Measurements to Improve AI/ML Training Data Sets

Rick Kuhn*, M S Raunak*, Raghu Kacker*, Jaganmohan Chandrasekaran**, Erin Lanus**, Tyler Cody**, Laura Freeman** * National Institute of Standards and Technology, ** Virginia Tech National Security Institute

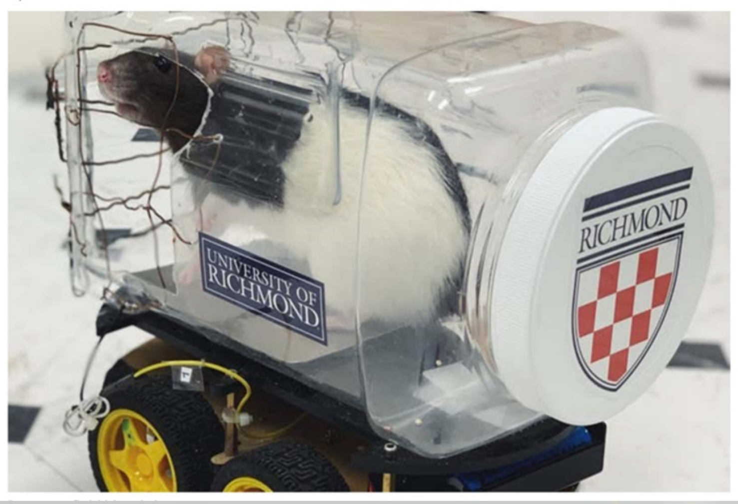
https://csrc.nist.gov/acts

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

Multiple conditions involved in accidents

"The camera failed to recognize the white truck against a bright sky" (2 factors)

"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" (3 factors)

We need to understand what combinations of conditions are included in testing

Conventional critical software testing is based on structural coverage – ensuring that conditions, decisions, paths are covered in testing

Life-critical aviation software requires MCDC testing, white-box criterion that doesn't fit neural nets and other black-box methods where input is what matters

We may have perfect structural coverage of code, but what does that tell us about response to rare inputs

How can we measure combination coverage?

rows of input

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

19 combinations included in test set

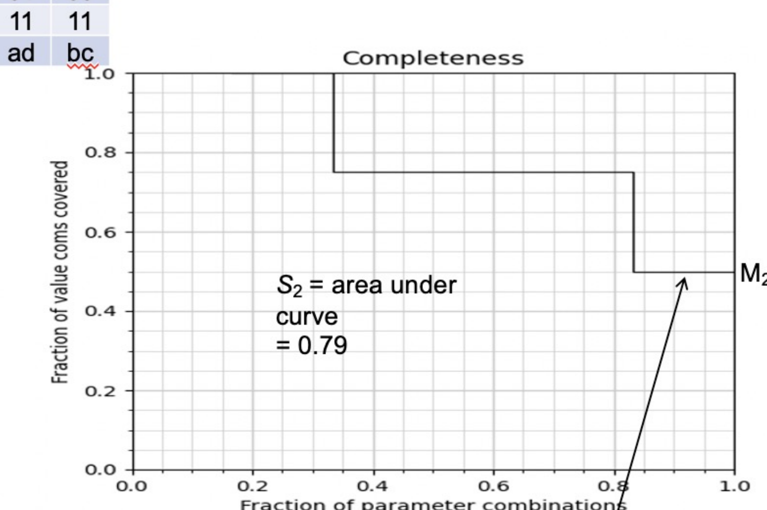
Vars	Combination values	Coverage
a b	00, 01, 10	.75
a c	00, 01, 10	.75
a d	00, 01, 11	.75
b c	00, 11	.50
b d	00, 01, 10, 11	1.0
c d	00, 01, 10, 11	1.0

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*

Graphing Coverage Measurement

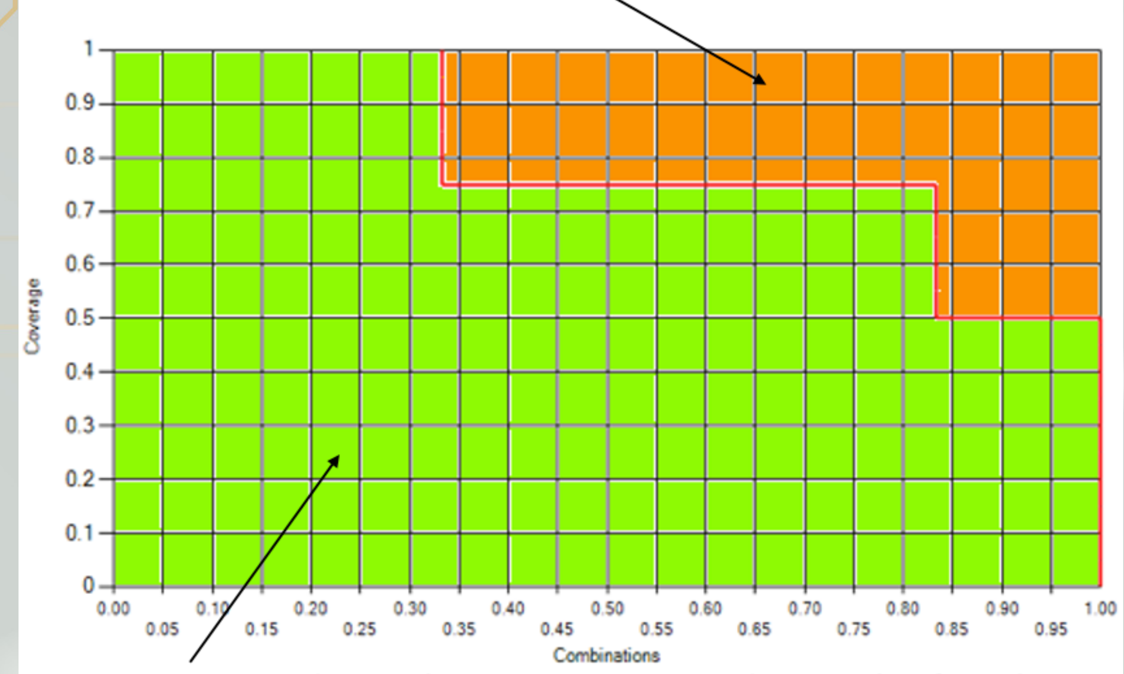
1.00	00	00				
.75	01	01	00	00	00	
.50	10	10	01	01	01	00
.25	11	11	10	10	11	11
	bd	cd	ab	ac	ad	bc



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

$1 - S_t$ = **Untested combinations** (look for problems here)



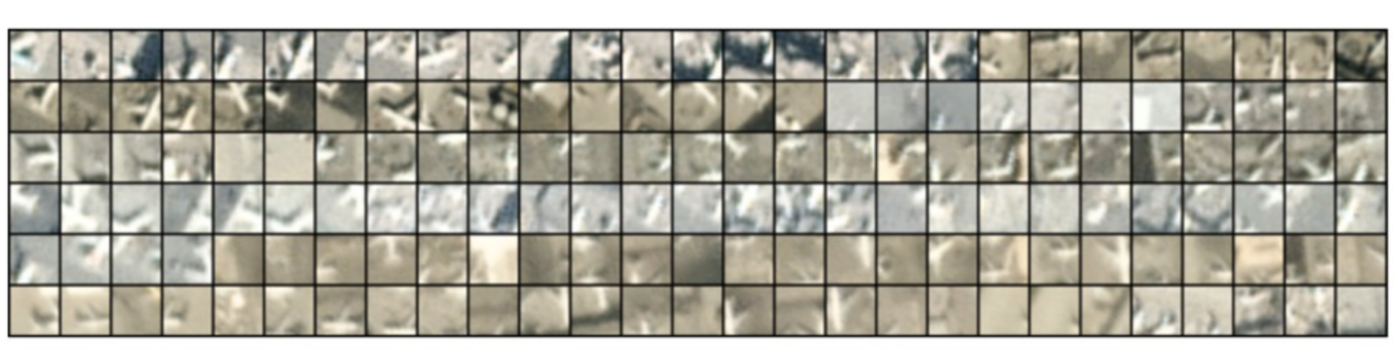
S_t = Tested combinations => code works for these

Using these measurements

Transfer learning – predict performance of a model trained on one data set when applied to another

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 measure, explain and predict it next time?

Density of combinations in one versus the other data set, 2-way

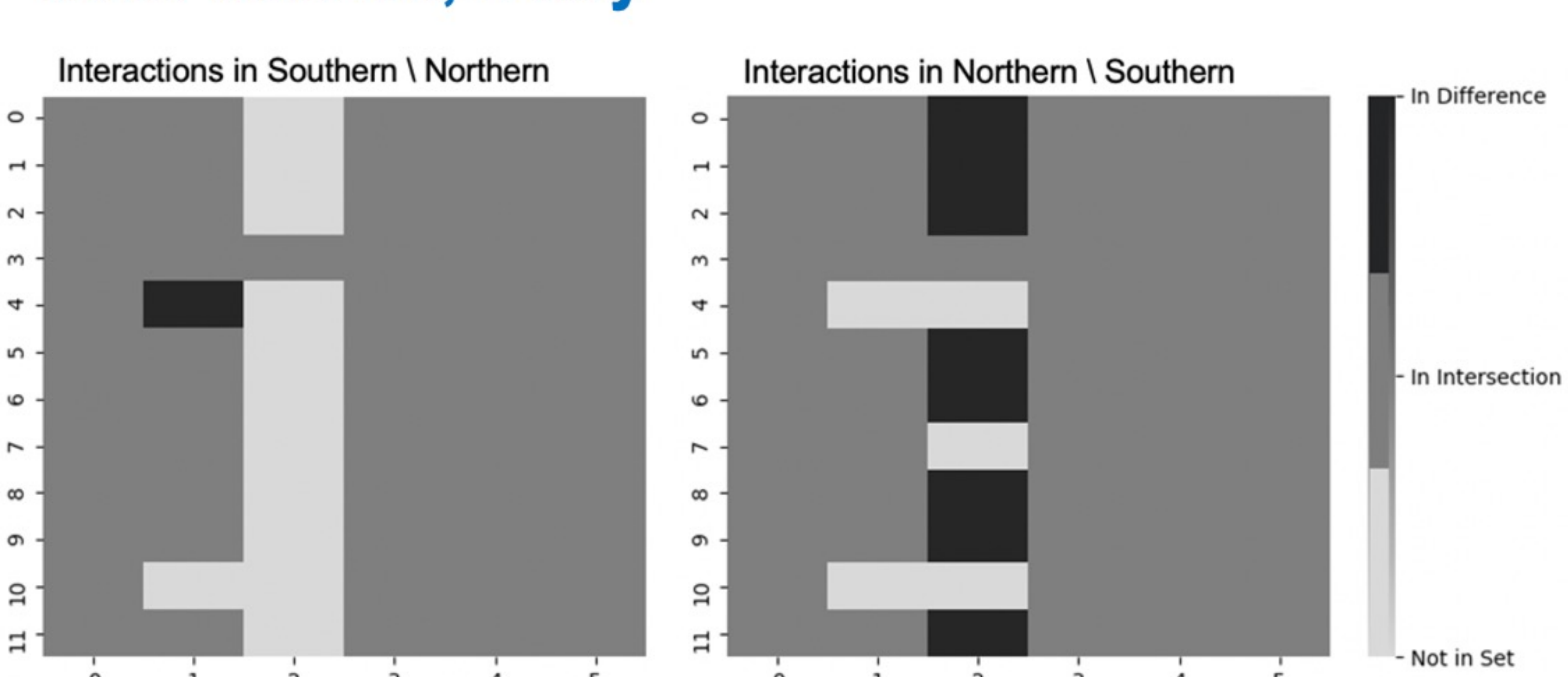


Image from Combinatorial Testing Metrics for Machine Learning, Lanus, Freeman, Kuhn, Kacker, IWCT 2021

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

→ The smaller data set has fewer "never seen" combinations, even with half as many observations

Assured autonomy – key points & current state

- Autonomous components are becoming routine in software engineering
- Methods used in high assurance conventional systems
  - Not sufficient for 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?
- Understanding and measuring interaction coverage is necessary

THE TWENTY-FOURTH ANNUAL HIGH CONFIDENCE SOFTWARE AND SYSTEMS CONFERENCE
MAY 6 - 8, 2024 | http://cps-hcss.org