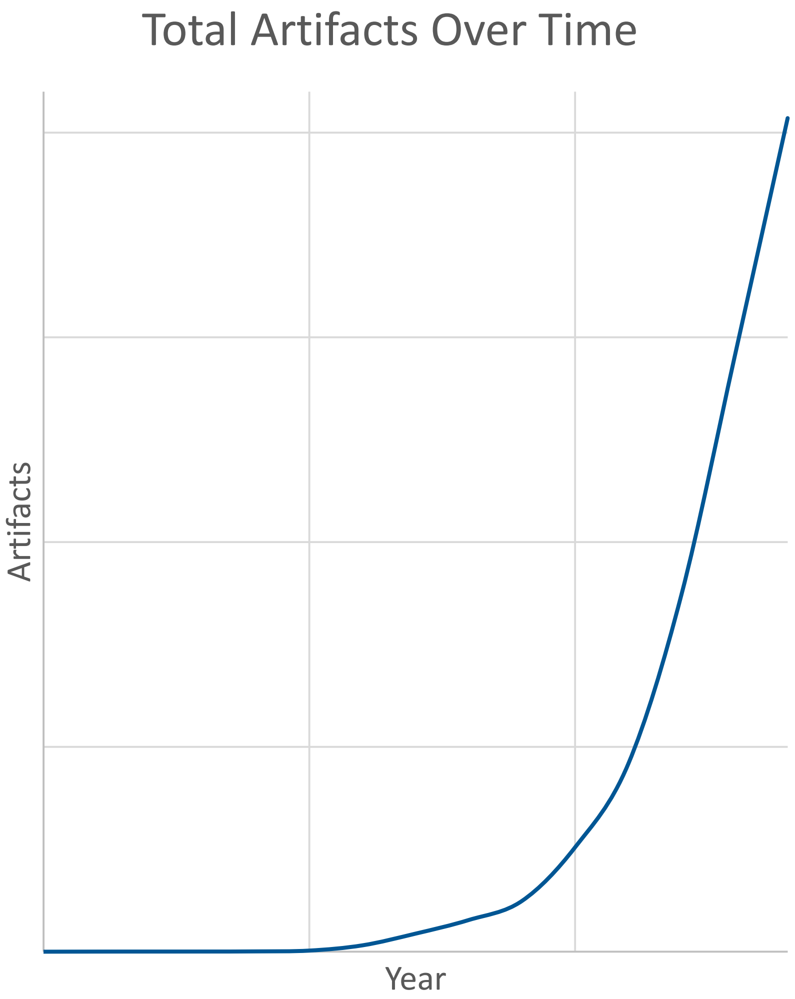
Human-Computer Decision Systems for Cybersecurity

N. VanHoudnos, S. Moon, D. French, Brian Lindauer*
P. Jansen, J. Carbonell, C. Hines, W. Casey

The Problem – Malware Classification



- Task: Given an exemplar, find other malware artifacts of the same class (family, behavior, etc.) in our existing catalog.
- Problem: Diversity and volume of incoming malware
  - Human analysis is far too expensive
  - Can't run all tools on all samples
- Malware variation is unpredictable in mode or frequency
  - "I'll know it when I see it" hard to quantify

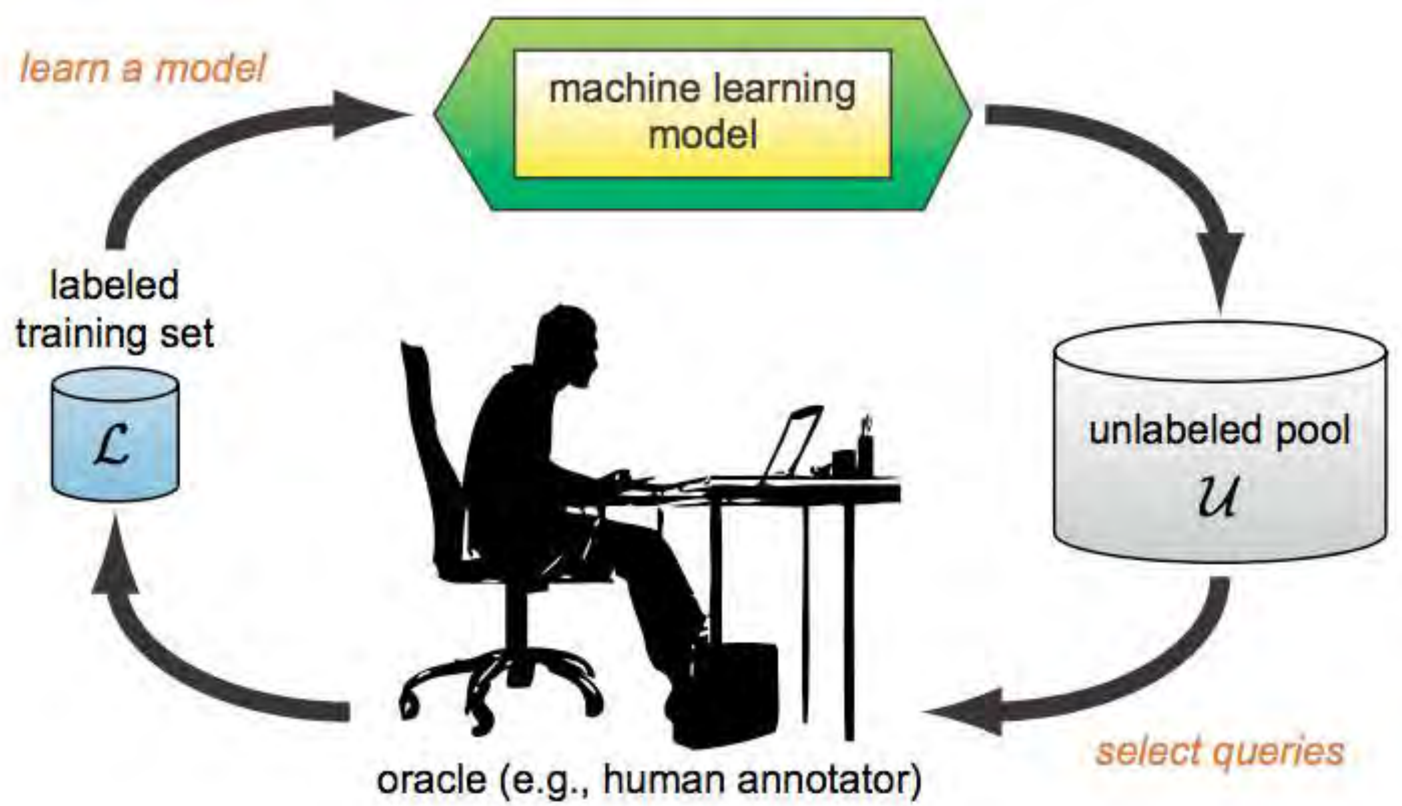
Features for Malware Classification

- Features from static analysis
  - Decompositional Techniques
    - *Section hashes
    - Resource hashes
  - Interpretive Techniques
    - Function hashes
    - *Mnemonic class histograms
    - *Import address table (IAT) hashes
- Features from runtime analysis
  - Host-based
    - *System call traces / call graphs
    - Filesystem operations
    - Registry operations
  - Network-based

* Explored in this project

First Id

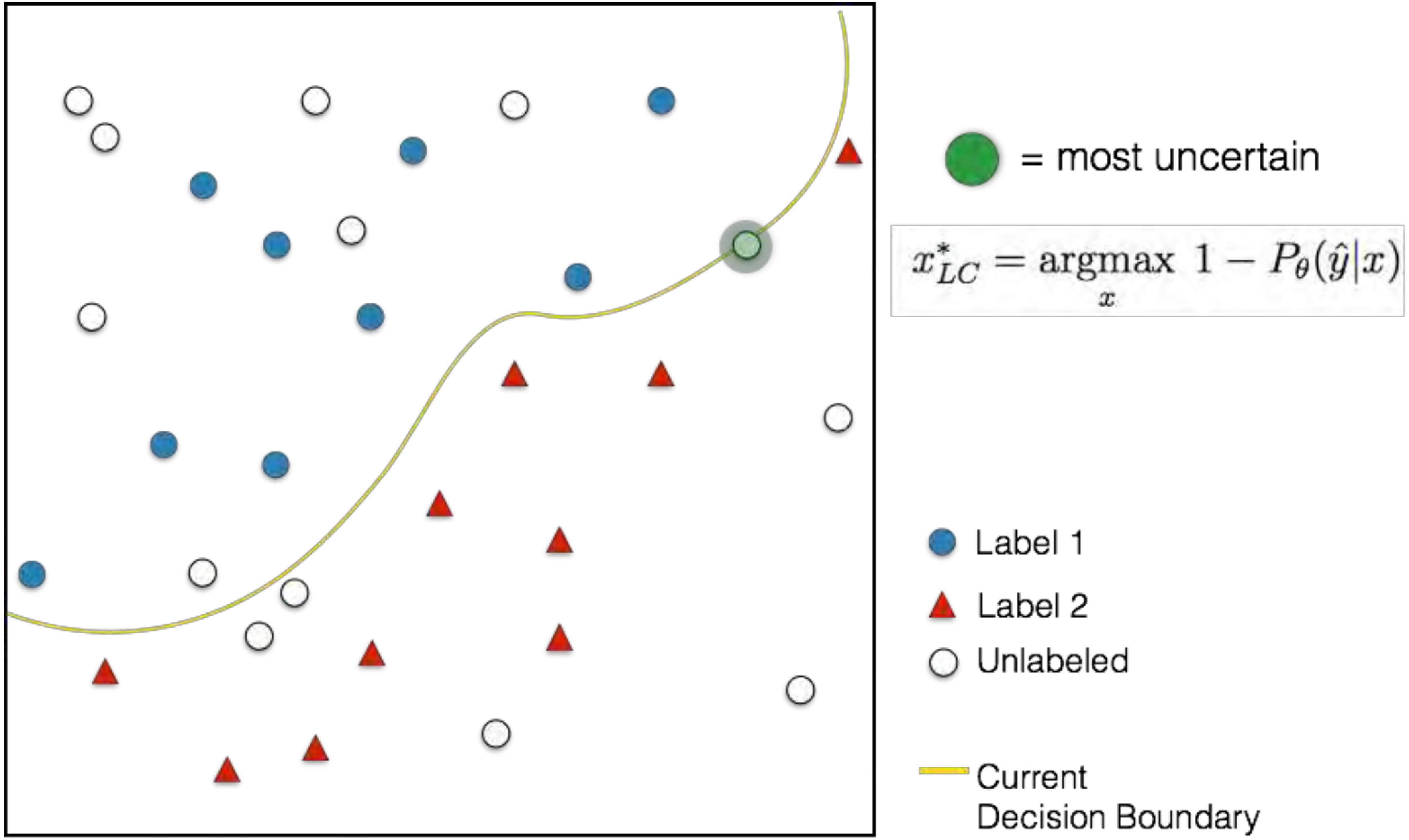
Active Learning



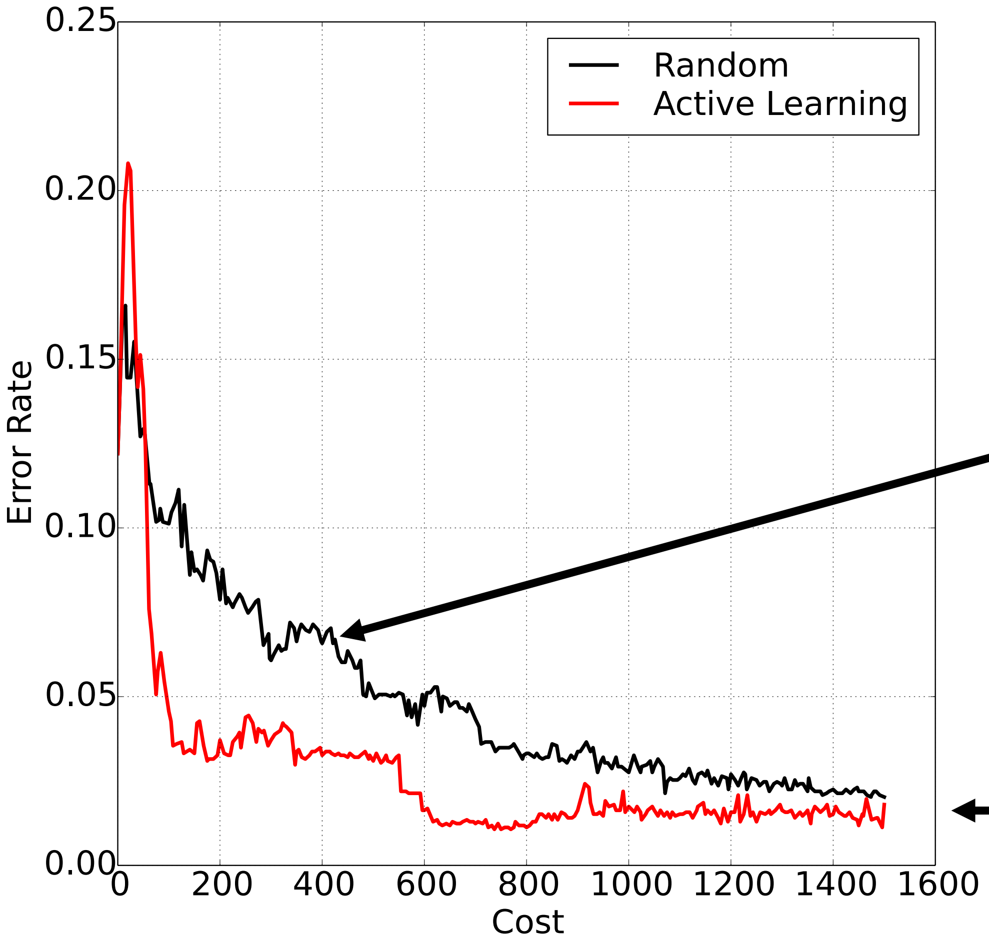
[Settles, 2011]

[Settles 2011]

Traditional Active Learning (Uncertainty-Based)

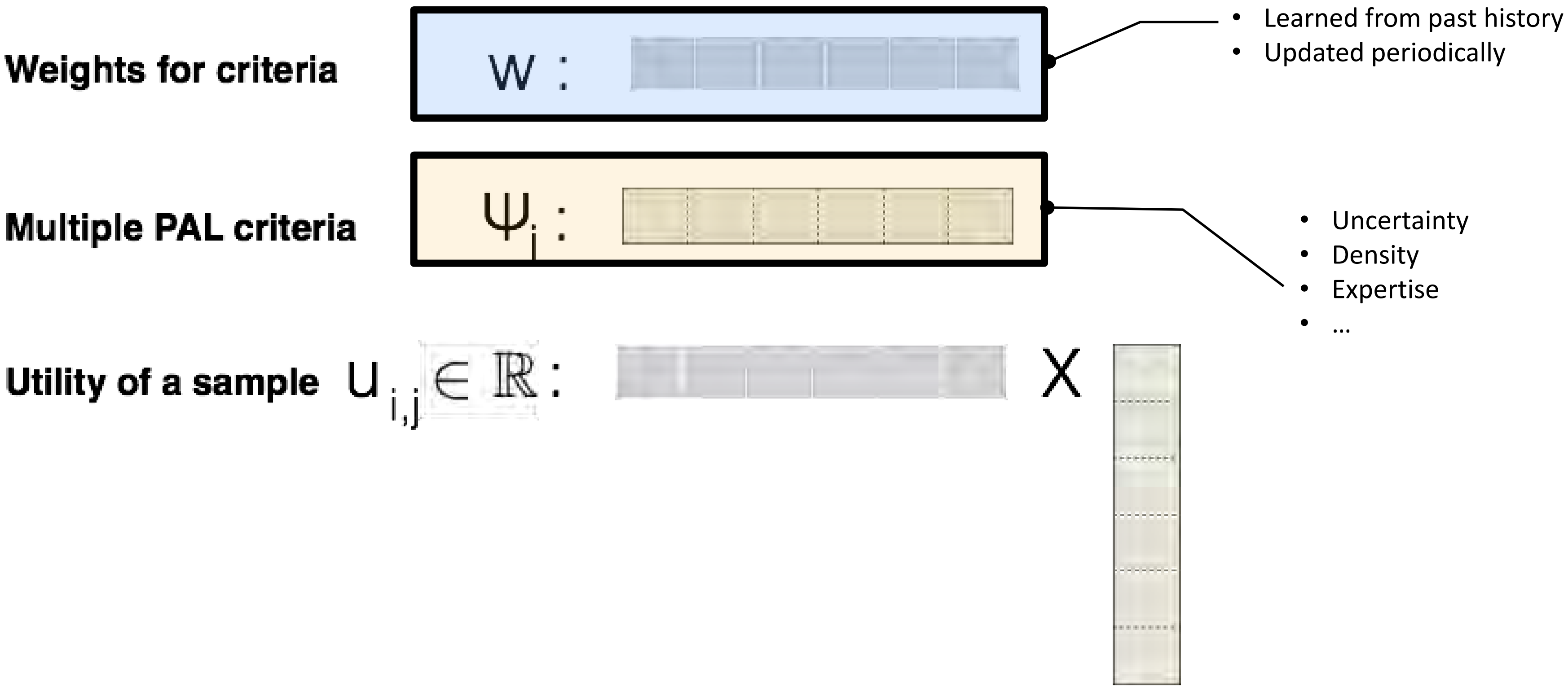


Simulated Active Learning



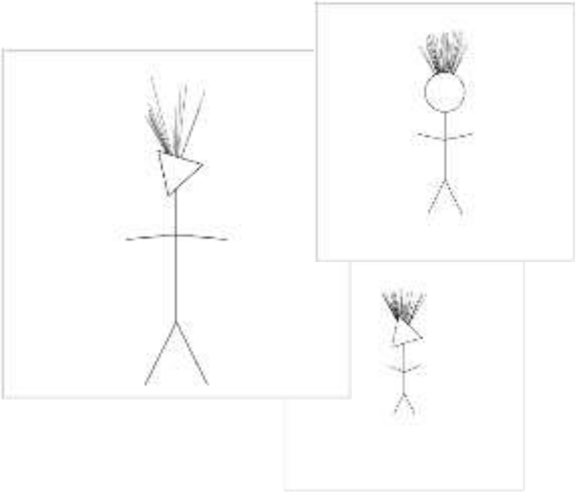
Learn a difficult task with less training data.

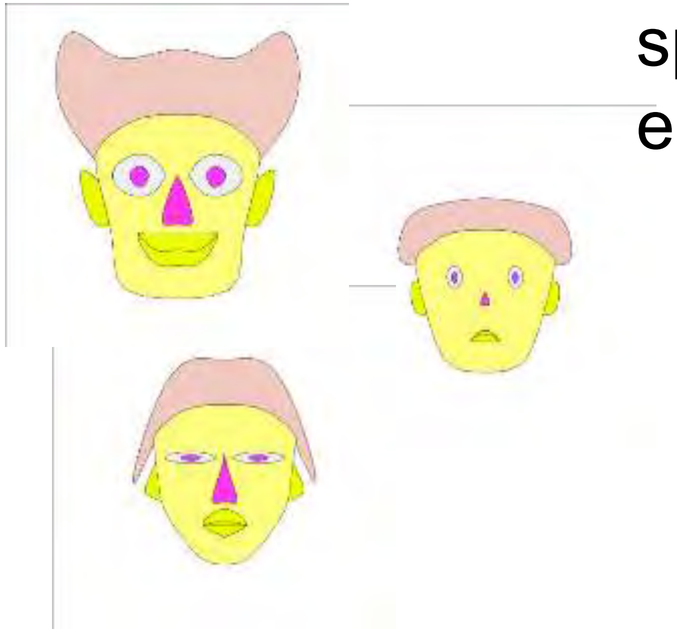
Dynamic Proactive Learning (DPAL)



DPAL is a mathematical framework to support active learning using many simultaneous criteria.

A Proxy Task

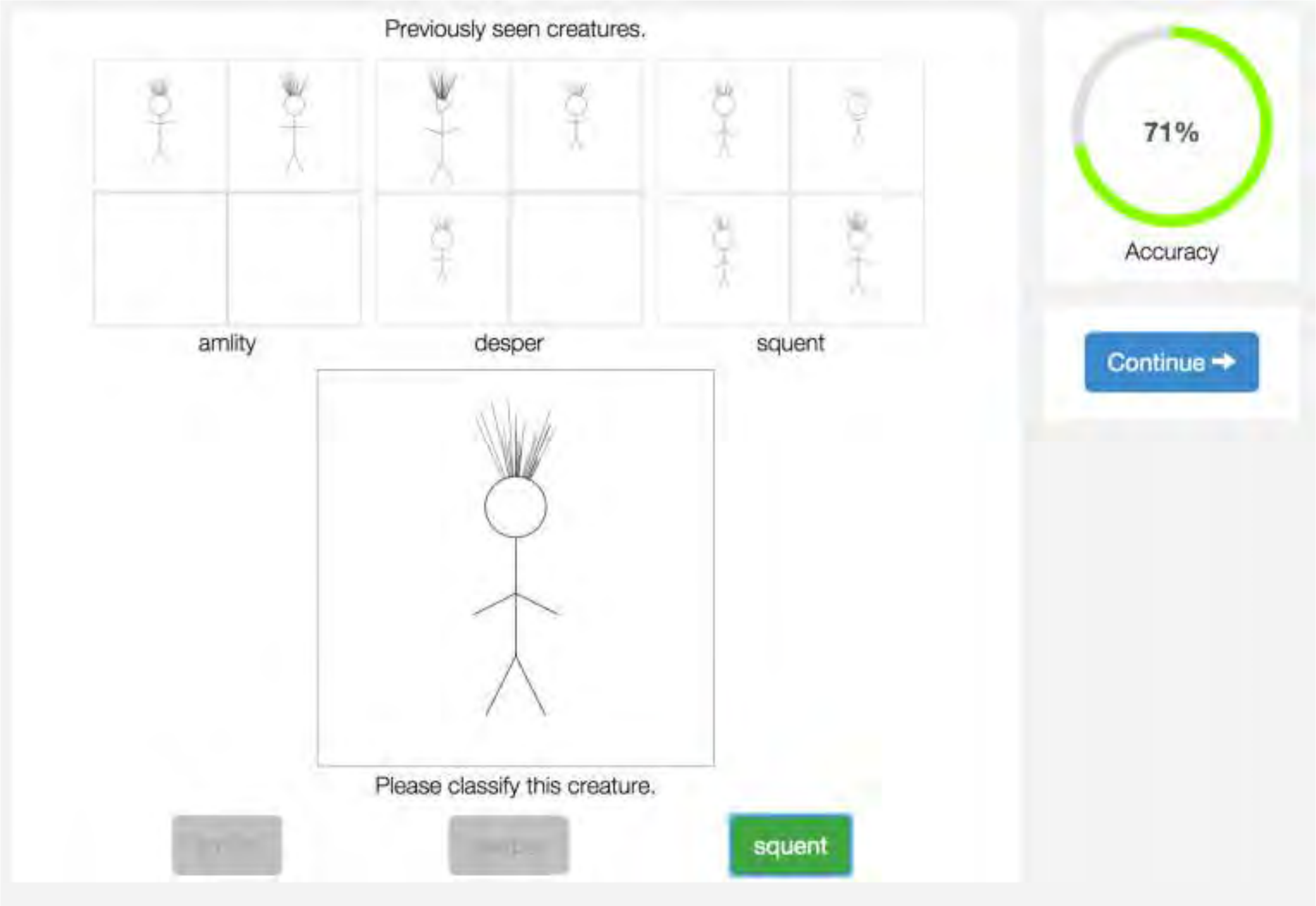




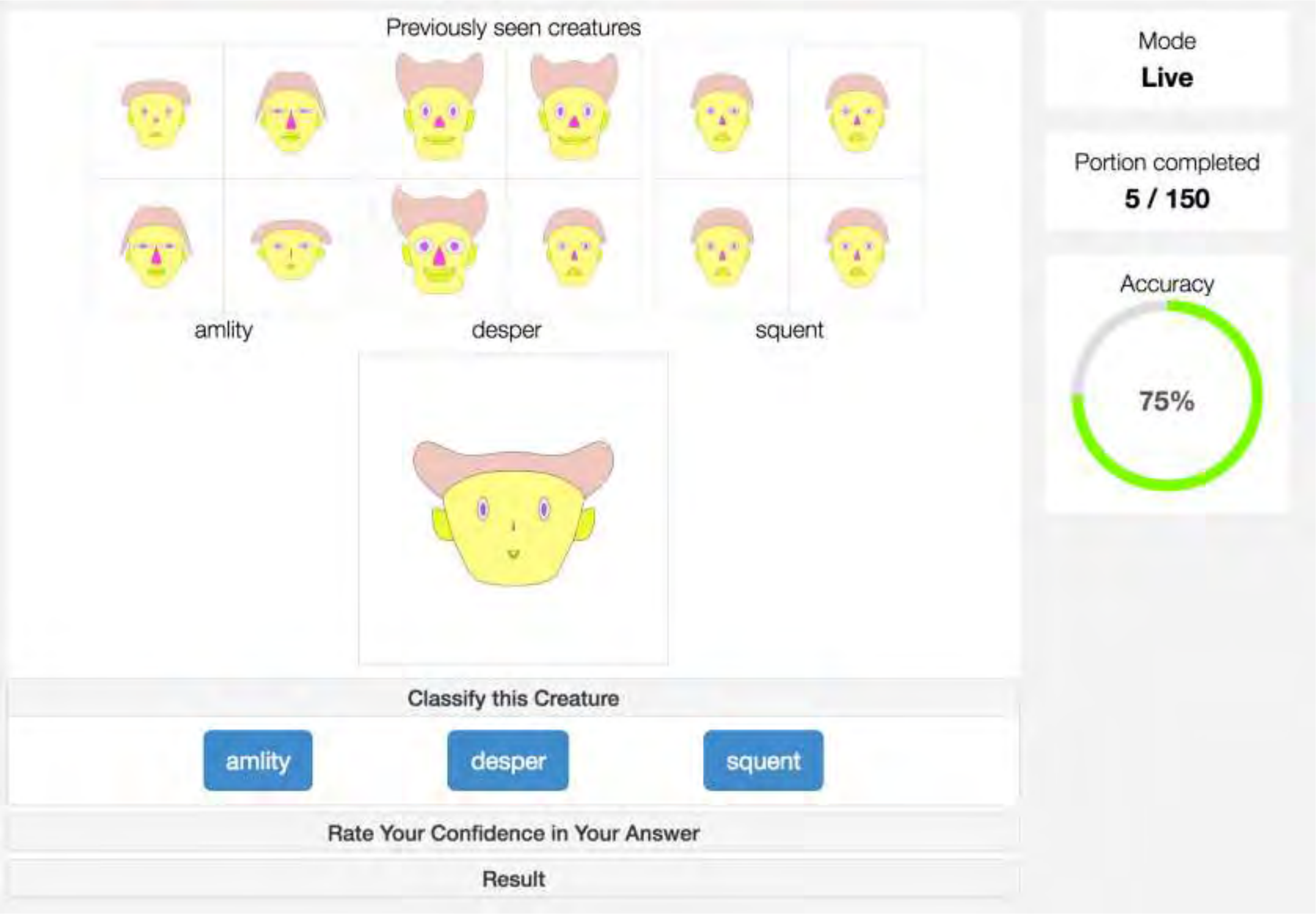
For validation, we need a **learnable task** and **ground truth**.

To stay close to the real data, we projected the samples into a four dimensional PCA space, and mapped those dimensions onto either stick figures or Chernoff faces.

Creature Classification on AMT

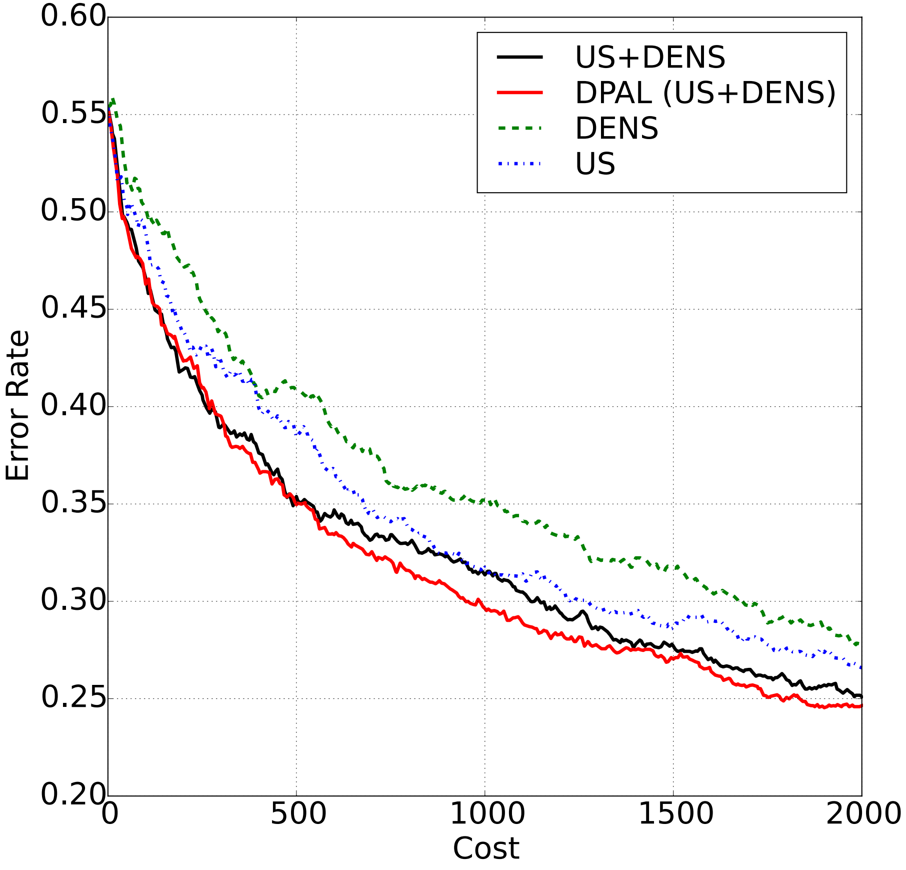


Creature Classification on AMT

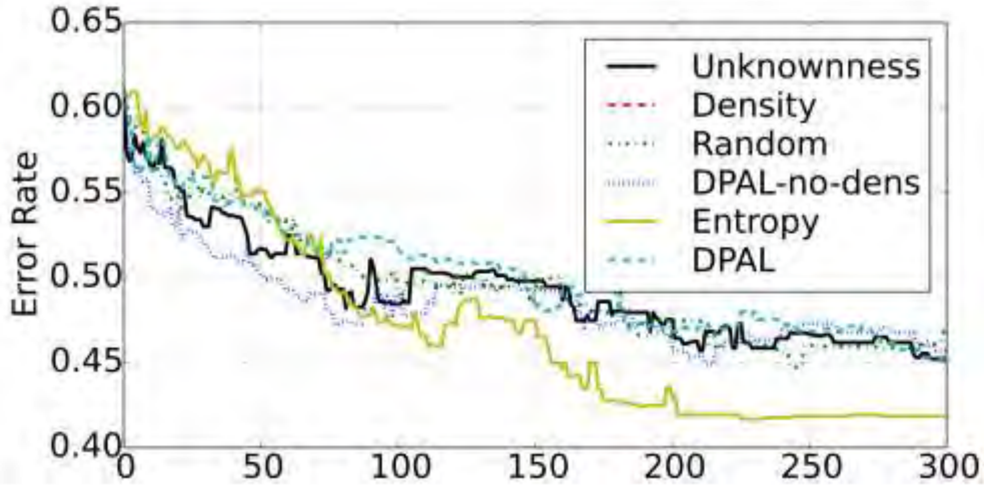


DPAL Only Helps In Simulation

DPAL in Simulation



DPAL with Real Users

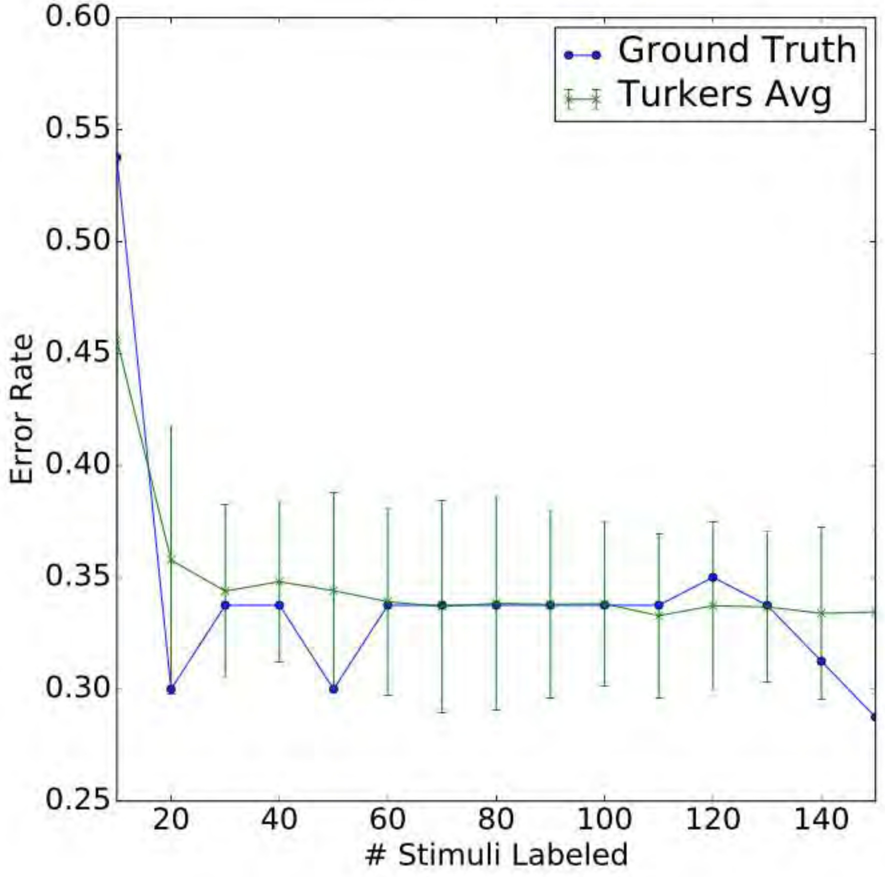


- Entropy (very simple!) wins.
- **Runtime features are too discriminative** for DPAL to gain an advantage.

Result: A framework for dynamically re-balancing the selection of points for labeling, extending and generalizing existing active learning methods.

(But it doesn't help on our task.)

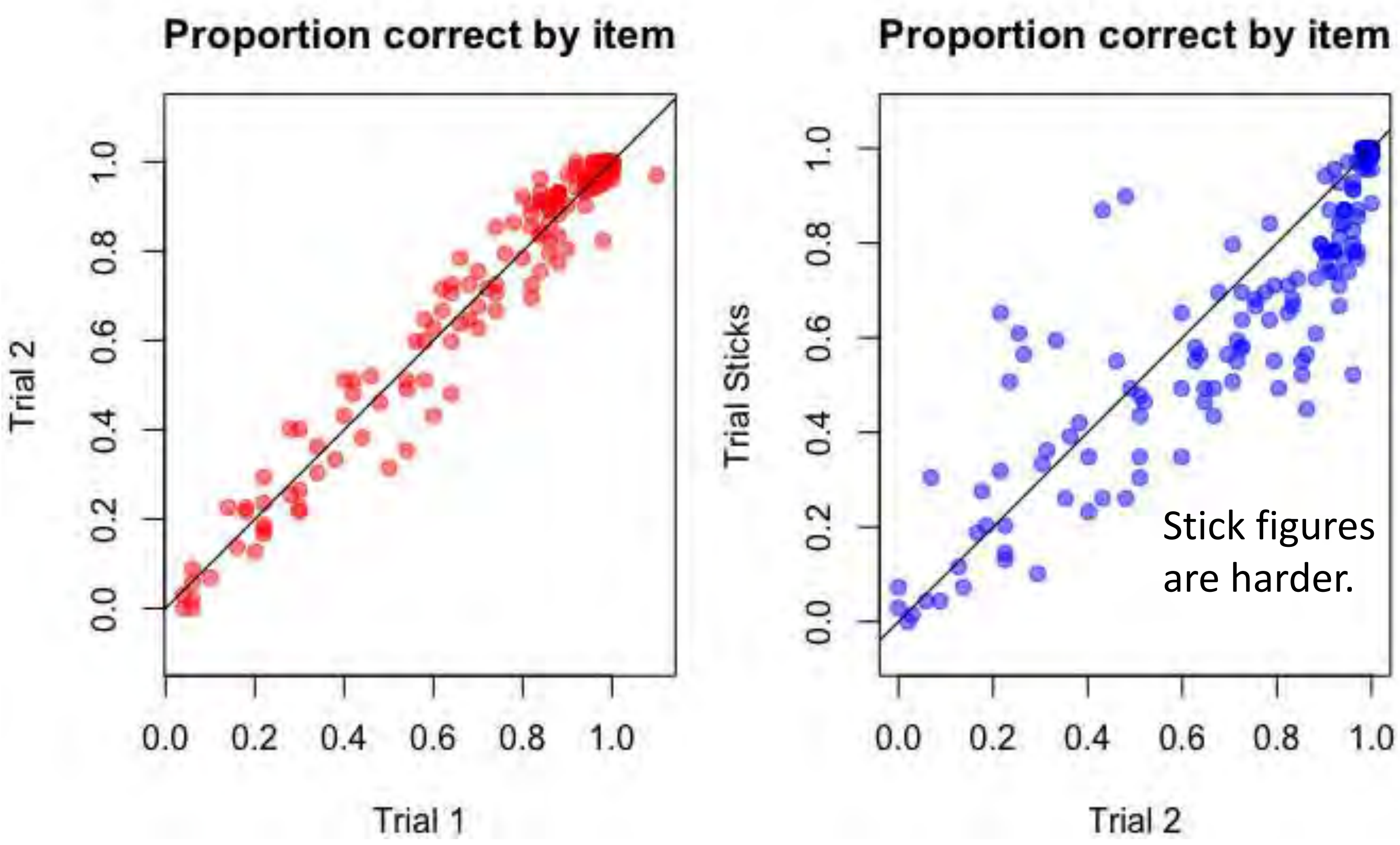
Expert Labels vs. Turker Labels



- “Ground Truth” shows an SVM trained with expert ground truth labels.
- ”Turkers Avg” trains the classifier with layperson labels instead.

Results

- The difficulty of a given stimulus is consistent across runs.



Result: Simple visualizations can allow even completely untrained people to differentiate malware families.

Questions:

- If visualizations are that effective for laypeople, can we use them to help experts?
- What if we could only use cheaper static features?

Static Analysis of a Large Scale Malware Repo

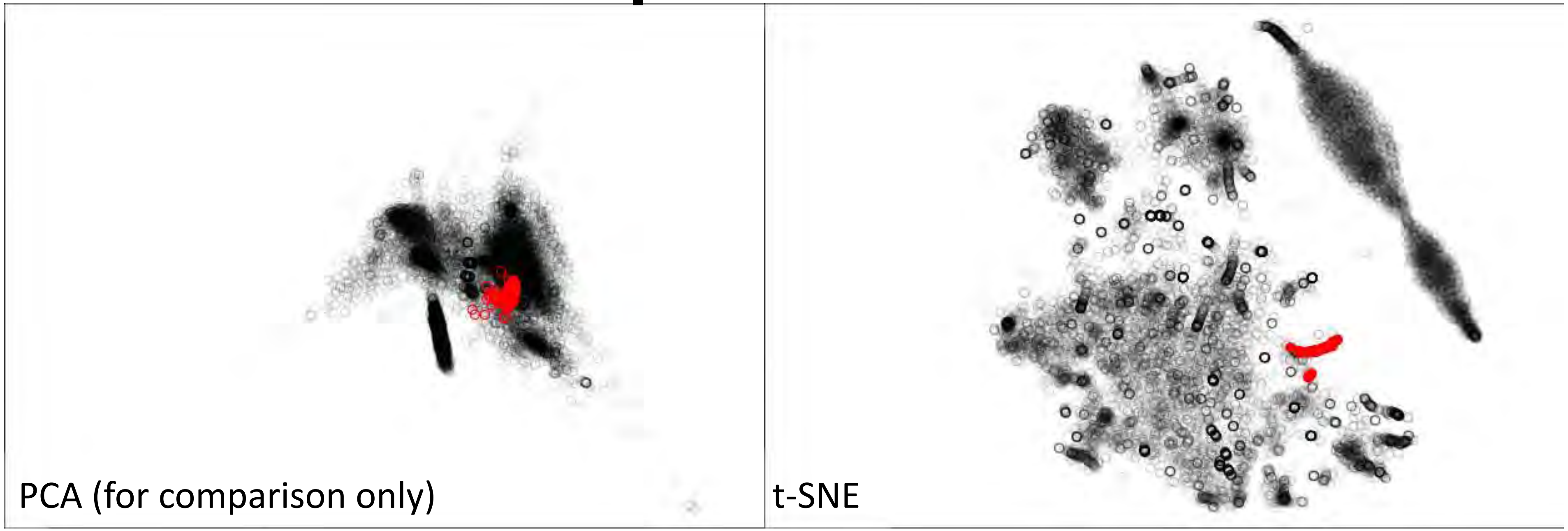
Data

- Approximately 50k suspicious binaries from Virus Total collected in January 2016.
- No ground truth – individual reverse engineering as needed

Two kinds of static analysis features computed:

- Cheap, broad brush features
  - Entry-Point (EP) section hash
  - Import Address Table (IAT) hash
- Expensive, fine grained features from disassembly
  - Mnemonic counts per function (add, mov, jmp, and others)
  - Approximate sequence of mnemonics per file

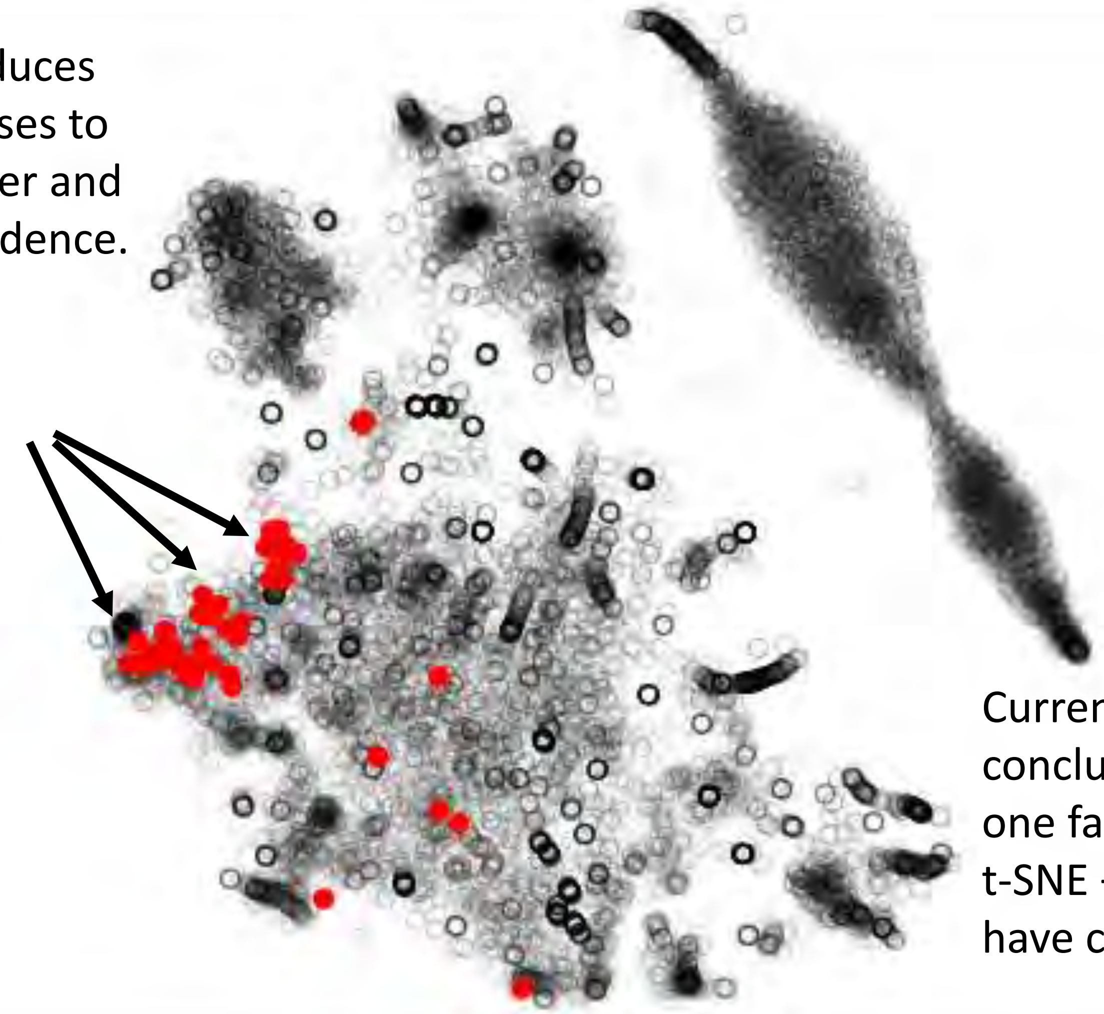
How Good is Your Cheap Feature?



- 20k observations of 545 mnemonic counts reduced to two dimensions.
- Red points are a specific IAT hash of interest.
- This IAT hash (cheap) is well localized in t-SNE space (expensive)
- Knowing this IAT hash is likely good enough to define this family.
- Expert analysis concludes this is a single family.

Cheap Can Be Noisy… A Different IAT Hash

Embedding reduces the number cases to reverse engineer and increases confidence.



Current analysis methods conclude this IAT hash is one family.
t-SNE + IAT = family would have cost less.

Result: t-SNE-based visualizations paired with IAT hashes greatly reduce the number of manual binary analyses required to understand new groups of binaries.

Future work

Operationalize this proof-of-concept

- Improve quality of t-SNE embedding to better localize groups
- Implement a predictive model to embed binaries as they stream in
- Measure if this tool increases effectiveness of early career reverse engineers.

Add dynamic, runtime features as a third level of visualization.

- Section hashing -> t-SNE static analysis -> t-SNE runtime features
- Learn when to stop, and when to move on to the next level.
- Increase confidence in findings from prior levels.

Conclusions

- Machine learning and human analysts provide complementary capabilities in malware analysis.
- Visualization of runtime features is surprisingly powerful – so much so that laypeople can label malware.
- Using more advanced dimensionality reduction, we can combine IAT hashes and t-SNE over mnemonic counts to achieve an order-of-magnitude reduction in analyst workload.

Questions?

Brian Lindauer, CERT Division
lindauer@cert.org