Assured Autonomy
- problems and possible solutions -

Hill Air Force Base

Ground Based Strategic Deterrence Program
Course 4: August 16-20, 2021

Rick Kuhn
National Institute of Standards and Technology
Gaithersburg, Maryland 20899
kuhn@nist.gov

Outline

- Why current safety-critical testing won't work
- Assurance based on input space coverage,
- Explainable AI as part of validation, and
- Transfer learning

Short overview of assured autonomy,
and NIST focus (measurement and test) in this area

(Slide from Darryl Ahner, US Air Force Institute of Technology)

Defense Science Board Study

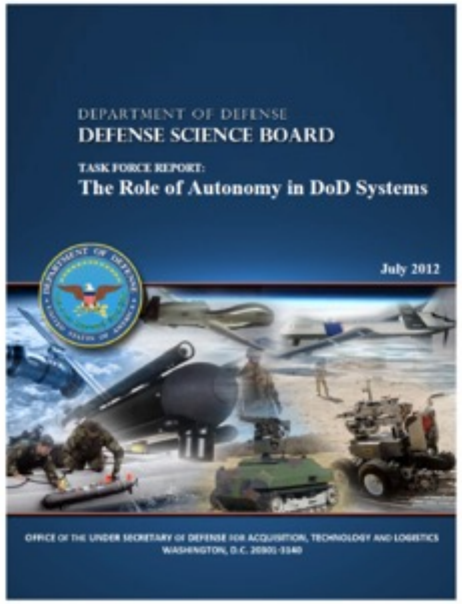
DSB 2012 The Role of Autonomy in DoD Systems Study recommends:

"USD(AT&L) to create developmental and operational T&E techniques that focus on the unique challenges of autonomy (to include developing operational training techniques that explicitly build trust in autonomous systems)."

Recommendation:

USD(AT&L) establish developmental and operational T&E techniques that focus on the unique challenges of autonomy

- Coping with the difficulty of enumerating all conditions and non-deterministic responses
- Basis for system decisions often not apparent to user
- Measuring trust that the autonomous system will interact with its human supervisor as intended
- Leverage the benefits of robust simulation



Software safety assurance is already *very* expensive

Consumer level software cost: about 50% development, 50% verification

For aviation life-critical, 12% development, 88% verification (Software is about 30% of cost for new civilian aircraft, higher for military)

Autonomy makes the problem even harder!

V&V cost and Certification

For FAA compliant DO-178B Level A software, the industry usually spends 7 times as much on verification (reviews, analysis, test). So that's about 12% for development and 88% for verification.

Level B reduces the verification cost by approximately 15%. The mix is then 25% development, 75% verification.

Randall Fulton
FAA Designated Engineering Representative
(private email to L. Markosian, July 2008)

13 April 2010 NFM 2010 10

Autonomy makes the problem even more expensive!

Assurance for Autonomous Systems is Hard

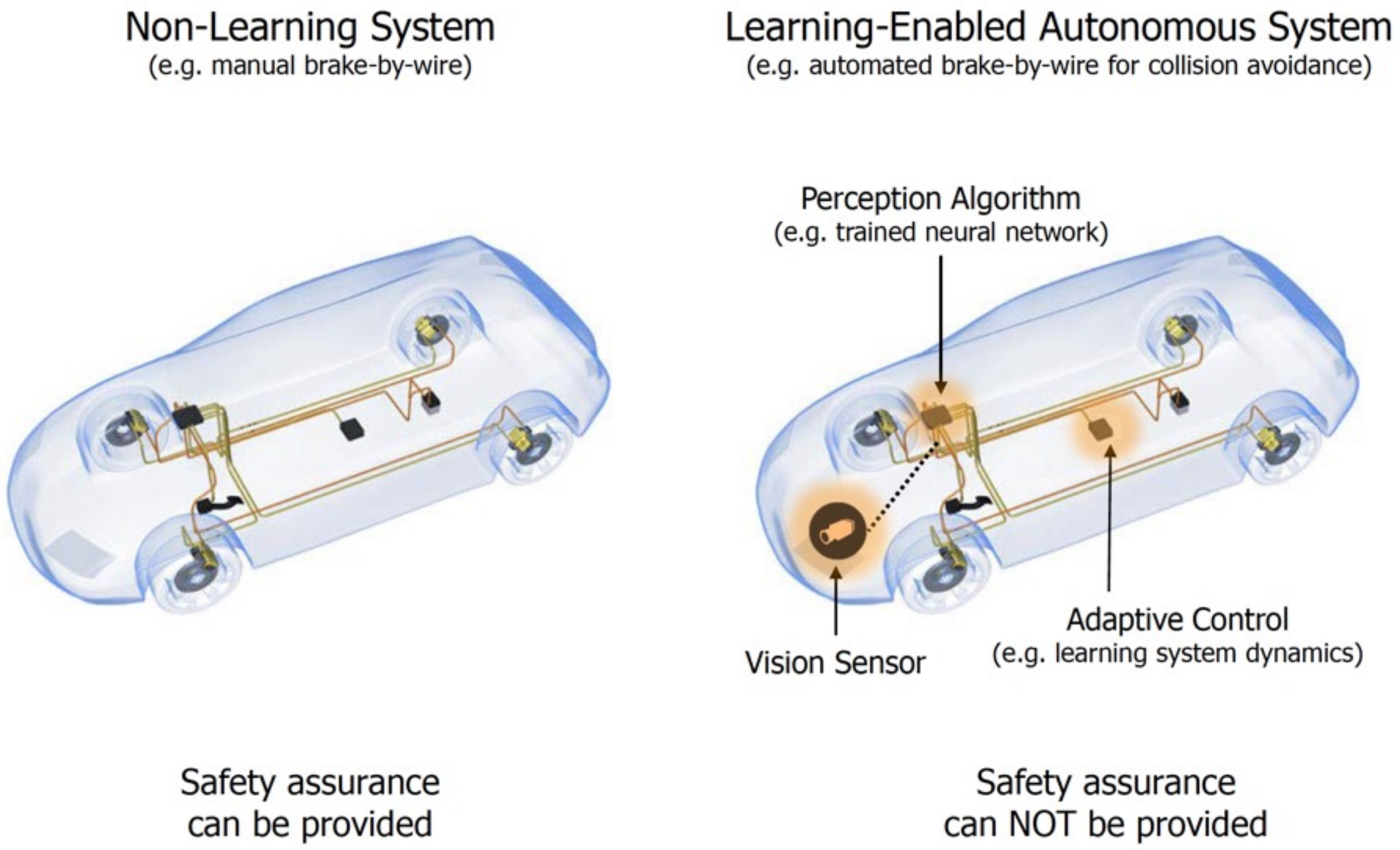
Traditional testing will require exorbitant time and money:
11B miles, 500 years, $6B
- *Driving to Safety, RAND Corp. Report, 2016*

Table 1. Examples of Miles and Years Needed to Demonstrate Autonomous Vehicle Reliability

		Benchmark Failure Rate		
Statistical Question	**How many miles (years[a]) would autonomous vehicles have to be driven...**	**(A) 1.09 fatalities per 100 million miles?**	**(B) 77 reported injuries per 100 million miles?**	**(C) 190 reported crashes per 100 million miles?**
	(1) without failure to demonstrate with 95% confidence that their failure rate is at most...	275 million miles (12.5 years)	3.9 million miles (2 months)	1.6 million miles (1 month)
	(2) to demonstrate with 95% confidence their failure rate to within 20% of the true rate of...	8.8 billion miles (400 years)	125 million miles (5.7 years)	51 million miles (2.3 years)
	(3) to demonstrate with 95% confidence and 80% power that their failure rate is 20% better than the human driver failure rate of...	11 billion miles (500 years)	161 million miles (7.3 years)	65 million miles (3 years)

[a] We assess the time it would take to compete the requisite miles with a fleet of 100 autonomous vehicles (larger than any known existing fleet) driving 24 hours

Illustrating the challenge



Components of autonomous systems can't use conventional safety assurance

Why can’t we use same processes as other safety-critical software ?

- Nearly all conventional software testing is based on structural coverage – ensuring that statements, decisions, paths are covered in testing
- Life-critical aviation software requires MCDC testing, white-box criterion that cannot be used for neural nets and other black-box methods

DARPA Program objectives

High level DARPA Goals

- Increase scalability of design-time assurance
  - What is the baseline capability of the proposed methods, in terms of the hybrid state-space and number and complexity of learning-enabled components
  - How do you plan to scale up by an order of magnitude?
  - How will you characterize the tradeoffs between fidelity of your modeling abstractions and scalability of the verification approach.
- Reduce overhead of operation-time assurance
  - What is the baseline overhead of the operation-time assurance monitoring techniques?
  - How do you plan to minimize it to be below 10% of the nominal system resource utilization?
- Scale up dynamic assurance
  - What is the size and scale of dynamic assurance case that can be developed and dynamically evaluated with your tools?
- Reduce trials to assurance
  - How will your approach quantifiably reduce the need for statistical testing?

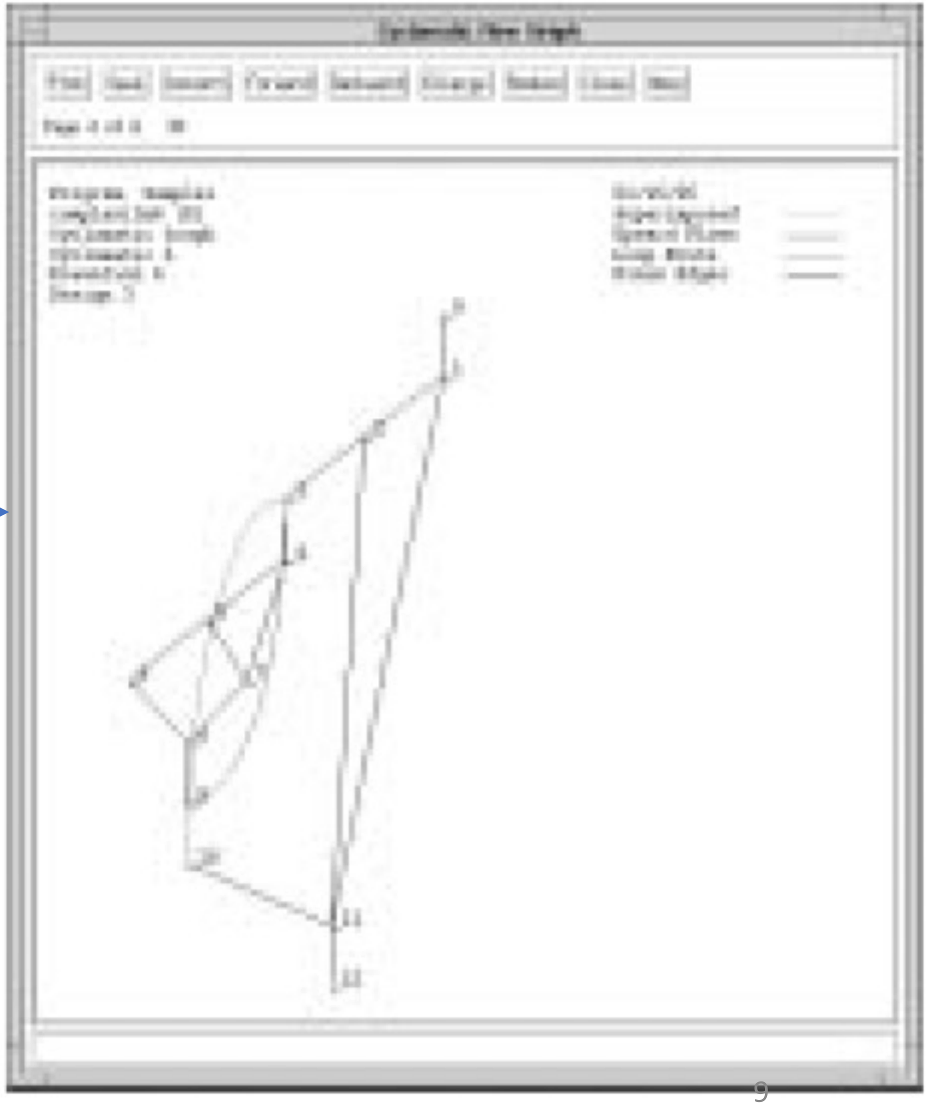
Code coverage works well - for conventional software

Annotated Source Listing

01/

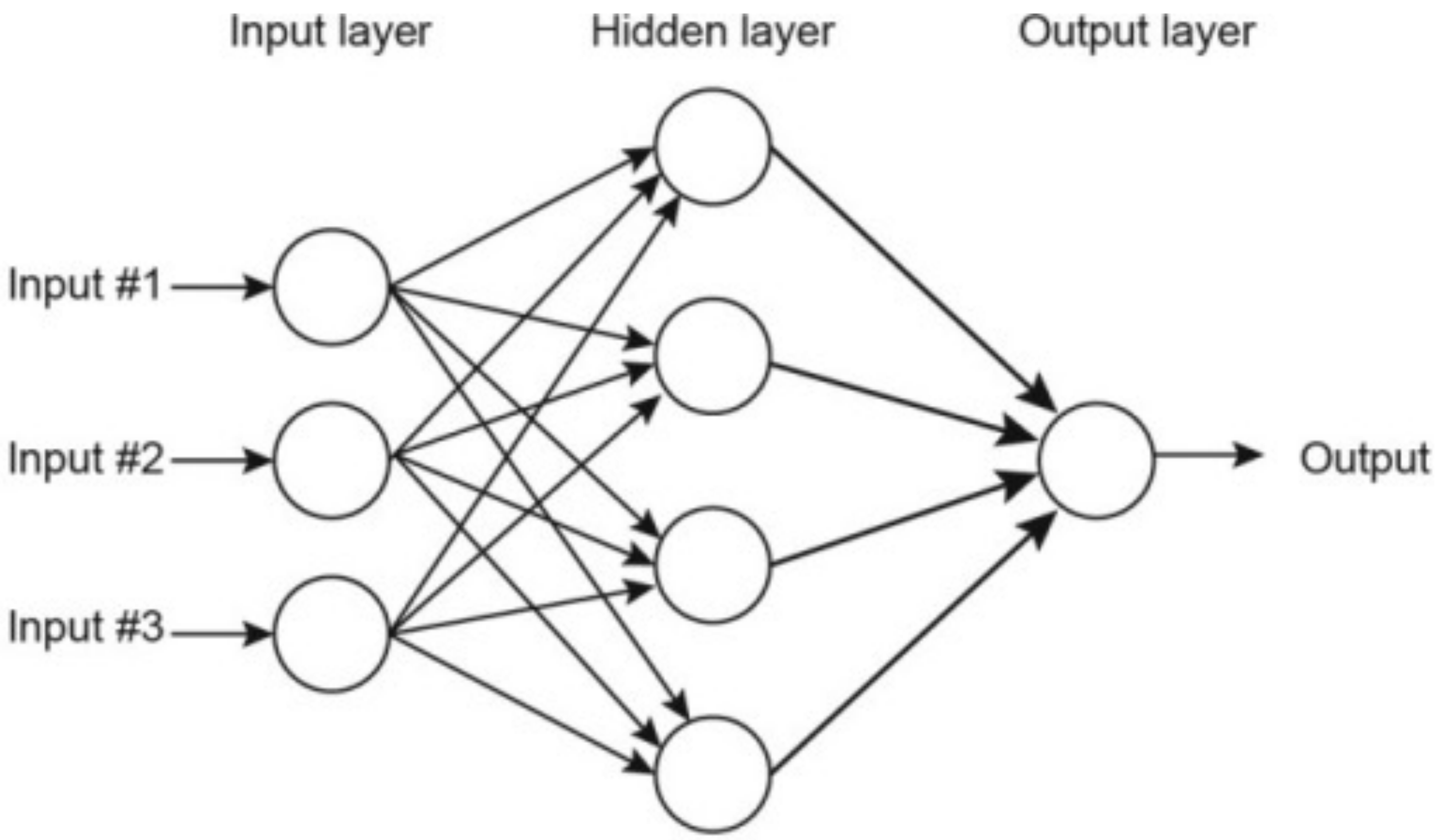
	v(G)	ev(G)	iv(G)	Start Line
	6	6	3	40

```
complexity6(int i, int j)
{/* Sample module with complexity 6 */
    if (i > 0 && j > 0) {
        while (i > j) {
            if (i % 2 && j % 2)
                printf("%d\n", i);
            else
                printf("%d\n", j);
            i--;
        }
    }
}
```

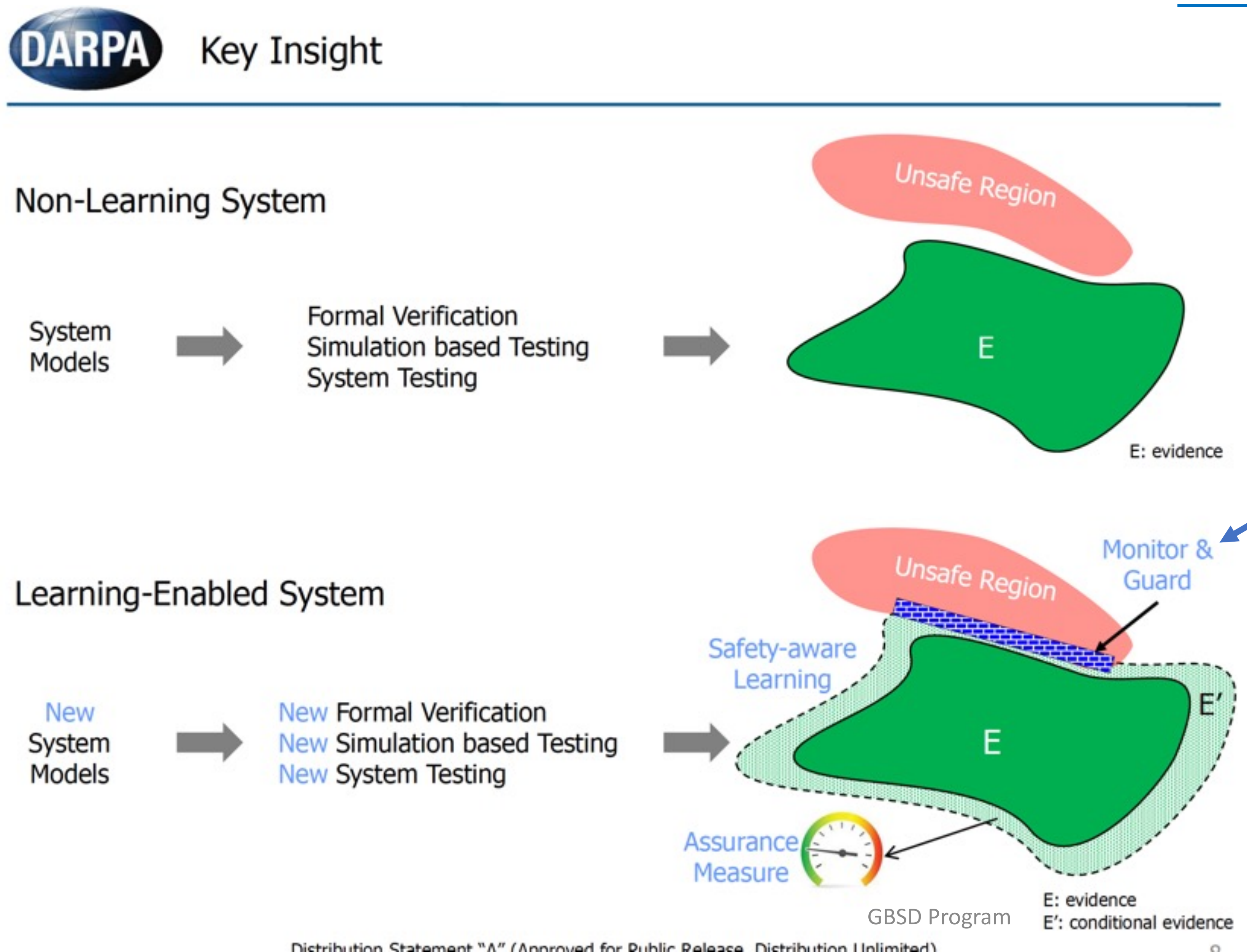


Can we use code coverage for machine learning?

- Much of AI/ML depends on various neural nets
- Algorithm and code stays the same
- Connections and weights vary
- Behavior changes depending on inputs used in training



DARPA approach



Monitor and guard the non-deterministic, unpredictable region of input space

To monitor and guard input space, need to measure

- Gold standard of assurance and verification of life-critical software can't be used for much of new life-critical autonomy software

- We can measure "neuron coverage", but indirect measure and not clear how closely related to accuracy and ability to correctly process all of the input space

- Measure the input space directly

- Then see if the AI system handles all of it correctly

Outline

- Why current safety-critical assurance won't work
- Assurance based on input space coverage
- Explainable AI as part of validation, and
- Transfer learning

Major DoD investment in assured autonomy

"The notion that autonomous systems can be fully tested is becoming increasingly infeasible as higher levels of self governing systems become a reality...*the standard practice of testing all possible states and all ranges of inputs to the system becomes an unachievable goal*. Existing TEVV methods are, by themselves, insufficient for TEVV of autonomous systems; therefore *a fundamental change is needed in how we validate and verify these systems*."
- OSD TEV&V Strategy Report, May 2015

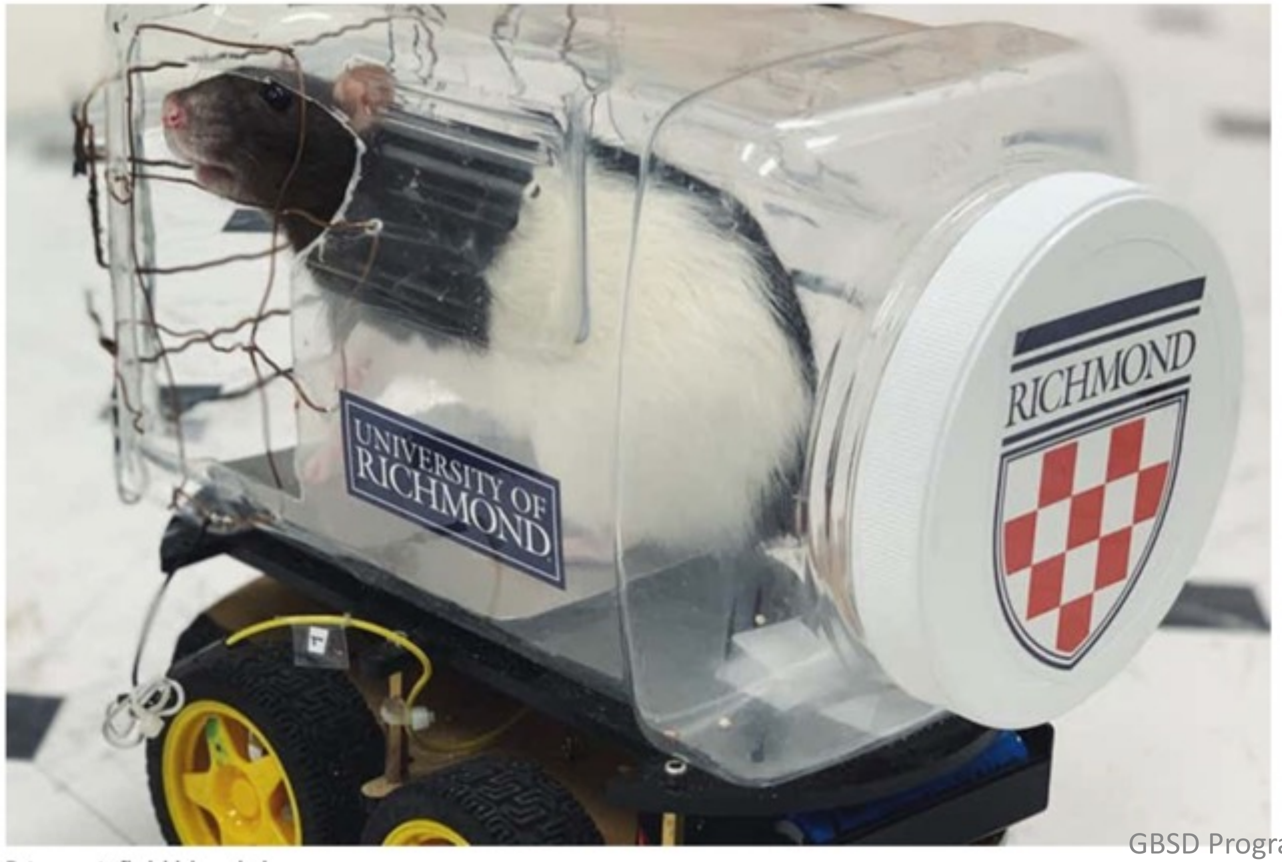
(Note that "testing all possible states and all ranges of inputs" was already unachievable, but the point holds.)

NewScientist

Scientists have trained rats to drive tiny cars to collect food

LIFE 22 October 2019

By Alice Klein



It doesn't take much intelligence to drive a car. Even rats can do it!

But can they do it under all kinds of conditions ?

The problem is harder outside of a constrained environment

Things get tricky as the scene becomes complex

- Multiple conditions involved in accidents
  - "The camera failed to recognize the white truck against a bright sky"
  - "The sensors failed to pick up street signs, lane markings, and even pedestrians due to the angle of the car shifting in rain and the direction of the sun"
- <u>We need to understand what combinations of conditions are included in testing</u>

Combinatorial value coverage - review

a	b	c	d
0	0	0	0
0	1	1	0
1	0	0	1
0	1	1	1

Vars	Combination values	Coverage
a b	00, 01, 10	.75
a c	00, 01, 10	.75
a d	00, 01, 11	.75
b c	00, 11	.50
b d	00, 01, 10, 11	1.0
c d	00, 01, 10, 11	1.0

19 combinations included in test set

100% coverage of 33% of combinations
75% coverage of half of combinations
50% coverage of 16% of combinations

Kuhn, D. R., Mendoza, I. D., Kacker, R. N., & Lei, Y. (2013). Combinatorial coverage measurement concepts and applications. *2013 IEEE Sixth Intl Conference on Software Testing, Verification and Validation Workshops*

GBSD Program

Vars	Combination values	Coverage
a b	00, 01, 10	.75
a c	00, 01, 10	.75
a d	00, 01, 11	.75
b c	00, 11	.50
b d	00, 01, 10, 11	1.0
c d	00, 01, 10, 11	1.0

Total possible 2-way combinations = $2^2\binom{4}{2} = 24$

S_2 = fraction of 2-way combinations covered = 19/24 = 0.79

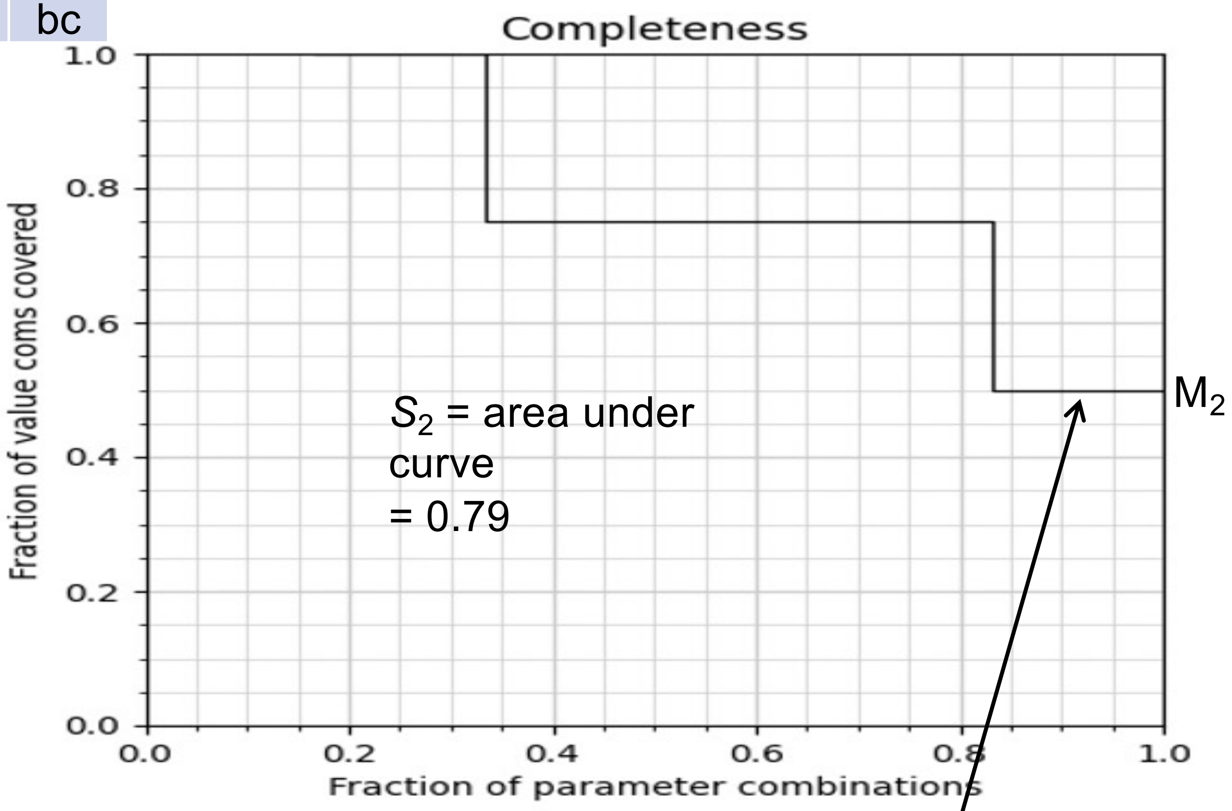
Rearranging the table:

1.00	00	00				
.75	01	01	00	00	00	
.50	10	10	01	01	01	00
.25	11	11	10	10	11	11
	bd	cd	ab	ac	ad	bc

GBSD Program

Graphing Coverage Measurement

1.00	00	00				
.75	01	01	00	00	00	
.50	10	10	01	01	01	00
.25	11	11	10	10	11	11
	bd	cd	ab	ac	ad	bc



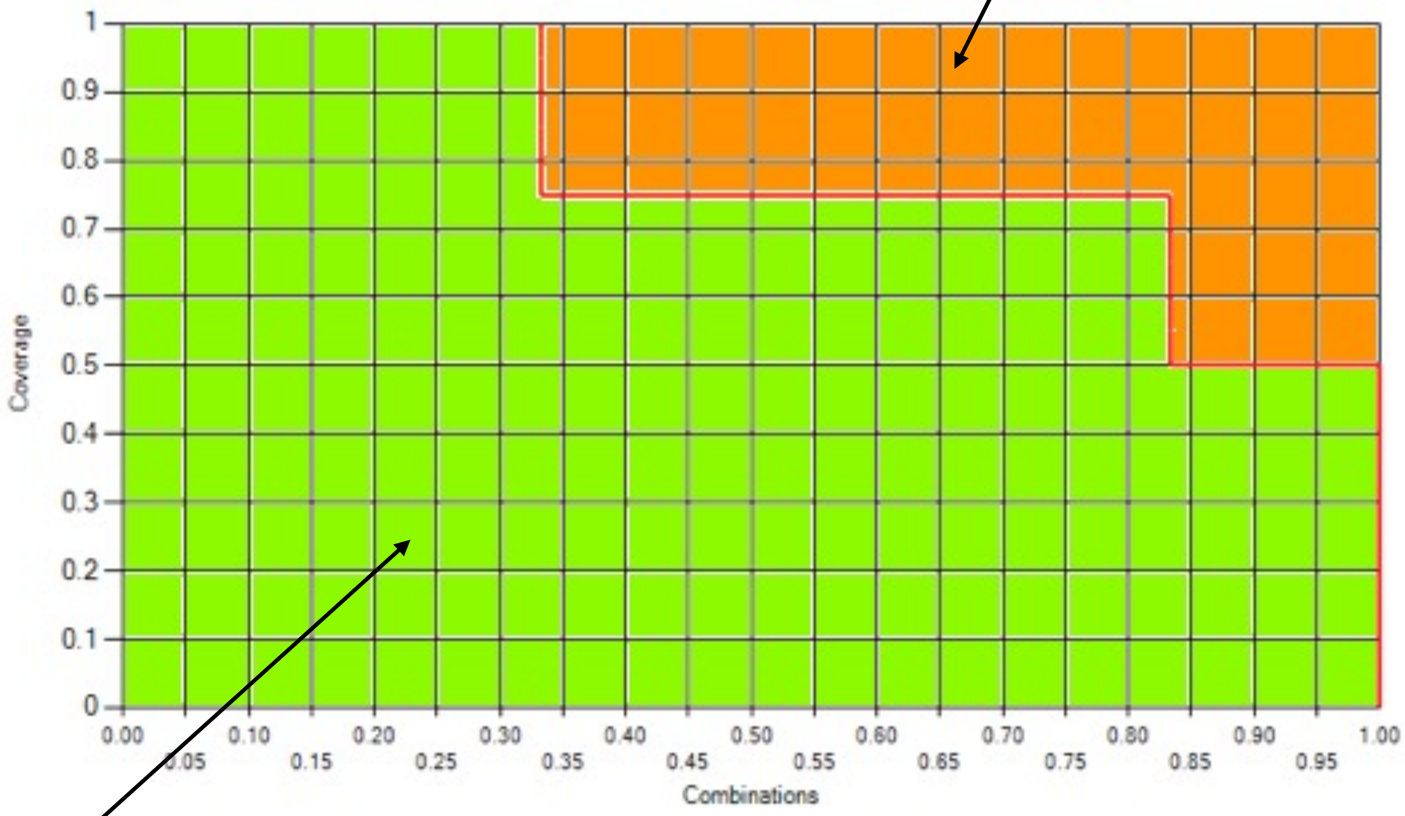
100% coverage of .33 of combinations
75% coverage of .50 of combinations
50% coverage of .16 of combinations

Bottom line:
All combinations covered to at least .50

What else does this chart show?

$1 - S_t$ = **Untested combinations**

(look for problems here)

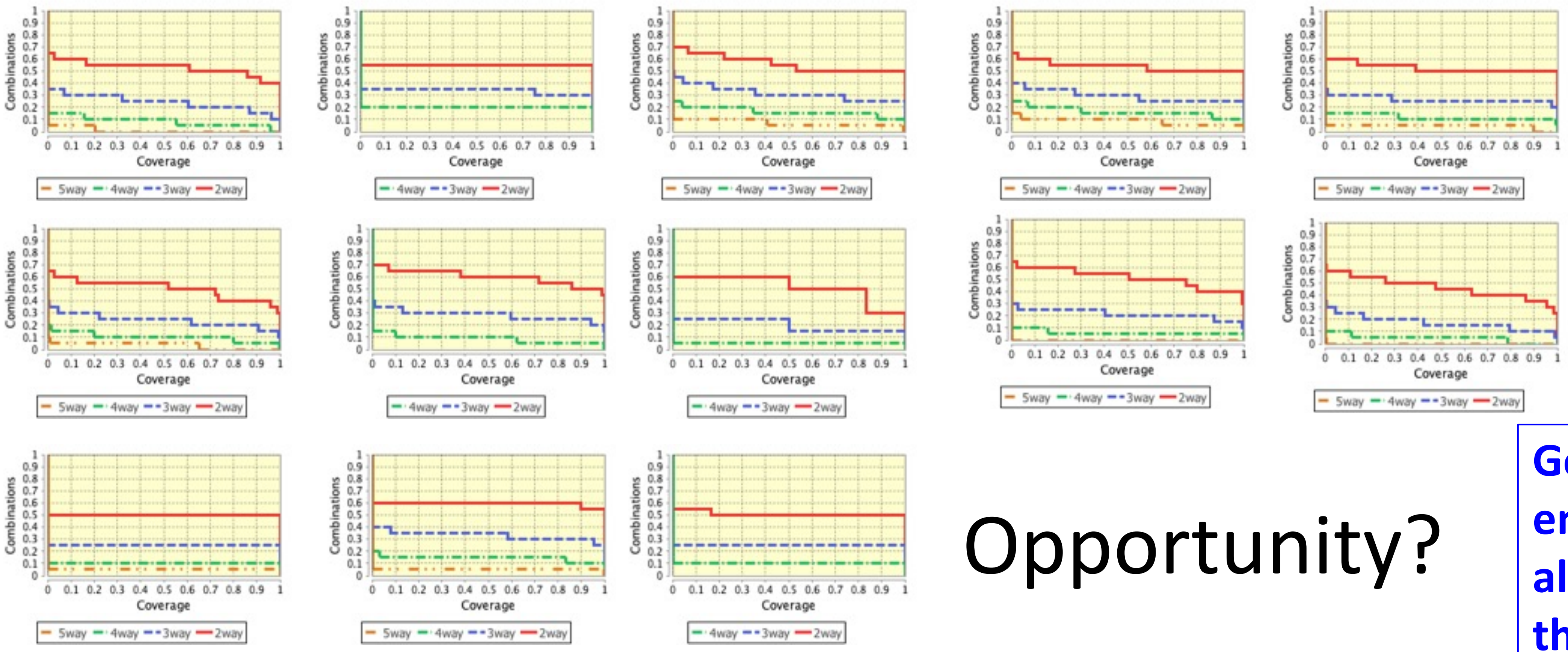


S_t = Tested combinations => code works for these

GBSD Program

What levels of input space coverage are seen in practical ML data sets?

Examples from WEKA data mining demo set



Opportunity?

Goal: enumerating all conditions that matter

Research questions

- Practical ML examples don’t seem to have very high input space coverage (previous slide)
- Can we improve results with better input space coverage?
- Empirical data show that small numbers of factors are involved in system failures (generally 1 to 6).
- Is this also true of autonomous systems?
- How are input space coverage and classification/prediction accuracy related?
- Can we apply some of these methods to temporal aspects? (sequence covering arrays)

Outline

- Why current safety-critical testing won’t work
- Assurance based on input space coverage
- Explainable AI as part of validation, and
- Transfer learning

What is the explainability problem?

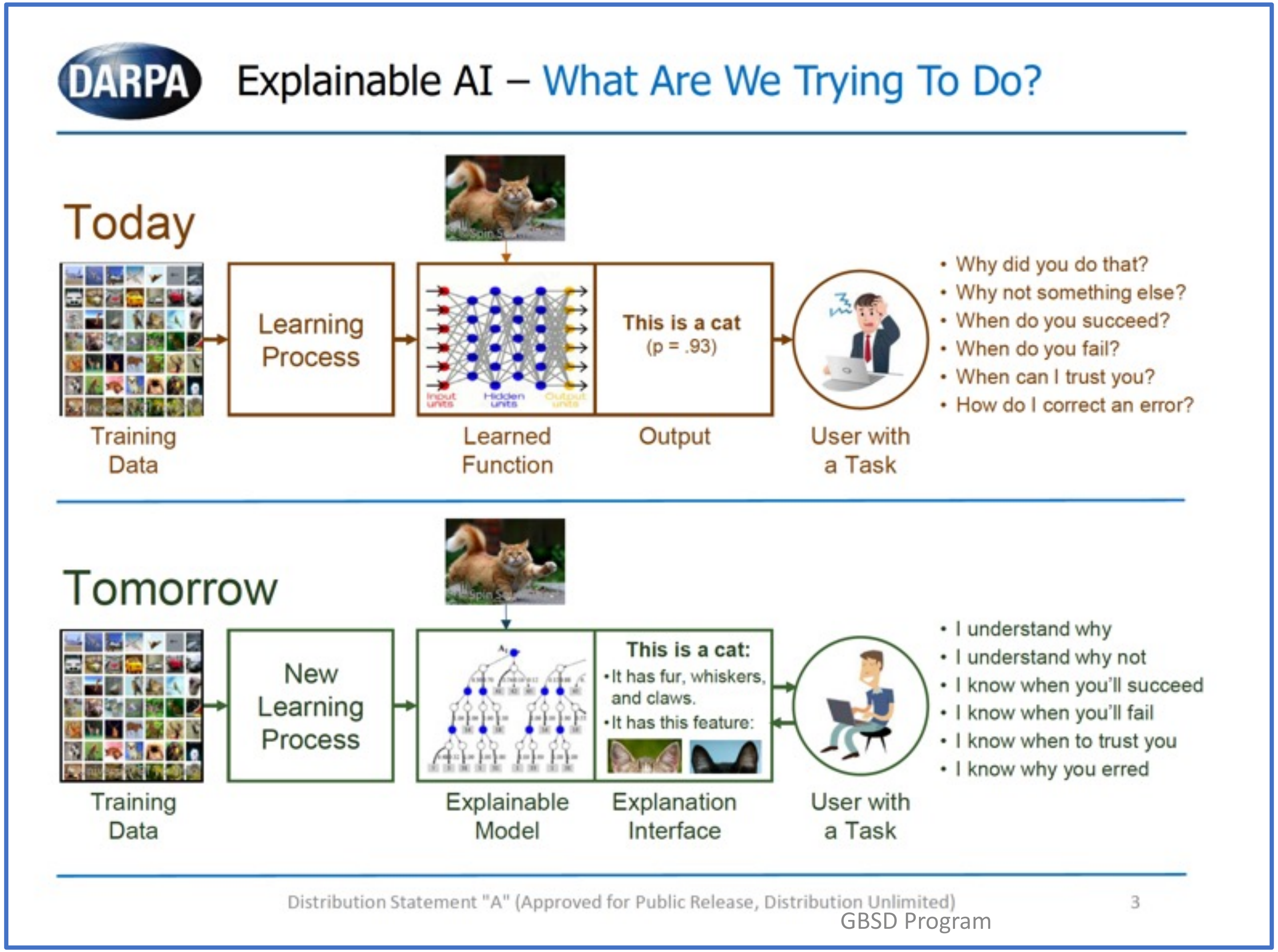
- AI systems are good, but sometimes make mistakes, and human users will not trust their decisions without explanation or justification
  → <u>assurance and explainability are closely tied</u>

- There is a tradeoff between AI accuracy and explainability: the most accurate methods, such as convolutional neural nets (CNNs), provide no explanations; understandable methods, such as rule-based, tend to be less accurate

- The black-box nature of these systems that makes explanation difficult also makes assurance and testing even harder

- Life-critical aviation software requires MCDC testing, white-box criterion that cannot be used for neural nets and other non-explainable methods

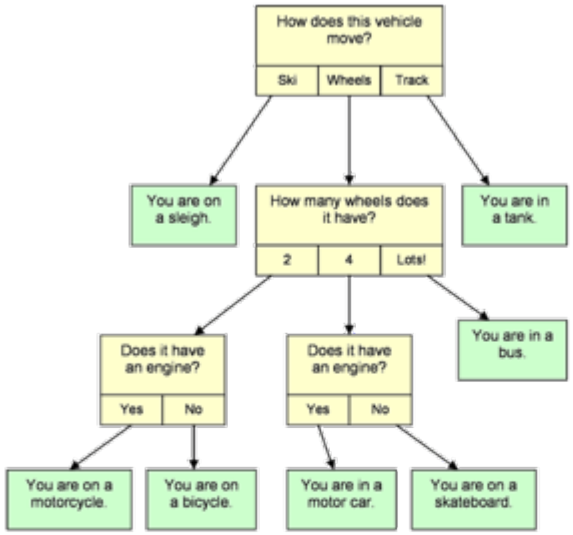
Explainability – what's current state of the art?



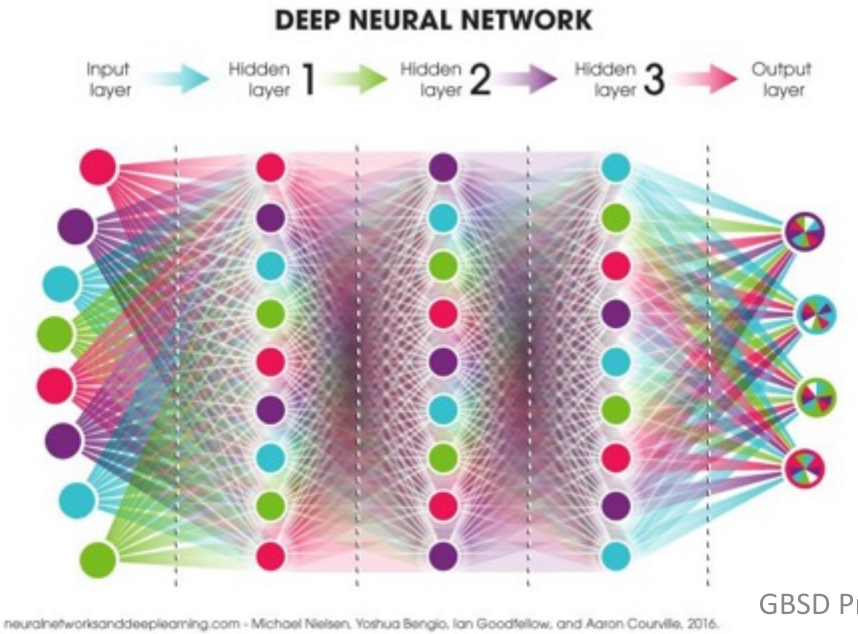
Black-box statistical predictions are inadequate

Explanations must be understandable to non-specialist

Tradeoff:



- OR -



Expert system:
Good for explanations,
not so good for accuracy

Neural nets:
Good for accuracy,
not so good for explanations

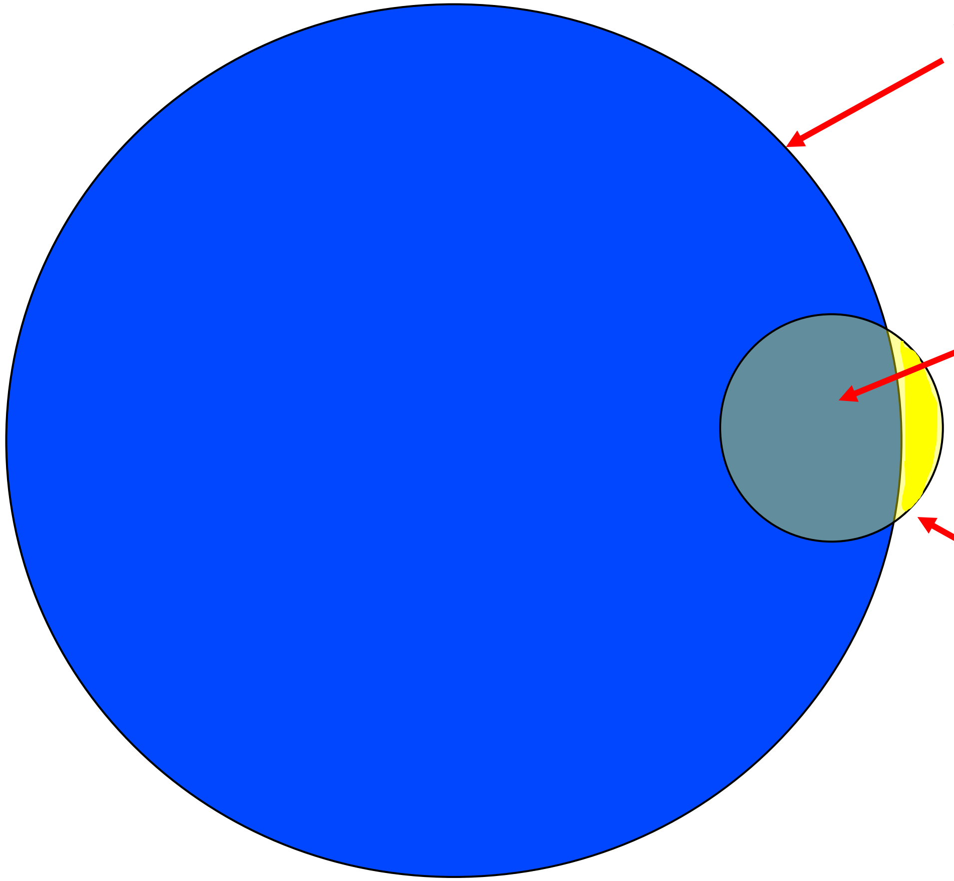
How do we get the best of both worlds?

What has been tried?

- Interpretable models – e.g. rule-based expert systems: “if patient has symptoms A and B, or has B with C and D, then illness is X”
  - best for explanations
  - hard to find rules
  - less accurate than other approaches
- Modify neural nets etc. to add explanations
  - reduces accuracy, complicates the system
  - explanations still not very understandable
- Model induction - infer explainable model from black-box
  - flexible for application, good explanations using only input, output
  - hard to produce the explainable model
- Our approach – derive rule predicates from inputs and outputs to CNNs and other black-box functions

Fault location – identify fault-triggering input

Given: a set of tests that the SUT fails, which combinations of variables/values triggered the failure?

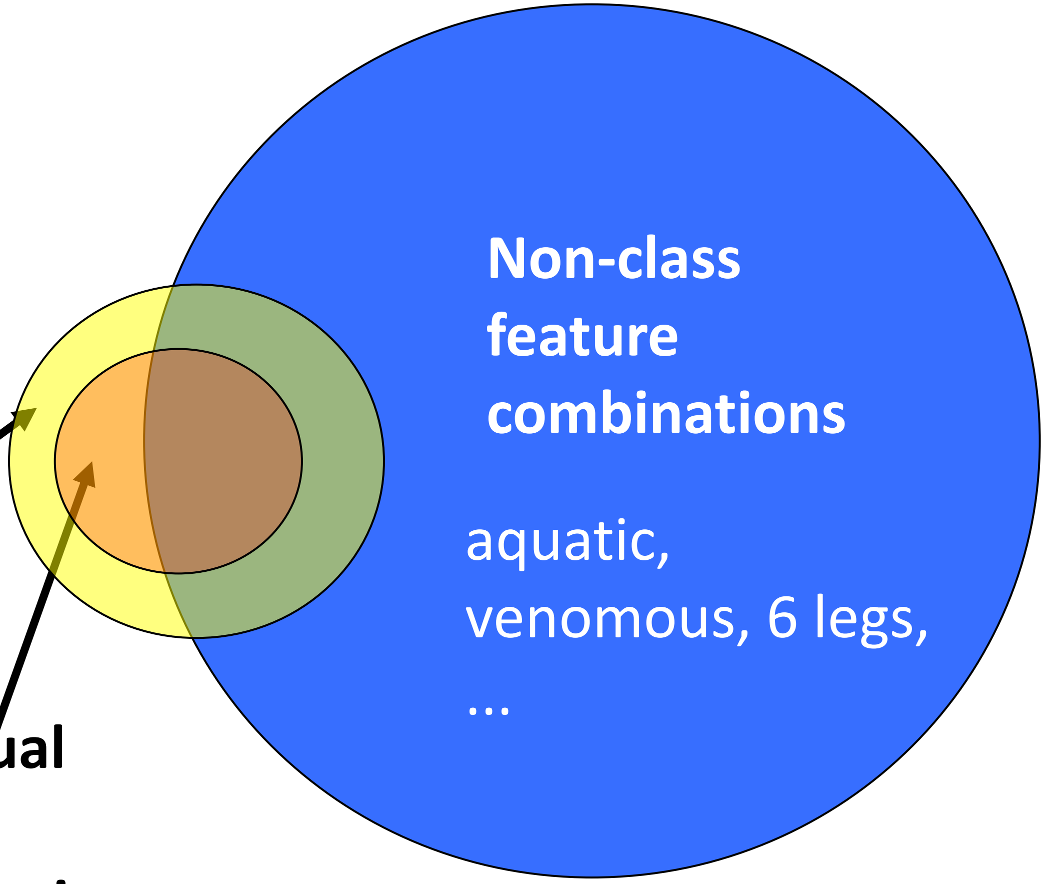


variable/value combinations in passing tests

variable/value combinations in failing tests

Combinations in failing but not in passing tests
<u>These are the ones we want</u>

Relevance to explainable AI



Class feature combinations -
brown & furry, black & furry, whiskers, claws, **...**not aquatic, not venomous, not 6 legs,

Individual feature combinations –
brown & furry, whiskers, claws, not aquatic, not venomous, not 6 legs, **...**

Animal shares features with <u>cat</u> class

Animal does not share features with <u>non-cat</u> classes

Kuhn, D. R., Kacker, R. N., Lei, Y., & Simos, D. E. (2020). Combinatorial methods for explainable AI. In *2020 IEEE Intl Conference on Software Testing, Verification and Validation Workshops (ICSTW)*

Why is this creature recognized as a reptile?

Input configuration $2^{15}6^1$

Class File: Class file rep1.csv; rows=1; cols=16

Nominal File: Nominal file notreptile.csv; rows=96; cols=16 || 2-way: 120 3-way: 560 4-way: 1,820 5-way: 4,368 6-way: 8,008

Class File Contents:

hair	feathers	eggs	milk	airborne	aquatic	predator	toothed	backbone	breathes	venomous	fins	nlegs	tail	domestic	catsize
0	0	1	0	0	0	0	0	1	1	0	0	4	1	0	1

```
---------------
0053 occurrences = 0.552 of cases, hair = 0
0076 occurrences = 0.792 of cases, feathers = 0
0055 occurrences = 0.573 of cases, eggs = 1
0055 occurrences = 0.573 of cases, milk = 0
0072 occurrences = 0.750 of cases, airborne = 0
0061 occurrences = 0.635 of cases, aquatic = 0
0044 occurrences = 0.458 of cases, predator = 0
0039 occurrences = 0.406 of cases, toothed = 0
0078 occurrences = 0.813 of cases, backbone = 1
0076 occurrences = 0.792 of cases, breathes = 1
0090 occurrences = 0.938 of cases, venomous = 0
0079 occurrences = 0.823 of cases, fins = 0
0036 occurrences = 0.375 of cases, nlegs = 4
0070 occurrences = 0.729 of cases, tail = 1
0083 occurrences = 0.865 of cases, domestic = 0
0043 occurrences = 0.448 of cases, catsize = 1
```

No single feature is sufficient explanation – shares features with non-reptiles

No pair of features sufficient – shares 2-way combinations w/ non-reptiles

```
0002 occurrences = 0.021 of cases, toothed,nlegs = 0,4
0005 occurrences = 0.052 of cases, hair,nlegs = 0,4
0005 occurrences = 0.052 of cases, milk,nlegs = 0,4
0006 occurrences = 0.063 of cases, eggs,nlegs = 1,4
0008 occurrences = 0.083 of cases, toothed,catsize = 0,1
0011 occurrences = 0.115 of cases, milk,catsize = 0,1
0012 occurrences = 0.125 of cases, eggs,catsize = 1,1
0013 occurrences = 0.135 of cases, hair,catsize = 0,1
0015 occurrences = 0.156 of cases, predator,catsize = 0,1
```

3-way combinations produce rules to explain recognition of Testudo as a reptile

```
00000 occurrences = 0.000 of cases, aquatic,toothed,nlegs = 0,0,4
00000 occurrences = 0.000 of cases, eggs,aquatic,nlegs = 1,0,4
00000 occurrences = 0.000 of cases, hair,aquatic,nlegs = 0,0,4
00000 occurrences = 0.000 of cases, hair,nlegs,catsize = 0,4,1
00000 occurrences = 0.000 of cases, milk,aquatic,nlegs = 0,0,4
00000 occurrences = 0.000 of cases, milk,nlegs,catsize = 0,4,1
00000 occurrences = 0.000 of cases, predator,toothed,nlegs = 0,0,4
00001 occurrences = 0.010 of cases, eggs,nlegs,catsize = 1,4,1
00001 occurrences = 0.010 of cases, eggs,predator,nlegs = 1,0,4
00001 occurrences = 0.010 of cases, feathers,toothed,backbone = 0,0,1
```

Non-reptiles in the database do not have these 3-way combinations

Only reptiles have these combinations of features:

not aquatic AND not toothed AND four legs

egg-laying AND not aquatic AND four legs

not hairy AND four legs AND cat size

not milk-producing AND not aquatic AND four legs

not milk-producing AND four legs AND cat size

not predator AND not toothed AND four legs

Mapping combinations to expressions

- Report identifies t-way combinations that distinguish the predicted class from others
- Combinations can be mapped to expressions to produce a rule-based type of explanation

if (not aquatic AND not toothed AND four legs)
OR (egg-laying AND not aquatic AND four legs)
OR (not hairy AND four legs AND cat size)
OR (not milk-producing AND not aquatic AND four legs)
OR (not milk-producing AND four legs AND cat size)
OR (not predator AND not toothed AND four legs)
then reptile;
else not reptile;

As noted, none of the single factors above is sufficient for explanation

Example: empty vs. occupied rooms, using sensor data

File Information

Class File: Class file o1.csv; rows=1; cols=5

Nominal File: Nominal file empty.csv; rows=7703; cols=5 || 2-way: 10 3-way: 10 4-way: 5 5-way: 1 6-way: 0

Class File Contents:

Temperature	Humidity	Light	CO2	HumidityRatio
B3	B3	B2	B2	B4

2-Way | 3-Way | 4-Way | 5-Way | 6-Way

Enabled

```
Combinations = 10, Settings = 210

0016 occurrences = 0.002 of cases, Humidity,Light = B3,B2
0016 occurrences = 0.002 of cases, Light,CO2 = B2,B2
0036 occurrences = 0.005 of cases, Temperature,Light = B3,B2
0040 occurrences = 0.005 of cases, CO2,HumidityRatio = B2,B4
0043 occurrences = 0.006 of cases, Light,HumidityRatio = B2,B4
0054 occurrences = 0.007 of cases, Temperature,CO2 = B3,B2
0078 occurrences = 0.010 of cases, Humidity,CO2 = B3,B2
0205 occurrences = 0.027 of cases, Temperature,HumidityRatio = B3,B4
0247 occurrences = 0.032 of cases, Temperature,Humidity = B3,B3
0495 occurrences = 0.064 of cases, Humidity,HumidityRatio = B3,B4
---------------
0523 occurrences = 0.068 of cases, Temperature = B3
2415 occurrences = 0.314 of cases, Humidity = B3
0085 occurrences = 0.011 of cases, Light = B2
0534 occurrences = 0.069 of cases, CO2 = B2
2190 occurrences = 0.284 of cases, HumidityRatio = B4
```

Why do we conclude this room is occupied?

These levels of humidity and lighting are strong indication

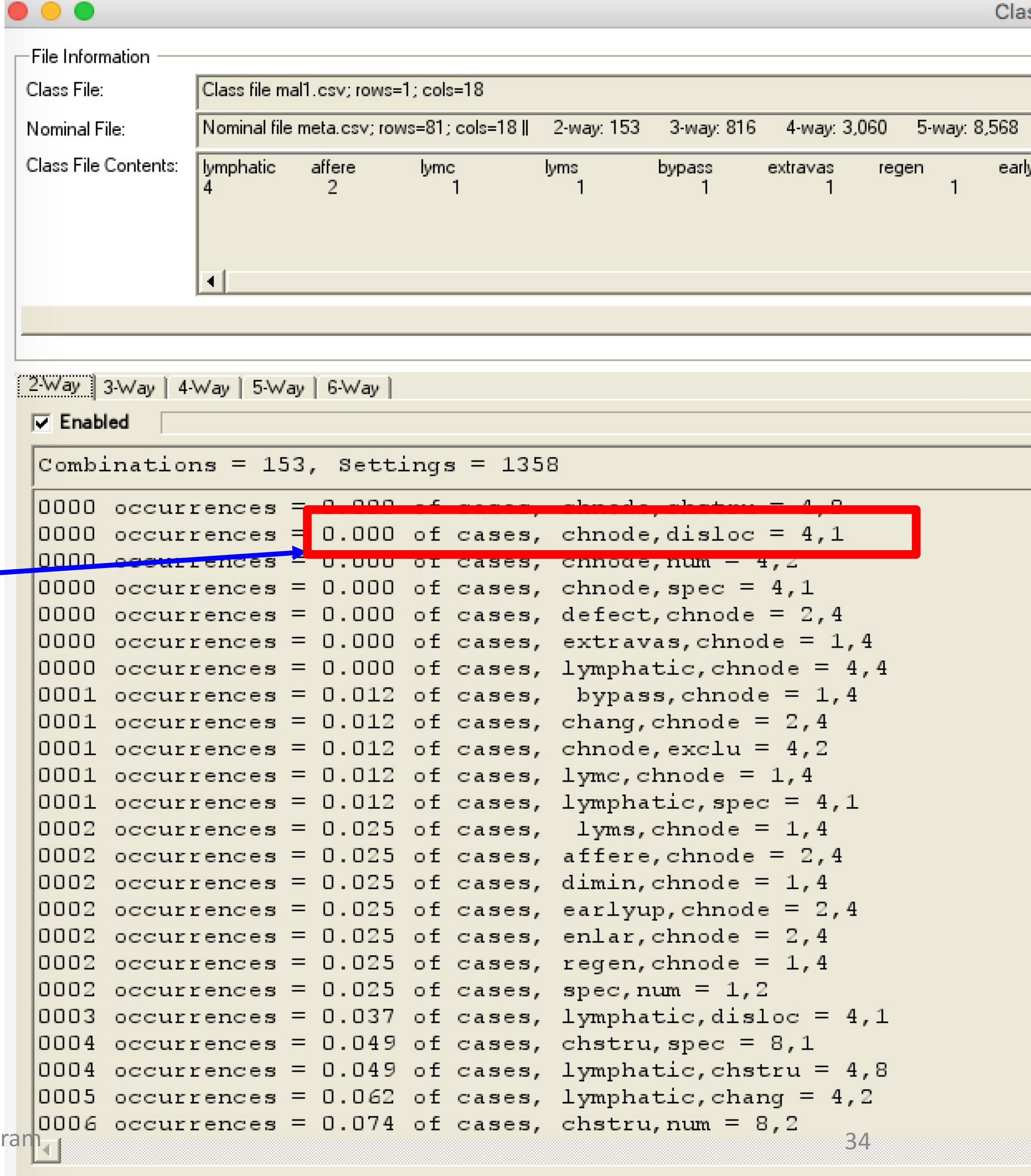
Considering levels of lighting, CO2, and humidity ratio provide even stronger evidence:

<u>Empty rooms don't have these levels</u>

```
00003 occurrences = 0.000 of cases, Light,CO2,HumidityRatio = B2,B2,B4
00005 occurrences = 0.001 of cases, Humidity,Light,CO2 = B3,B2,B2
00008 occurrences = 0.001 of cases, Temperature,Light,CO2 = B3,B2,B2
00011 occurrences = 0.001 of cases, Humidity,Light,HumidityRatio = B3,B2,B4
```

A different example: lymph node pathology – why is this classified as malignant not metastatic?

- These combinations are characteristic of lymphoma that arises in lymph node instead of metastatic that spread to node from somewhere else



Summary - explainable AI

- Combinatorial methods can provide explainable AI
- We have prototype that applies this approach
  - Determine combinations of variable values that differentiate an example from other possible conclusions
    - ➔ Feature combinations present shared with class
    - ➔ Feature combinations not shared with class not present
- Method can be applied to black-box functions such as CNNs
- Present explanation in the preferred form of rules, “if A & B, or C with D & E, then conclusion is X”

Outline

- Why current safety-critical testing won’t work
- Assurance based on input space coverage
- Explainable AI as part of validation, and
- Transfer learning

Transfer learning – what is the problem?

- Differences inevitably exist between training data sets, test data sets, and real-world application data
- Further differences exist between data from two or more different environments
- How do we predict performance of a model trained on one data set when applied to another?
  - New environment
  - Changed environment
  - Additional possible values
  - etc.

Lanus, E., Freeman, L. J., Kuhn, D. R., & Kacker, R. N. (2021, April). Combinatorial Testing Metrics for Machine Learning. In *2021 IEEE Intl Conference on Software Testing, Verification and Validation Workshops (ICSTW)*

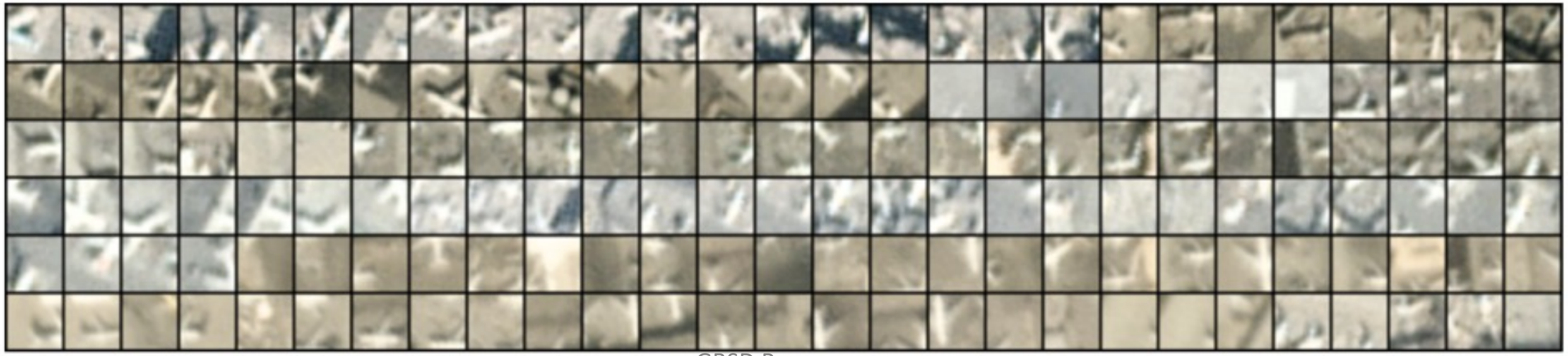
Transfer learning – conventional practice

- Randomized selection – but will randomization be sufficient, especially with smaller data sets?
- Ensure at least one of each object type – but this may not be representative of object attribute distributions

- Interactions are critical to consider in most ML problems, especially for safety, but conventional practice does little to ensure data sets are adequately representative of interactions

Example – image analysis

- Planes in satellite imagery – Kaggle ML data set – determine if image contains or does not contain an airplane
- Two data sets – Southern California (SoCal, 21,151 images) or Northern California (NorCal, 10,849 images)
- 12 features, each discretized into 3 equal range bins



Transfer learning problem

- Train model on one set, apply to the other set
- Problem –
  - Model trained on larger, SoCal data applied to smaller, NorCal data → performance drop
  - Model trained on smaller, NorCal data applied to larger, SoCal data → NO performance drop
- This seems backwards!
- Isn’t it better to have more data?
- Can we explain this and predict it next time?

Density of combinations in one but not the other data set, 2-way

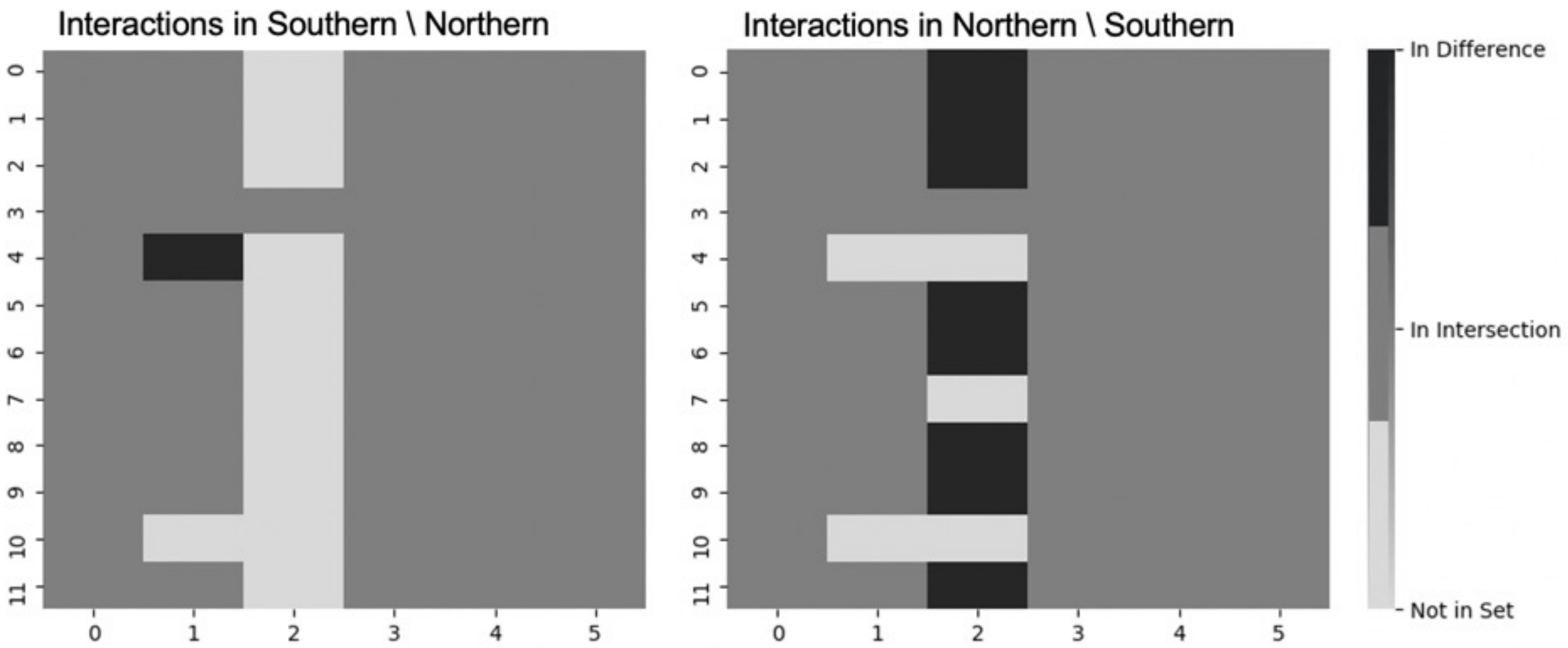


Image from Combinatorial Testing Metrics for Machine Learning, Lanus, Freeman, Kuhn, Kacker, IWCT 2021

For C = SoCal, N = NorCal,
|C\N| / |C| = 0.02
|N\C| / |N| = 0.12

→ The NorCal data set has fewer “never seen” combinations, even with half as many observations

Summary – Transfer learning

- Current approaches to estimating success for transfer learning are largely ad-hoc and not highly effective
- Combinatorial methods show promise for improvements – measurable quantities directly related to determining if one data set is representative of the field of application
- Much additional work is needed to evaluate this idea, and to understand the link between combinatorial difference values and prediction accuracy
- Empirical studies planned

Assured autonomy – more questions than answers

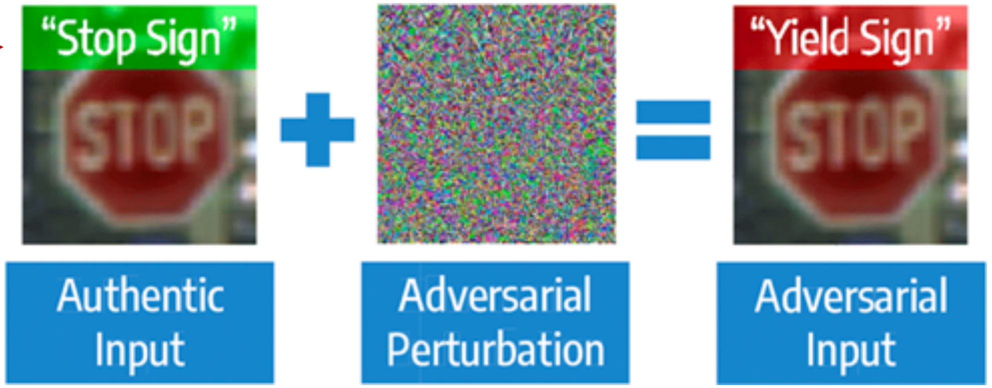
- How to classify bug types – in learning systems that are programmed by their inputs, *continuously*
  - Are they mostly aging-related bugs?
  - Or something else not yet defined?
- Interactions of learning components with programmed components – especially replacing humans
  - Changes the nature of system failures
  - More like failures involving human factors issues? ☺ Turing test for bugs! Distinguish between human-triggered and AI-triggered system failures?

Assured autonomy – key points & current state

- For capability and cost reasons, <u>autonomous components are becoming routine</u> in software engineering
- Many, or most, <u>methods used in high assurance conventional systems do not apply</u> to many autonomous components
  - Structural coverage – not for neural nets, and others
  - Formal proofs – for some parts but limited
- How to deal with learning, dynamic changes in system, routine non-determinism?
- Developing appropriate measures of test adequacy

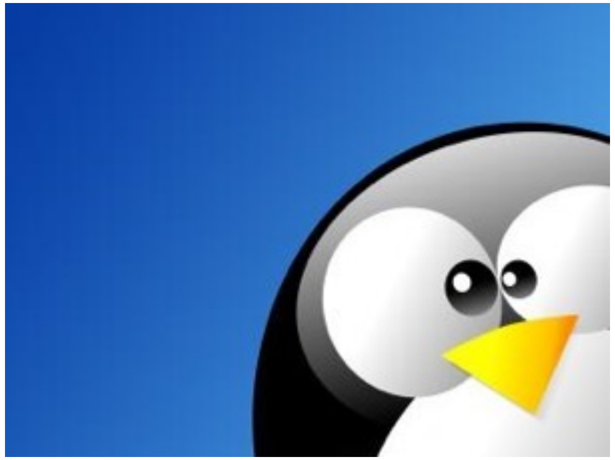
Where are we going?

- Need new approaches in:
  - Design
  - Simulation
  - Validation
  - Formal verification
  - Testing
  - Explainability
- *Security – much bigger problem than safety assurance – solvable?*
  - *All the old vulnerabilities apply – with greater consequences*
  - *And new vulnerabilities*



- *Leading to … AI vs. AI?*

Please contact us if you're interested!



Rick Kuhn, Raghu Kacker, M.S. Raunak
{kuhn, raghu.kacker, raunak}@nist.gov

http://csrc.nist.gov/acts