Towards Verified Artificial Intelligence

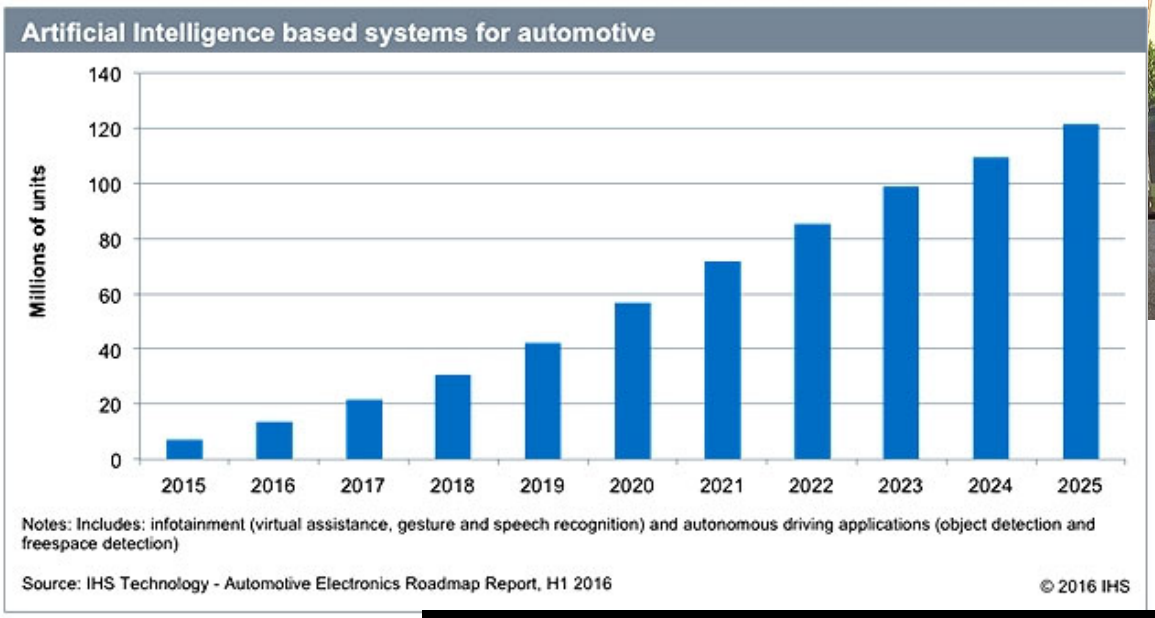
Sanjit A. Seshia

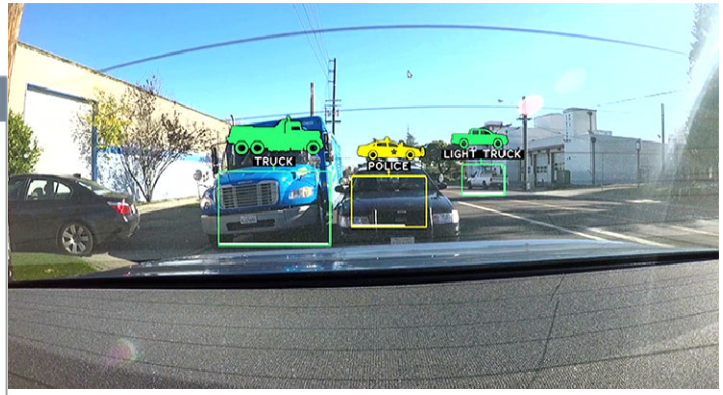
UC Berkeley

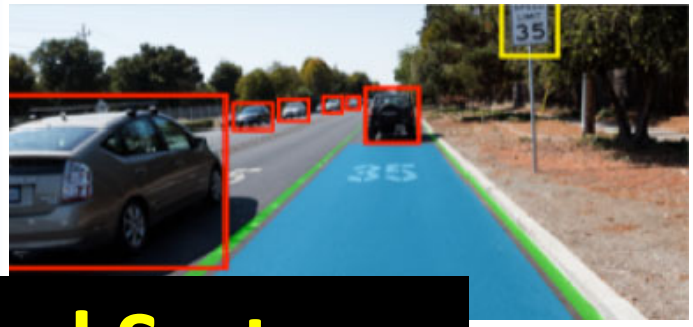
EECS 219C
April 24, 2019

VeriCal

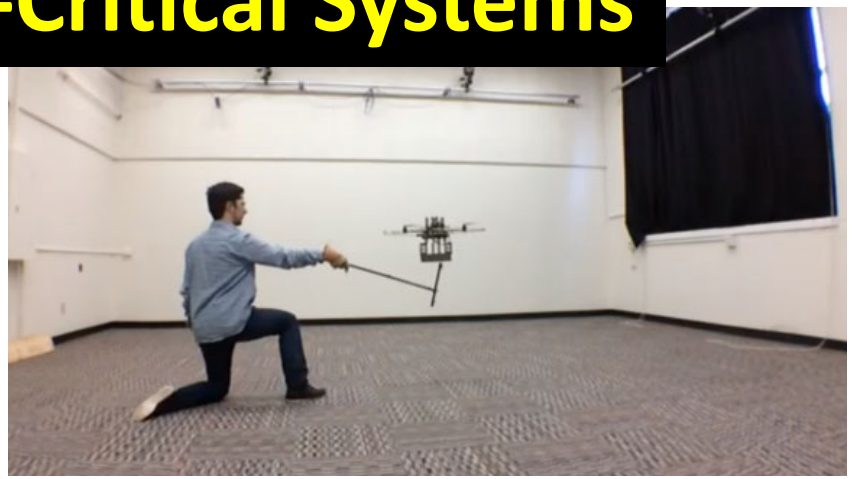
Growing Use of Machine Learning/AI in Cyber-Physical Systems

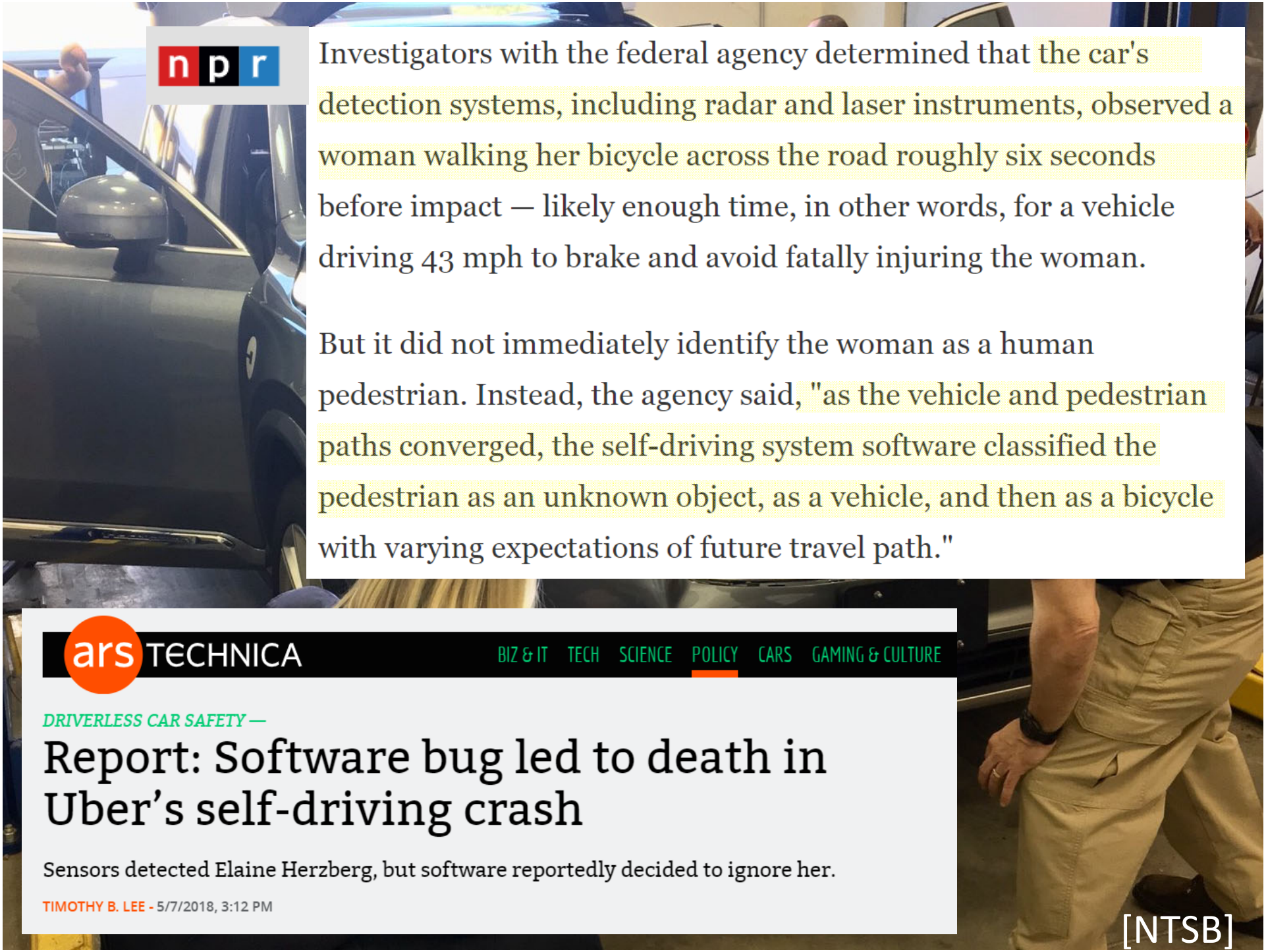






Many Safety-Critical Systems





npr

Investigators with the federal agency determined that the car's detection systems, including radar and laser instruments, observed a woman walking her bicycle across the road roughly six seconds before impact — likely enough time, in other words, for a vehicle driving 43 mph to brake and avoid fatally injuring the woman.

But it did not immediately identify the woman as a human pedestrian. Instead, the agency said, "as the vehicle and pedestrian paths converged, the self-driving system software classified the pedestrian as an unknown object, as a vehicle, and then as a bicycle with varying expectations of future travel path."

ars TECHNICA

BIZ & IT TECH SCIENCE POLICY CARS GAMING & CULTURE

DRIVERLESS CAR SAFETY —

Report: Software bug led to death in Uber's self-driving crash

Sensors detected Elaine Herzberg, but software reportedly decided to ignore her.

TIMOTHY B. LEE - 5/7/2018, 3:12 PM

[NTSB]

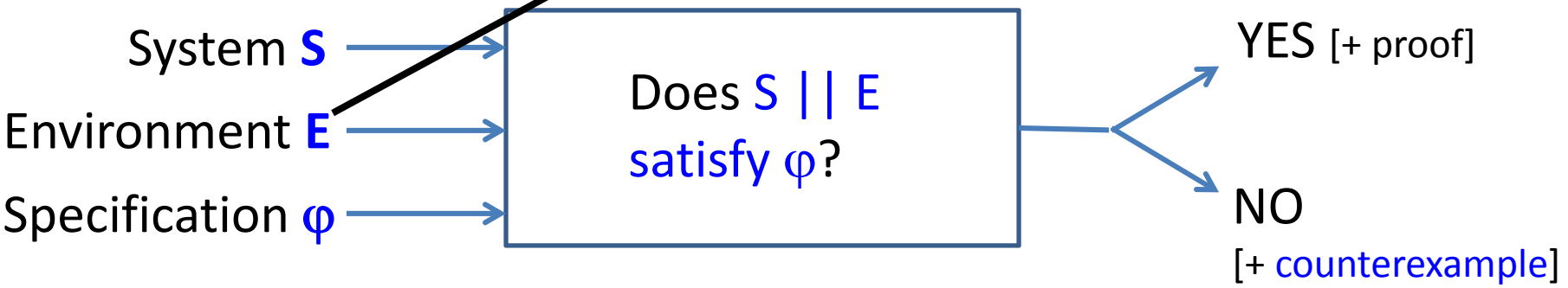
Artificial Intelligence (AI)

Computational Systems that attempt to mimic aspects of human intelligence, including especially the ability to learn from experience.

How do we ensure that AI/ML-based systems are Dependable?

Challenges for Verified AI

S. A. Seshia, D. Sadigh, S. S. Sastry.
Towards Verified Artificial Intelligence. July 2016. https://arxiv.org/abs/1606.08514.



Environment Modeling Challenge – Uncertainty and Unknowns

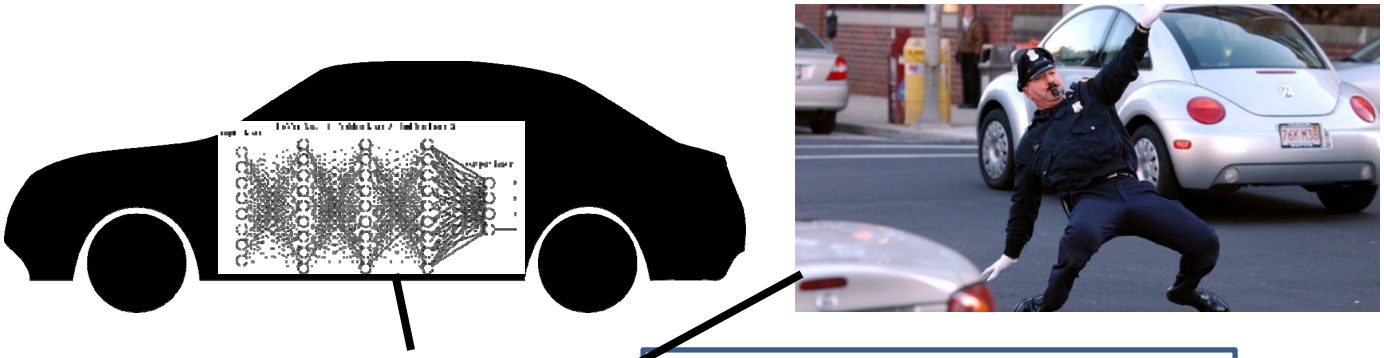
Self-Driving Vehicles: Interact with Humans in Complex Environments; Significant use of machine learning!

Known Unknowns and Unknown Unknowns!!

Cannot represent all possible environment scenarios

Challenges for Verified AI

S. A. Seshia, D. Sadigh, S. S. Sastry.
Towards Verified Artificial Intelligence. July 2016. https://arxiv.org/abs/1606.08514.



System **S**

Environment **E**

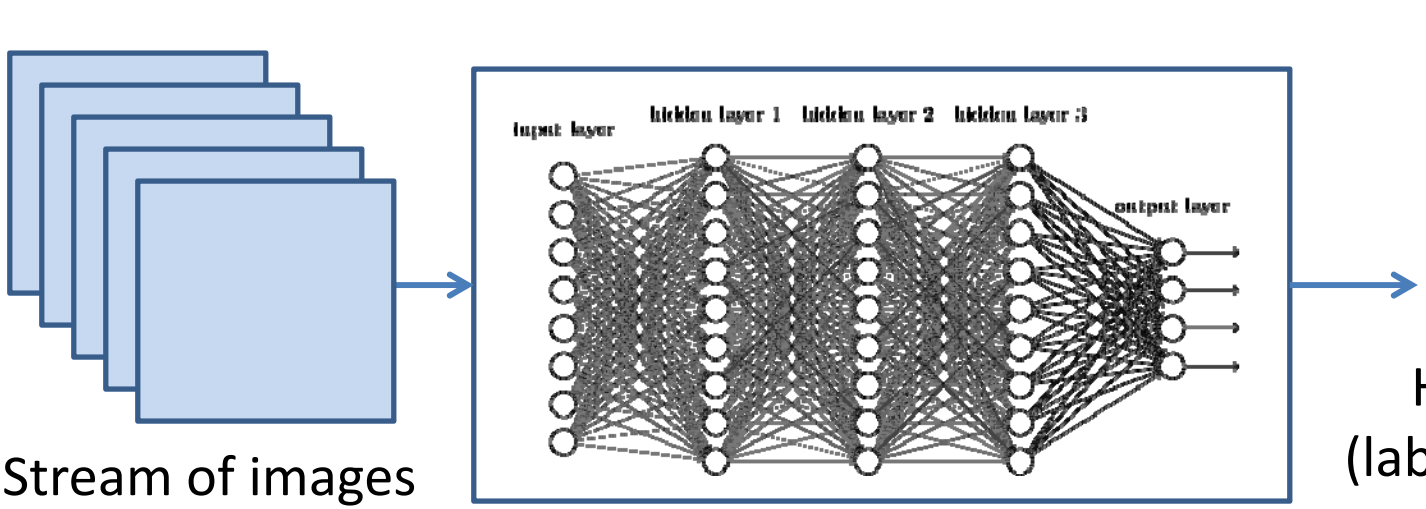
Specification φ

Does S || E satisfy φ?

YES [+ proof]

NO
[+ counterexample]

Modeling Learning Systems with High-Dimensional Input & State Space



Stream of images

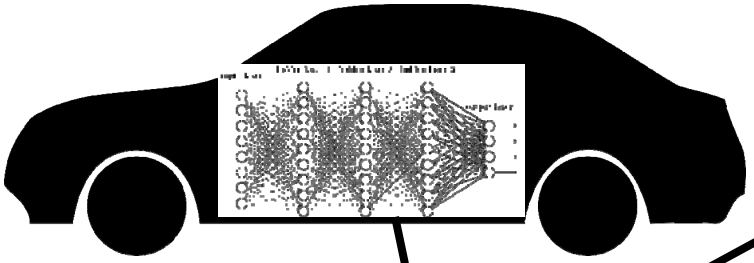
Histogram of (label, confidence)

Input Space: ~10^6 dimensions for single time point
System Parameters: >1M, continuous+discrete

Need New Methods for *Abstraction* and *Modular Reasoning*!

Challenges for Verified AI

S. A. Seshia, D. Sadigh, S. S. Sastry.
Towards Verified Artificial Intelligence. July 2016. https://arxiv.org/abs/1606.08514.



System **S**

Environment **E**

Specification φ

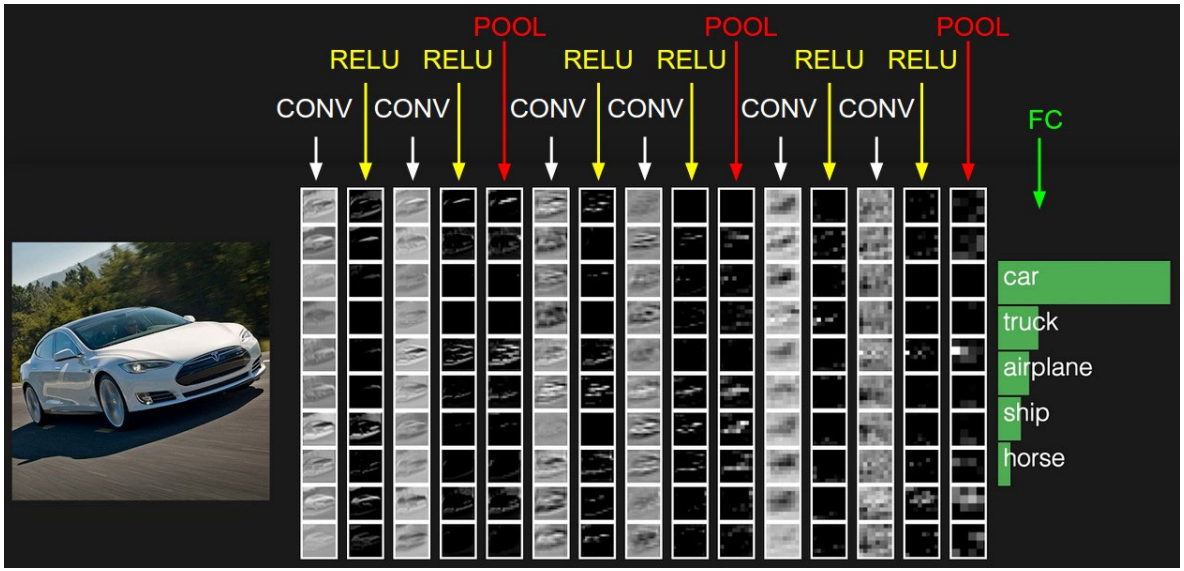
Does S || E satisfy φ?

YES [+ proof]

NO
[+ counterexample]

What's the Specification for Perception Tasks?

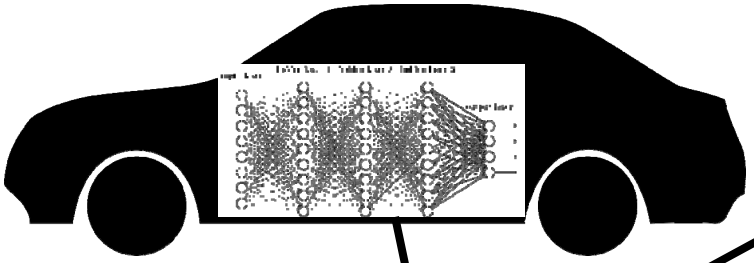
Convolutional Neural Network trained to recognize cars



How do you formally specify "a car"?

Challenges for Verified AI

S. A. Seshia, D. Sadigh, S. S. Sastry.
Towards Verified Artificial Intelligence. July 2016. https://arxiv.org/abs/1606.08514.



System **S**

Environment **E**

Specification φ

Does S || E satisfy φ?

YES [+ proof]

NO
[+ counterexample]

Design Correct-by-Construction instead? How?

Counterexamples, Inputs, etc. from High-Dimensional Signal Spaces

Need Design Principles for Verified AI

Challenges **Principles**

1. Environment (incl. Human) Modeling →
2. Formal Specification →

?

3. Learning Systems Representation →
4. Scalable Training, Testing, Verification →
5. Design for Correctness →

S. A. Seshia, D. Sadigh, S. S. Sastry. *Towards Verified Artificial Intelligence*. July 2016. https://arxiv.org/abs/1606.08514.

Outline

- Challenges for Verified AI

- Design of Closed-Loop Cyber-Physical Systems with Machine Learning Components
  - Specification, Verification, Synthesis
  - Autonomous Vehicles
  - Deep Learning

- Principles for Verified AI
  - Summary of Ideas
  - Future Directions

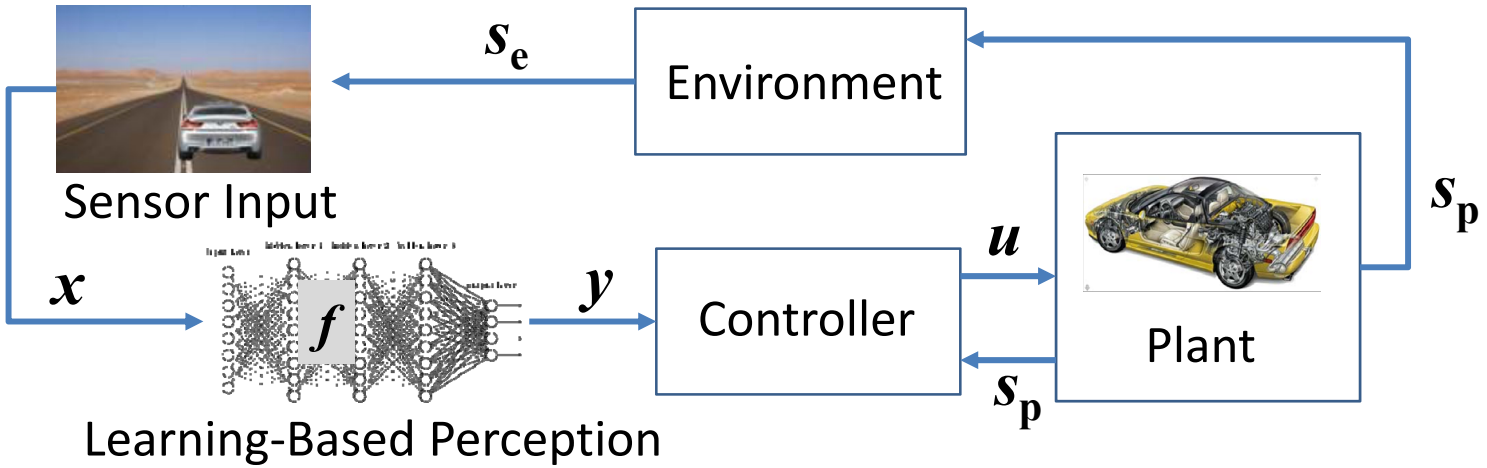
Challenge: Formal Specification

Principle: Start at System Level (i.e. Specify Semantic Behavior of the Overall System)

Falsification of Cyber-Physical Systems with Machine Learning Components

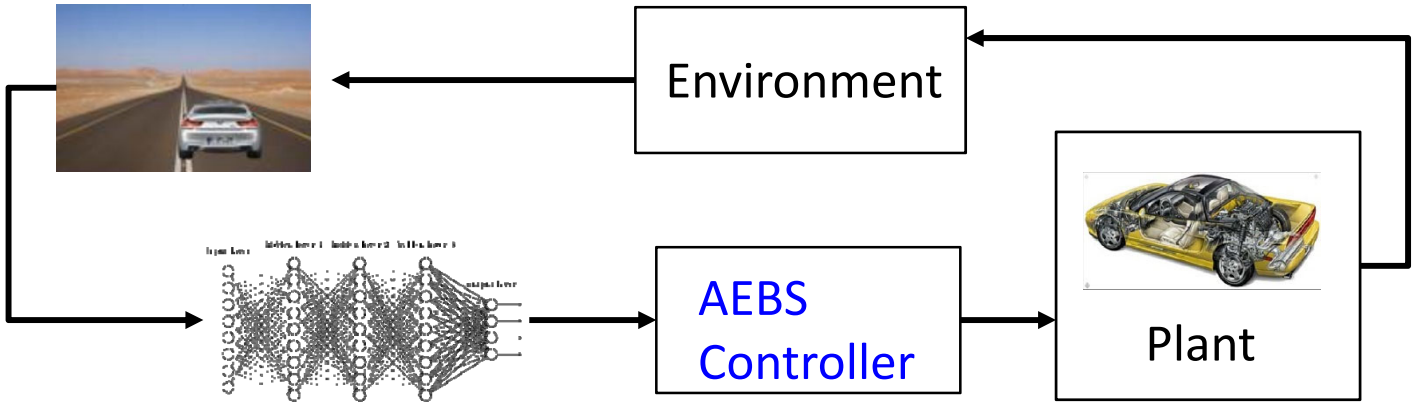
T. Dreossi, A. Donze, and S. A. Seshia. *Compositional Falsification of Cyber-Physical Systems with Machine Learning Components*, In NASA Formal Methods Symposium, May 2017.

Problem: Verify Cyber-Physical System that uses ML-based Perception (CPSML system)



- Initial Focus:
  - Falsification: finding scenarios that violate safety properties
  - Test (Data) Generation: generate "interesting" data for training / testing → improve accuracy
  - Deep Neural Networks, given the increasing interest and use in the automotive context.

Automatic Emergency Braking System (AEBS)



Deep Learning-Based Object Detection

- Goal: Brake when an obstacle is near, to maintain a minimum safety distance
  - Initial: Controller, Plant, Env models in Matlab/Simulink
  - More recent: other driving simulators
- Perception: Object detection/classification system based on deep neural networks
  - Inception-v3, AlexNet, … trained on ImageNet
  - more recent: squeezeDet, Yolo, … trained on KITTI

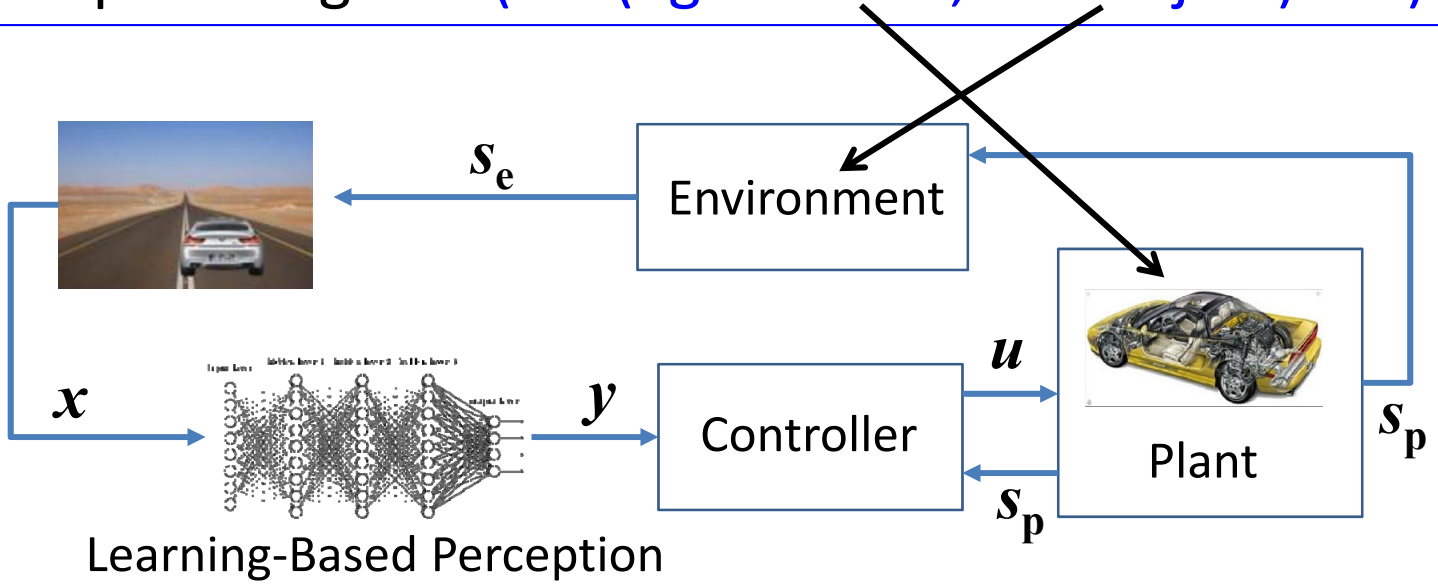
Our Approach: Use a System-Level Specification

✗ "Verify the Deep Neural Network Object Detector"

✓ "Verify the System containing the Deep Neural Network"

Formally Specify the *End-to-End Behavior* of the System

Temporal Logic: **G** (*dist*(ego vehicle, env object) > Δ)



Property does not mention inputs/outputs of the neural network $\varphi(s_p, s_e, u)$

Temporal Logic Requirements

Signal Temporal Logic (STL) [Maler & Nickovic, '04]

Predicates over continuous signals, Propositional Formulas φ ($\wedge$,$\vee$, $\neg$ of the predicates), Temporal Operators ($\boldsymbol{G}, \boldsymbol{F}, \boldsymbol{X}, \boldsymbol{U}$), real-time interval τ.

$\boldsymbol{G}_\tau\ \varphi$	φ is true at all future moments in τ.
$\boldsymbol{F}_\tau\ \varphi$	φ is true in some future moment in τ.
$\varphi_1 \boldsymbol{U}_\tau\ \varphi_2$	φ_1 is true until φ_2 becomes true in τ.

Safety (invariance): Vehicle maintains specified min distance from obstacles.

$$\boldsymbol{G}_{[0,\tau]}\ [\ \text{dist(vehicle, obstacle)} > \Delta\ ]$$

From Logical Formulas to Objective Functions

- STL formula has both
  - Boolean semantics: true/false
  - Quantitative semantics: value in $\mathbb{R}$

- Example:

  $\boldsymbol{G}_{[0,\tau]}$(dist(vehicle, obstacle) > Δ)

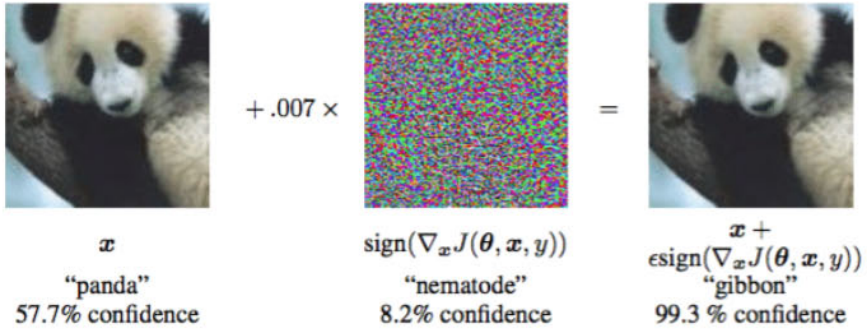
  $\inf_{[0,\tau]}$ [dist(vehicle, obstacle) - Δ]

Specification Spectrum

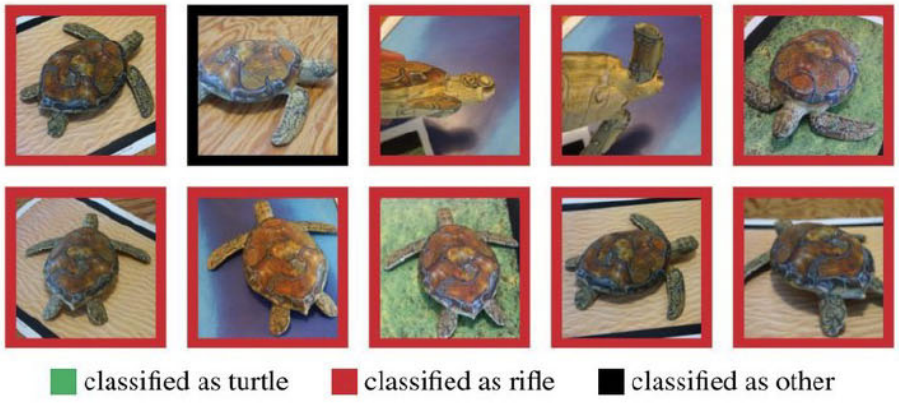
[S. A. Seshia, et al., "Formal Specification for Deep Neural Networks", TR May 2018; ATVA'18]

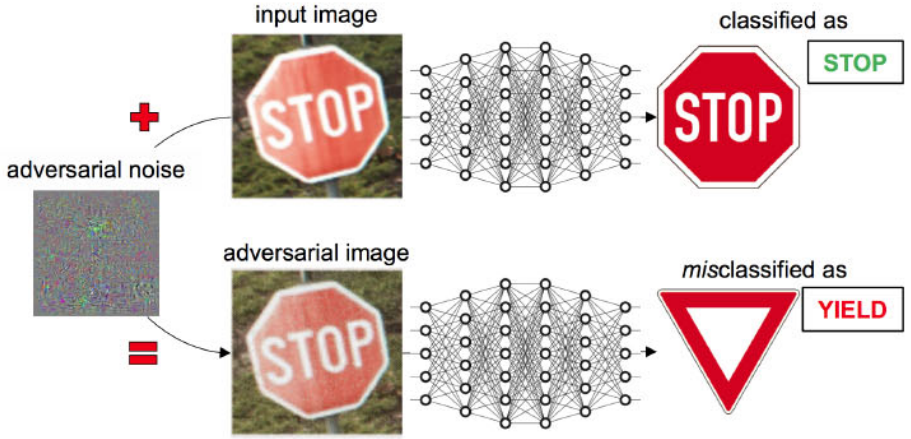
- System Level ("Semantic")
  - Typical properties of reactive systems (safety, liveness, etc.)
- Component Level
  - Robustness
  - Input-Output Relations
  - Monotonicity
  - Fairness
  - Coverage
  - …

Robustness Analysis

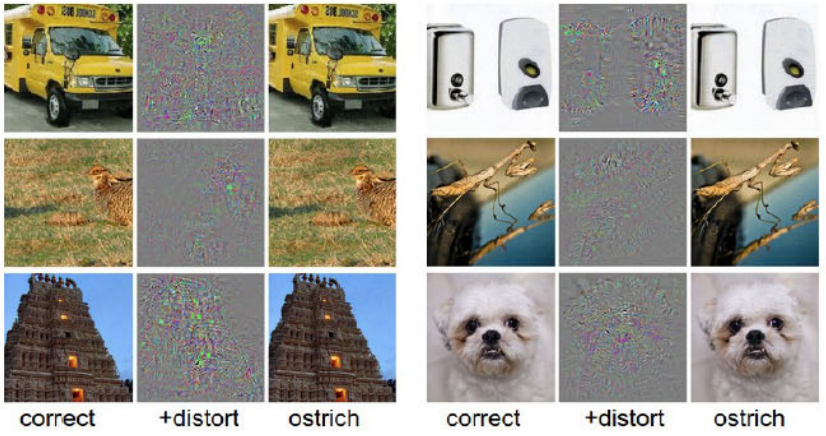


Explaining and Harnessing Adversarial Examples, Goodfellow et. al





Slides by Andrej Karpathy



Intriguing Properties of Neural Networks, Szegedy et. al

Robustness as a Property

[Dreossi, Ghosh, Sangiovanni-Vincentelli, Seshia, VNN 2019]

Given: Input $x \in X$

NN $f : X \rightarrow Y$

Find: Adversarial example x^* which satisfy,

(1) Admissibility constraint: $x^* \in \tilde{X}$

(2) Distance constraint: $D(\mu(x, x^*), \alpha)$

} Characterize the adversary E

(3) Target behavior constraint: $A(x, x^*, \beta)$ Depends on the specification φ

Optimization Versions →

Minimizing Perturbation

$$x^* = \underset{x' \in \tilde{X}}{\text{argmin}} \ \alpha \quad \text{s.t.} \quad \mu(x, x') \leq \alpha, \ f(x') \neq f(x)$$

Maximizing Loss

$$x^* = \underset{x' \in \tilde{X}}{\text{argmax}} \ \beta \quad \text{s.t.} \quad L(f(x), f(x')) \geq \beta$$

Summary of Robustness Properties

[Dreossi, Ghosh, Sangiovanni-Vincentelli, Seshia, VNN 2019]

Adversary Type	**Admissibility Constraint** $x^* \in \tilde{X}$	**Distance Constraint** $D(\mu(x, x^*), \alpha)$	**Target Constraint** $A(x, x^*, \beta)$
Minimum Perturbation Adversary, Targeted Attacks	$x^* = x + r \in X$	$\|r\|_p \leq \alpha$	$f(x^*) = y$
Minimum Perturbation Adversary, Untargeted Attacks	$x^* = x + r \in X$	$\|r\|_p \leq \alpha$	$f(x^*) \neq f(x)$
Maximize Loss Adversary, Untargeted Attacks	$x^* = x + r \in X$	$\|r\|_\infty \leq \alpha = \epsilon$	$L(\theta, x^*, y) \geq \beta$
Robust Adversarial Examples	$x^* = x + r \in X$	$\mathbb{E}_{t \in T}[d(t(x), t(x^*))] \leq \alpha = \epsilon$	$\mathbb{E}_{t \in T}[\log P(y_t \mid t(x^*))] \geq \beta$
Input Output Relations	$x^* = x + r \in X$	$x^* \in S_{in}(x)$	$f(x^*) \notin S_{out}(f(x))$
Black-Box Transferable Attacks	$x^* = x + r \in X$	$\|r\|_2 \leq \alpha$	$f_{sub}(x^*) = y, f_{sub}(x^*) \neq f_{sub}(x)$ $f_{sub}(x^*) \neq f_{sub}(x) \rightarrow f(x^*) \neq f(x)$
Neuron Coverage	$x^* \in X$	$x^* \in \{\gamma x, x + r\}$	$f_1(x) = \ldots = f_k(x) \Rightarrow f_i(x^*) \neq f_j(x^*)$ $F_n(x^*) \geq \beta$
Semantic Robustness	$s^* \in S \Rightarrow x^* = R(s^*) \in X$	$\mu(s_i^*, s_j^*) \leq \alpha$	$f(R(s^*)) \Rightarrow \mathscr{M}(s^*) \not\models \varphi$

Specification Spectrum

[S. A. Seshia, et al., "Formal Specification for Deep Neural Networks", TR May 2018; ATVA'18]

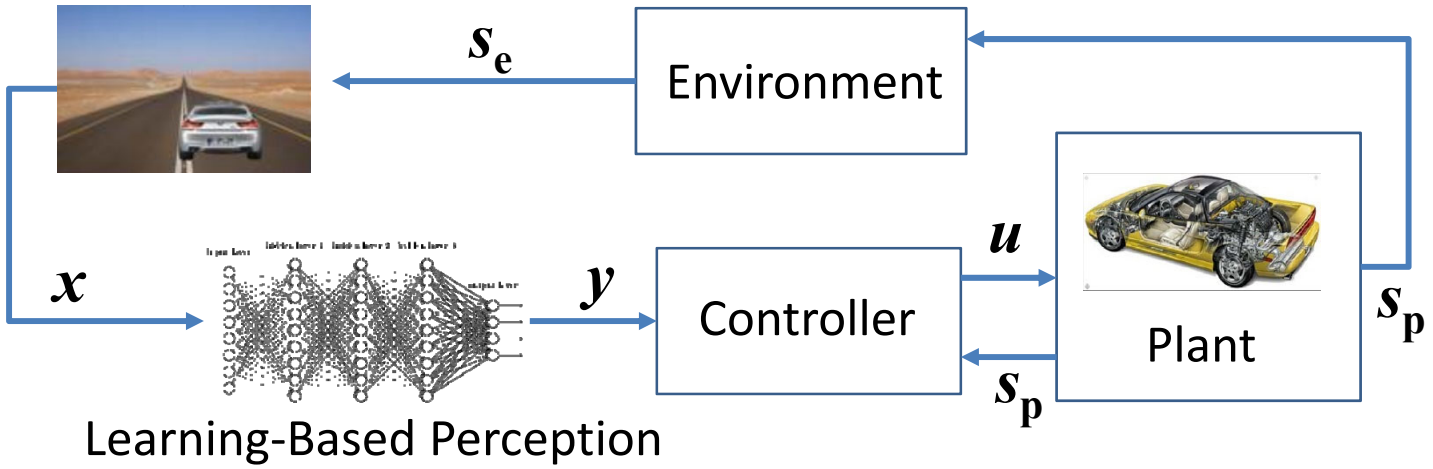
- System Level ("Semantic")
  - Typical properties of reactive systems (safety, liveness, etc.)

- Component Level
  - Robustness
  - Input-Output Relations
  - Monotonicity
  - Fairness
  - Coverage
  - …

Need a notion of Semantic Robustness
→ Robustness in the *Semantic Feature Space*
[Dreossi, Jha, Seshia CAV 2018]

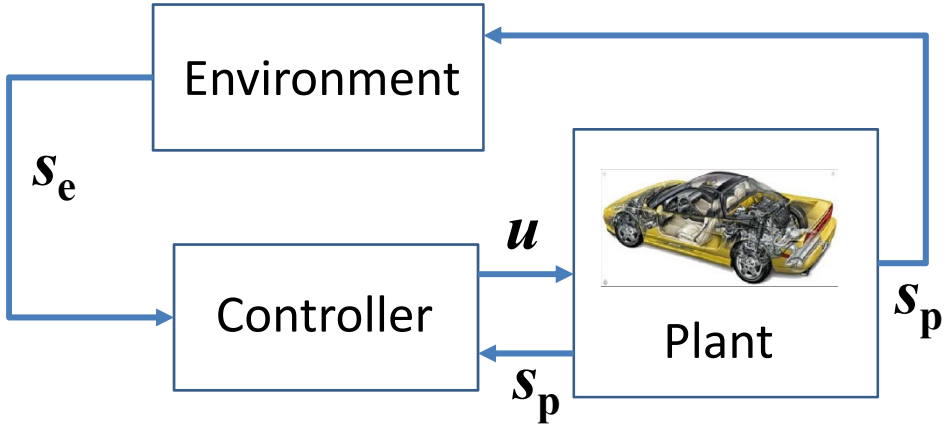
Challenge: Scalability of Verification →
Principle: Compositional Simulation-Based Verification (Falsification)

CPSML model



Challenge: $s_e << x$

Traditional CPS model



Three Key Ideas

1. Reduce CPSML falsification problem to combination of CPS falsification and ML analysis

2. Simulation-based temporal logic falsification for CPS model

3. ML analysis via systematic exploration of "semantic feature space"

[Dreossi et al., NFM'17; Dreossi et al., CAV'18]

Simulation-Based Falsification of Signal Temporal Logic for CPS

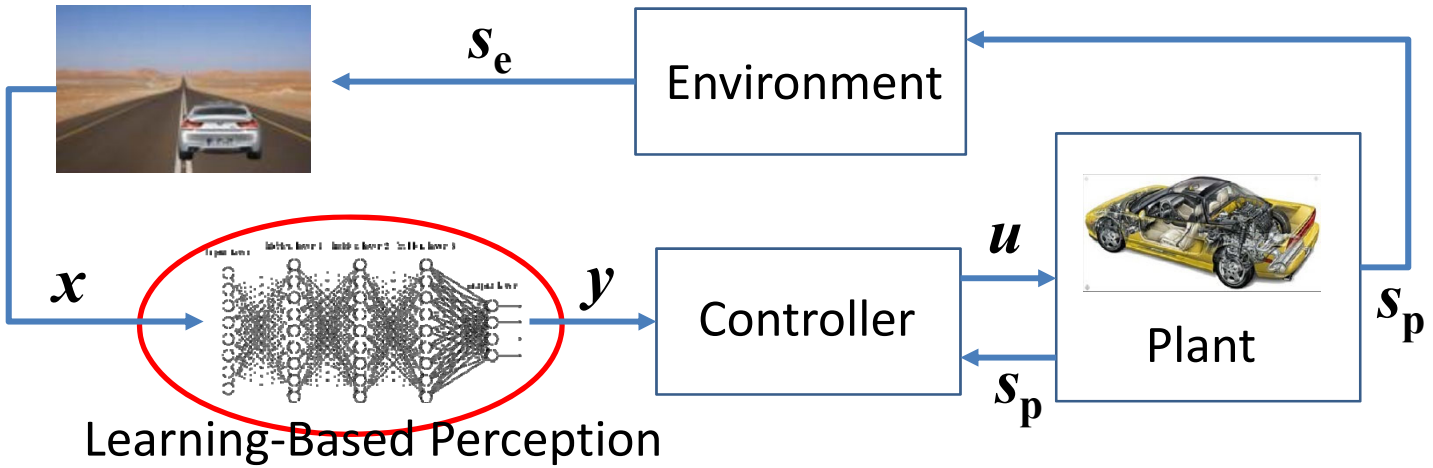
- STL has quantitative semantics
  - Logical formula → Cost Function ρ
  - Quantifies "how much" a trace satisfies a property

- *Advantage:* Finding a bug (property violation) → minimizing ρ and checking if value < 0.
  - View of "verification as optimization" underlies simulation-based falsification tools
  - Used by some production groups in automotive industry

- Can also apply to MTL and other TL variants

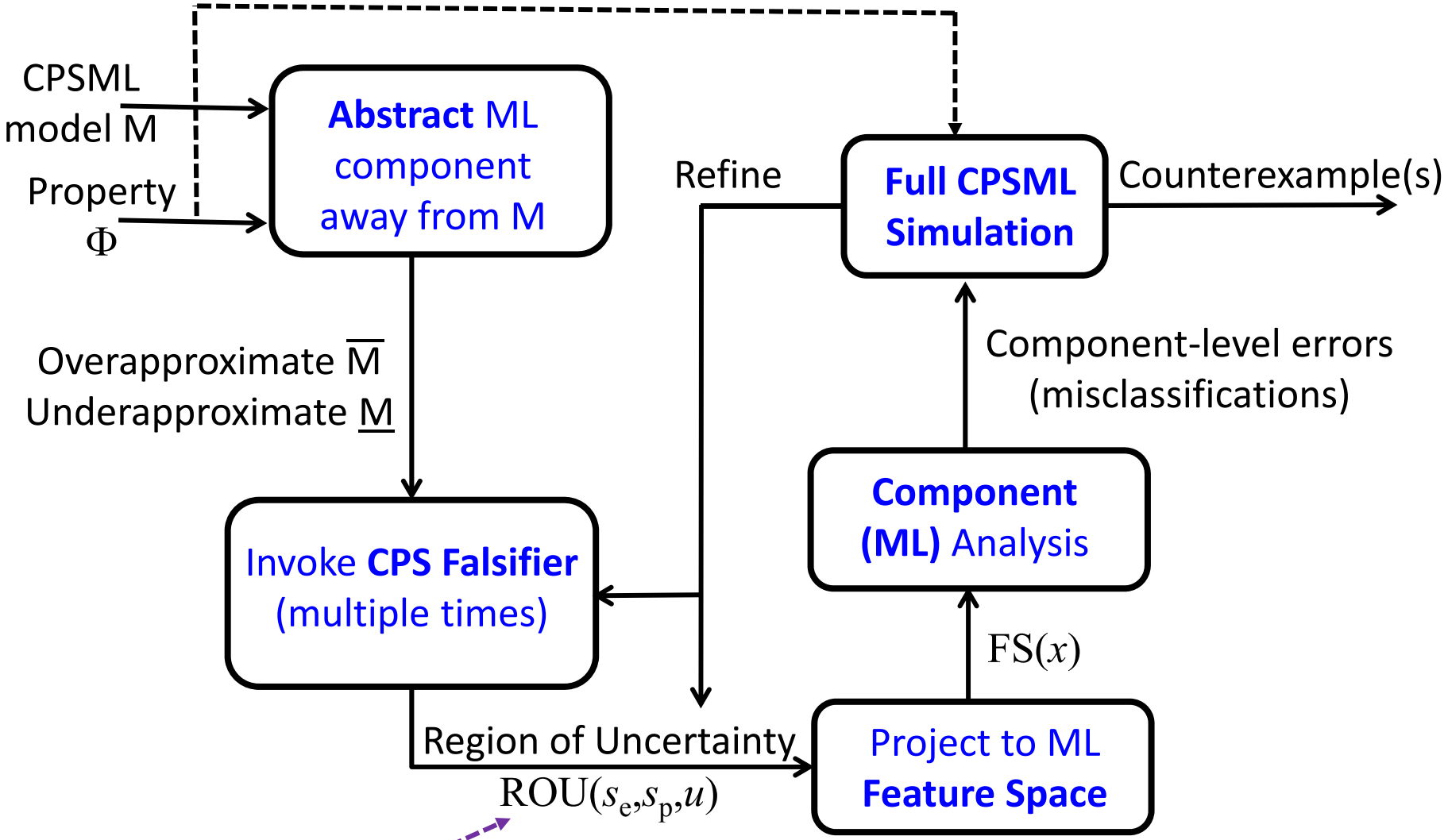
Need *Compositional* Falsification of CPSML

- *Challenge:* Very High Dimensionality of Input Space!
- Standard solution: Use *Compositional (Modular)* Verification



- However: *no formal spec*. for neural network!
- Compositional Verification *without* Compositional Specification?!! (see [Seshia, UCB/EECS TR 2017])

Compositional Approach: Combine Temporal Logic CPS Falsifier with ML Analyzer

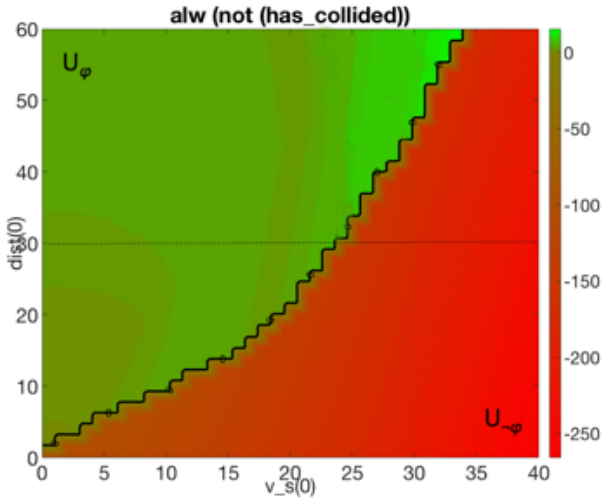


[Dreossi, Donze, Seshia, NFM 2017]

Identifying Region of Uncertainty (*ROU*) for Automatic Emergency Braking System

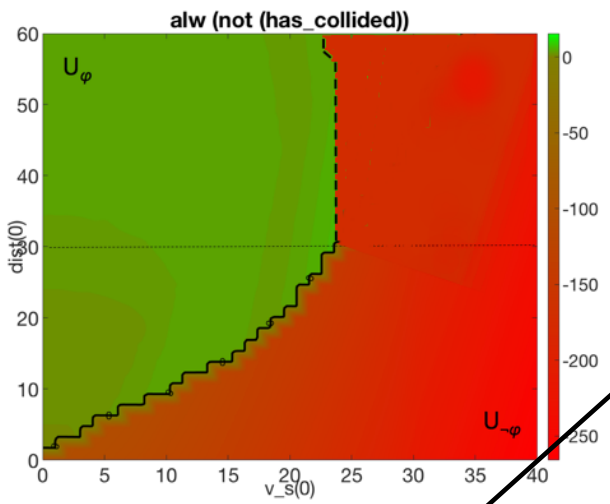
Green → environments where the property is satisfied

Underapproximate $\underline{M}$

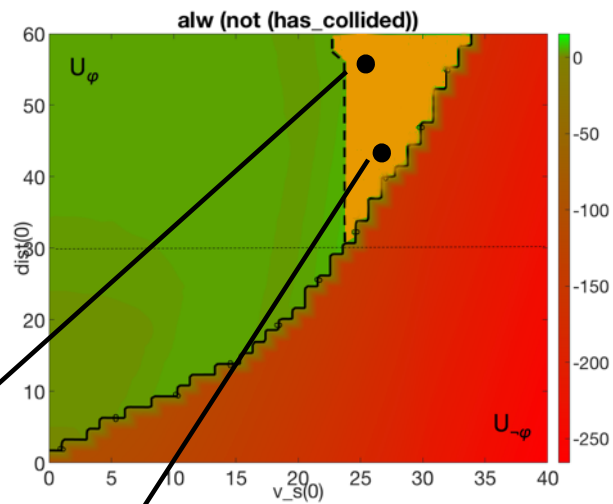


ML always correct

Overapproximate $\overline{M}$



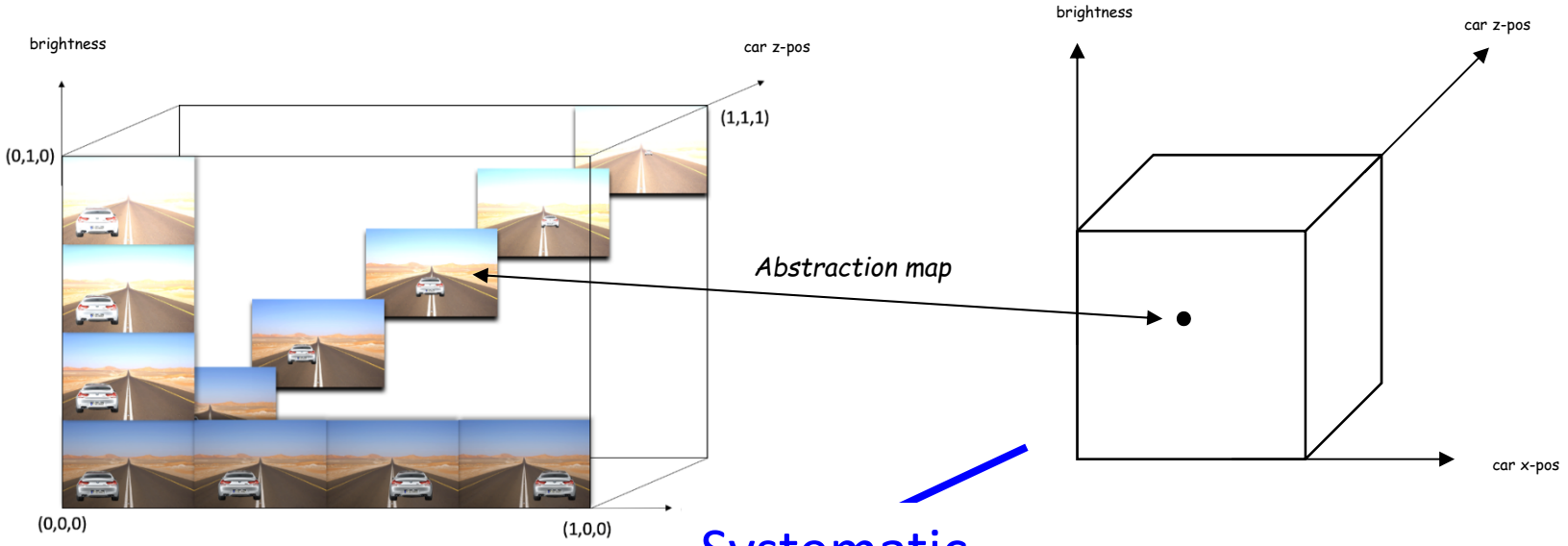
ML always wrong



Potentially unsafe region depending on ML component (yellow)

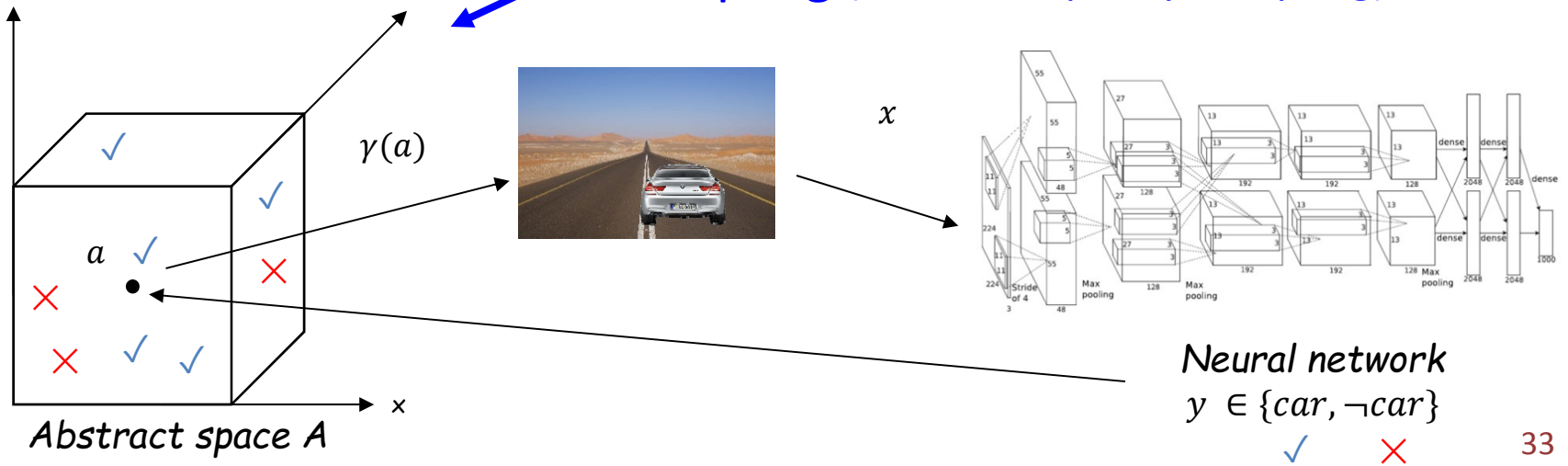
Machine Learning Analyzer

Systematically Explore ROU in the Image (Sensor) Space



Semantic Feature space $\tilde{X}$

Systematic Sampling (low-discrepancy sampling)



Abstract space A

Neural network

$y \in \{car, \neg car\}$

Sample Result

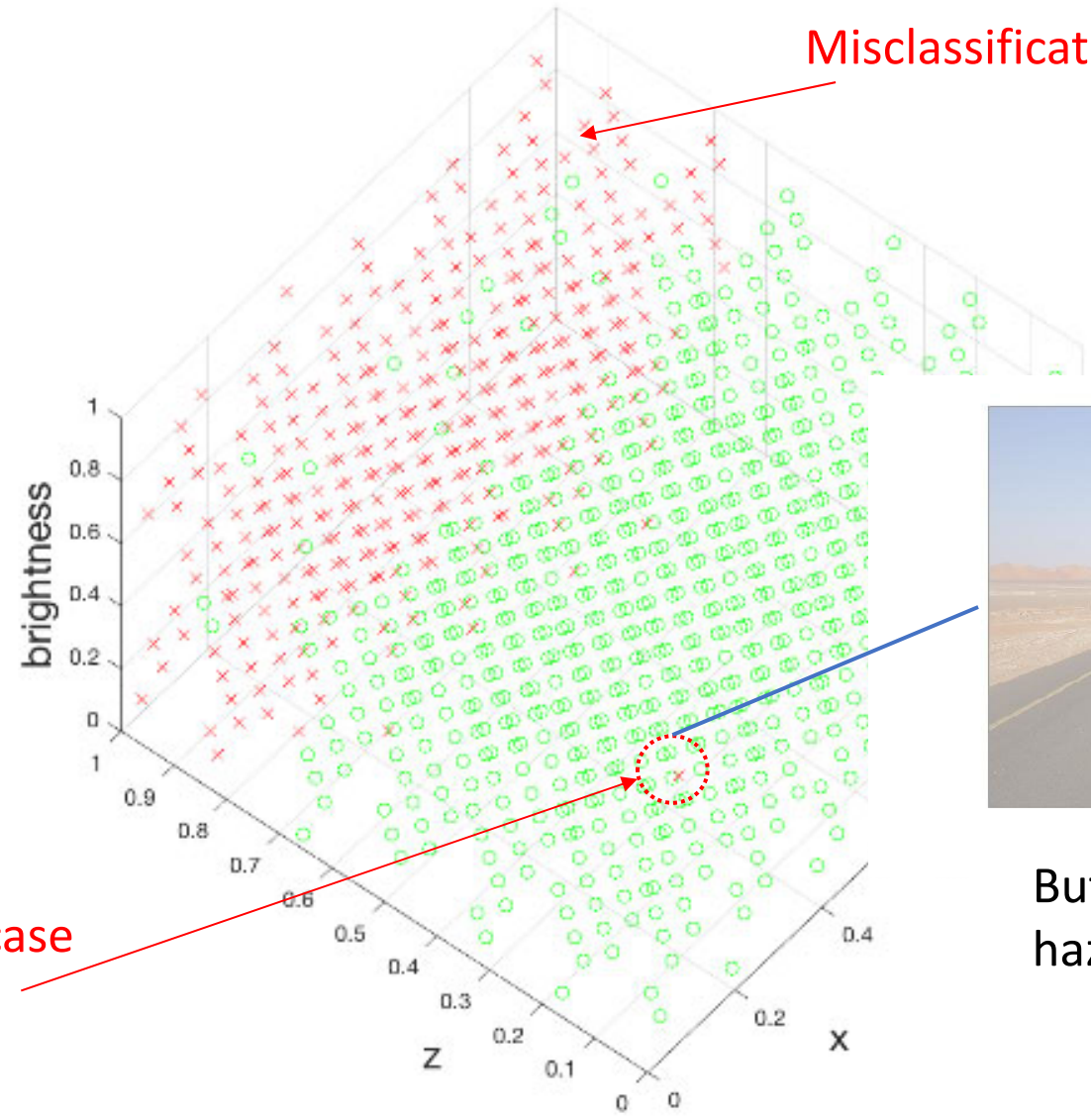
This misclassification may not be of concern

Misclassifications

Inception-v3 *Neural Network* *(pre-trained on ImageNet using TensorFlow)*

Sample Result

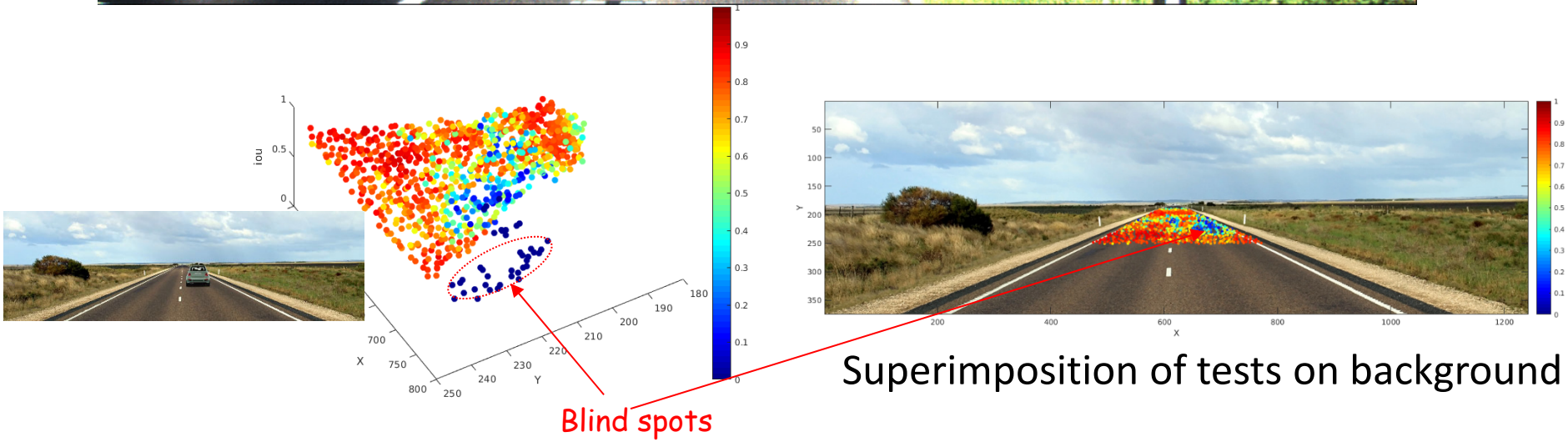
Inception-v3 *Neural Network* *(pre-trained on ImageNet using TensorFlow)*



But this one is a real hazard!

Image Streams

[Dreossi, Ghosh, et al., ICML 2017 workshop]



Superimposition of tests on background

Results on squeezeDet NN and KITTI dataset for autonomous driving

Summary

- Write System-Level Specifications
- Use Compositional Approach
  - Falsification on the CPSML Model
  - Extends to Verification in some cases
- Approach Generates Counterexamples

➢ How do we use these counterexamples?

➢ *Semantic* Adversarial Machine (Deep) Learning [Dreossi, Jha, Seshia, CAV 2018]

Challenge: How to (Re)Design Learning Components

Principle: Use Oracle-Guided Inductive Synthesis

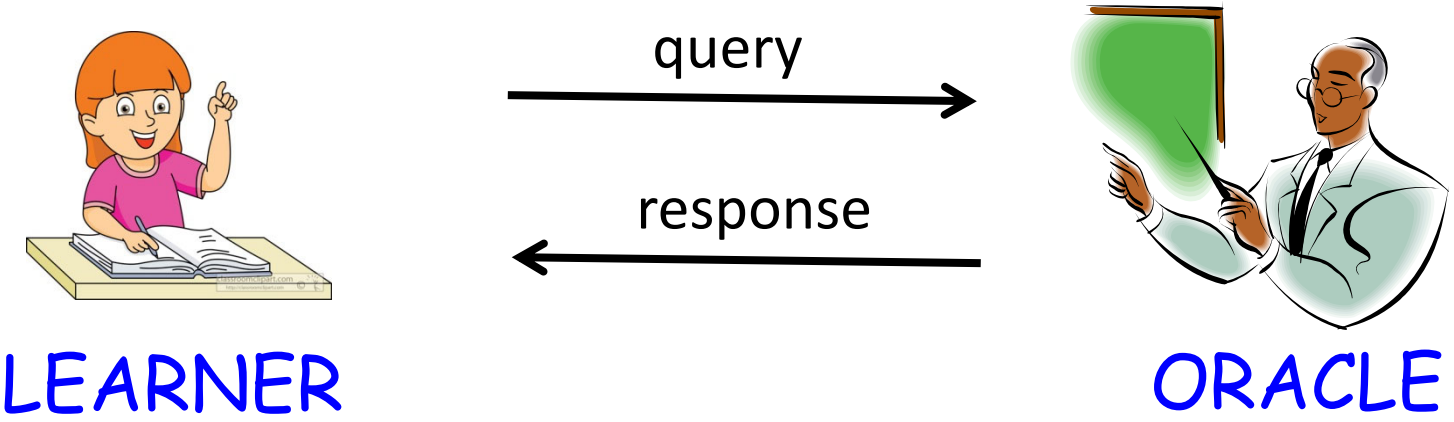
Correct-by-Construction Design with Formal (Oracle-Guided) Inductive Synthesis/Learning

Inductive Synthesis: Learning from Examples (ML)

Formal Inductive Synthesis: Learn from Examples *while satisfying a Formal Specification*

Key Idea: **Oracle-Guided Learning**

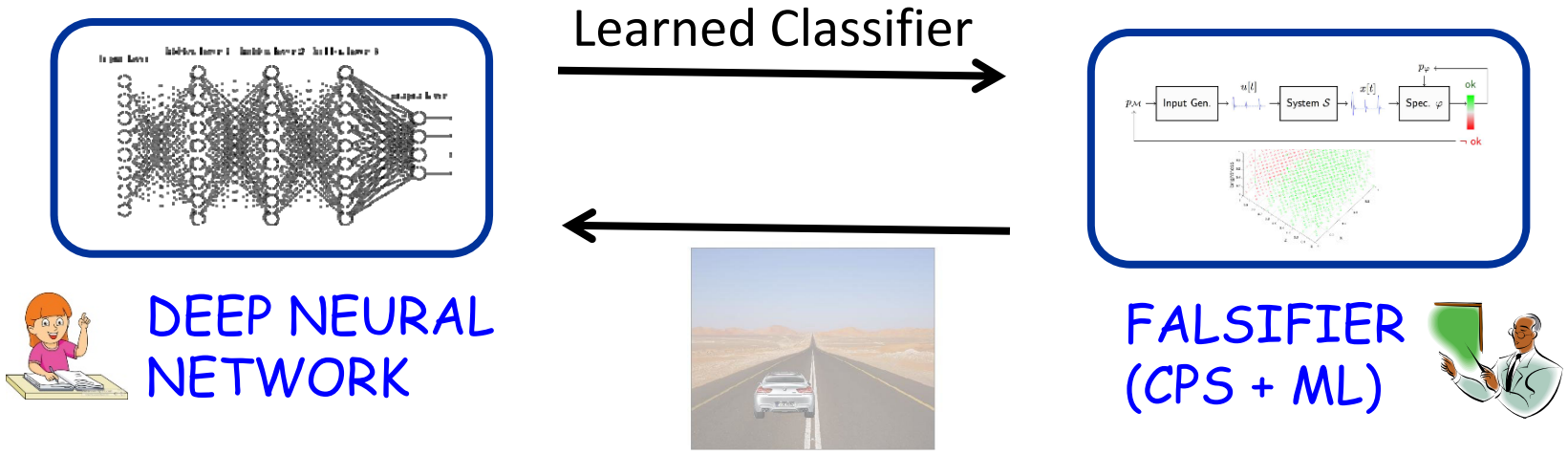
Combine Learner with Oracle (e.g., Verifier) that answers Learner's Queries



[Jha & Seshia, "A Theory of Formal Synthesis via Inductive Learning", 2015, Acta Informatica 2017.]

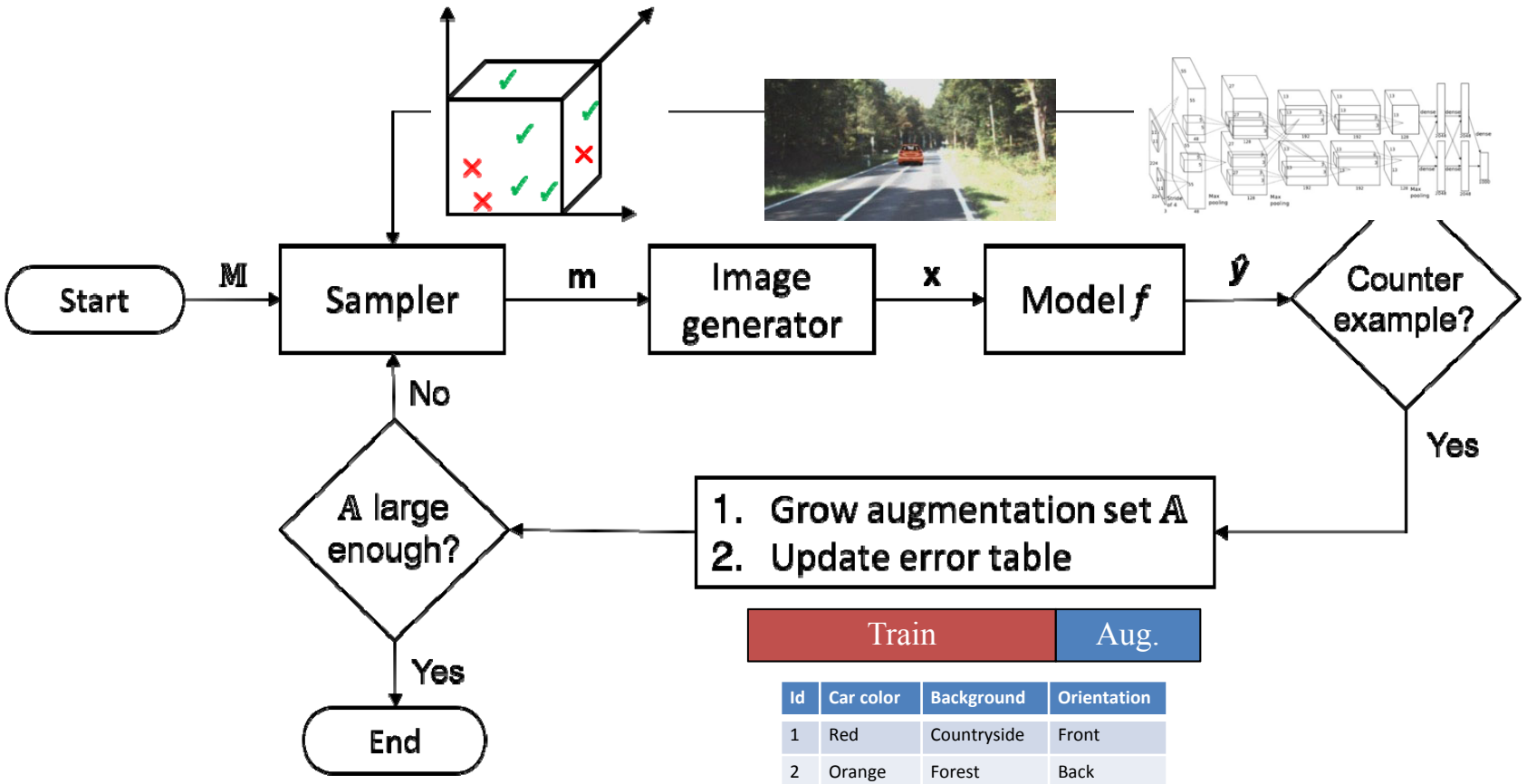
Counterexample-Guided Training of Deep Neural Networks

- Instance of Oracle-Guided Inductive Synthesis
- Oracle is Verifier (CPSML Falsifier) used to find counterexample inputs to DNN
- Substantially increase accuracy with relatively few additional examples



"Counterexample-Guided Data Augmentation", T. Dreossi, S. Ghosh, X. Yue, K. Keutzer, A. Sangiovanni-Vincentelli, S. A. Seshia, IJCAI 2018.

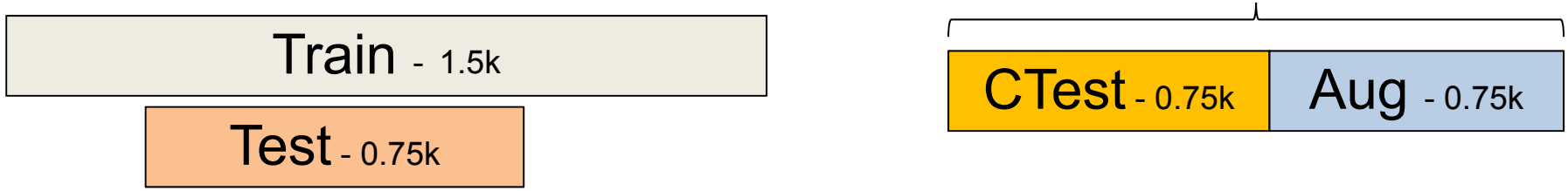
Counterexample-Guided Data Augmentation



Id	Car color	Background	Orientation
1	Red	Countryside	Front
2	Orange	Forest	Back
3	White	Forest	Front
4	Green	Forest	Back

"Counterexample-Guided Data Augmentation", T. Dreossi, S. Ghosh, X. Yue, K. Keutzer, A. Sangiovanni-Vincentelli, S. A. Seshia, IJCAI 2018.

Experimental Results

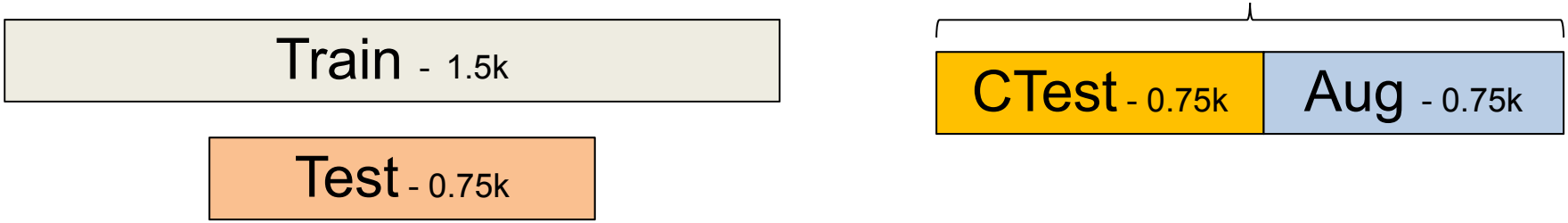


Comparison of Augmentation Methods

	Model	Precision	Recall
	Original	.61	.75
	Standard augmentation	.69	.80
Counterexample-guided augmentation	Uniform random	.76	.87
	Constrain	.75	.86
	Low-discrepancy	**.79**	**.87**
	Cross-entropy	.78	.78

"Counterexample-Guided Data Augmentation", T. Dreossi, S. Ghosh, X. Yue, K. Keutzer, A. Sangiovanni-Vincentelli, S. A. Seshia, IJCAI 2018.

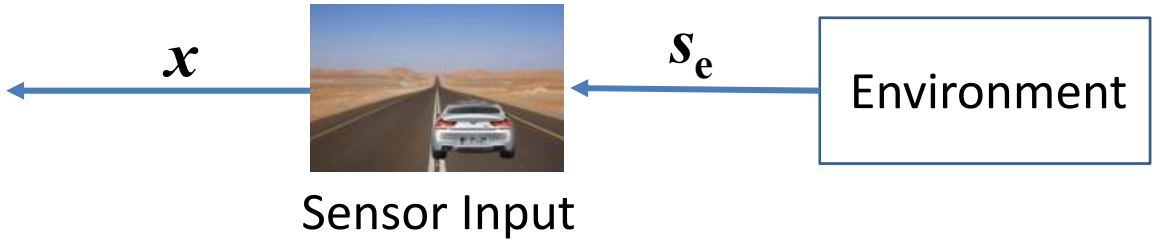
Experimental Results



Precision – Low-discrepancy sampling

Model	Test	CTest 1	CTest 2	CTest 3	Time (hrs)
Original	0.98	0.69			0.02
Aug_1	0.99	0.82	0.59		~6
Aug_2	0.99	0.86	0.82	0.53	~14
Aug_3	0.99	0.84	0.80	0.88	~26

Much harder to find counterexamples after retraining!!!



Challenge: Environment Modeling → capturing assumptions

Principles: Data-Driven, Introspective, Stochastic Modeling

Generating Meaningful (Sensor) Data

- Large and unstructured input space
- We want scenes that make physical sense and are *interesting and useful* for training/ testing or design
- How can we guide data generation towards such scenes?

Images created with GTA-V

➢ Our approach: Scene Improvisation

[D. Fremont et al., "Scenic: Language-Based Scene Generation", 2018.]

SCENIC: Scenario Description Language

- *Scenic* is a probabilistic programming language defining *distributions over scenes*

- Example scenario: a badly-parked car

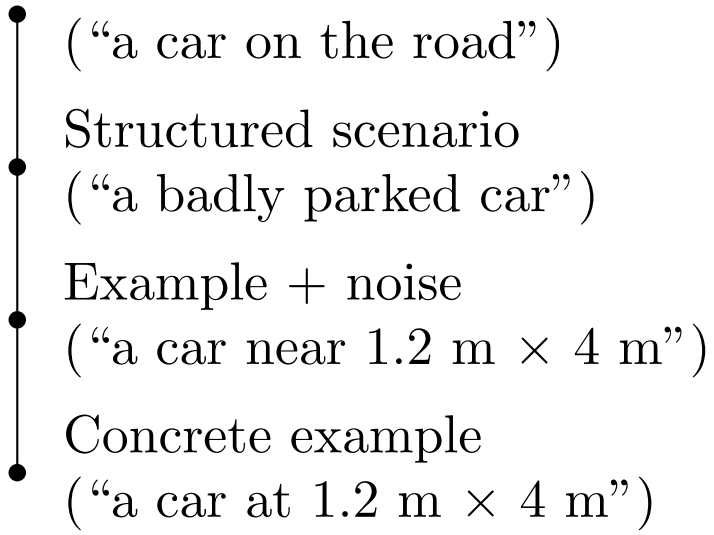
```
from gta import Car, curb, roadDirection

ego = Car

spot = OrientedPoint on visible curb
badAngle = Uniform(1.0, -1.0) * (10, 20) deg
Car left of (spot offset by -0.5 @ 0),
    facing badAngle relative to roadDirection
```

[D. Fremont et al., "Scenic: Language-Based Scene Generation", 2018.]

SCENIC: a Scenario Description Language

- Defines a *distribution over scenes*
- Readable, concise syntax fo common geometric relationships
- Declarative hard and soft constraints
- Scenarios on a spectrum from very broad classes of scenes to small variations on a single scene

```
scenario := (import file)* (statement)*
boolean := True | False | booleanOperator
scalar := number | distribution | scalarOperator
distribution := baseDist | resample(distribution)
vector := scalar @ scalar | Point | vectorOperator
heading := scalar | OrientedPoint | headingOperator
direction := heading | vectorField
value := boolean | scalar | vector | direction
       | region | instance | instance.property
classDefn := class class[(superclass)]:
           (property: defaultValueExpr)*
instance := class specifier, ...
specifier := with property value | posSpec | headSpec
```



SCENIC: Scenario Description Language

Scenic makes it possible to specify broad scenarios with complex structure, then generate many concrete instances from them automatically:

Platoons

Bumper-to-Bumper Traffic

Use Case: Retraining with Hard Cases

e.g. for car detection, one car partially occluding another:

```
from gta import Car

ego = Car with roadDeviation (-10, 10) deg

c = Car visible,
        with roadDeviation (-10, 10) deg

leftRight = Uniform(1.0, -1.0) * (1.25, 2.75)
Car beyond c by leftRight @ (4, 10),
    with roadDeviation (-10, 10) deg
```

Use Case: Retraining with Hard Cases

Improves accuracy on hard cases without compromising accuracy on original training set

e.g. for car detection, one car partially occluding another:

Use Case: Generalizing a Known Failure

Use Case: Generalizing a Known Failure

Scenic makes it easy to generalize along different dimensions:

Add noise

Change car model

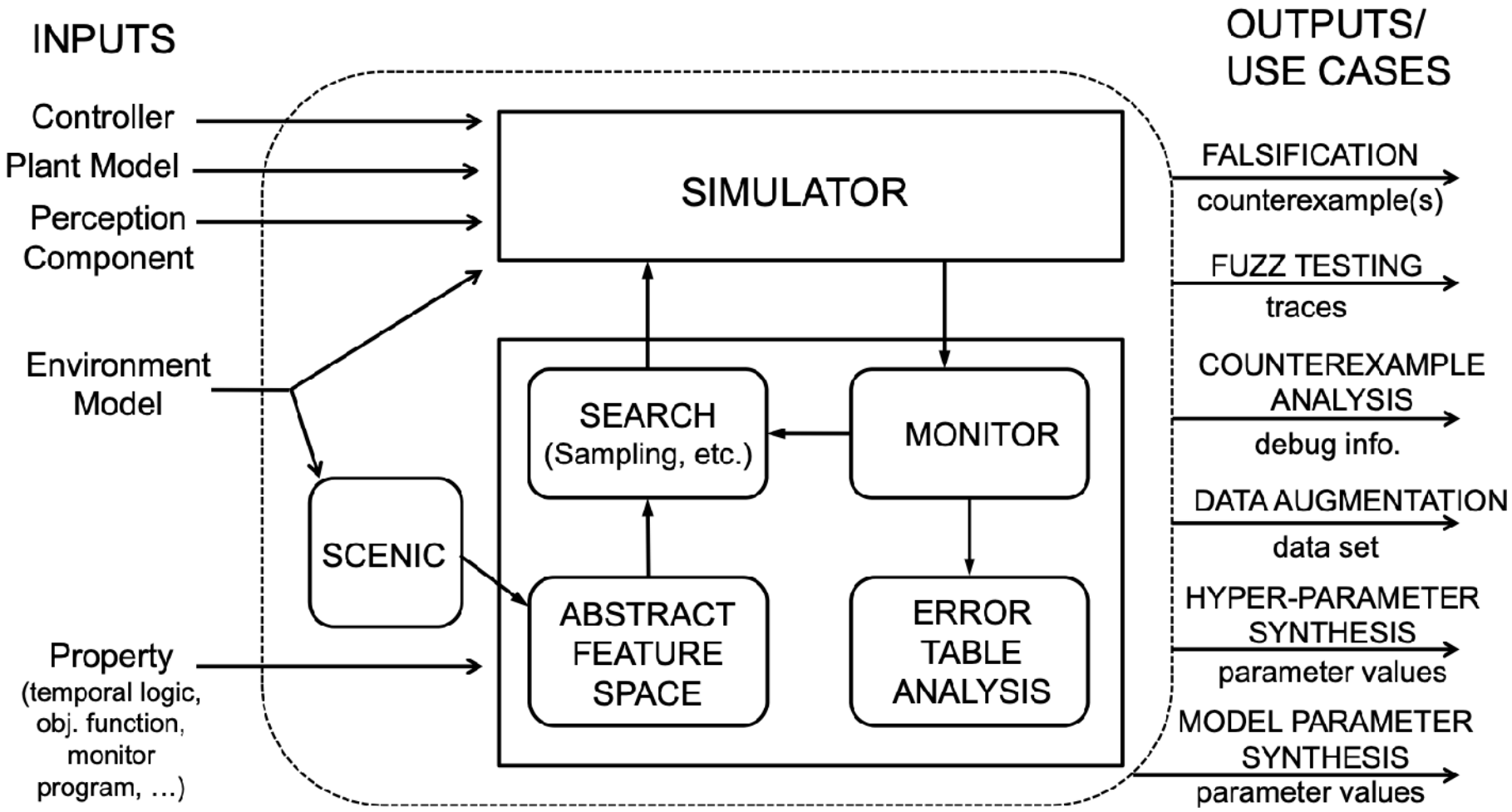
Change global position

Outline

- Sample Challenges for Verified AI
- Design of Closed-Loop Cyber-Physical Systems with Machine Learning Components
  - Specification, Verification, Synthesis
  - Autonomous Vehicles
  - Deep Learning
- Principles for Verified AI
  - Summary of Ideas
  - Future Directions

VERIFAI: A Toolkit for the Design and Analysis of AI-Based Systems [Dreossi, Fremont, Ghosh, et al., CAV 2019]

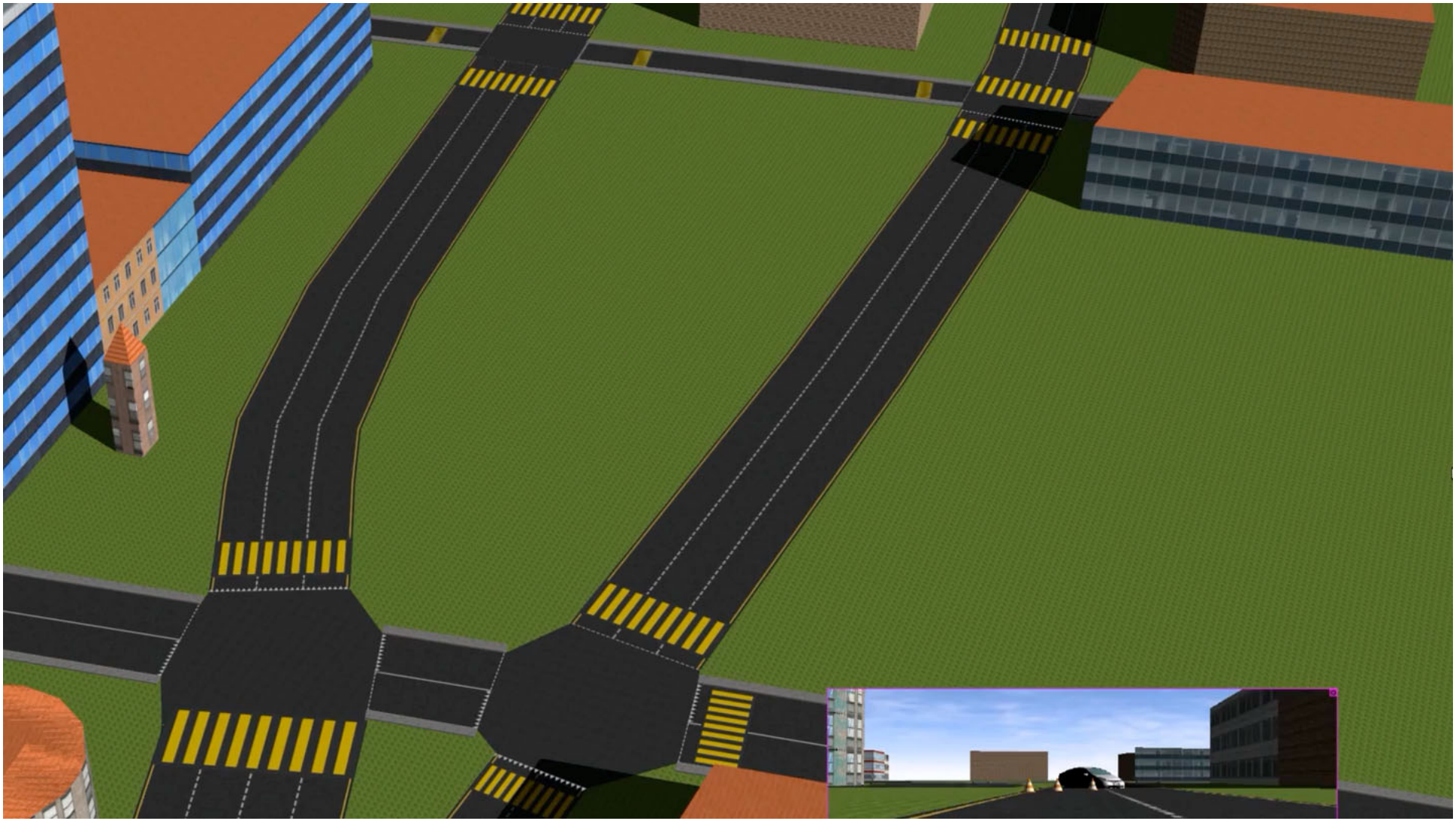
https://github.com/BerkeleyLearnVerify/VerifAI

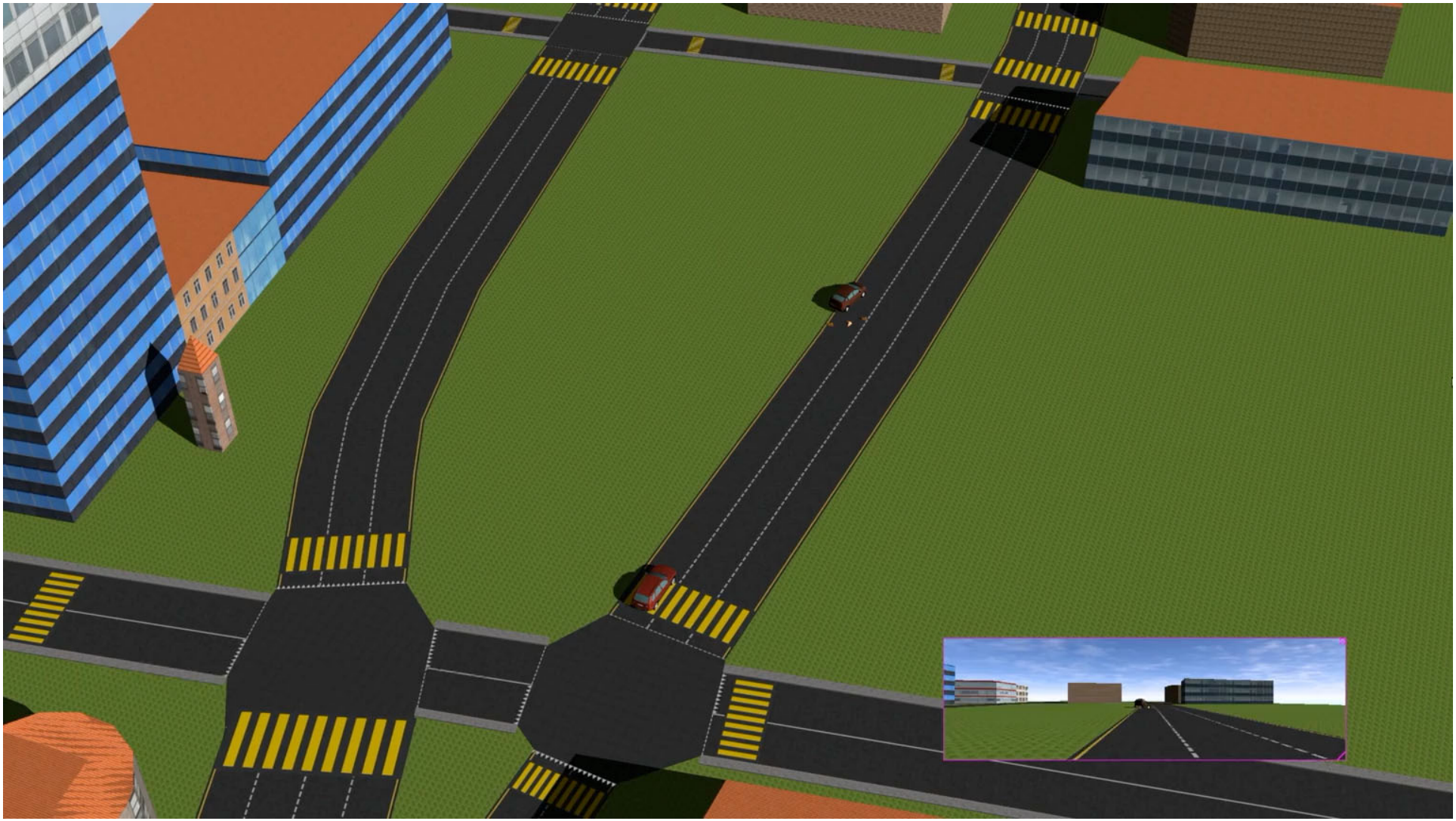


Environment Modeling: the SCENIC Language

```
# Pick location for blockage randomly along curb
blockageSite = OrientedPoint on curb

# Place traffic cones
spot1 = OrientedPoint left of blockageSite by (0.3, 1)
cone1 = TrafficCone at spot1,
                     facing (0, 360) deg

...

# Place disabled car ahead of cones
SmallCar ahead of spot2 by (-1, 0.5) @ (4, 10),
          facing (0, 360) deg
```

Fremont et al., *Scenic: A Language for Scenario Specification and Scene Generation*, PLDI 2019 (to appear).





Towards Verified Artificial Intelligence

Challenges		Principles
1. Environment (incl. Human) Modeling	→	Data-Driven, Introspective, Stochastic Modeling
2. Specification	→	Start with System-Level Spec.; Derive Component Specs.
3. Learning Systems Complexity	→	Abstraction, Semantic Feature Analysis, Explanations
4. Scalable Training, Testing, Verification	→	Compositional Analysis and Algorithmic Improvisation
5. Design for Correctness	→	Formal, Oracle-Guided Inductive Synthesis; Run-time Assurance

S. A. Seshia, D. Sadigh, S. S. Sastry. *Towards Verified Artificial Intelligence*. July 2016. https://arxiv.org/abs/1606.08514.