: A platform for Characterizing Artificial Intelligence Safety and Robustness. In AISafety.

  Resolved: I^P, $I^\int$. Unresolved: I^O, E^G.

Vehicle's Aim...

... is to resolve the problems I^O, I^P, $I^\int$, E^G

Vehicle's Aim...

... is to resolve the problems I^O, I^P, $I^\int$, E^G

... and support community's effort towards resolution of the "Grand Challenges":

Vehicle's Aim...

... is to resolve the problems I^O, I^P, $I^\int$, E^G

... and support community's effort towards resolution of the "Grand Challenges":

- Theory: finding appropriate verification properties
- Verification: undecidability of non-linear real arithmetic and scalability of neural network verifiers
- Machine-learning: understanding and integrating property-driven training
- Programming: finding the right languages to support these developments
- Complex systems: integration of neural net verification into complex systems

Sources


M. Daggitt, R. Atkey, W. Kokke, E. Komendantskaya, L. Arnaboldi: Compiling Higher-Order Specifications to SMT Solvers: How to Deal with Rejection Constructively. CPP 2023


N. Slusarz, E. Komendantskaya, M. Daggitt, R. Stewart, K. Stark: Logic of Differentiable Logics: Towards a Uniform Semantics of DL. LPAR 2023.


Matthew L. Daggitt, Wen Kokke, Robert Atkey, Luca Arnaboldi, Ekaterina Komendantskaya: Vehicle: Interfacing Neural Network Verifiers with Interactive Theorem Provers. FOMLAS 2022


Vehicle Team: The Vehicle language: `https://github.com/vehicle-lang` 2023.


M.Daggitt and W.Kokke: Vehicle User Manual. `https://vehicle-lang.readthedocs.io` 2023.


The Vehicle team: Vehicle Tutorial `https://vehicle-lang.github.io`. 2023. Tutorial code repository `https://github.com/vehicle-lang/vehicle-tutorial`.

Purpose of this Tutorial...

- Discuss challenges in NN verification ("grand" and technical)
- Understand how logic and PL semantics can contribute in this domain
- Introduce **Vehicle** specification language at the user level
- Give you a bit of practice with NN verification and gather feedback

Plan for the rest of this tutorial

- Before coffee break:
  - Brief introduction to **Vehicle** specification language
  - Neural Network Robustness: an iconic verification case
- During and after the break:
  - Exercise session: write and verify a **Vehicle** spec
    Tutorial pages: `https://vehicle-lang.github.io`
    Join tutorial Slack channel via the tutorial page, to ask questions
- After the break:
  - Property-driven training in Vehicle
  - Large system integration
  - Application areas for NN Verification

Setup

- Download the tutorial repository:

```
git clone https://github.com/vehicle-lang/tutorial
```

- Install Vehicle:

```
pip install vehicle-lang
```

- Install Marabou verifier:

```
pip install marabou
```