
Logic Extraction for Explainable AI

Susmit Jha

Computer Science Laboratory

SRI

July, 2019

AI reaches human-level accuracy on benchmark datasets

Going deeper with convolutions. (Inception) C Szegedy et al, 2014

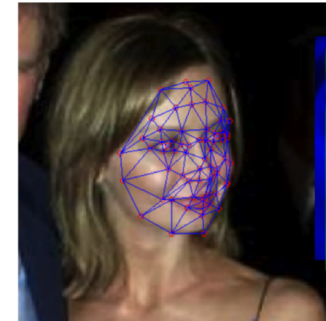
Team	Year	Place	Error (top-5)	Uses external data
SuperVision	2012	1st	16.4%	no
SuperVision	2012	1st	15.3%	Imagenet 22k
Clarifai	2013	1st	11.7%	no
Clarifai	2013	1st	11.2%	Imagenet 22k
MSRA	2014	3rd	7.35%	no
VGG	2014	2nd	7.32%	no
GoogLeNet	2014	1st	6.67%	no



(a) Siberian husky



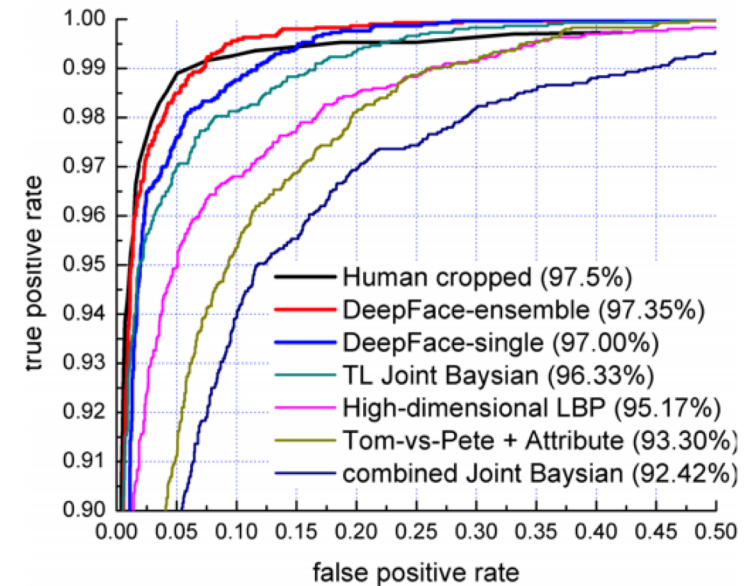
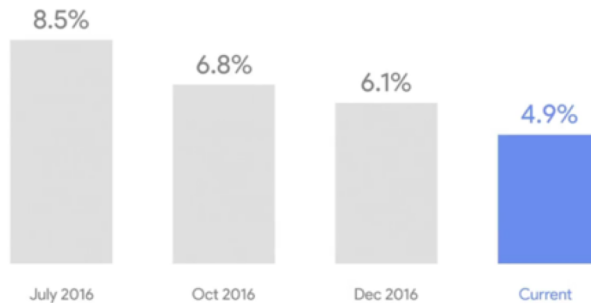
(b) Eskimo dog



Calista_Flockhart_0002.jpg
Detection & Localization

Speech Recognition
Word Error Rate

Switchboard
benchmark

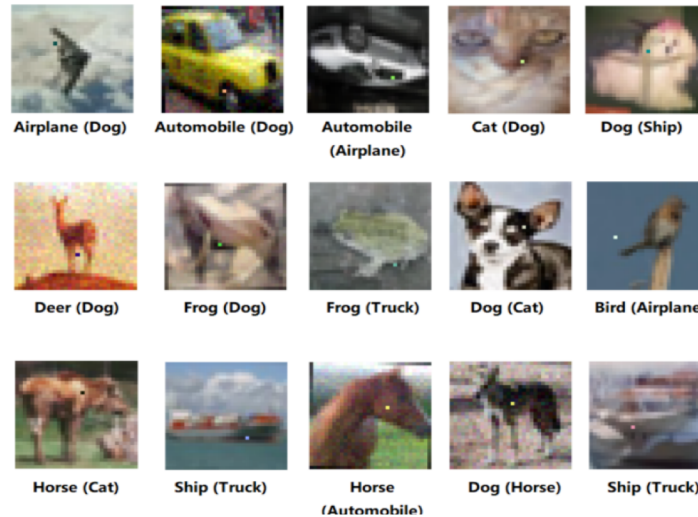


Face Detection. Taigman et al, 2014

Beyond aggregate numbers

Machine learning very susceptible to adversarial attacks.

Szegedy et al, 2013, 2014

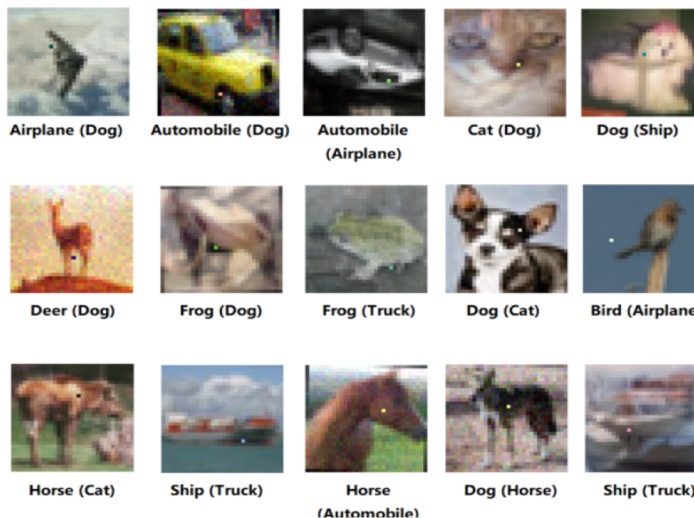


Only allowed to modify the value of 1 pixel.
70.97% of the natural images can be perturbed to at least one target class by modifying just one pixel with 97.47% confidence on average.

Beyond aggregate numbers

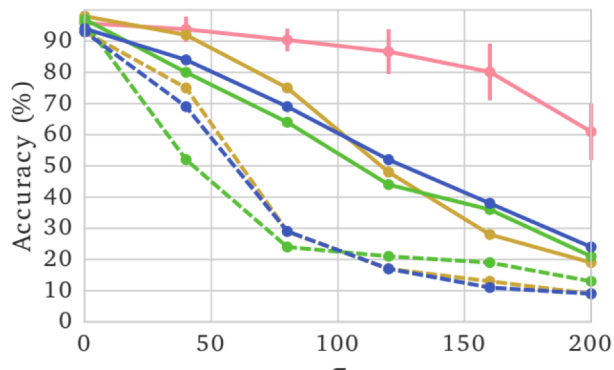
Machine learning very susceptible to adversarial attacks.

Szegedy et al, 2013, 2014



Only allowed to modify the value of 1 pixel.
70.97% of the natural images can be perturbed to at least one target class by modifying just one pixel with 97.47% confidence on average.

Actual Class	x	\tilde{x}	$\text{pred}_{\text{human}}(\tilde{x})$	$\text{pred}_{\text{VGG16}}(\tilde{x})$
Basenji			Basenji	Yorkshire Terrier
Dalmatian			Dalmatian	Basenji
Border Collie			Border Collie	German Shepherd
Staffordshire Bullterrier			Staffordshire Bullterrier	Border Collie

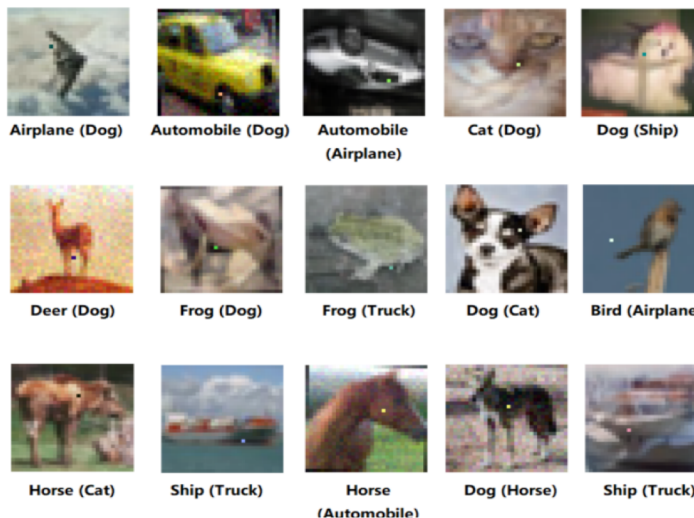


Low robustness to benign noise Dodge et al. 2017

Beyond aggregate numbers

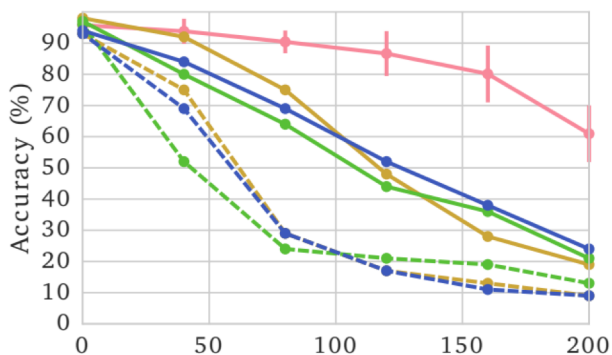
Machine learning very susceptible to adversarial attacks.

Szegedy et al, 2013, 2014



Only allowed to modify the value of 1 pixel.
70.97% of the natural images can be perturbed to at least one target class by modifying just one pixel with 97.47% confidence on average.

Actual Class	x	\tilde{x}	pred _{human} (\tilde{x})	pred _{VGG16} (\tilde{x})
Basenji			Basenji	Yorkshire Terrier
Dalmatian			Dalmatian	Basenji
Border Collie			Border Collie	German Shepherd
Staffordshire Bullterrier			Staffordshire Bullterrier	Border Collie



Statistically good doesn't mean logically/conceptually good.



Low robustness to benign noise Dodge et al. 2017

Understanding deep learning requires rethinking generalization.
C. Zhang, S. Bengio, M. Hardt, B. Recht, O. Vinyals

TRINITY: Trust, Resilience and Interpretability

Trust

- Global Assume/Guarantee Contracts on DNNs
- Closed-loop verification of NN controllers
- Extracting and Integrating Temporal Logic into Learned Control



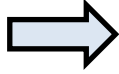
Resilience

- Adversarial Robustness

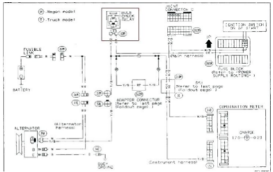
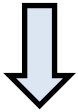
Interpretability

- Explaining Decisions as Sparse Boolean Formula Learning
- Inverse Reinforcement Learning of Temporal Specifications

TRINITY: Trust, Resilience and Interpretability

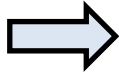


Demonstrations



Specifications

TRINITY: Trust, Resilience and Interpretability



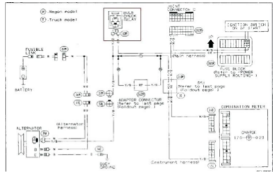
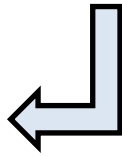
TRUSTED



Demonstrations

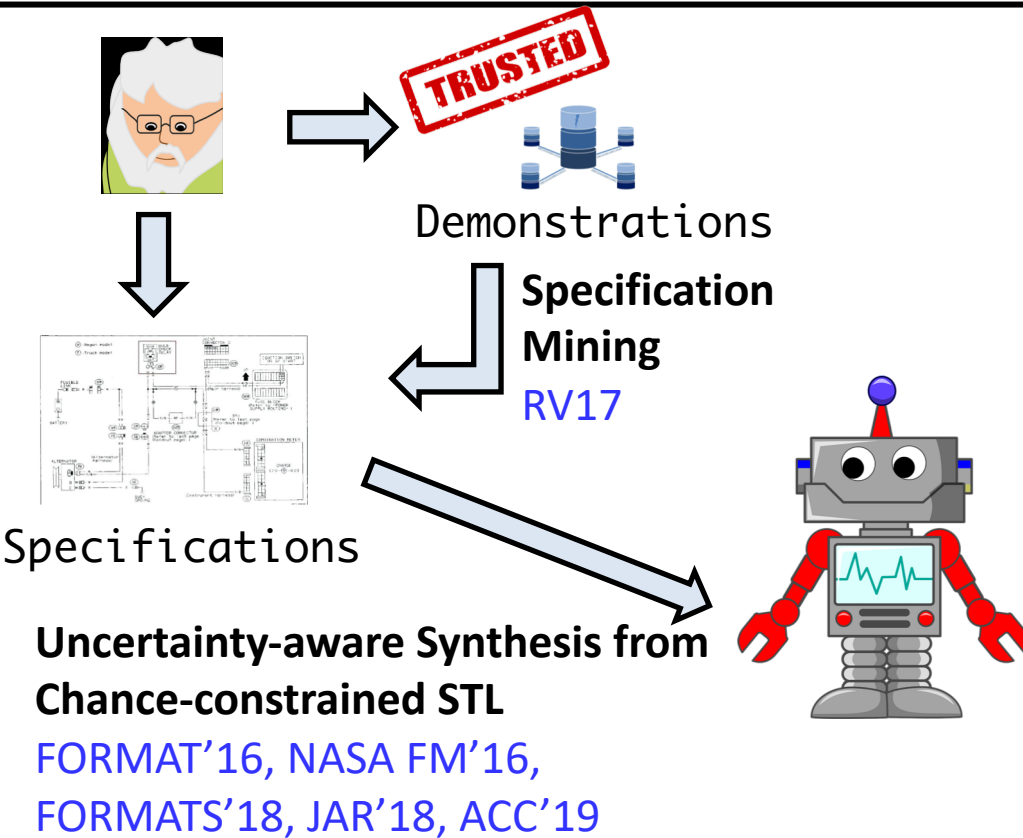
Specification
Mining

RV17

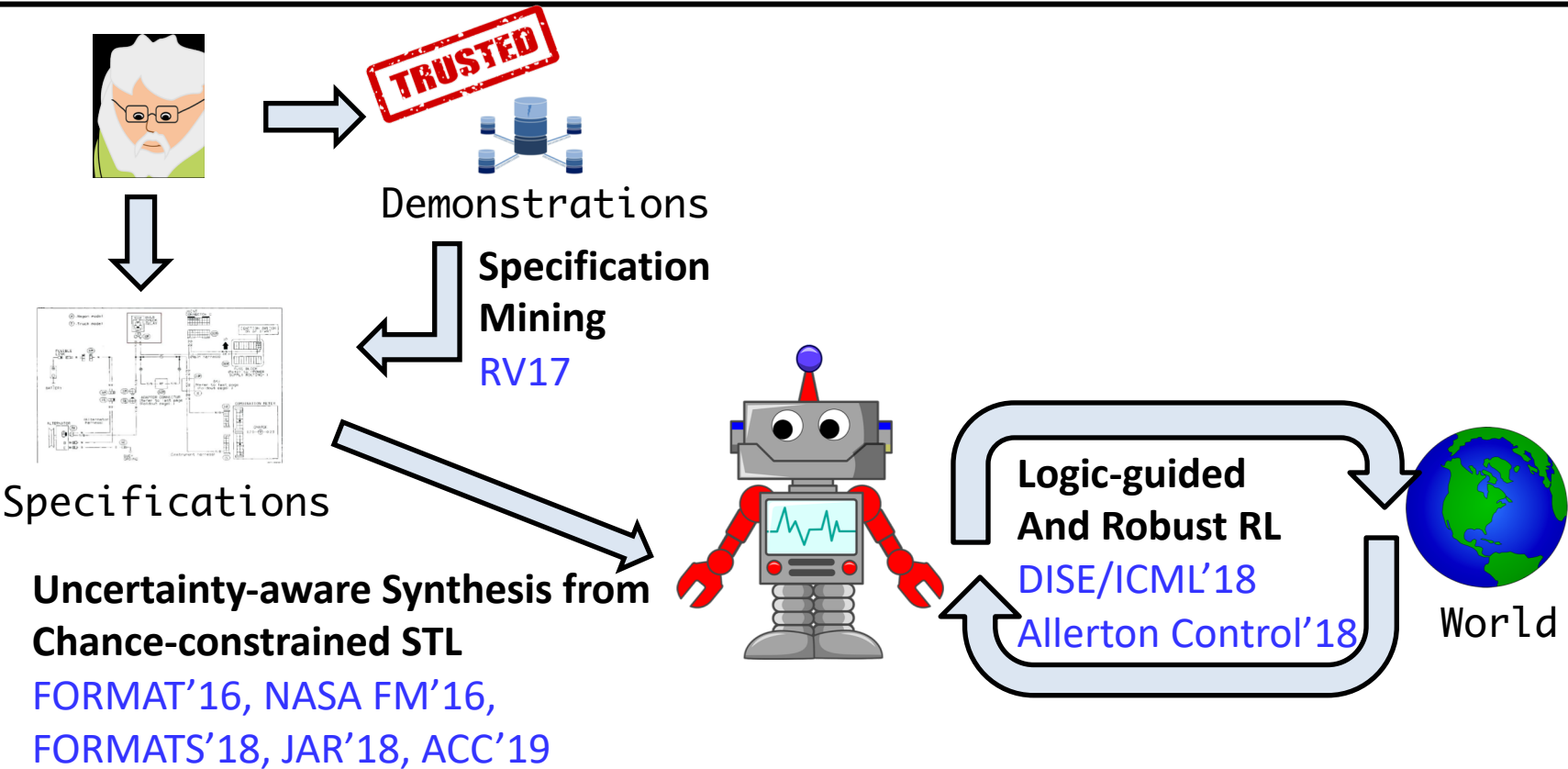


Specifications

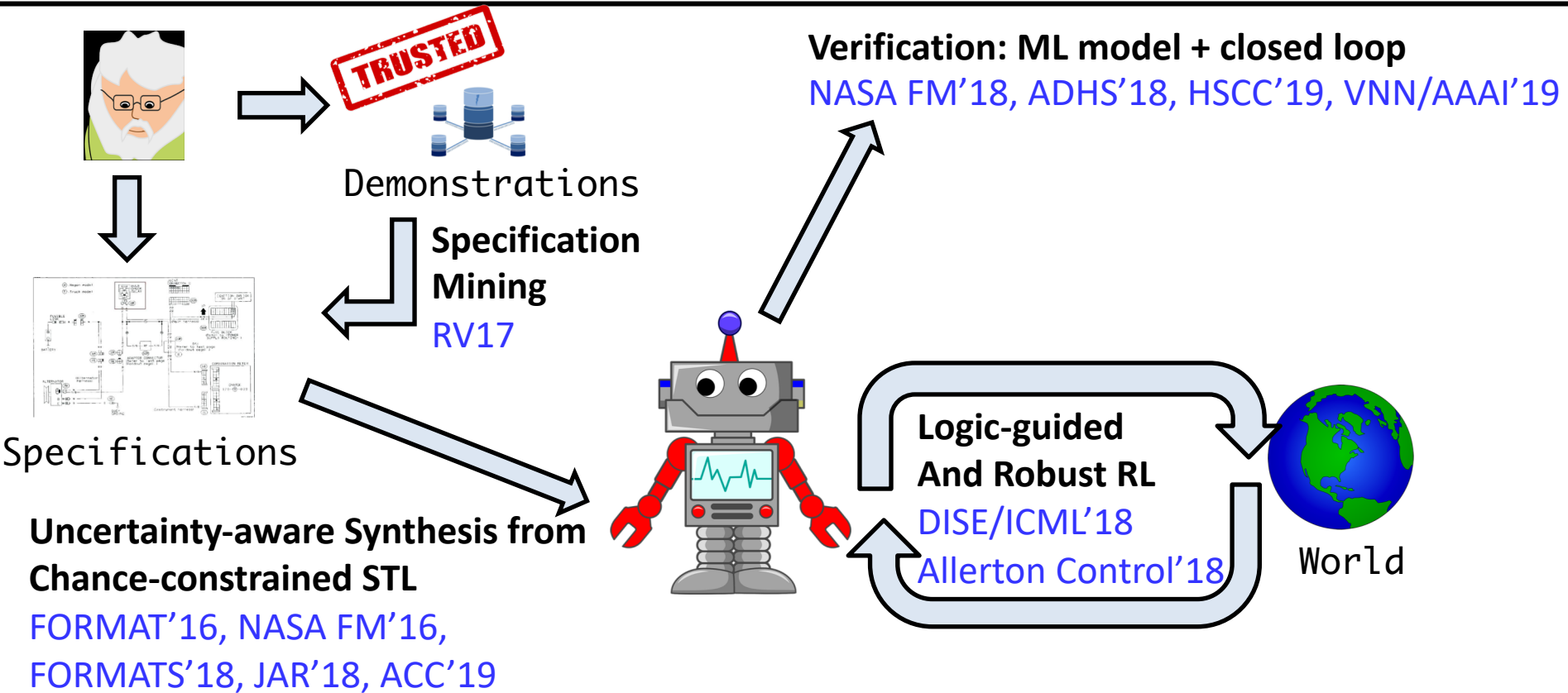
TRINITY: Trust, Resilience and Interpretability



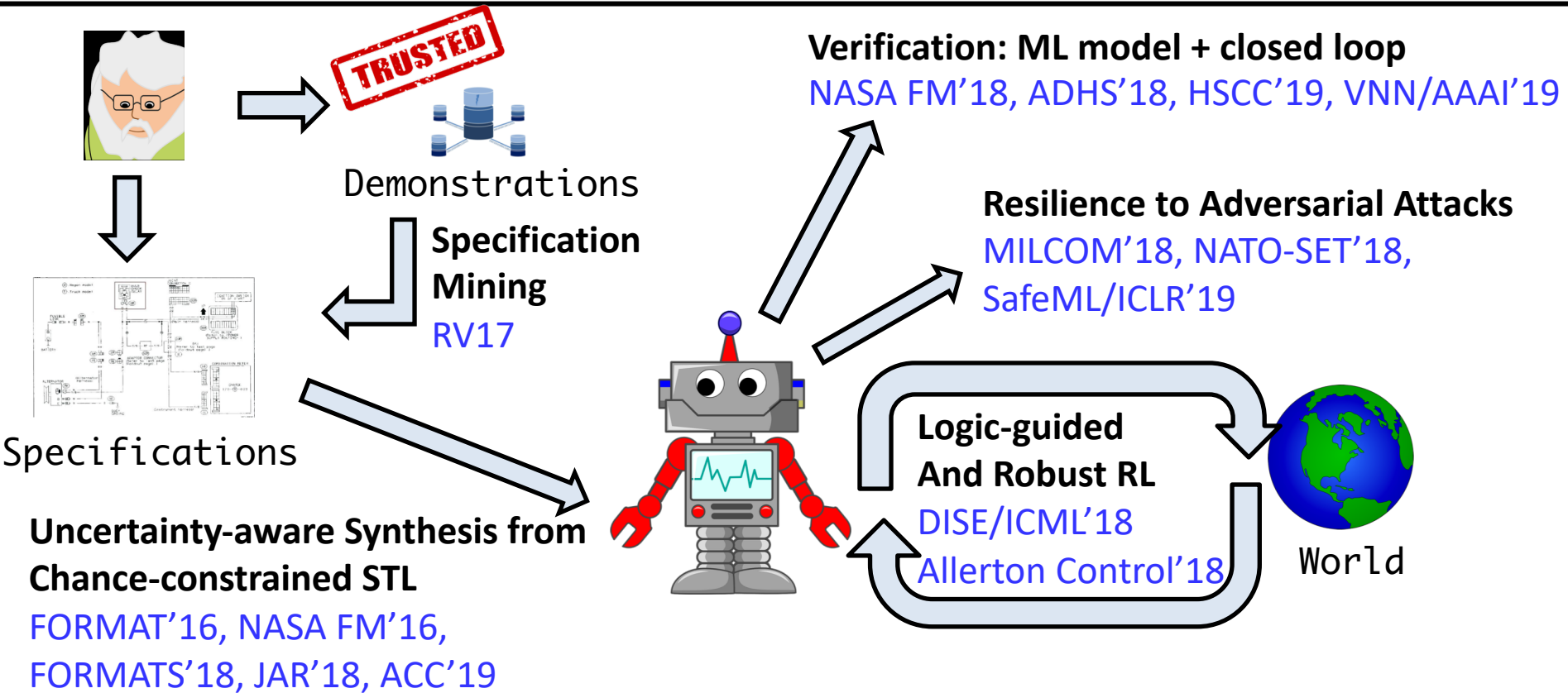
TRINITY: Trust, Resilience and Interpretability



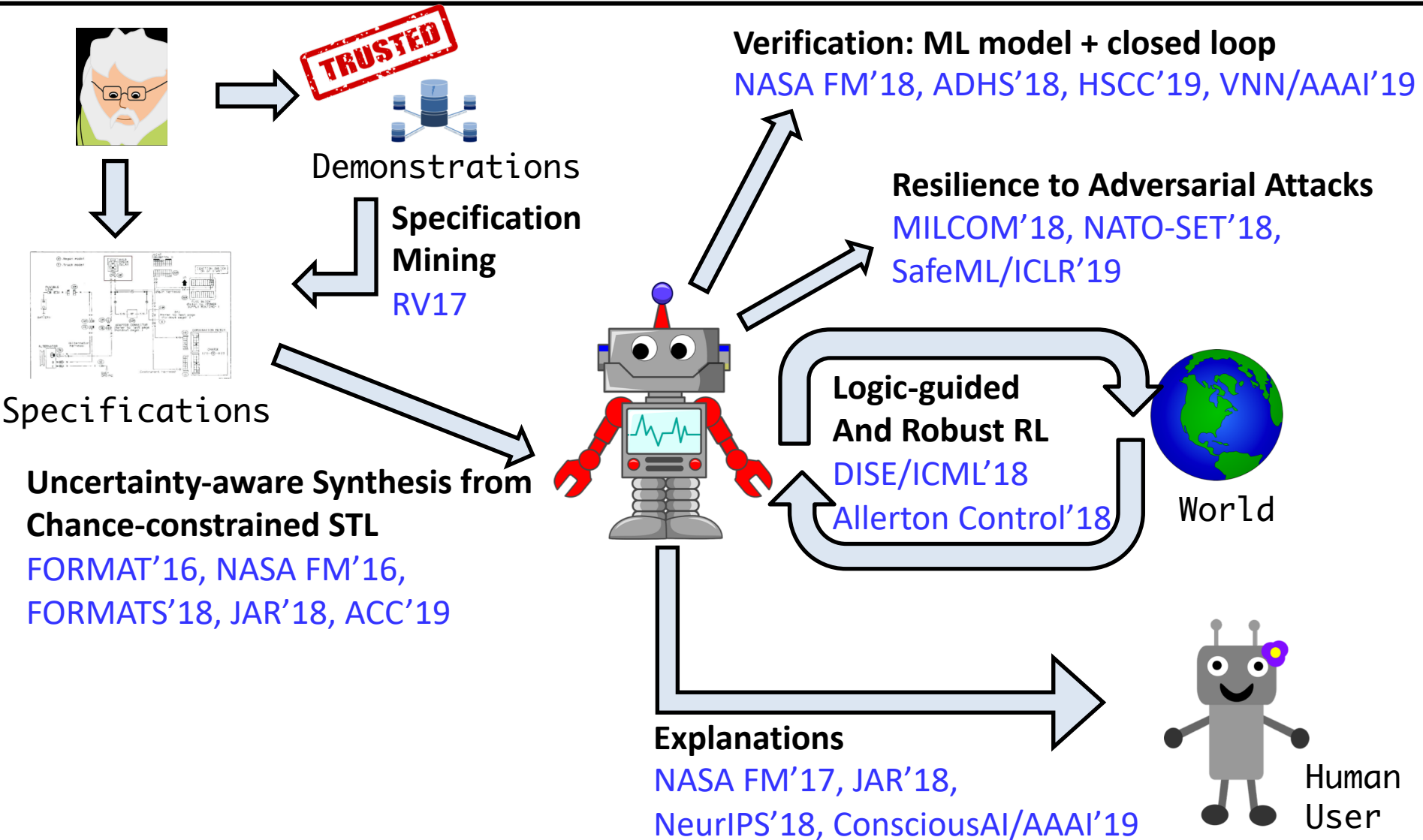
TRINITY: Trust, Resilience and Interpretability



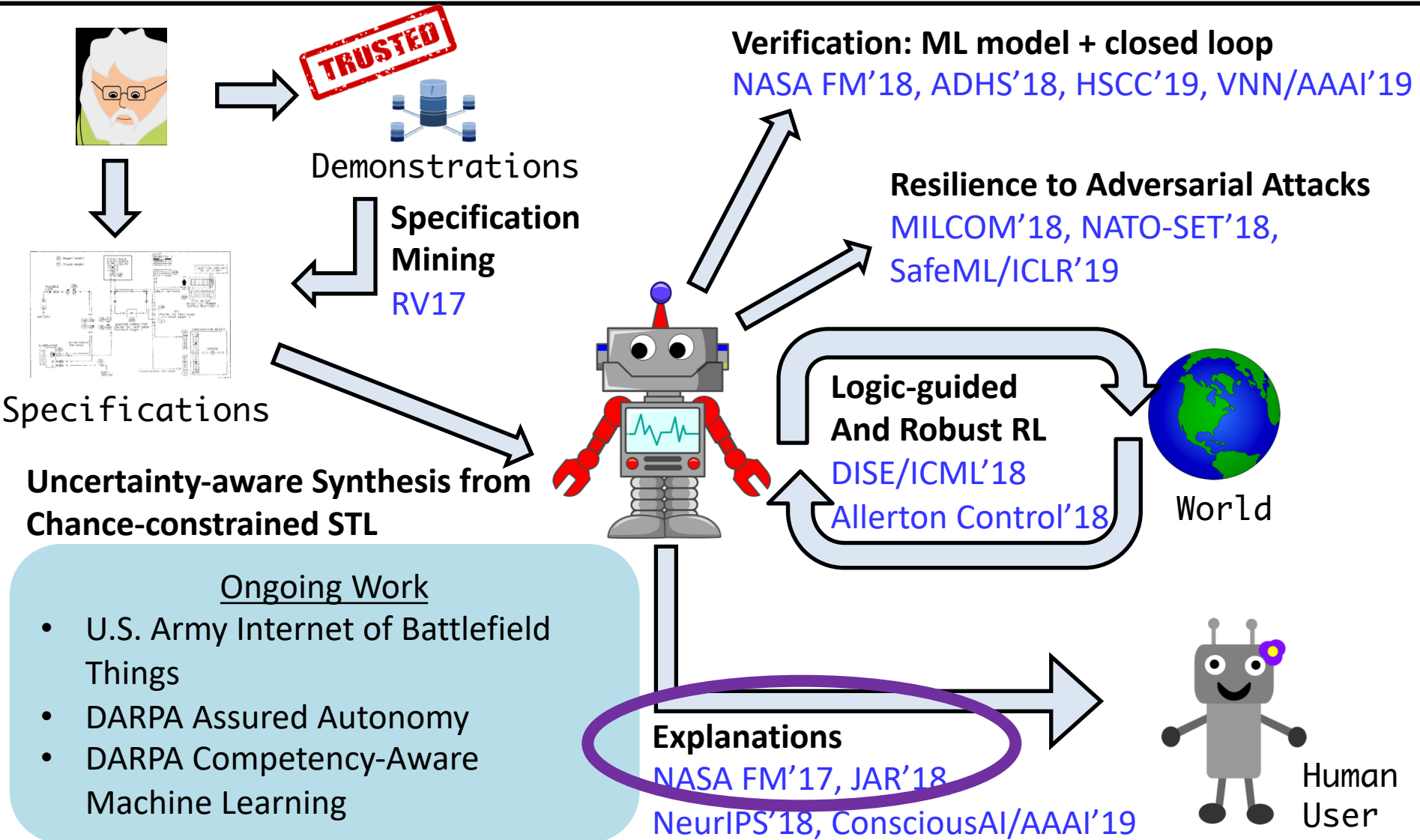
TRINITY: Trust, Resilience and Interpretability



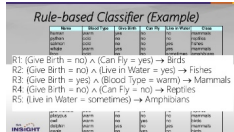
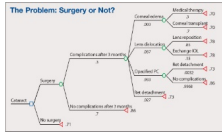
TRINITY: Trust, Resilience and Interpretability



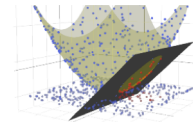
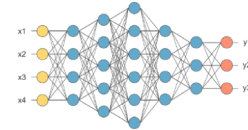
TRINITY: Trust, Resilience and Interpretability



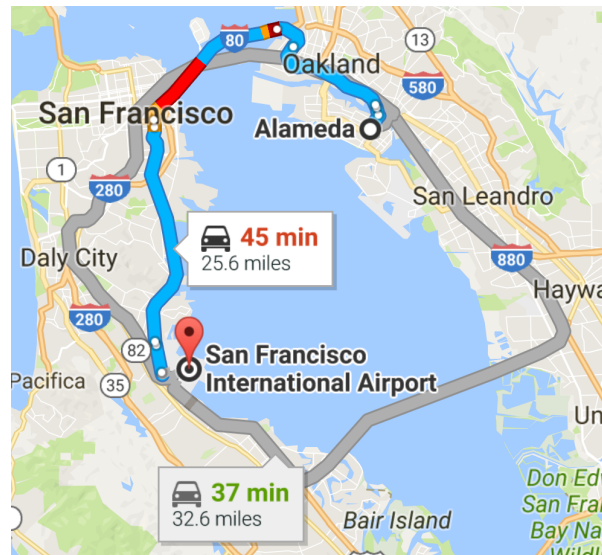
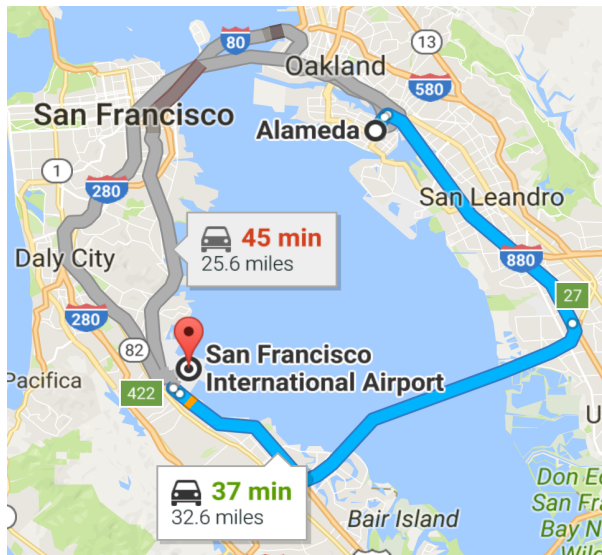
Need for explanation



Interpretable but less scalable:
Decision Trees, Linear Regression



Scalable but less interpretable :
Neural Networks, Support Vector
Machines



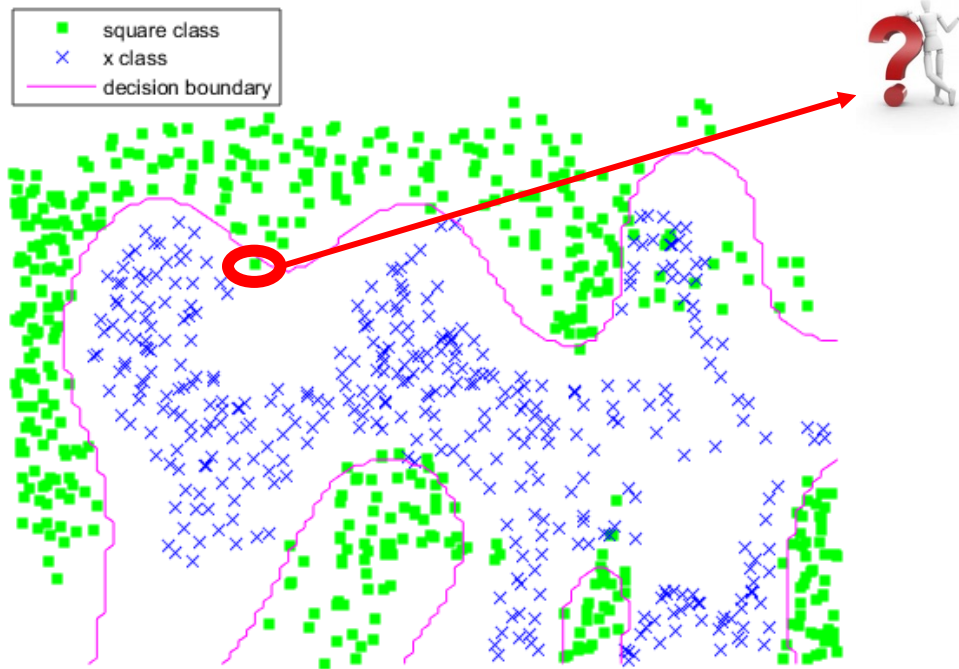
- This route is faster.
- There is traffic on Bay Bridge.
- There is an accident just after Bay Bridge backing up traffic.



Why did we take the San Mateo bridge instead of the Bay Bridge ?

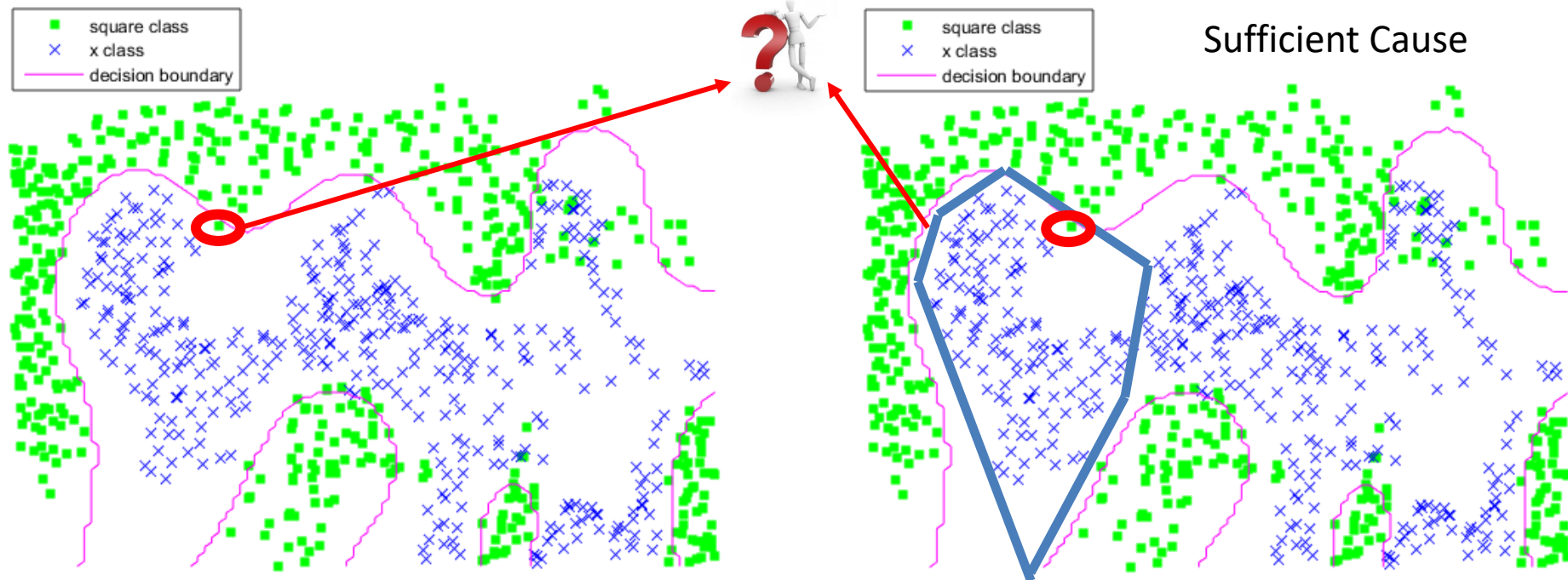
Local Explanations of Complex Models

Not reverse engineering an ML model but finding explanation locally for one decision.



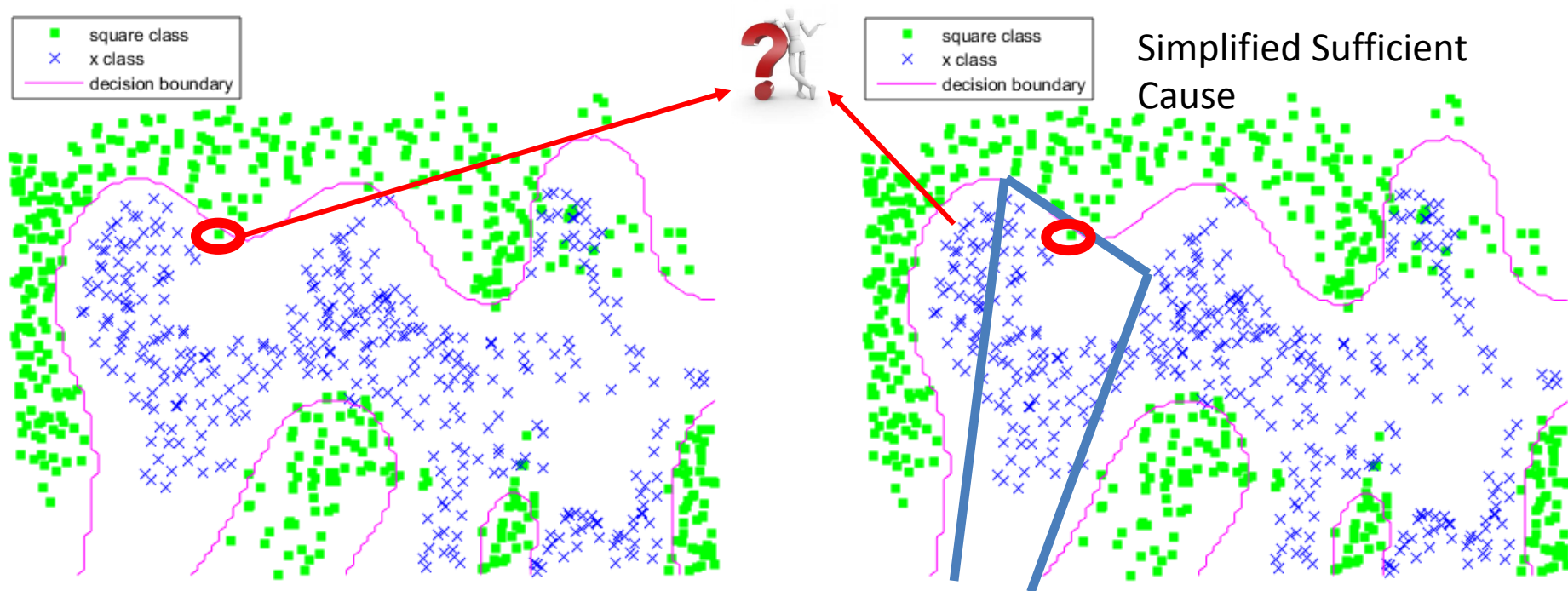
Local Explanations of Complex Models

Not reverse engineering an ML model but finding explanation locally for one decision.



Local Explanations of Complex Models

Not reverse engineering an ML model but finding explanation locally for one decision.



Local Explanations in AI

Not reverse engineering an ML model but finding explanation locally for one decision.

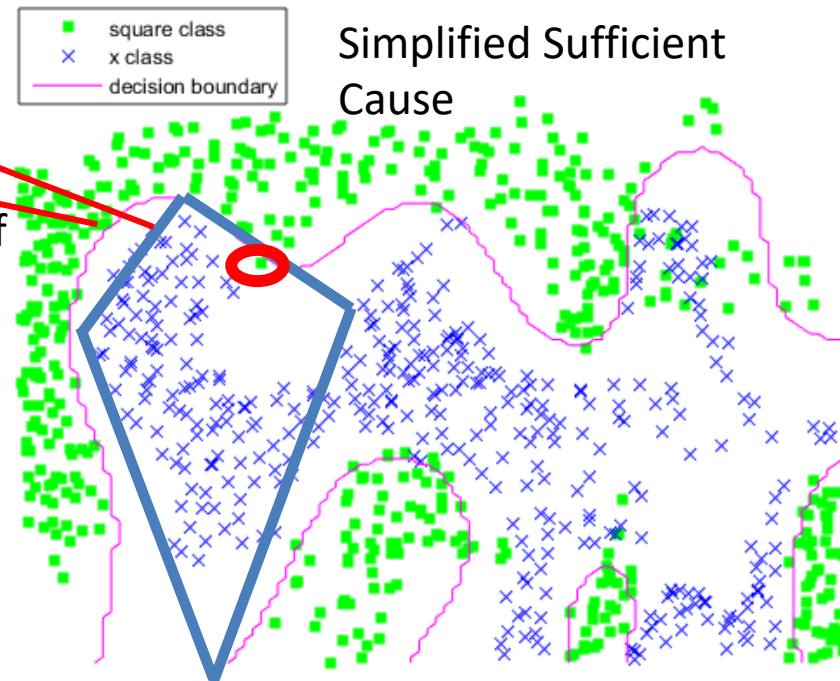
$$\xi(x) = \operatorname{argmin}_{g \in G} \mathcal{L}(f, g, \pi_x) + \Omega(g)$$

$\mathcal{L}(f, g, \pi_x)$ Measure of how well g approximates f

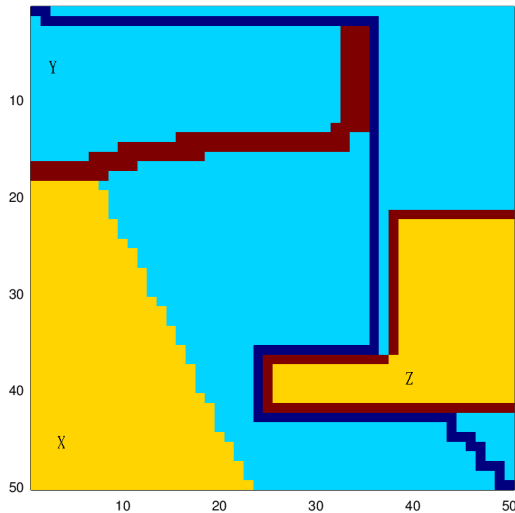
$\Omega(g)$ Measure of complexity of g

Formulation in AI:

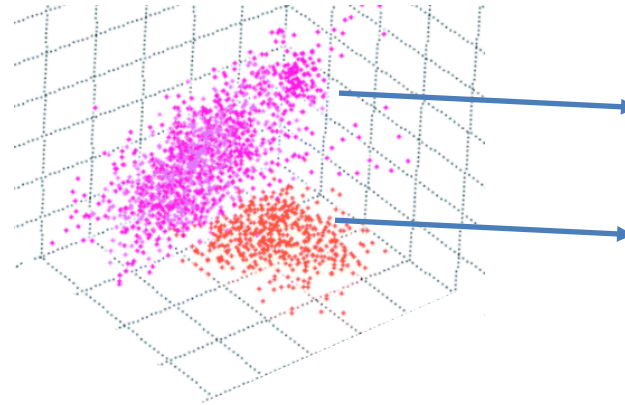
- Ribeiro, Marco Tulio, Sameer Singh, and Carlos Guestrin. "Why Should I Trust You?: Explaining the Predictions of Any Classifier." *International Conference on Knowledge Discovery and Data Mining*. ACM, 2016.
- Hayes, Bradley, and Julie A. Shah. "Improving Robot Controller Transparency Through Autonomous Policy Explanation." *International Conference on Human-Robot Interaction*. ACM, 2017.



Model Agnostic Explanation through Boolean Learning



Why does the path not go through Green?



Let each point in k-dimensions (for some k) correspond to a map.

Maps in which optimum path goes via green

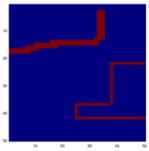
Maps in which optimum path does not go via green

Find a Boolean formula ϕ such that

$$\phi \Leftrightarrow \text{Path contain } z$$

$$\phi \Rightarrow \text{Path contain } z$$

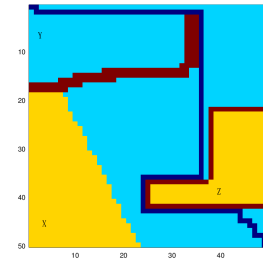
Explanations as Learning Boolean Formula



```

Algorithm 1:  $A^*$ 
Input: start, goal( $x_g, y_g$ ), open( $x_s, y_s$ )
Output: path
1 if goal(start) = true then return path(start)
2
3 open ← start
4 closed ←  $\emptyset$ 
5 while open  $\neq \emptyset$  do
6   sort(open)
7    $n_s = \text{pop}(open)$ 
8    $kids = \text{expand}(n_s)$ 
9   for all the kid  $\in kids$  do
10     $dist_f = (g(n_s) + 1) + h(kid)$ 
11    if goal(kid) = true then return path(Path(n_s, kid))
12    if  $dist_f < cost(n_s)$  then open ← kid
13  closed ←  $n_s$ 
14 return  $\emptyset$ 
    
```

A^*



$\Phi_{explain}$:
Using explanation vocabulary
Ex: Obstacle presence

Φ_{query} :
Some property of the output
Ex: Some cells not selected

$$\Phi_{explain} \Rightarrow \Phi_{query}$$

$$\Phi_{explain} \Leftrightarrow \Phi_{query}$$

How difficult is it? Boolean formula learning

$\phi_{explain} \Rightarrow \phi_{query}$

$\phi_{explain} \Leftrightarrow \phi_{query}$

50x50 grid has 2^{2500} possible explanations even if vocabulary only considers presence/absence of obstacles.

Scalability: Usually the feature space or vocabulary is large. For a map, its order of features in the map. For an image, it is order of the image's resolution.

Guarantee: Is the sampled space of maps enough to generate the explanation with some quantifiable probabilistic guarantee?

How difficult is it? Boolean formula learning

$\Phi_{explain} \Rightarrow \Phi_{query}$

$\Phi_{explain} \Leftrightarrow \Phi_{query}$

50x50 grid has $2^{2^{50 \times 50}}$ possible explanations even if vocabulary only considers presence/absence of obstacles.

Scalability: Usually the feature space or vocabulary is large. For a map, its order of features in the map. For an image, it is order of the image's resolution.

Guarantee: Is the sampled space of maps enough to generate the explanation with some quantifiable probabilistic guarantee?

On PAC learning algorithms for rich Boolean function classes

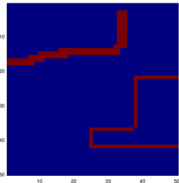
Rocco A. Servedio*

Department of Computer Science
Columbia University
New York, NY U.S.A.
rocco@cs.columbia.edu

Theoretical Result:

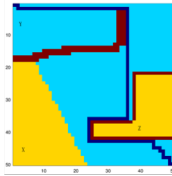
Learning Boolean formula even approximately is hard. 3-DNF is not learnable in Probably Approximately Correct framework unless $RP = NP$.

Two Key Ideas



```
Algorithm 1: A*  
Input: start, goal(x), P(x), openSet  
Output: null  
1 If goal(x) = true then return solvePath(start)  
2  
3 openSet ← start  
4 closed ← ∅  
5 While openSet ≠ ∅ do  
6     u ← choose(u) in openSet  
7     u ← solvePath(u)  
8     Add all the neighbors to openSet  
9     If solvePath(u) = true then return solvePath(u)  
10    If solvePath(u) = false then return null  
11 closed ← u  
12 return null
```

A*



ϕ_{query} :
Some property of the output
Ex: Some cells not selected

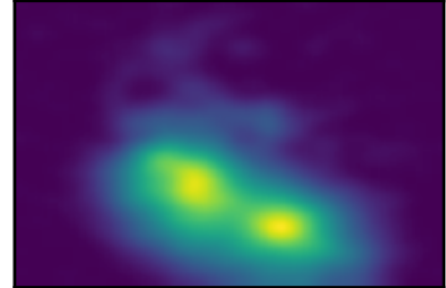
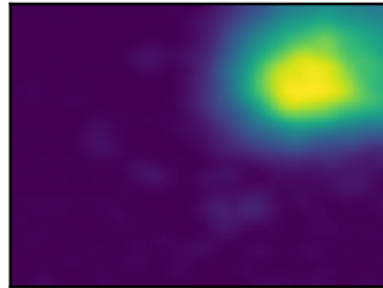
1. Vocabulary is large.
2. How many samples (and what distribution) to consider for learning explanation ?
3. Learning Boolean formula with PAC guarantees is hard.



Active learning Boolean formula $\phi_{explain}$ and not learning from fixed sample.

Explanations are often short and involve only few variables !

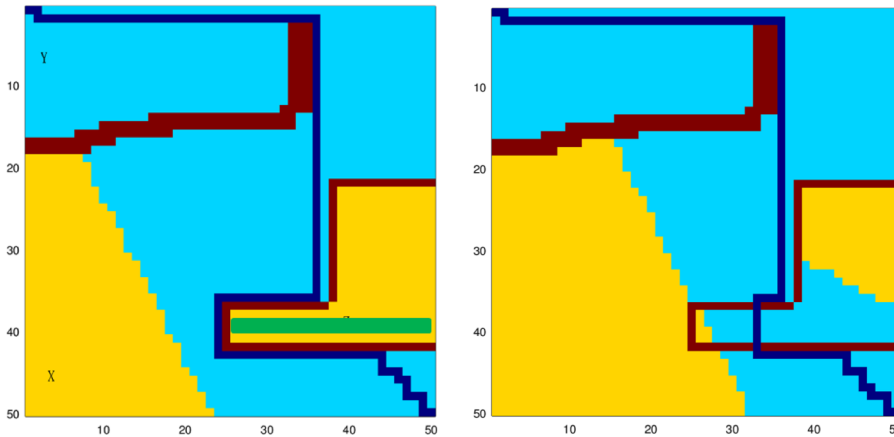
Two Key Ideas



Active learning Boolean formula $\phi_{explain}$ and not learning from fixed sample.

Explanations are often short and involve only few variables !

Two Key Ideas



Involves only two variables.
If we knew which two, we had
only $2^{2^2} = 16$
possible explanations.

**How do we find these relevant
variables?**

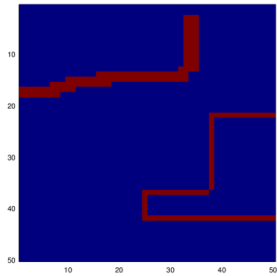


Active learning Boolean formula $\phi_{explain}$ and not learning from fixed sample.

Explanations are often short and involve only few variables !

Actively Learning Boolean Formula

Oracle



```

Algorithm 1: A*
Input: start, goal(x), h(x), open(0)
Output: path
1 if goal(start) = true then return path(start)
2
3 open ← start
4 closed ← ∅
5 while open ≠ ∅ do
6   sort(open)
7   n ← open.pop()
8   kids ← expand(n)
9   for all the kid ∈ kids do
10    | h(kid) = (g(n) + 1) + h(kid)
11    | If goal(kid) = true then return path(Path(n), kid)
12    | If h(kid) < closed then open ← kid
13   closed ← n
14 return ∅
    
```

ϕ_{query} :
Some property of the output
Ex: Some cells not selected



ϕ_{query} :
Some property of the output
Ex: Some cells not selected

ϕ
Evaluates assignments and returns T,F



Assignments to V
m1 = (0,0,0,1,1,0,1)
m2 = (0,0,1,1,0,1,0)

$\phi_{explain}(V)$:
Using explanation vocabulary
Ex: Obstacle presence



Actively Learning Relevant Variables

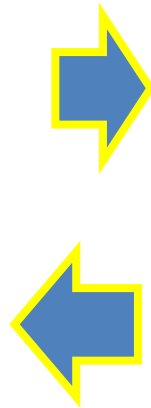
Find U such that $\phi_{\text{explain}}(V) \equiv \phi_{\text{explain}}(U)$ where $|U| \ll |V|$

ϕ_{explain} is sparse

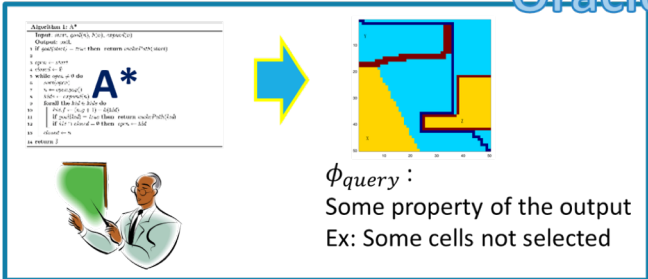
Actively Learning Relevant Variables

Find U such that $\phi_{\text{explain}}(V) \equiv \phi_{\text{explain}}(U)$ where $|U| \ll |V|$

Assignments to V
 $m1 = (0,0,0,1,1,0,1)$



Oracle



```
Algorithm 1: A*
Input: start, goal(s), f, g, h, openSet
Output: path
1 if start == goal then return (start)
2 openSet ← {start}
3 while openSet ≠ ∅ do
4   select (u) ← argmin_{u ∈ openSet} f(u)
5   openSet ← openSet \ {u}
6   if u == goal then return (u)
7   for each v in neighbors(u) do
8     if v not in openSet and h(v) < h(u) then
9       g(v) ← g(u) + 1
10      h(v) ← h(v)
11      f(v) ← g(v) + h(v)
12      openSet ← openSet ∪ {v}
13 return ∅
```

ϕ_{query} :
Some property of the output
Ex: Some cells not selected

$m1 : \text{True}$

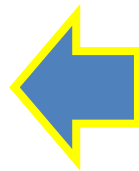
Actively Learning Relevant Variables

Find U such that $\phi_{\text{explain}}(V) \equiv \phi_{\text{explain}}(U)$ where $|U| \ll |V|$

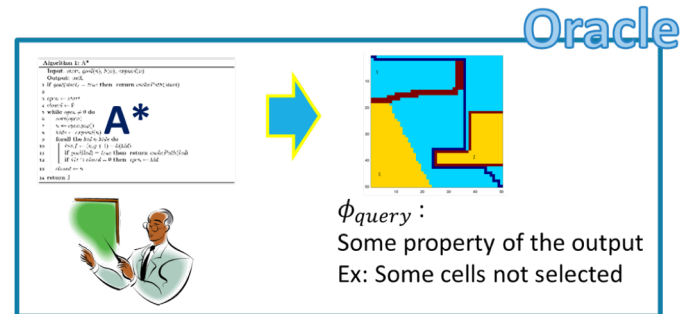
Assignments to V

$m1 = (0,0,0,1,1,0,1)$

$m2 = (0,0,1,1,0,1,0)$



Random Sample Till
Oracle differs



$m1$: True, $m2$: False

Actively Learning Relevant Variables

Find U such that $\phi_{\text{explain}}(V) \equiv \phi_{\text{explain}}(U)$ where $|U| \ll |V|$

Assignments to V

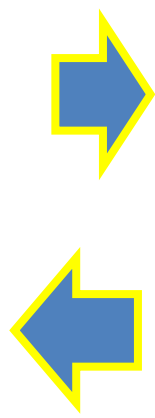
$m_1 = (0,0,0,1,1,0,1)$

$m_2 = (0,0,1,1,0,1,0)$

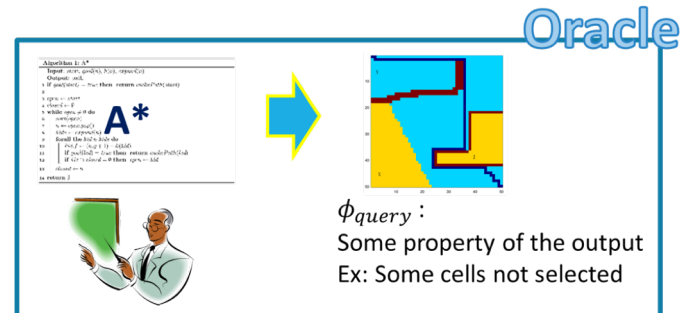
$m_3 = (0,0,0,1,1,1,0)$



Binary Search Over
Hamming Distance



Oracle



```
Algorithm 1: A*
Input: start, goal(s), f, g, h, open(C)
Output: path
1 if start == goal then return start
2 if not open(C) then return failure
3 select s ← argmin_{s ∈ C} f(s)
4 add s to path
5 open ← open(C) \ {s}
6 if not open(C) then return failure
7 if h(s) < h(goal) then return start-PATH-goal
8 if not open(C) then return failure
9 if not open(C) then return failure
10 if not open(C) then return failure
11 if not open(C) then return failure
12 if not open(C) then return failure
13 if not open(C) then return failure
14 if not open(C) then return failure
15 if not open(C) then return failure
16 if not open(C) then return failure
17 if not open(C) then return failure
18 if not open(C) then return failure
19 if not open(C) then return failure
20 if not open(C) then return failure
21 if not open(C) then return failure
22 if not open(C) then return failure
23 if not open(C) then return failure
24 if not open(C) then return failure
25 if not open(C) then return failure
26 if not open(C) then return failure
27 if not open(C) then return failure
28 if not open(C) then return failure
29 if not open(C) then return failure
30 if not open(C) then return failure
31 if not open(C) then return failure
32 if not open(C) then return failure
33 if not open(C) then return failure
34 if not open(C) then return failure
35 if not open(C) then return failure
36 if not open(C) then return failure
37 if not open(C) then return failure
38 if not open(C) then return failure
39 if not open(C) then return failure
40 if not open(C) then return failure
41 if not open(C) then return failure
42 if not open(C) then return failure
43 if not open(C) then return failure
44 if not open(C) then return failure
45 if not open(C) then return failure
46 if not open(C) then return failure
47 if not open(C) then return failure
48 if not open(C) then return failure
49 if not open(C) then return failure
50 if not open(C) then return failure
51 if not open(C) then return failure
52 if not open(C) then return failure
53 if not open(C) then return failure
54 if not open(C) then return failure
55 if not open(C) then return failure
56 if not open(C) then return failure
57 if not open(C) then return failure
58 if not open(C) then return failure
59 if not open(C) then return failure
60 if not open(C) then return failure
61 if not open(C) then return failure
62 if not open(C) then return failure
63 if not open(C) then return failure
64 if not open(C) then return failure
65 if not open(C) then return failure
66 if not open(C) then return failure
67 if not open(C) then return failure
68 if not open(C) then return failure
69 if not open(C) then return failure
70 if not open(C) then return failure
71 if not open(C) then return failure
72 if not open(C) then return failure
73 if not open(C) then return failure
74 if not open(C) then return failure
75 if not open(C) then return failure
76 if not open(C) then return failure
77 if not open(C) then return failure
78 if not open(C) then return failure
79 if not open(C) then return failure
80 if not open(C) then return failure
81 if not open(C) then return failure
82 if not open(C) then return failure
83 if not open(C) then return failure
84 if not open(C) then return failure
85 if not open(C) then return failure
86 if not open(C) then return failure
87 if not open(C) then return failure
88 if not open(C) then return failure
89 if not open(C) then return failure
90 if not open(C) then return failure
91 if not open(C) then return failure
92 if not open(C) then return failure
93 if not open(C) then return failure
94 if not open(C) then return failure
95 if not open(C) then return failure
96 if not open(C) then return failure
97 if not open(C) then return failure
98 if not open(C) then return failure
99 if not open(C) then return failure
100 if not open(C) then return failure
```

ϕ_{query} :
Some property of the output
Ex: Some cells not selected

m_1 : True, m_2 : False

Actively Learning Relevant Variables

Find U such that $\phi_{\text{explain}}(V) \equiv \phi_{\text{explain}}(U)$ where $|U| \ll |V|$

Assignments to V

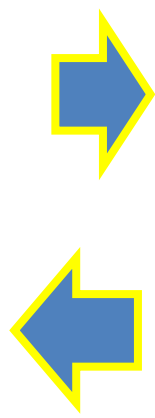
$m_1 = (0,0,0,1,1,0,1)$

$m_2 = (0,0,1,1,0,1,0)$

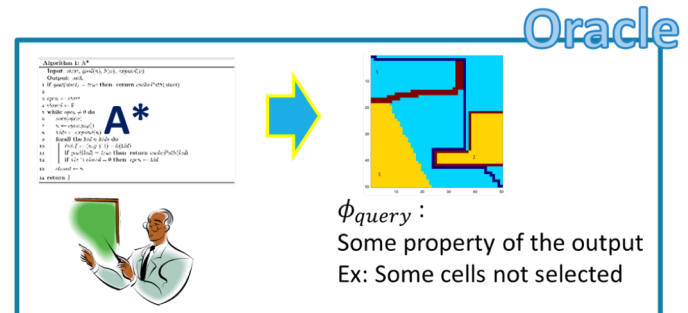
$m_3 = (0,0,0,1,1,1,0)$



Binary Search Over
Hamming Distance



Oracle



```
Algorithm 1: A*
Input: start, goal, h, g, open, closed
Output: path
1 if start == goal then return start
2 if start not in open then return failure
3 open.add(start)
4 while open not empty
5   u = pop(open)
6   if u == goal then return u
7   for each v in neighbors(u)
8     if v not in open and v not in closed
9       g[v] = g[u] + 1
10      h[v] = h[v]
11      f[v] = g[v] + h[v]
12      open.add(v)
13   closed.add(u)
14 return failure
```

ϕ_{query} :
Some property of the output
Ex: Some cells not selected

m_1 : True, m_2 : False

m_3 : True

Actively Learning Relevant Variables

Find U such that $\phi_{\text{explain}}(V) \equiv \phi_{\text{explain}}(U)$ where $|U| \ll |V|$

Hamming Distance = 4

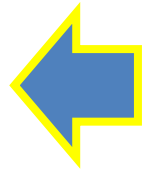
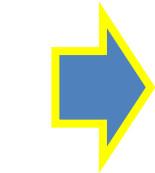
Assignments to V

~~$m_1 = (0,0,0,1,1,0,1)$~~

$m_2 = (0,0,1,1,0,1,0)$

$m_3 = (0,0,0,1,1,1,0)$

Hamming Distance = 2



Binary Search Over Hamming Distance

Oracle

```

Algorithm A*
Input: start, goal, h, g, open, closed
Output: path
if start == goal
  return []
open.add(start)
while open not empty
  u ← pop(open)
  if u == goal
    return path(u)
  for each v in neighbors(u)
    if v not in closed and h(v) < h(u)
      open.add(v)
      closed.add(v)
  if open.empty()
    return failure
return path(u)
        
```

→

ϕ_{query} :
Some property of the output
Ex: Some cells not selected

~~m_1 : True, m_2 : False~~
 m_3 : True

Actively Learning Relevant Variables

Find U such that $\phi_{\text{explain}}(V) \equiv \phi_{\text{explain}}(U)$ where $|U| \ll |V|$

Hamming Distance = 2

Assignments to V

$m_2 = (0,0,1,1,0,1,0)$

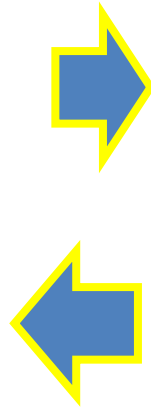
$m_3 = (0,0,0,1,1,1,0)$

$m_4 = (0,0,1,1,1,1,0)$

Hamming Distance = 1



Binary Search Over Hamming Distance



Oracle

```

Algorithm 1: A*
Require: start, goal(s), f(s), g(s)
function A*
    S ← start
    while S ≠ ∅
        s ← argmin_{s ∈ S} f(s)
        if s = goal(s)
            return s
        S ← S ∪ expand(s)
    return ∅
    
```

→

ϕ_{query} :
Some property of the output
Ex: Some cells not selected

m_2 : False, m_3 : True
 m_4 : True

Actively Learning Relevant Variables

Find U such that $\phi_{\text{explain}}(V) \equiv \phi_{\text{explain}}(U)$ where $|U| \ll |V|$

Hamming Distance = 2

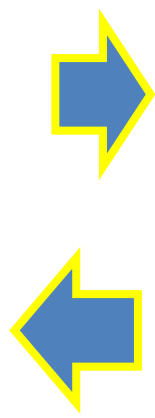
Assignments to V

$m_2 = (0,0,1,1,0,1,0)$

~~$m_3 = (0,0,0,1,1,1,0)$~~

$m_4 = (0,0,1,1,1,1,0)$

Hamming Distance = 1



Binary Search Over Hamming Distance

Oracle

```

Algorithm 1: A*
Require: start, goal, h, g, open, closed
Initialize: open, closed
if start = goal then return start
open.add(start)
while open not empty do
    u ← pop(open)
    if u = goal then return u
    expand(u)
    if u is closed then continue
    if u is better than closed then
        closed.add(u)
        open.add(u)
return -1
        
```

→

ϕ_{query} :
Some property of the output
Ex: Some cells not selected

m_2 : False, ~~m_3 : True~~
 m_4 : True

Actively Learning Relevant Variables

Find U such that $\phi_{\text{explain}}(V) \equiv \phi_{\text{explain}}(U)$ where $|U| \ll |V|$

Hamming Distance = 1

Assignments to V

$m_2 = (0,0,1,1,0,1,0)$

$m_4 = (0,0,1,1,1,1,0)$


Fifth variable v_5 is relevant !!



Binary Search Over Hamming Distance



Oracle



ϕ_{query} :
Some property of the output
Ex: Some cells not selected

m_2 : False, m_4 : True

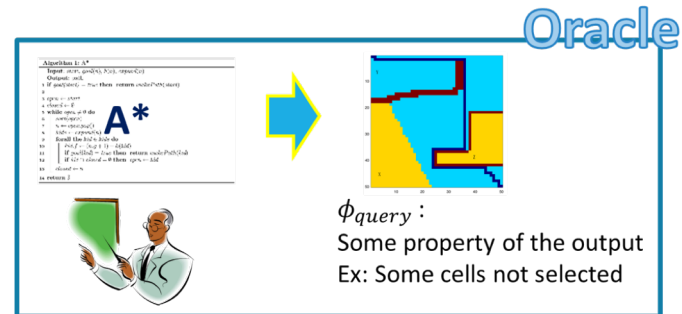
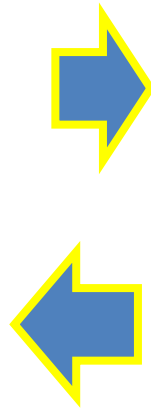
Actively Learning Relevant Variables

Find U such that $\phi_{\text{explain}}(V) \equiv \phi_{\text{explain}}(U)$ where $|U| \ll |V|$

Repeat to find all relevant variables



Binary Search Over Hamming Distance



m2: False, m4: True

Actively Learning Relevant Variables

Find U such that $\phi_{\text{explain}}(V) \equiv \phi_{\text{explain}}(U)$ where $|U| \ll |V|$

For each assignment
to relevant variables



Random Sample
Till Oracle differs

$$\ln(1/(1 - \kappa))$$



Binary Search Over
Hamming Distance

$$\ln(|V|)$$

Relevant variables of ϕ_{explain} found with confidence κ in

$$2^{|U|} \ln(|V|/(1 - \kappa))$$

Actively Learning Boolean Formula

Find U such that $\phi_{\text{explain}}(V) \equiv \phi_{\text{explain}}(U)$ where $|U| \ll |V|$

Used distinguishing example based approach from ICSE'10

Susmit Jha, Sumit Gulwani, Sanjit A Seshia, and Ashish Tiwari. Oracle-guided component-based program synthesis. In *2010 ACM/IEEE 32nd International Conference on Software Engineering*, volume 1, pages 215–224. IEEE, 2010.

Scales to ~200 variables

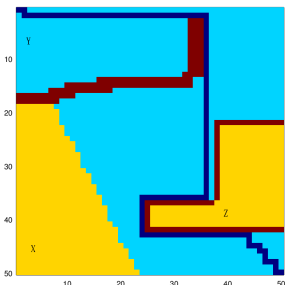
ϕ_{explain} found with confidence κ in
 $O(2^{|U|} \ln(|V|/(1 - \kappa)))$



Build Truth Table for the relevant variables U

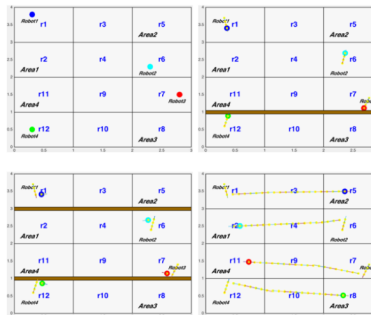
Worst Case: $2^{|U|}$

Experiments



A* Planning
 $|V| = 2500$
 $|U| \leq 4$
 Runtime < 3 minutes

10^{153}



Reactive Exploration Strategy
 $|V| = 96$

10^{28}

$|U| \leq 2$
 Runtime < 5 seconds

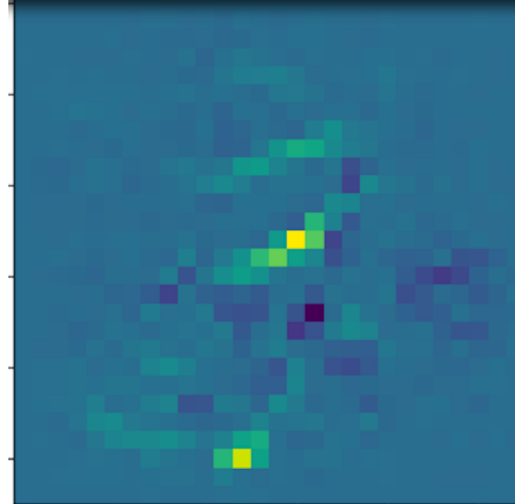
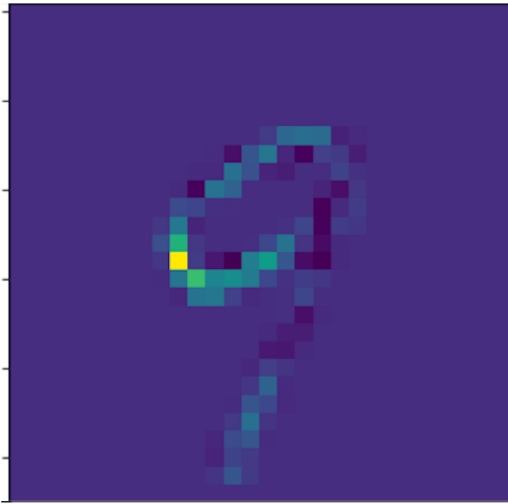


Image Classification: MNIST

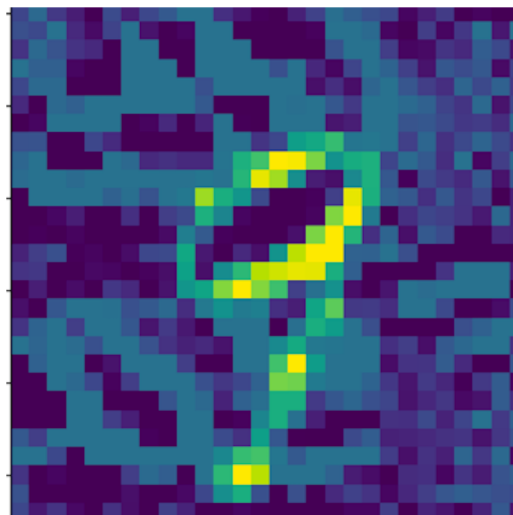
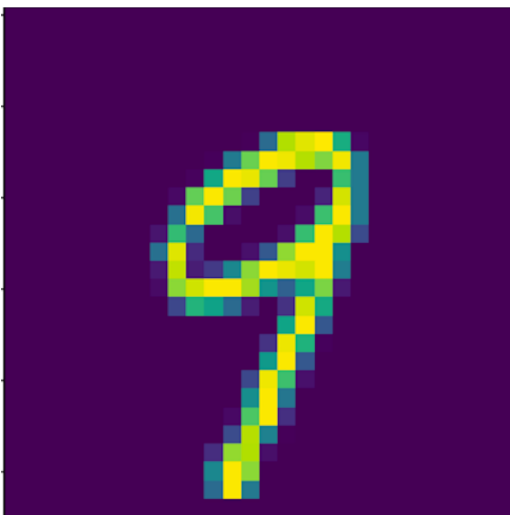
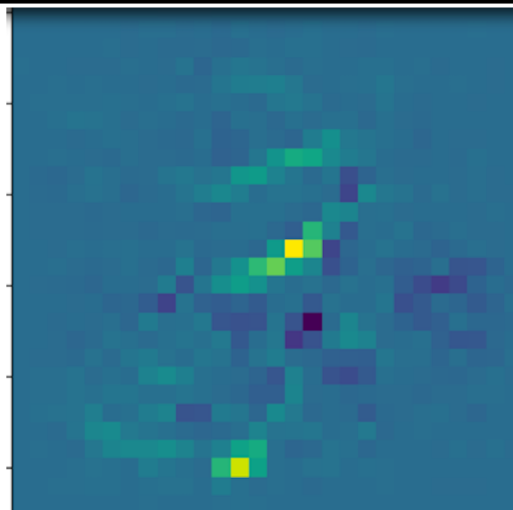
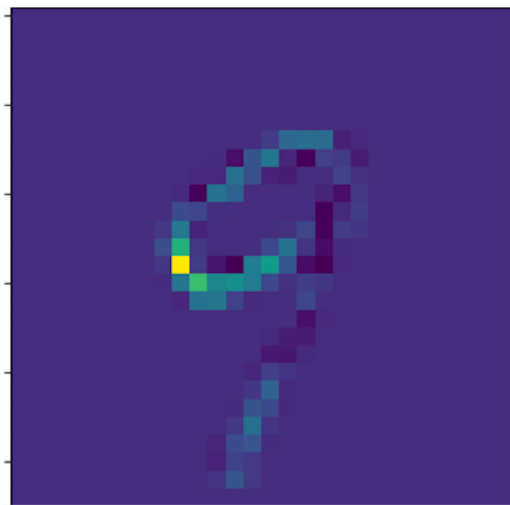


Image Classification: ImageNet
 with Carlini-Wagner
 Adversarial Attacks

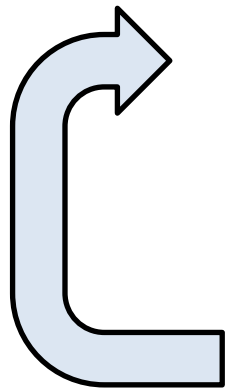
Experiments



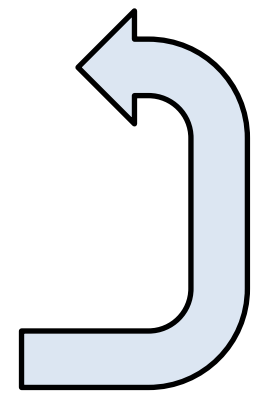
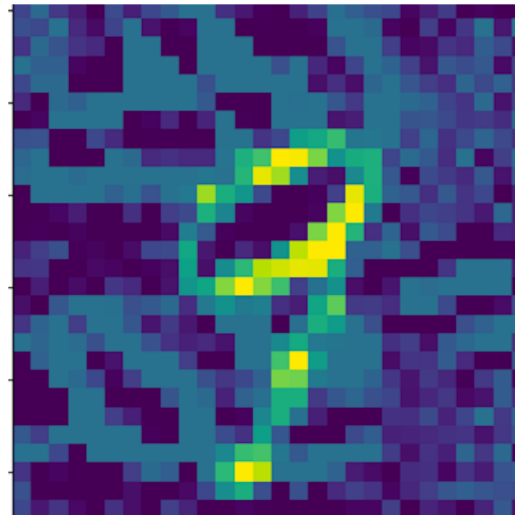
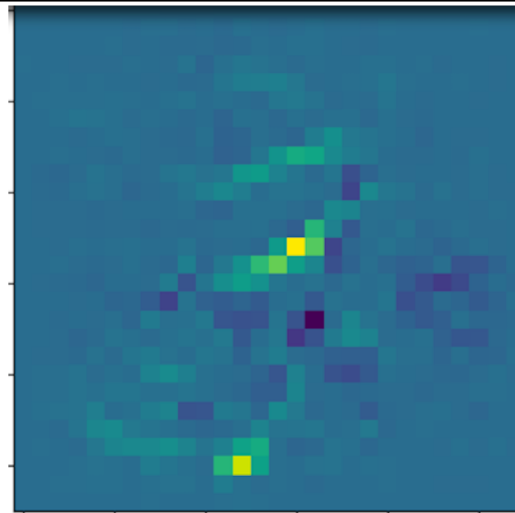
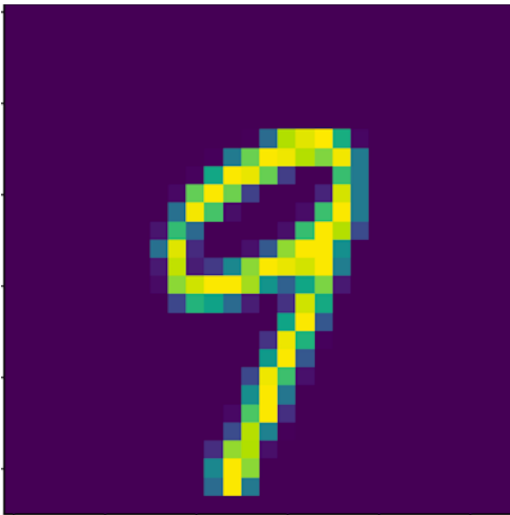
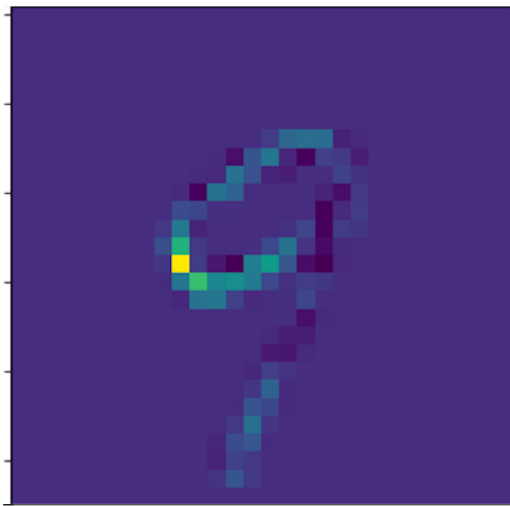
Experiments



Experiments

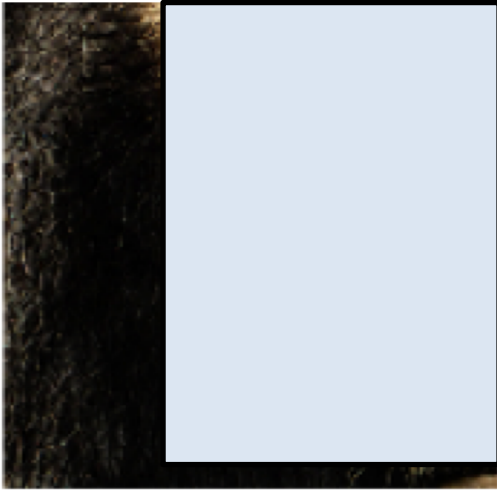


Why 9



Why 3

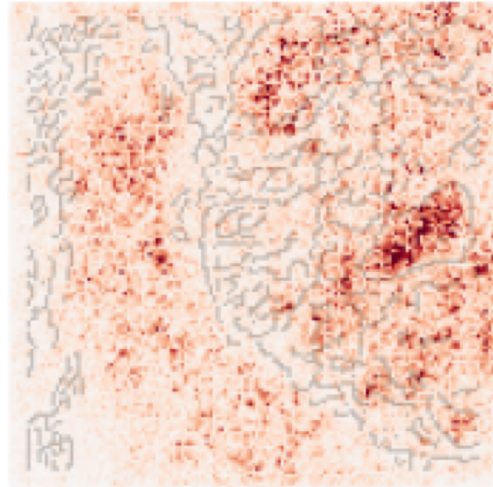
Why not just do sensitivity analysis?



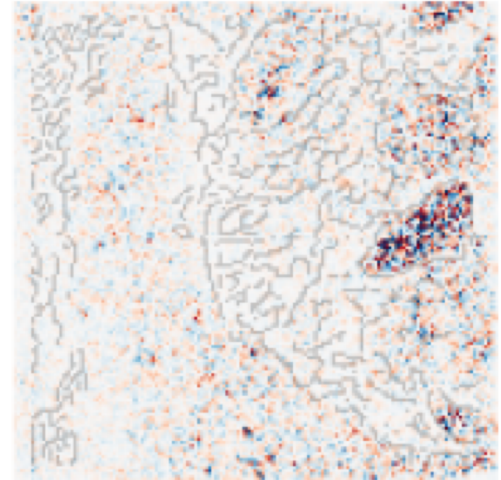
Why not just do sensitivity analysis?



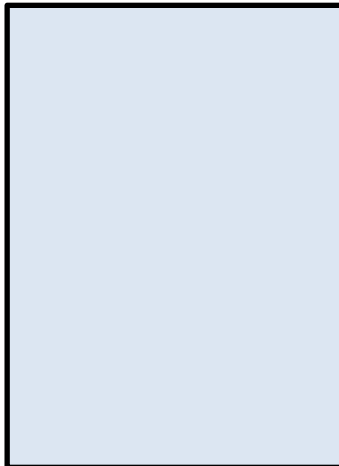
Why not just do sensitivity analysis?



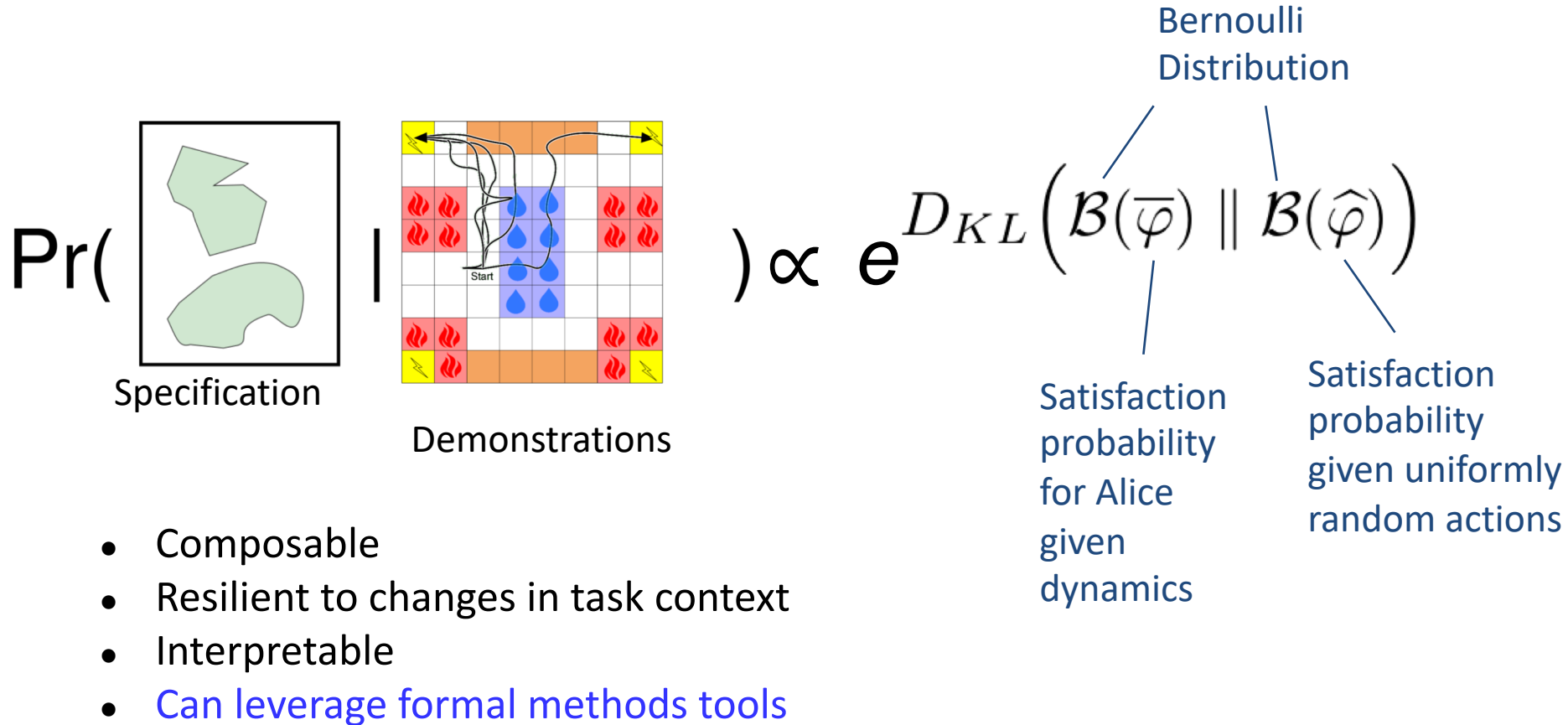
Sensitivity (IG)



Sparse Boolean
Formula Learning



Learning Temporal Logic Properties from Noisy Time Traces



Marcell Vazquez-Chanlatte, Susmit Jha, Ashish Tiwari, Mark K. Ho and Sanjit A. Seshia.
Learning Task Specifications from Demonstrations. NeurIPS, 2018

Communicating Using Demonstrations: More involved example

1. Avoid fire (red).
2. Eventually Recharge (yellow).
3. If you touch the water (blue) then dry off (brown) before recharging (yellow).

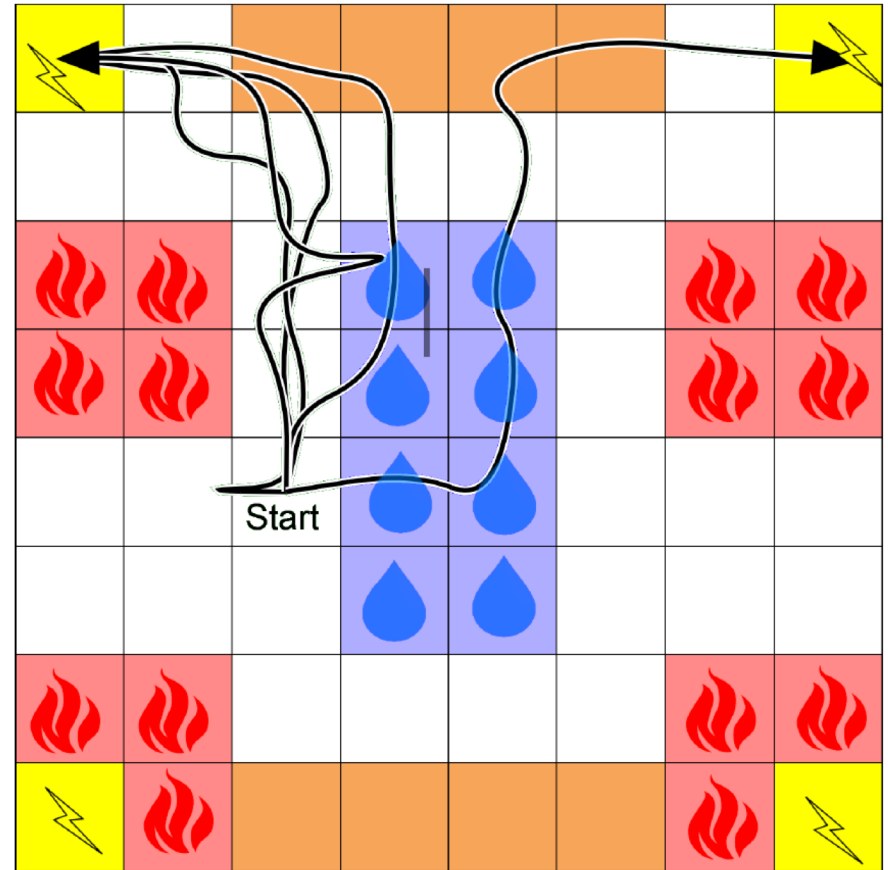
Temporal Logic Specification

H: Historically

O: Once

S: Since

$$(H \neg \text{red} \wedge O \text{ yellow}) \wedge H((\text{yellow} \wedge O \text{ blue}) \Rightarrow (\neg \text{blue} S \text{ brown}))$$



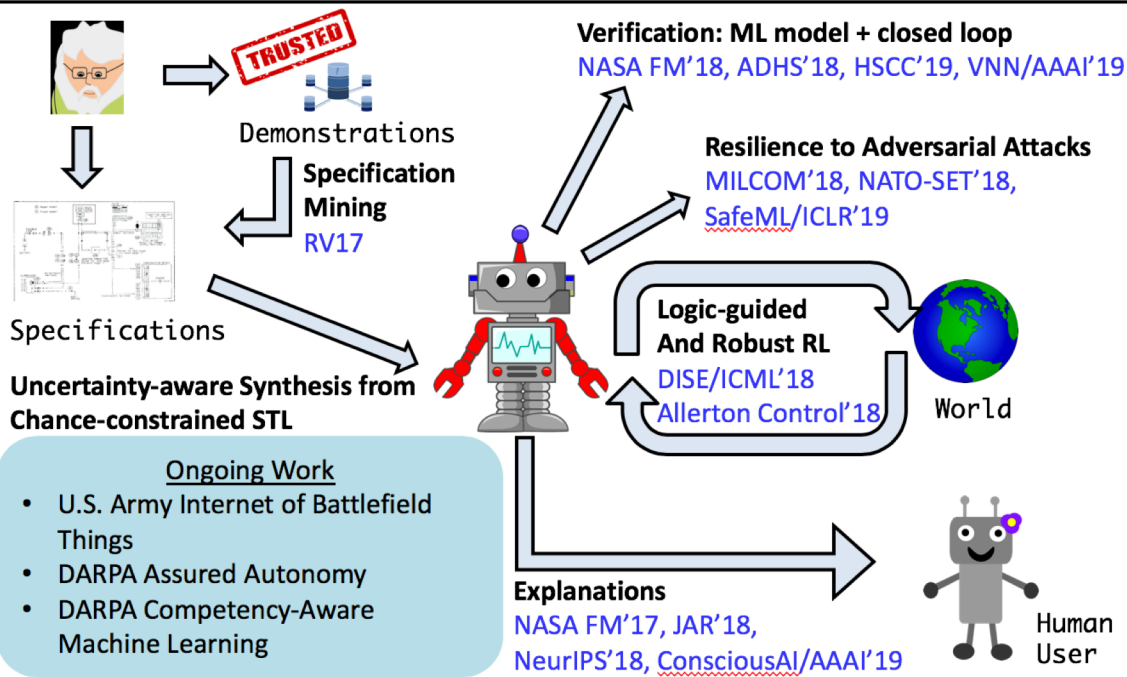
Interpretability / Explanation Generation in TRINITY

- **Inferring and Conveying Intentionality: Beyond Numerical Rewards to Logical Intentions.** Susmit Jha and John Rushby.
AAAI Spring Symposium, Towards Conscious AI Systems, 2019
- **Learning Task Specifications from Demonstrations.** Marcell Vazquez-Chanlatte, Susmit Jha, Ashish Tiwari, Mark K. Ho and Sanjit A. Seshia.
Neural Information Processing Systems (NeurIPS), 2018
- **Explaining AI Decisions Using Efficient Methods for Learning Sparse Boolean Formulae.** Susmit Jha, Tuhin Sahai, Vasumathi Raman, Alessandro Pinto and Michael Francis.
Journal of Automated Reasoning, 2018
- **On Learning Sparse Boolean Formulae For Explaining AI Decisions.** Susmit Jha, Vasumathi Raman, Alessandro Pinto, Tuhin Sahai, and Michael Francis.
NASA Formal Methods (NFM), 2017

Thanks!

If you are interested in building *trusted, resilient and interpretable* AI, please contact me with your CV if you are interested.

TRINITY @ SRI



Co-travelers (Present and Past):

Brian Burns, Margaret Chapman, Ajay Divakaran, Sauradeep Dutta, Michael Francis, Mark K. Ho, Uyeong Jang, Brian Jalaian, Somesh Jha, Patrick Lincoln, Alessandro Pinto, Vasu Raman, John Rushby, Dorsa Sadigh, Sriram Sankaranarayanan, Sanjit A. Seshia, Natarajan Shankar, Ashish Tiwari, Claire Tomlin, Marcell Vazquez-Chanlatte, Gunjan Verma

Funding sources (Present and Past):

DARPA, US Army Research Laboratory, National Science Foundation

TRINITY @ SRI

