COLUMBIA UNIVERSITY
Data Science Institute

ALFRED P. SLOAN
FOUNDATION

NSF

J.P.Morgan

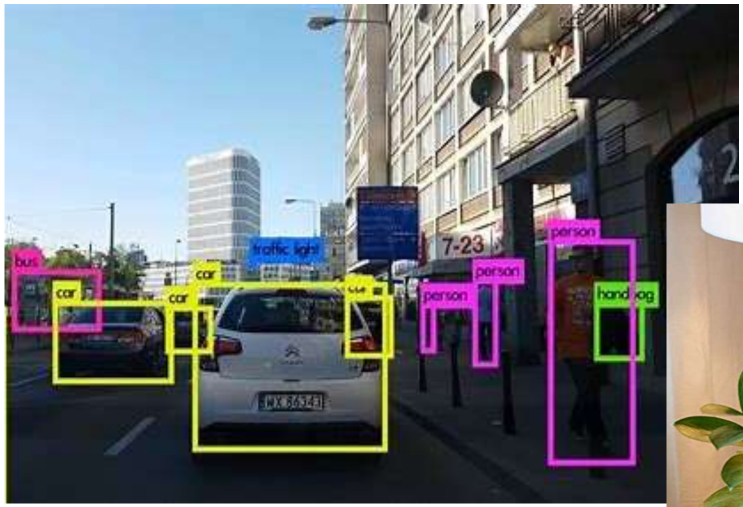
Trustworthy AI

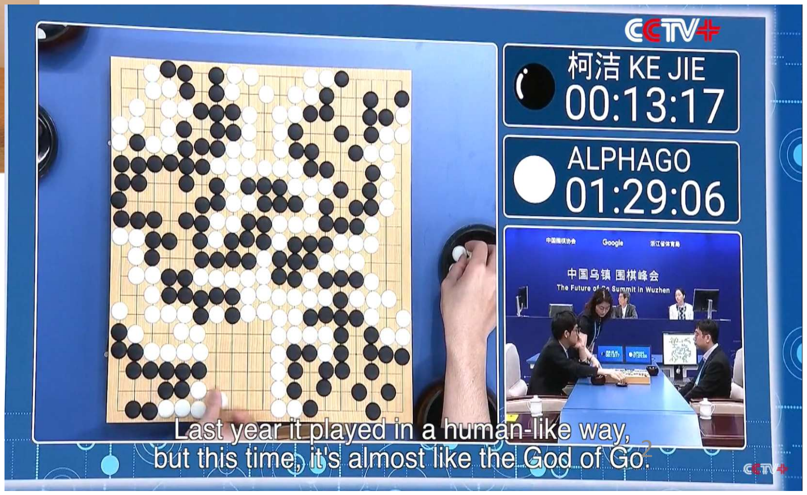
Jeannette M. Wing
Executive Vice President for Research and Professor of Computer Science, Columbia University
Adjunct Professor of Computer Science, Carnegie Mellon University

Trustworthy AI | October 2021 | Communications of the ACM

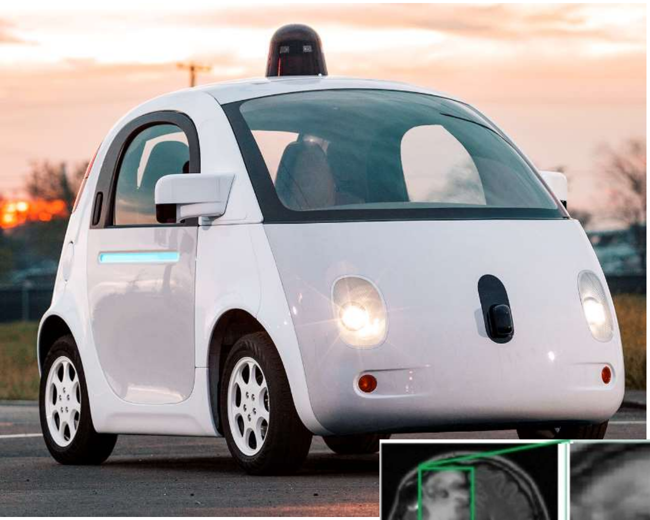
Hot Science of Security (HotSoS)
Virtual Keynote
April 6, 2022

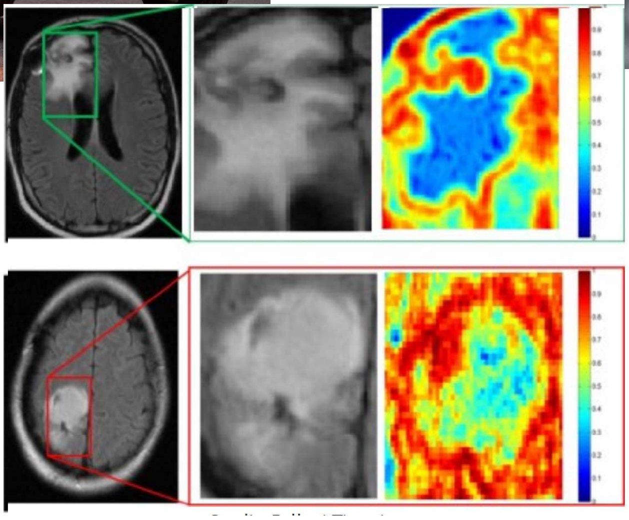
AI achieves or exceeds human performance




CCTV+
柯洁 KE JIE
00:13:17
ALPHAGO
01:29:06
Last year it played in a human-like way,
but this time, it's almost like the God of Go.

AI can benefit humanity and society

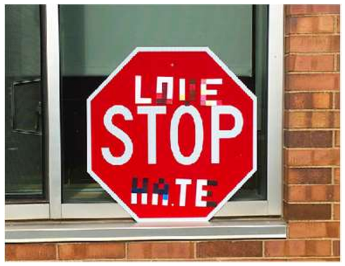




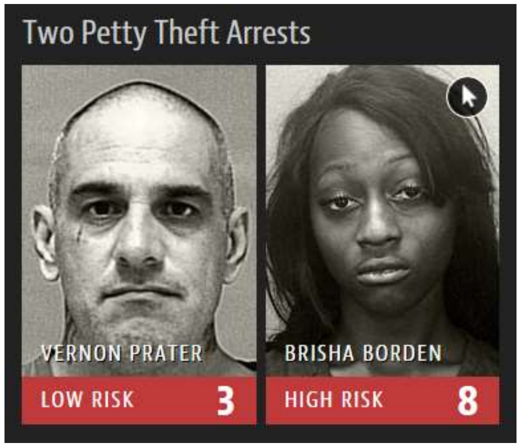
[Tiwari et al. 2016, *American J. of Neuroradiology*]

Hiring Process
Recruiting & Pre-Screening
Phase 1
Assessments
Phase 2
Drug Test & Background Check
Phase 3
Placement
Phase 4
Onboarding
Phase 5

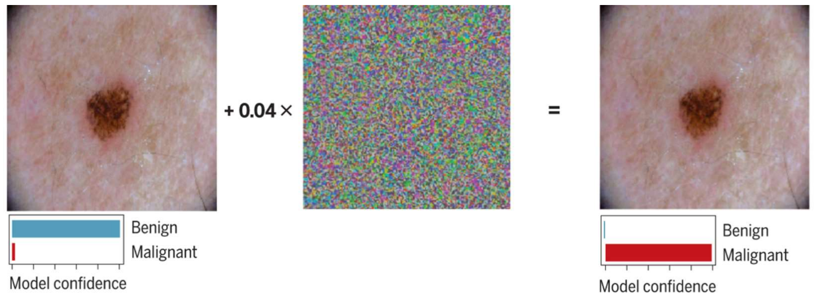
But, why should we trust AI-based systems?



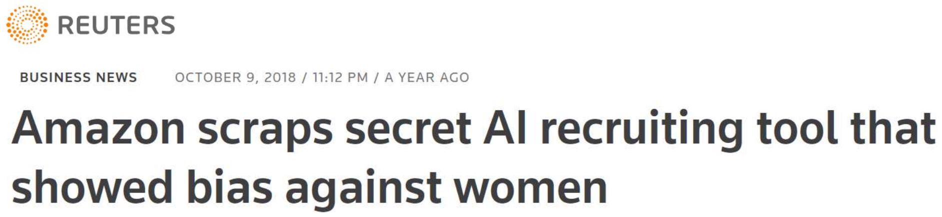
[Eykholt et al. 2017, *CVPR*]



[Angwin et al. 2016, *Pro Publica*]



[Finlayson et al 2019, *Science*]



REUTERS

BUSINESS NEWS OCTOBER 9, 2018 / 11:12 PM / A YEAR AGO

Amazon scraps secret AI recruiting tool that showed bias against women

[Dastin 2018, *Reuters*]

Question:

How then can we deliver on the promise of the benefits of AI but address these scenarios that have life-critical consequences for people and society?

In short, *how can we achieve trustworthy AI?*

From Trustworthy Computing…

- Trustworthy =
  - + Reliability
    - Does it do the right thing?
  - + Safety
    - Does it do no harm?
  - + Security
    - How vulnerable is it to attack?
  - + Privacy
    - Does it protect a person’s identity and data?
  - + Availability
    - Is the system up when I need to access it?
  - + Usability
    - Can a human use it easily?

- Computing = hardware + software + people

…to Trustworthy AI: Upping the Ante

- Trustworthy =
  - + Reliability
    - Does it do the right thing?
  - + Safety
    - Does it do no harm?
  - + Security
    - How vulnerable is it to attack?
  - + Privacy
    - Does it protect a person's identity and data?
  - + Availability
    - Is the system up when I need to access it?
  - + Usability
    - Can a human use it easily?

+ Accuracy
+ Robustness
+ Fairness
+ Accountability
+ Transparency
+ Interpretability/Explainability
+ Ethical

+ ...properties yet to be identified

- AI = data + ML model + task

Trustworthy AI = Trustworthy Computing +

+ Accuracy
  - How well does the AI system do on new (unseen) data compared to data on which it was trained and tested?

+ Robustness
  - How sensitive is the outcome to a change in the input?

+ Fairness
  - Are the outcomes unbiased?

+ Accountability
  - Who or what is responsible for the outcome?

+ Transparency
  - Is it clear to an external observer how the system's outcome was produced?

+ Interpretability/Explainability:
  - Can the system's outcome be justified with an explanation that a human can understand and/or that is meaningful to the end user?

+ Ethical
  - Was the data collected in an ethical manner?
  - Will the outcome be used in an ethical manner?

+ properties yet to be identified

Question:

How can we achieve trustworthy AI?

One Approach:

Through formal methods.

From Traditional Formal Verification…

$$\boxed{E, M \vDash P}$$

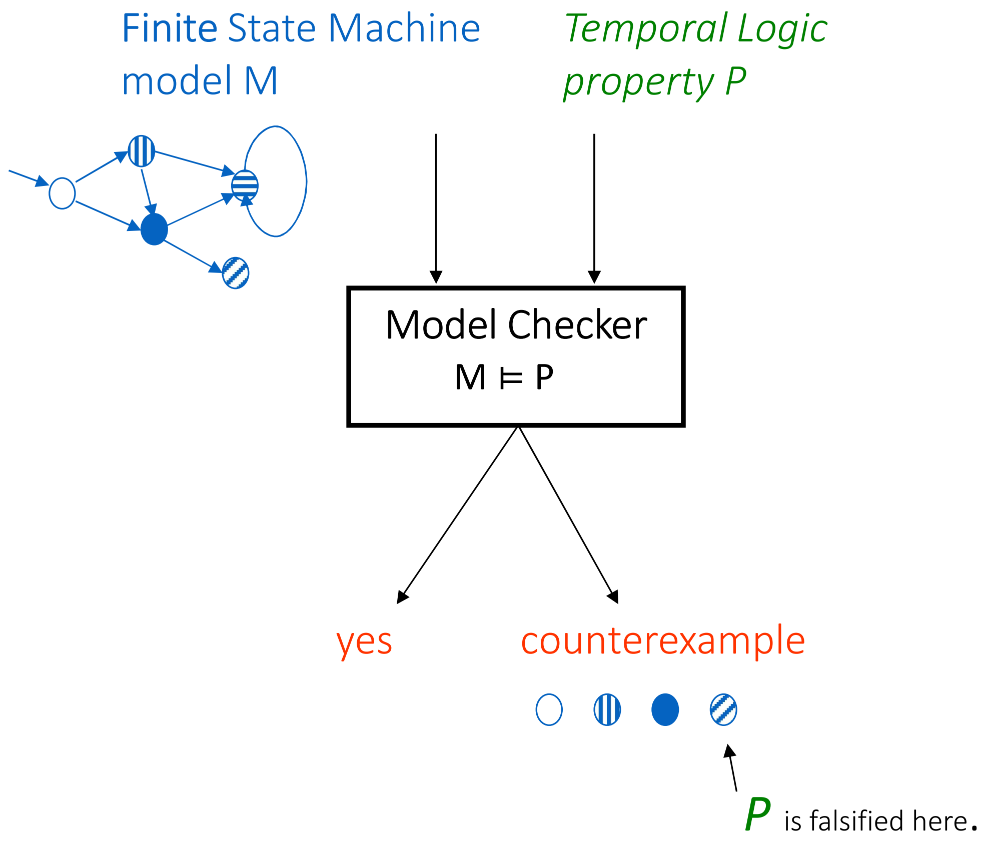
M: program (code), protocol, abstract model of concurrent or distributed system

⊨: logics and tools, e.g., model checkers, theorem provers, Satisfiability Modulo Theories (SMT) solvers

P: discrete (Boolean) logic, correctness properties (safety $\Box$ and liveness $\Diamond$)

E: system environment

Model Checking Primer



… to Verifying AI Systems: Upping the Ante

$\boxed{E, M \models P}$

M: program (code), …, abstract model of system
⊨: model checking, theorem proving, SMT
P : discrete (Boolean) logic
E : model of environment

$\boxed{\mathbf{D,}\ M \models P}$

M: machine-learned model, …, program (code)
⊨: interval analysis, probabilistic logics
P : probabilistic, stochastic
D : **model of data, e.g.,**
stochastic process or distribution that generates the data inputs on which M's outputs need to be verified

$$\boxed{D, M \models P}$$

Two Main Differences

$\boxed{D, M \models P}$

- Need for Probabilistic Reasoning
- The Role of Data
    - Collection and partitioning of data
    - Specifying “unseen” data
    - What do we quantify over?
    - How do we verify?

Need for Probabilistic Reasoning

$\boxed{M \vDash P}$

- M is semantically and structurally different from a typical computer program
    - M is inherently probabilistic
    - Internally, the model itself operates over probabilities and outputs results with assigned probabilities
    - Structurally, M is machine-generated and unlikely to be human-readable, another kind of “intermediate” code
    - Reasoning about uncertainty of M’s environment
- P may be formulated over continuous, not (just) discrete domains, and/or using expressions from probability and statistics.
    - Robustness properties for deep neural networks are characterized as predicates over continuous variables
    - Fairness properties are characterized in terms of expectations with respect to a loss function over reals
    - Differential privacy is defined in terms of a difference in probabilities with respect to a (small) real value
- $\vDash$: Probabilistic logics and hybrid logics
    - Need scalable and/or new verification techniques that work over reals, non-linear functions, probability distributions, stochastic processes, and so on.

Models: Hybrid Automata [Henzinger 1996]

Tools: HyTech, CheckMate, CEGAR+, PHAVer, SpaceEx, …

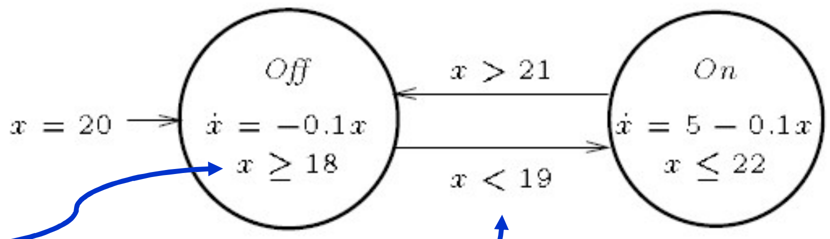


Figure 1: Thermostat automaton

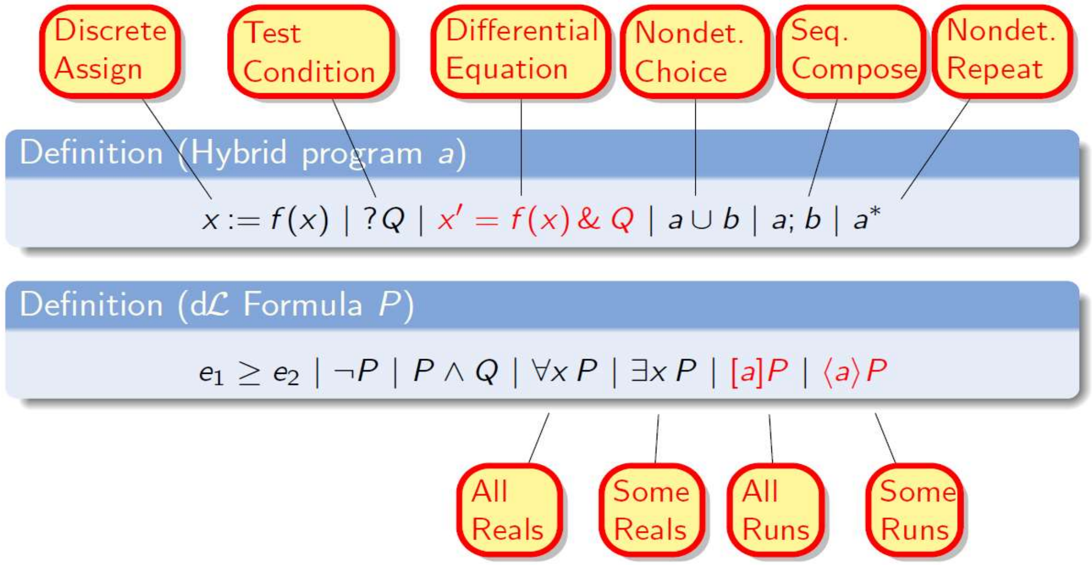
Continuous behavior described by differential equations (here, flow conditions)

Invariant implies that at the latest, it will go on when the temperature falls to 18 degrees.

Jump Condition implies that the heater can go on as soon as the temperature falls below 19 degrees.

Logics: Differential Dynamic Logic [Platzer 2008]

Tool: KeYmaera

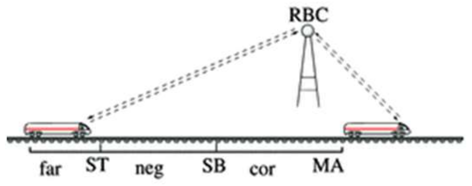


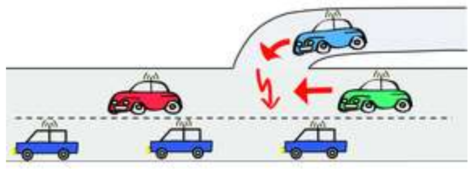
Definition (Hybrid program a)

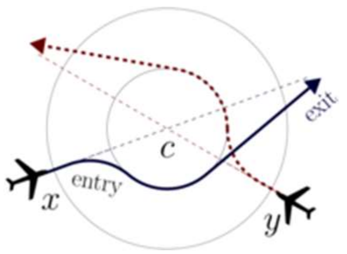
$$x := f(x) \mid ?Q \mid x' = f(x)\,\&\,Q \mid a \cup b \mid a; b \mid a^*$$

Definition (d$\mathcal{L}$ Formula P)

$$e_1 \geq e_2 \mid \neg P \mid P \wedge Q \mid \forall x\, P \mid \exists x\, P \mid [a]P \mid \langle a \rangle P$$







Reasoning about Uncertainty

Probabilistic automata
Probabilistic model checking
Probabilistic logics
Probabilistic programming

```
bool c1, c2;
c1 := Bernoulli(0.5);
c2 := Bernoulli(0.5);
observe(c1 || c2);
```

The Role of Data, D

$\boxed{D, M \models P}$

available data: data at hand, used for training and testing

unseen data: data over which M needs (or is expected) to operate
without having seen it before

Collection and Partitioning Data

$\boxed{D, M \models P}$

- How do we partition an available (given) dataset into a training set and a test set? What guarantees can we make of this partition with respect to a desired property P, in building a model M?

- How much data suffices to build a model M for a given property P? Does adding more data to train or test M make it more robust, fairer, etc. or does it not have an effect with respect to the property P? What new kind of data needs to be collected if a desired property does not hold?

Specifying Unseen Data

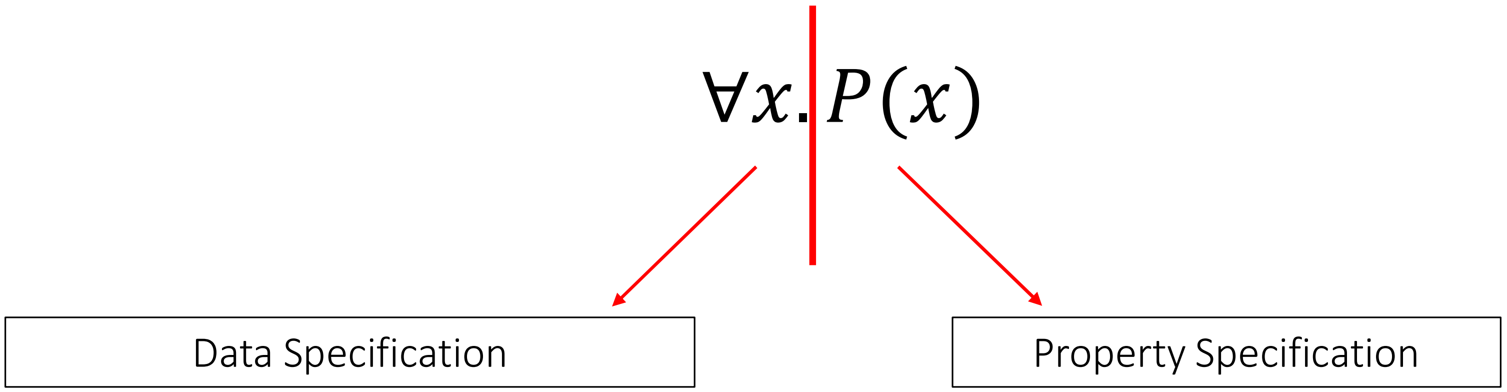
$D, M \models P$

- How do we specify the data and/or characterize properties of the data?
  - Specify D as a stochastic process or data distribution (e.g., via its parameters).
  - Probabilistic programming languages, e.g., Stan, used to specify statistical models
  - But what of large real-world datasets that do not fit common statistical models or which have thousands of parameters?
- Breaking the circular reasoning
  - To specify unseen data, we need to make certain assumptions about the unseen data. Would these assumptions not then be the same as those we would make to build the model M in the first place? That is, *how can we trust the specification of D?*
  - Approaches: (1) repertoire of statistical tools (see later slide); (2) assume that an initial specification is small or simple enough that it can be checked by (say, manual) inspection; then we use this specification to bootstrap an iterative refinement process (akin to counterexample-guided-abstraction-and-refinement in formal methods).
- How does the specification of unseen data relate to the specification of the data on which M was trained and tested?

What Do We Quantify Over?

$\boxed{\text{E, M} \vDash \text{P}}$ In traditional formal methods, we strive to prove $\forall x. P(x)$

$\boxed{\text{D, M} \vDash \text{P}}$ but for AI systems, we do not expect M to work for all input data or for all datasets D.



x

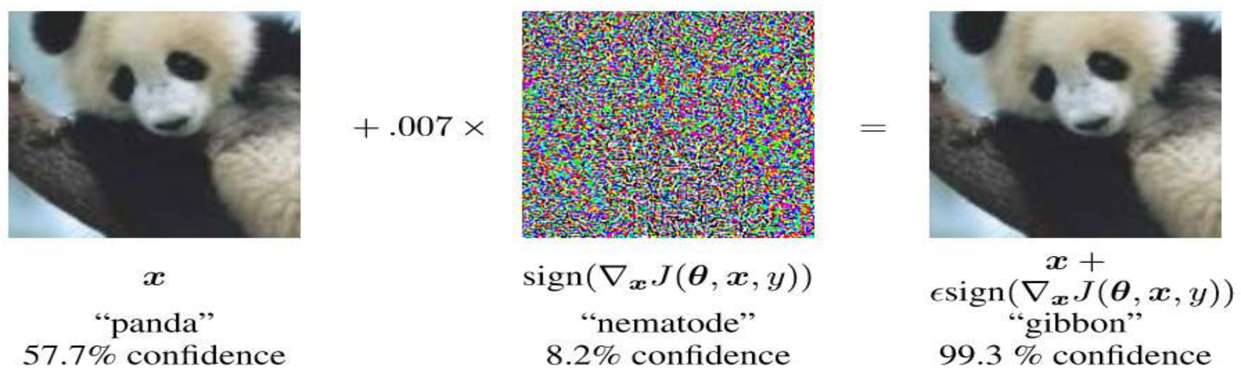
$\forall x$

$$\forall x.\, P(x)$$

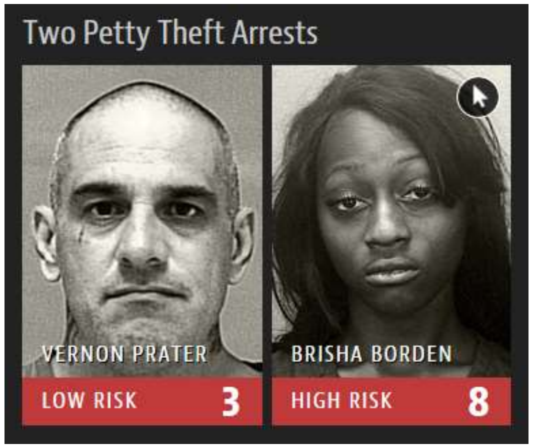
Data Specification	Property Specification
$x \sim D$	**Fairness**, e.g., statistical parity on a given (single) data distribution Example: COMPAS recidivism dataset
$\boxed{x \sim D, \forall D \in C}$	**Fairness**, e.g., nearby distributions **Robustness**, e.g., semantic perturbation
$x \sim D, \forall D$	**Robustness**, e.g., any arbitrary norm-bounded perturbation Example: changing pixels to an image

What Do We Quantify Over?

- How can we specify the class of distributions over which P should hold for a given M? It might be property-dependent.
  - For robustness, in the adversarial machine learning setting, we might want to show that M is robust to all norm-bounded perturbations D. More interestingly, we might want to show M is robust to all “semantic” or “structural” perturbations for the task at hand. For example, computer vision.


$+ .007 \times$
$=$
$\boldsymbol{x}$
"panda"
57.7% confidence
$\text{sign}(\nabla_{\boldsymbol{x}} J(\boldsymbol{\theta}, \boldsymbol{x}, y))$
"nematode"
8.2% confidence
$\boldsymbol{x} + \epsilon\text{sign}(\nabla_{\boldsymbol{x}} J(\boldsymbol{\theta}, \boldsymbol{x}, y))$
"gibbon"
99.3 % confidence

Robustness and Fairness

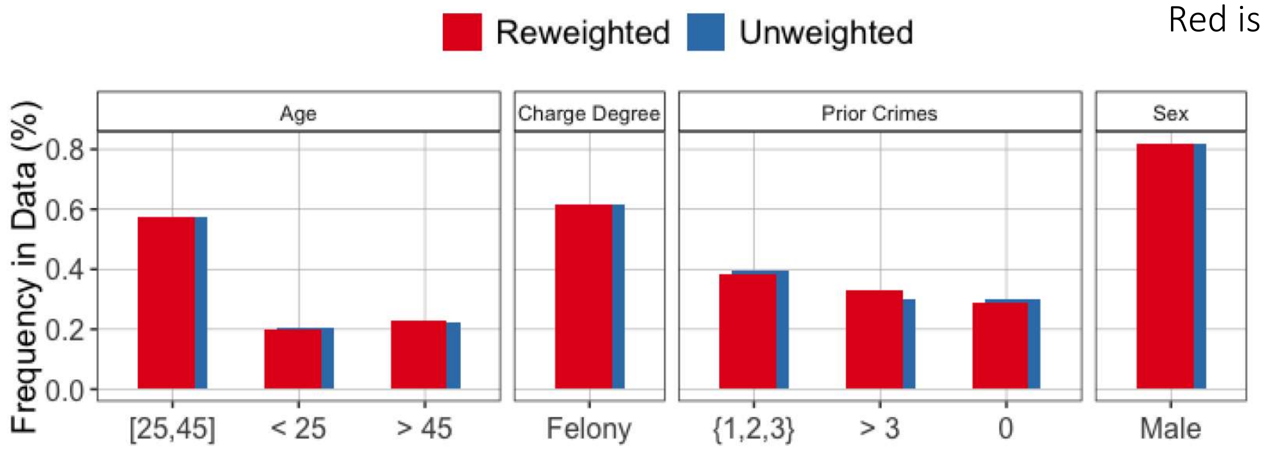


D. Mandal, S. Deng, D. Hsu, S. Jana, and J.M. Wing, "Ensuring Fairness Beyond the Training Data," to appear in *Proceedings of the 34th Conference on Neural Information Processing Systems* (NeurIPS), December 2020. arXiv:2007.06029, July 2020. July 2020.

Robust and Fair Classifiers

- State-of-the-art “fair” classifiers are not robust

Blue is unweighted (“fair”).
Red is reweighted (“unfair”).



- For **fairness**, we might want to show the ML model is fair on a given dataset and all unseen datasets that are “similar” (for some formal notion of “similar”).
- Use on-line algorithm (two-player game) to build a fair classifier that is robust to a *class* of distributions.

The Verification Task $\models$

- How do we check the available data for desired properties? For example, if we want to detect whether a dataset is fair or not, what should we be checking about the dataset?
- If we detect that the property does not hold, how do we fix the model, amend the property, or decide what new data to collect for retraining the model? What is the equivalent of a “counterexample” in the verification of an ML model and how do we use it?
- How do we exploit the explicit specification of unseen data to aid in the verification task?
- How can we extend standard verification techniques to operate over data distributions, perhaps taking advantage of the ways in which we formally specify unseen data?

Verification: e.g., Interval Analysis

Technique: Propagate bounds with symbolic intervals [E_{lower}, E_{upper}]

- Works over deep neural networks with ReLUs

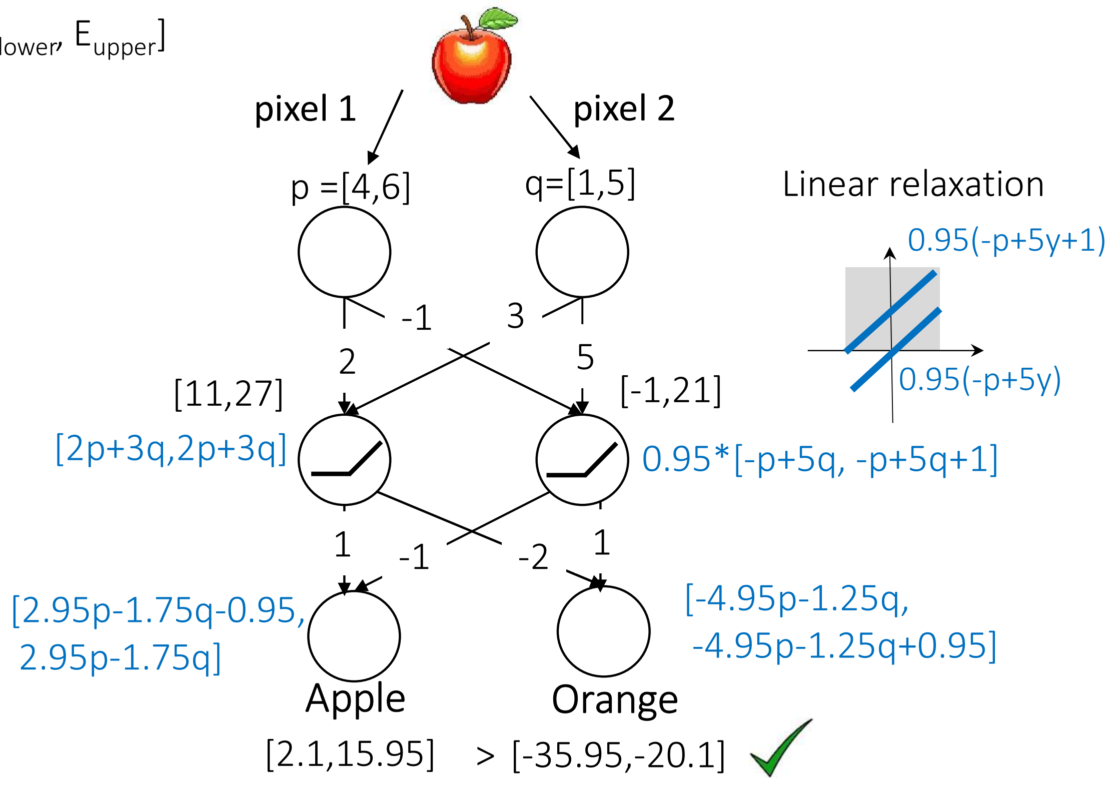
Advantages

- Efficient and sound over-approximation
- Highly parallelizable: *Over 200 times faster than state-of-the-art SMT solver-based approaches*

Applications

- Autonomous vehicles, aircraft control, malware

robustness property:
input ranges: pixel p=[4,6], pixel q=[1,5]
output: Apple

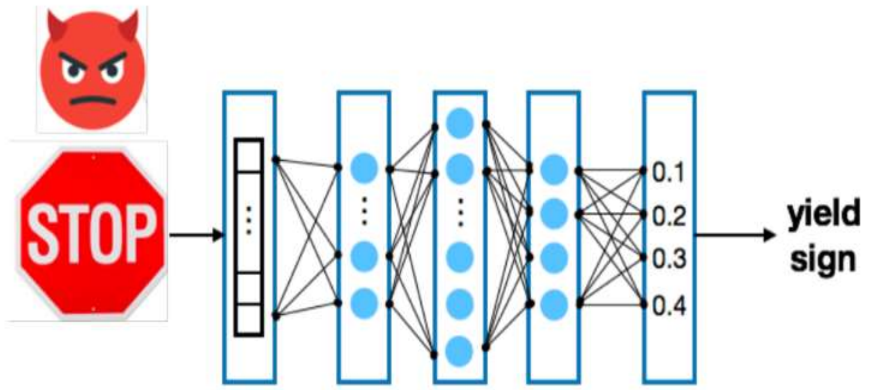


Opportunities for Formal Methods

- Task-specific
- Model synthesis: “Correct-by-construction” approach
- Compositionality
- Statistical methods for model evaluation and model checking
  - sensitivity analysis, prediction scoring, predictive checking, residual analysis, and model criticism

Robust by Construction

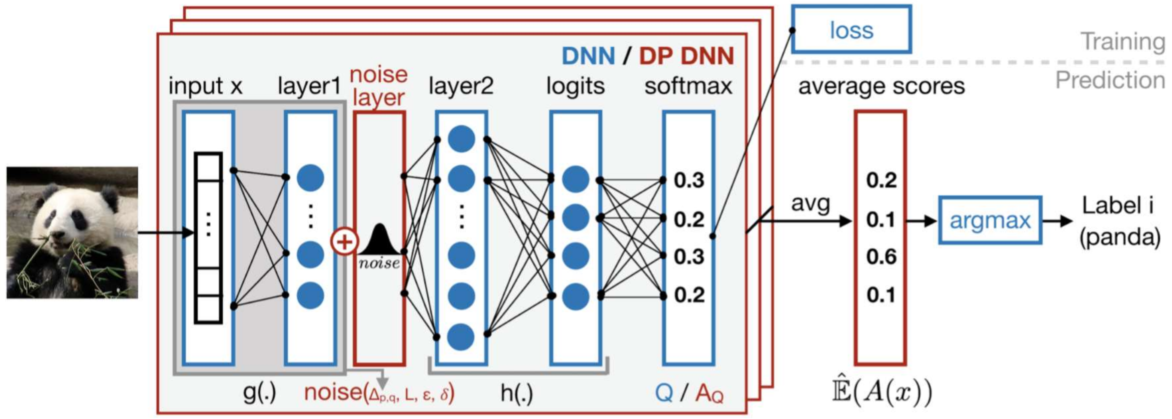
Problem



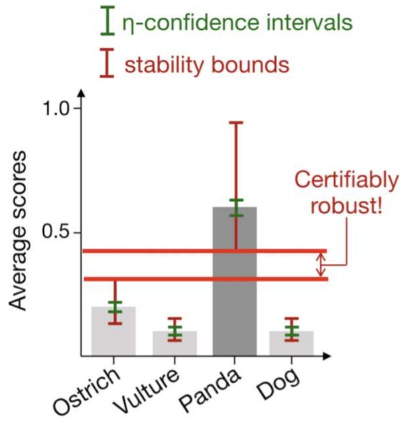
Mathias Lecuyer, Baggelis Atlidakis, Roxana Geambasu, Daniel Hsu, and Suman Jana, "Certified Robustness to Adversarial Examples with Differential Privacy, arXiv:1802.03471v2 , June 26, 2018, to appear IEEE Security and Privacy (''Oakland'') 2019.

Solution Inspired by Differential Privacy

1. Add a noise layer a la Differential Privacy



(a) PixelDP DNN Architecture



(b) Robustness Test Example

2. Provable guarantee from DP says classifier is robust to some degree of input perturbations.

Trustworthy AI meets Formal Methods

$$\boxed{D, M \vDash P}$$

Thank You