Whats and Whys of Neural Network Verification

Workshop on Safe ML, HWU, Scotland

Ekaterina Komendantskaya

Heriot-Watt University and University of Southampton

The Vehicle /LAIV Team

Matthew Daggitt

Wen Kokke

Bob Atkey

Katya Komendantskaya

Natalia Slusarz

Luca Arnaboldi

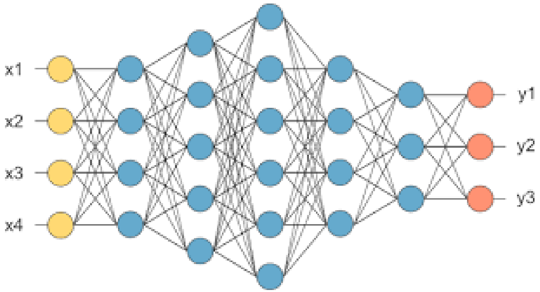
Marco Casadio

Rob Stewart

Remi Desmartin

Kathrin Stark

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

Can we verify properties of neural network-enhanced software?

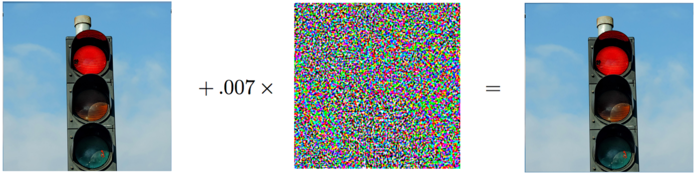
Can we verify properties of neural network-enhanced software?

1. Can we state the properties?

Property source 1: Universal properties

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

Example 1: Adversarial robustness



"red" — 97.6% confidence

noise

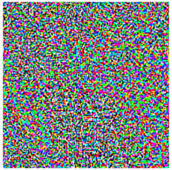
"green" — 81.2% confidence

Example 1: Adversarial robustness

"red"

97.6% confidence

$+ .007 \times$



noise

$=$

"green"

81.2% confidence

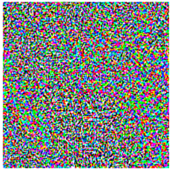
- the perturbations are imperceptible to human eye

Example 1: Adversarial robustness

"red"

97.6% confidence

$+ .007 \times$



noise

$=$

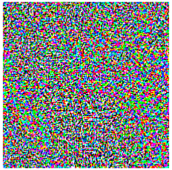
"green"

81.2% confidence

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

Example 1: Adversarial robustness

$+ .007 \times$



$=$

"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 $f \in \mathbb{R}^m \rightarrow \mathbb{R}^n$ 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

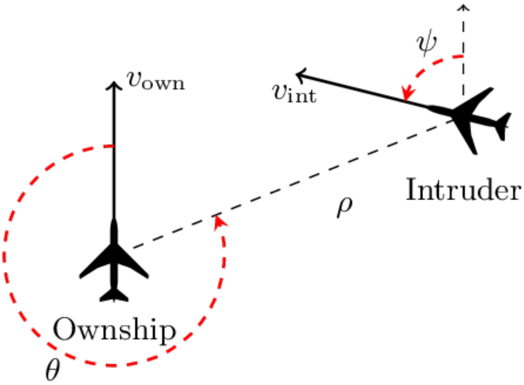
Sometimes neural networks are used to approximate "standard" algorithms.

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.

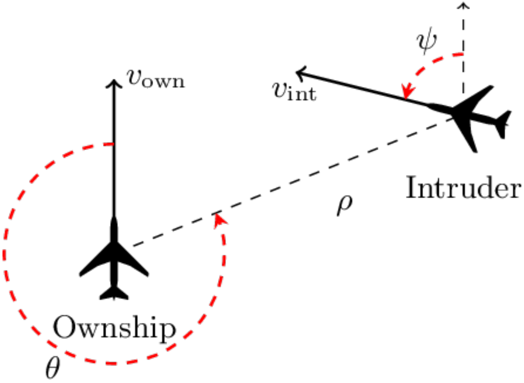


Example 2: ACAS Xu

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

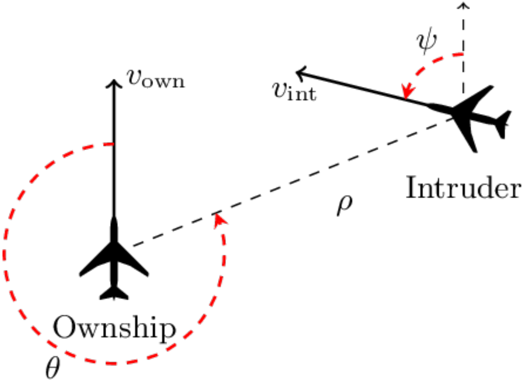
A collision avoidance system for unmanned autonomous aircraft.

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

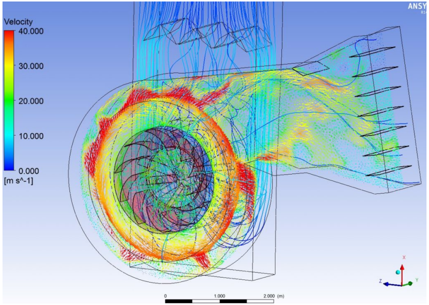
Often there may be properties specific to the domain being modelled.

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.



The lifecycle of neural network verification

$\boxed{\text{Property}}$

Challenges the area faces

- Theory: finding appropriate verification properties

Challenges the area faces

- Theory: finding appropriate verification properties

Challenges the area faces

- Theory: finding appropriate verification properties

A lot has to be done to even:

- … understand the properties we do have

M. Casadio, E. Komendantskaya, M. Daggitt, W. Kokke, G. Katz, G. Amir, I. Refaeli: Neural Network Robustness as a Verification Property: A Principled Case Study. CAV (1) 2022: 219-231

- … or specifying/programming them in a clear way

M. Daggitt, W. Kokke, E. Komendantskaya, R. Atkey, L. Arnaboldi, N. Slusarz, M. Casadio, B. Coke, and J. Lee. The Vehicle Tutorial: Neural Network Verification with Vehicle. The 6th Workshop on Formal Methods for ML-Enabled Autonomous Systems (FoMLAS'23).

Can we verify properties of neural network-enhanced software?

2. What sort of verification algorithms exist?

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/

The lifecycle of neural network verification

Property

Verification

Marabou
Eran
etc.

Challenges the area faces

- Theory: finding appropriate verification properties
- Verification: undecidability of non-linear real arithmetic and scalability of neural network verifiers; proof certificates and soundness.

Challenges the area faces

- Theory: finding appropriate verification properties
- Verification: undecidability of non-linear real arithmetic and scalability of neural network verifiers; proof certificates and soundness.

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

R. Desmartin, O. Isac, G. Passmore, K. Stark, E. Komendantskaya, G. Katz: Towards a Certified Proof Checker for Deep Neural Network Verification. LOPSTR 2023: 198-209

Can we verify properties of neural network-enhanced software?

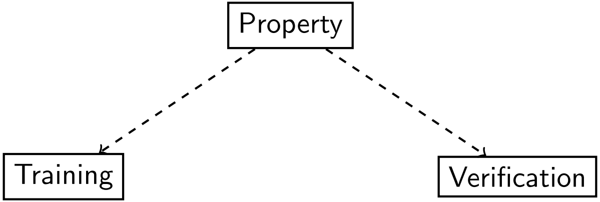
3. How do we ensure neural networks actually satisfy the properties we want?

The lifecycle of neural network verification

$$\boxed{\text{Property}} \dashrightarrow \boxed{\text{Verification}}$$

Marabou
Eran
etc.

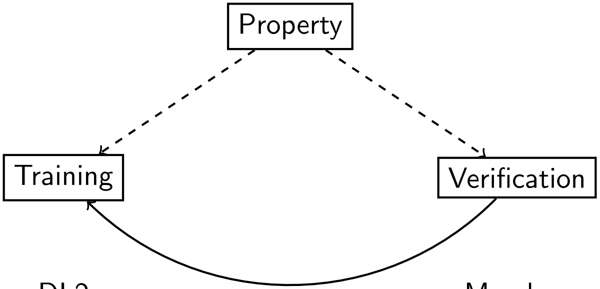
The lifecycle of neural network verification



DL2
ACT
etc.

Marabou
Eran
etc.

The lifecycle of neural network verification



DL2
ACT
etc.

Marabou
Eran
etc.

Challenges the area faces

- Theory: finding appropriate verification properties
- Verification: undecidability of non-linear real arithmetic and scalability of neural network verifiers; proof certificates and soundness
- 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; proof certificates and soundness
- Machine-learning: understanding and integrating property-driven training
  - General way to convert logical properties into loss functions: Differentiable logics

N. Slusarz, E. Komendantskaya, M. Daggitt, R. Stewart, K. Stark: Logic of Differentiable Logics: Towards a Uniform Semantics of DL. LPAR 2023: 473-493

Challenges the area faces

- Theory: finding appropriate verification properties
- Verification: undecidability of non-linear real arithmetic and scalability of neural network verifiers; proof certificates and soundness
- Machine-learning: understanding and integrating property-driven training
  - General way to convert logical properties into loss functions: Differentiable logics

    N. Slusarz, E. Komendantskaya, M. Daggitt, R. Stewart, K. Stark: Logic of Differentiable Logics: Towards a Uniform Semantics of DL. LPAR 2023: 473-493
  - How to handle universal quantifiers?

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

Challenges the area faces

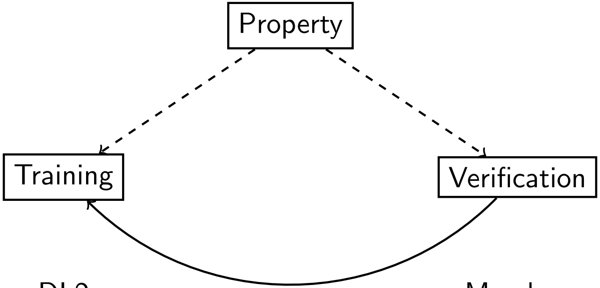
- Theory: finding appropriate verification properties
- Verification: undecidability of non-linear real arithmetic and scalability of neural network verifiers; proof certificates and soundness
- Machine-learning: understanding and integrating property-driven training
  - General way to convert logical properties into loss functions: Differentiable logics

    N. Slusarz, E. Komendantskaya, M. Daggitt, R. Stewart, K. Stark: Logic of Differentiable Logics: Towards a Uniform Semantics of DL. LPAR 2023: 473-493
  - How to handle universal quantifiers?

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

D. Kientiz: The Influence of Geometric Properties of Data Distributions on Artificial Neural Networks. PhD Thesis, Heriot-Watt University, 2023

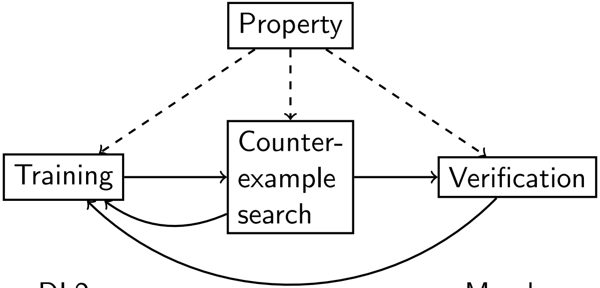
The lifecycle of neural network verification



DL2
ACT
etc.

Marabou
Eran
etc.

The lifecycle of neural network verification



DL2
ACT
etc.

PGD
FGSM
etc.

Marabou
Eran
etc.

Can we verify properties of neural network-enhanced software?

1. Can we state the properties?

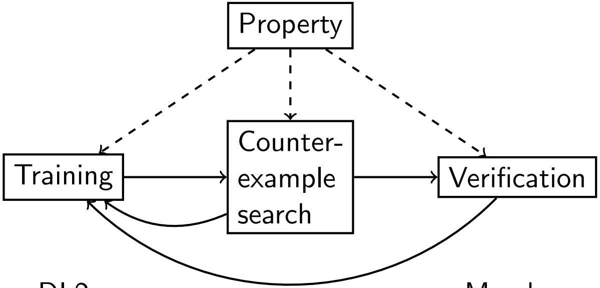
2. What sort of verification algorithms exist?

3. How do we ensure neural networks actually satisfy the properties we want?

Can we verify properties of neural network-enhanced software?

1. Can we state the properties?
2. What sort of verification algorithms exist?
3. How do we ensure neural networks actually satisfy the properties we want?
4. How do we verify complex systems that contain neural nets?

The lifecycle of neural network verification

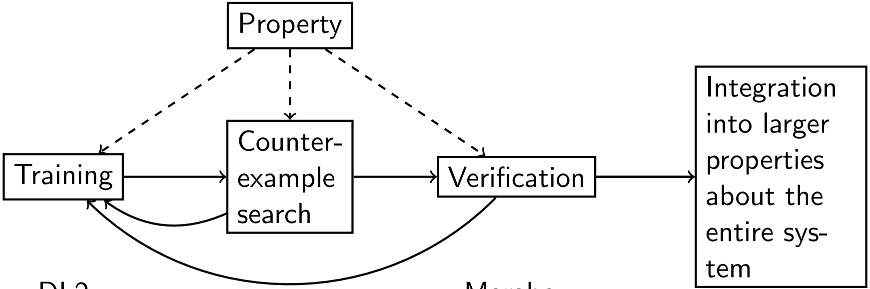


DL2
ACT
etc.

PGD
FGSM
etc.

Marabou
Eran
etc.

The lifecycle of neural network verification



DL2
ACT
etc.

PGD
FGSM
etc.

Marabou
Eran
etc.

Challenges the area faces

- Theory: finding appropriate verification properties
- Verification: undecidability of non-linear real arithmetic and scalability of neural network verifiers; proof certificates and soundness
- Machine-learning: understanding and integrating property-driven training
- Complex systems: integration of neural net verification into complex systems

Challenges the area faces

- Theory: finding appropriate verification properties
- Verification: undecidability of non-linear real arithmetic and scalability of neural network verifiers; proof certificates and soundness
- Machine-learning: understanding and integrating property-driven training
- Complex systems: integration of neural net verification into complex systems
  - Largely intersects with the area of Cyber-Physical Systems (CPS) verification
  - Probably the next "Grand Challenge" for the area
    - Work by NASA and Boeing researchers:

      Corina S. Pasareanu, Ravi Mangal, Divya Gopinath, Huafeng Yu: Assumption Generation for Learning-Enabled Autonomous Systems. RV 2023: 3-22
    - Our humble example

      M. Daggitt, W. Kokke, E. Komendantskaya, R. Atkey, L. Arnaboldi, N. Slusarz, M. Casadio, B. Coke, and J. Lee. The Vehicle Tutorial: Neural Network Verification with Vehicle. `https://vehicle-lang.github.io/tutorial/`

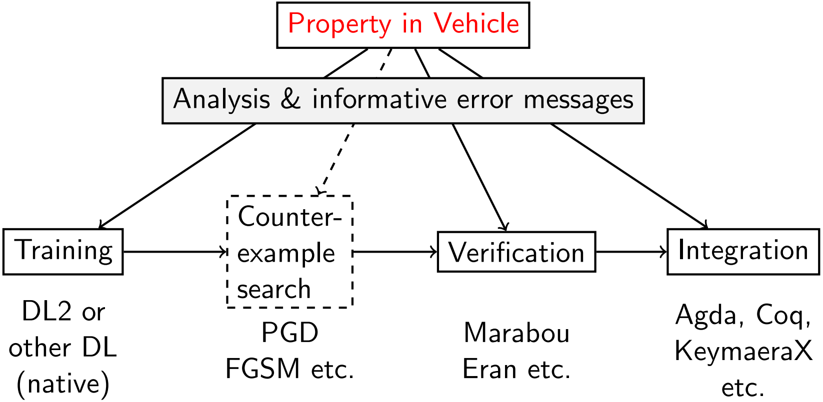
Can we verify properties of neural network-enhanced software?

1. Can we state the properties?
2. What sort of verification algorithms exist?
3. How do we ensure neural networks actually satisfy the properties we want?
4. How do we verify complex systems that contain neural nets?

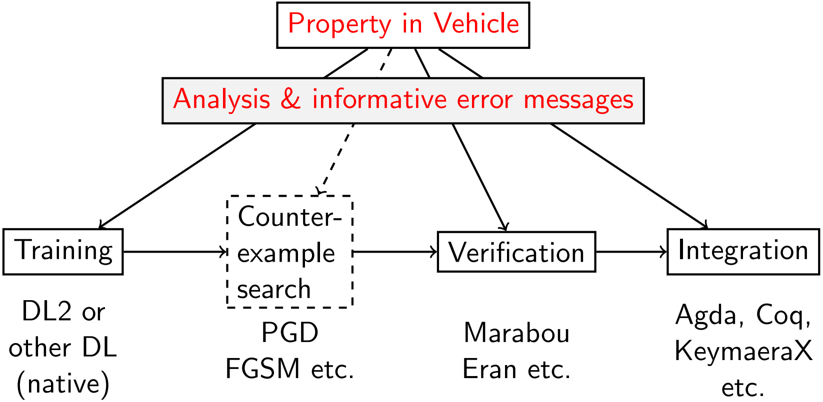
Can we verify properties of neural network-enhanced software?

1. Can we state the properties?
2. What sort of verification algorithms exist?
3. How do we ensure neural networks actually satisfy the properties we want?
4. How do we verify complex systems that contain neural nets?
5. And how to hold all of this together?

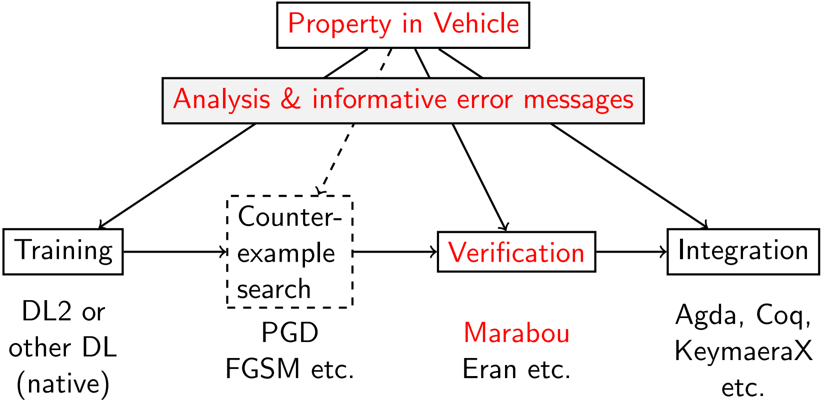
Programming Language Support



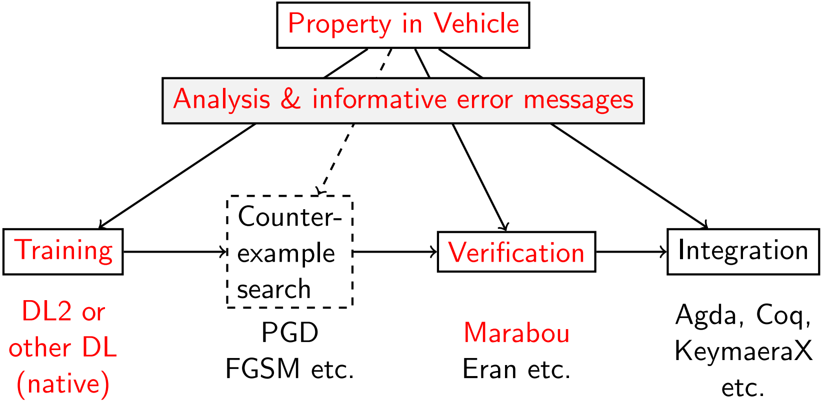
Programming Language Support



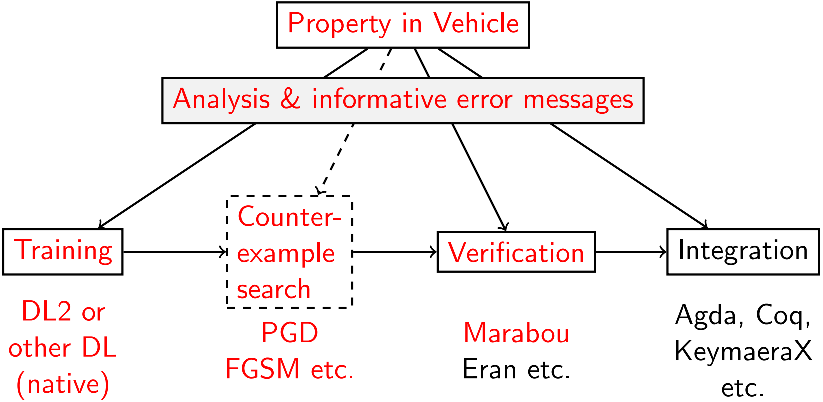
Programming Language Support



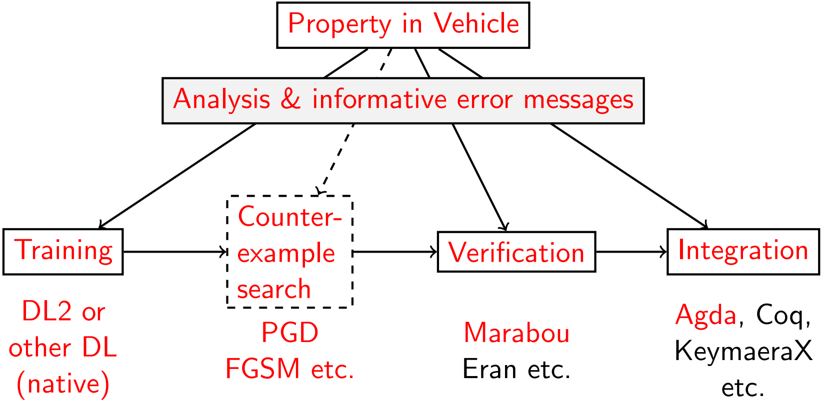
Programming Language Support



Programming Language Support



Programming Language Support



Challenges the area faces

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

Complexity is Good! (Thanks for your Attention)