On Removing Experimental Bias in ML-based Malware Detection

Fabio Pierazzi

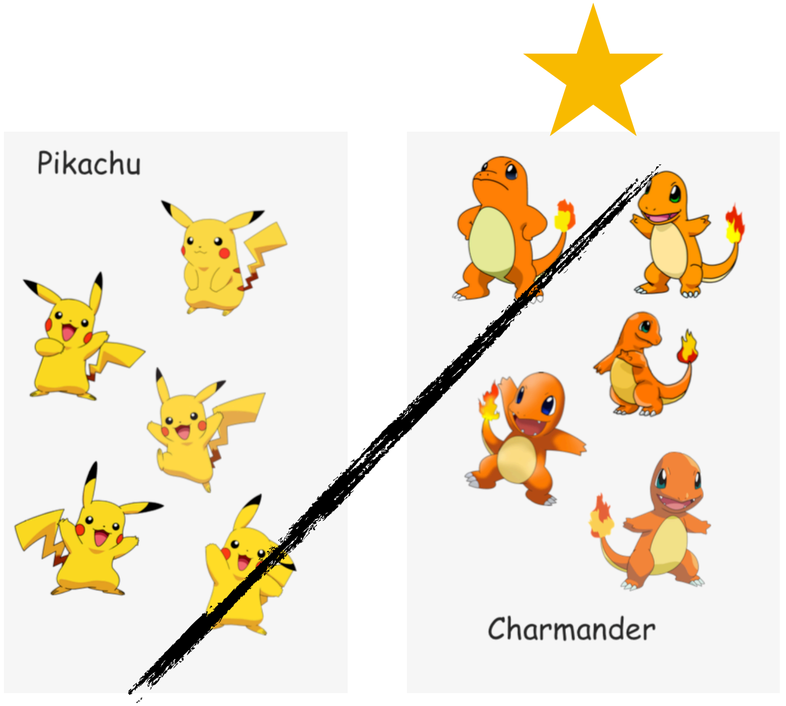
Assistant Professor at King's College London
<fabio.pierazzi@kcl.ac.uk> — @fbpierazzi
https://fabio.pierazzi.com

Mar 23, 2021
Software Engineering Forschungsmethodentraining
Humboldt-Universität zu Berlin, DE

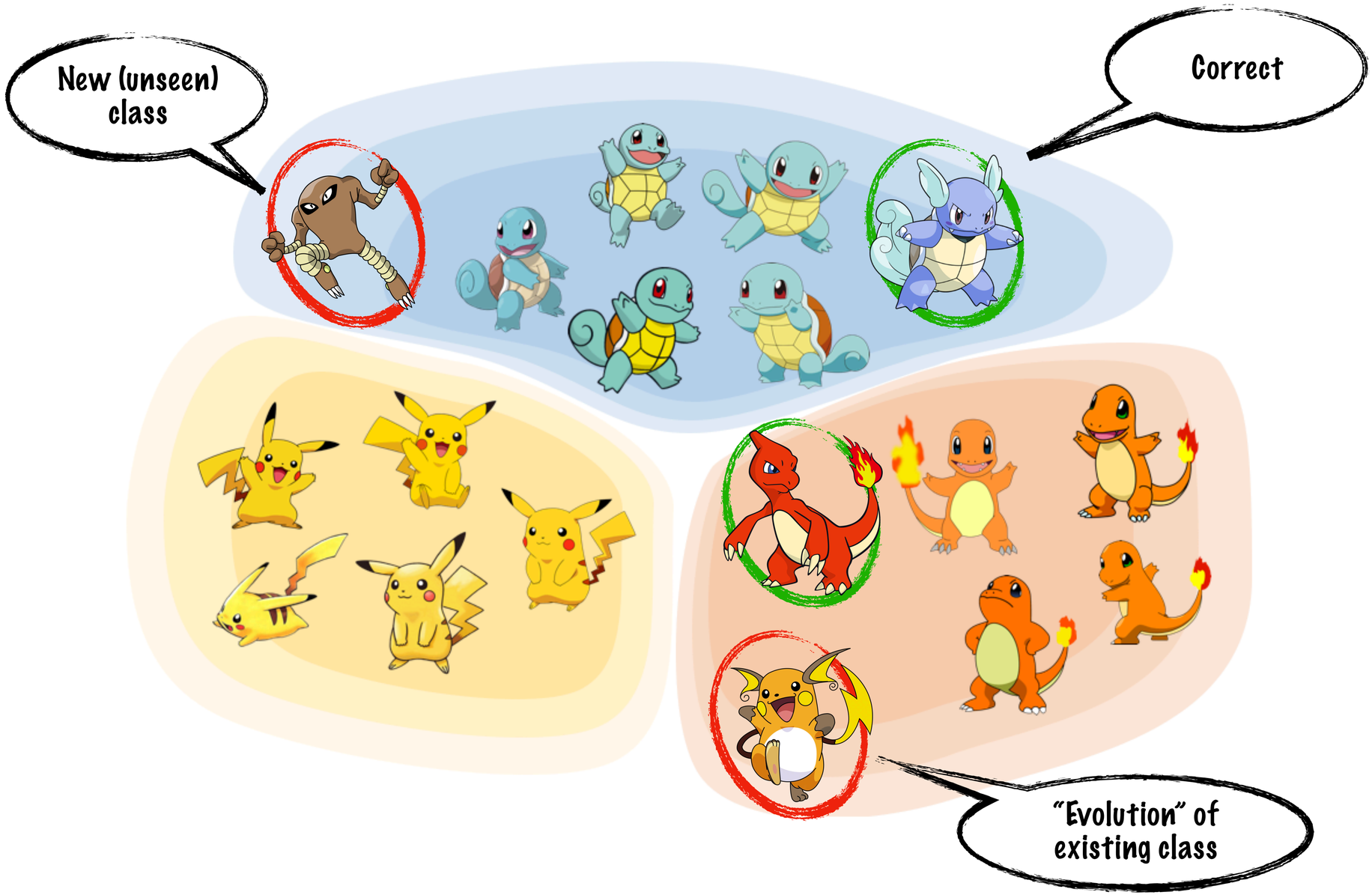
Machine Learning Classification

Usually, a 3-phase process:

1. **Feature Engineering**: to represent objects as numerical vectors
2. **Training**: build a model *M*, given labelled objects
3. **Testing**: given *M,* predict the labels of unknown objects



Concept Drift: Example



ML for Malware Detection

Approach X (Novel)

- Feature Engineering
- ML Algorithm

VirusShare Kharon
VIRUSTOTAL
DREBIN MalGenome
Google Play

10k gw
50k mw

2,3

Dataset Collection

Baseline Approaches

Testing
Training

1

k-fold CV

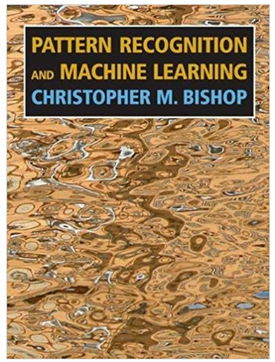
Compare Results

Three major unrealistic assumptions

Approach A

Approach B

ML Best Practices

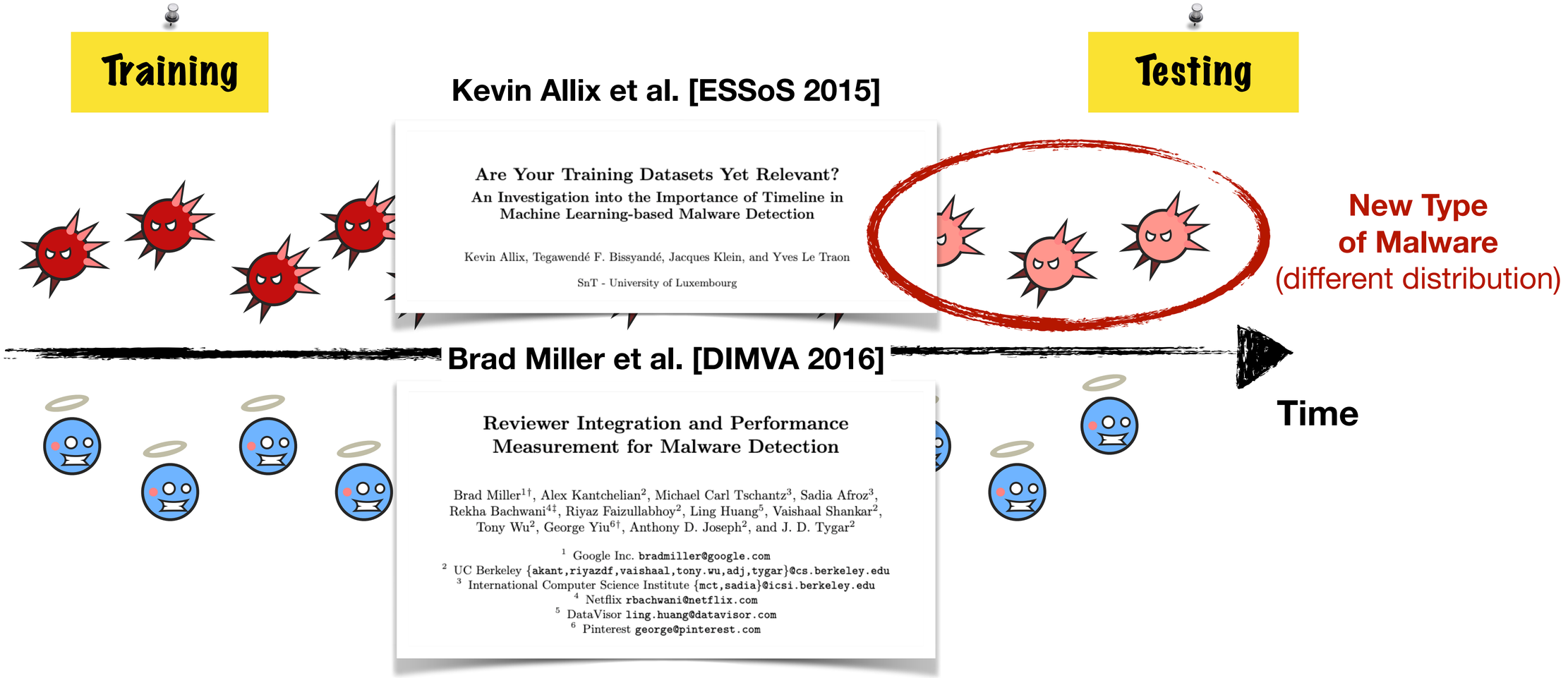


Approach	k-fold F1
X (New)	**0.99**
A	0.97
B	0.86

Sources of Experimental Bias (1/3)

Temporal Inconsistency in Train/Test Sets

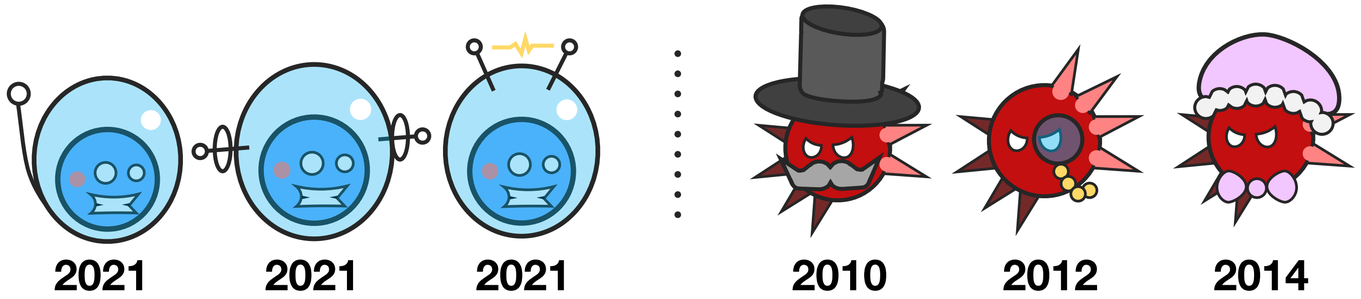
Violations use future knowledge in training



Sources of Experimental Bias (2/3)

Temporal {good|mal}ware inconsistency

Violations may learn artifacts



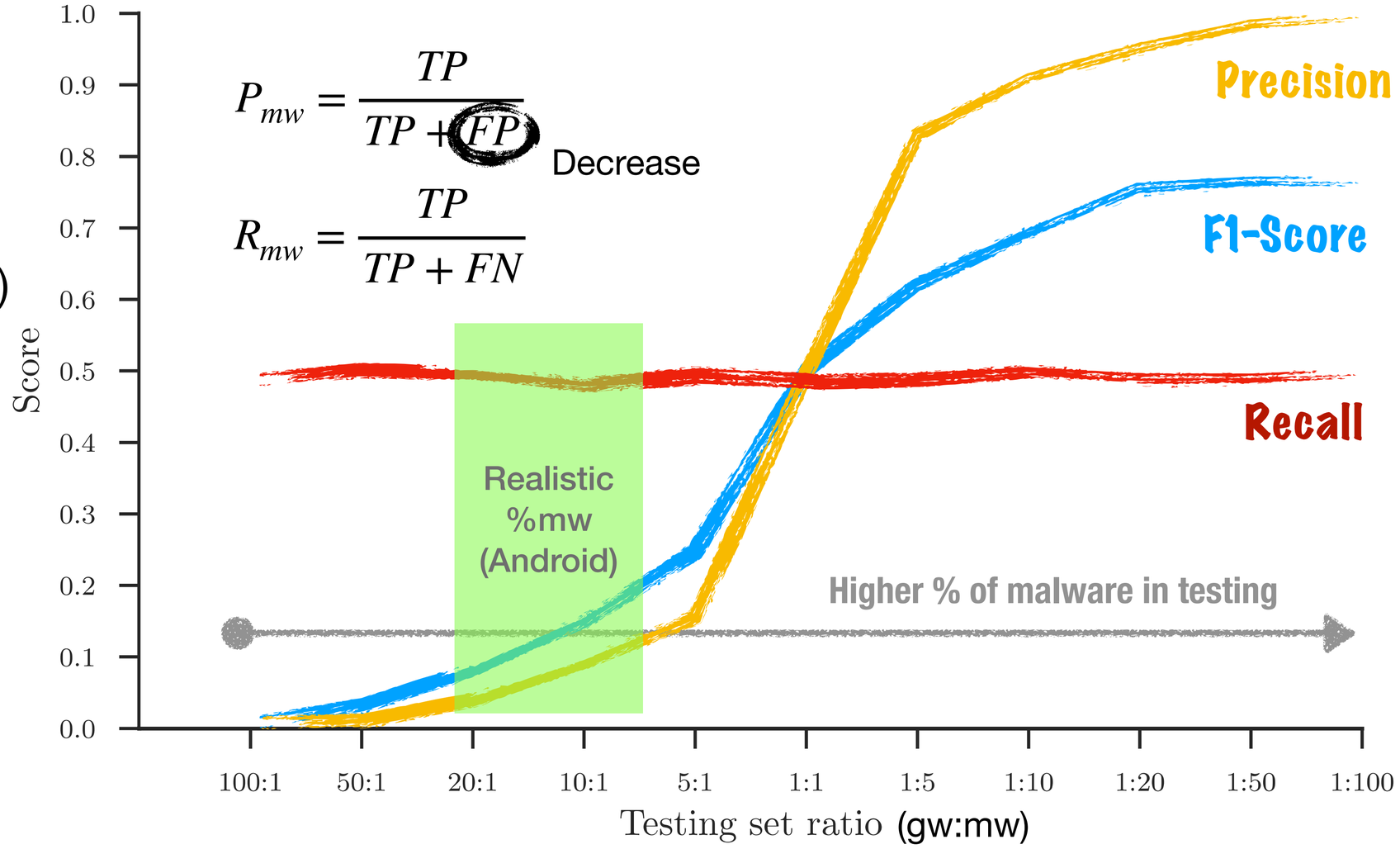

```
2020: new_method()
```

Sources of Experimental Bias (3/3)

Unrealistic Test Class Ratio

Violations produce unrealistic results

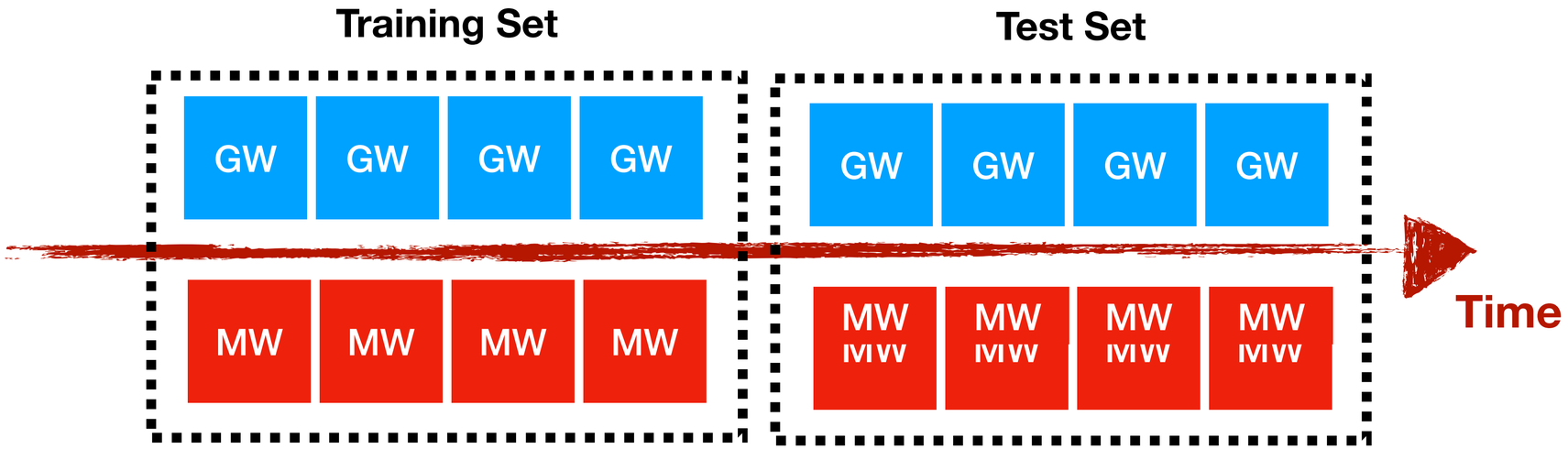
- **Training set**: Fixed
- **Testing set**: Varying % of mw (by downsampling gw)



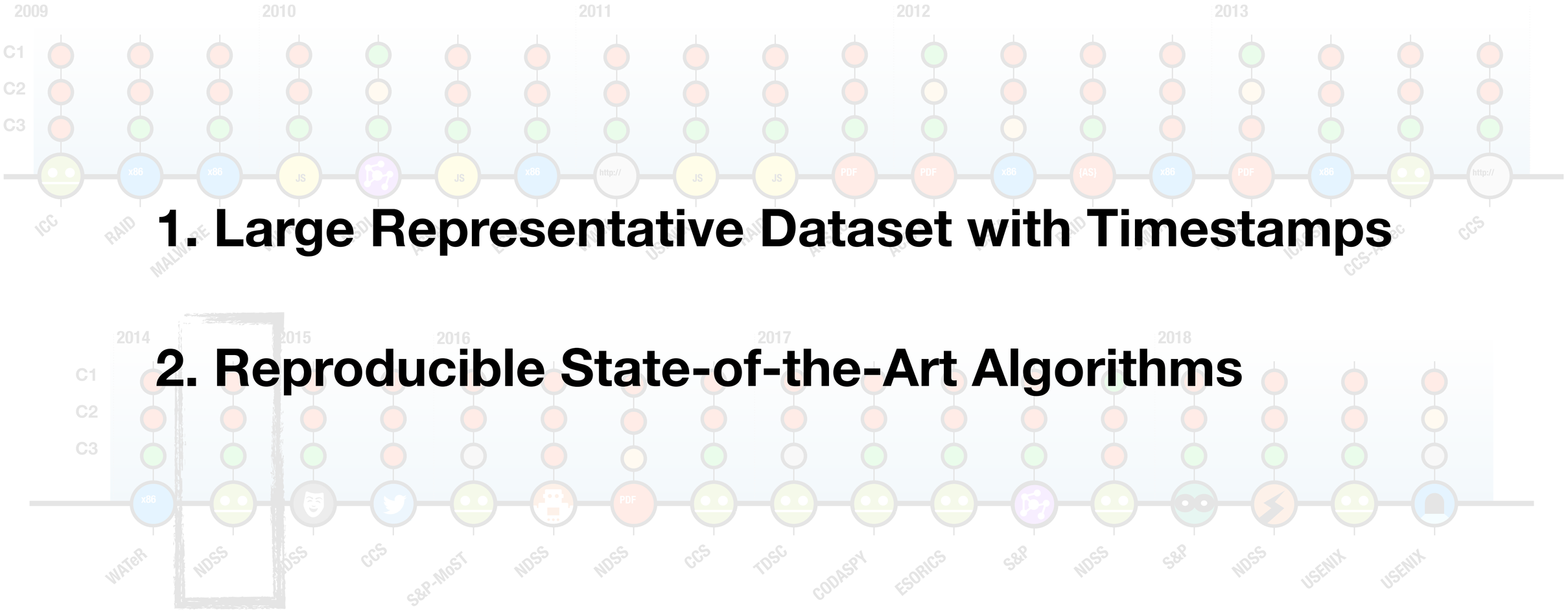
TESSERACT Framework

Experimental Constraints

C1	Temporal training consistency	→ time(training) < time(testing)
C2	{good|mal}ware temporal consistency	→ time(gw) = time(mw)
C3	Realistic testing classes ratio	→ realistic %mw in test



Endemic Problem



1. Large Representative Dataset with Timestamps

2. Reproducible State-of-the-Art Algorithms

Details: https://s2lab.kcl.ac.uk/projects/tesseract/poster-references.pdf

Dataset

- **129,729** Android applications from **AndroZoo** [1]
- **10%** malware
- Covering **3 years** (2014 to 2016)

[1] K. Allix, T. F. Bissyandé, J. Klein, and Y. Le Traon.
AndroZoo: Collecting Millions of Android Apps for the Research Community
In *Mining Software Repositories (MSR)*, 2016

Benchmark Algorithms

Representative

Reproducible

Algorithm 1: DREBIN [NDSS14]

- Statically extracted features are bit vectors, present (1) or not (0)
- Linear Support Vector Machine (**SVM**) classifier

Algorithm 2: MaMaDroid [NDSS17]

- Markov chains from static call graphs through flow analysis
- Features are transition matrices of the Markov chains
- Random Forest (**RF**) classifier

Algorithm 3: Deep Learning [ESORICS17]

- DREBIN features and dataset
- **Deep** Feed-Forward **Neural Network**

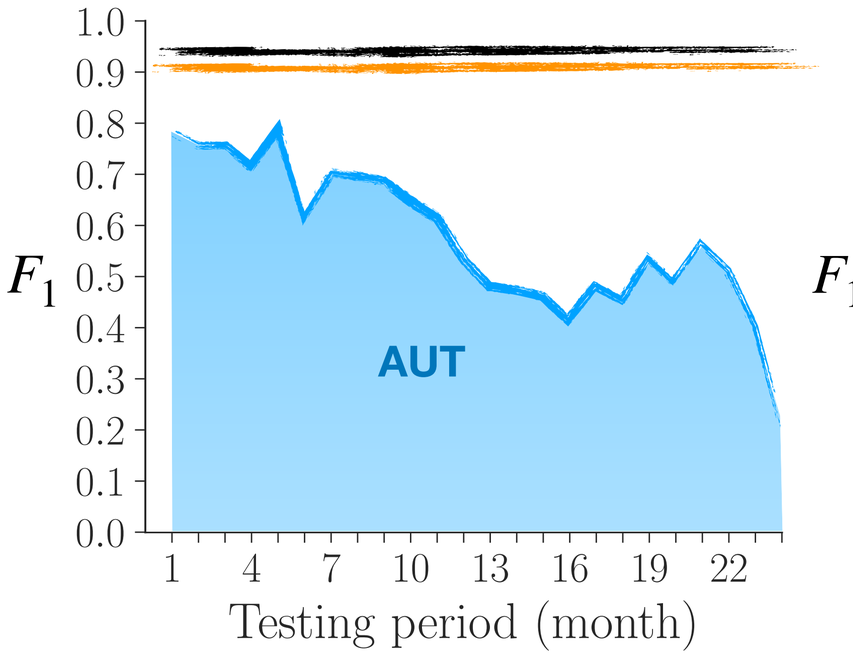
[NDSS14] Arp et al., *DREBIN: Effective and Explainable Detection of Android Malware in Your Pocket*.
[NDSS17] Mariconti et al., *MaMaDroid: Detecting Android Malware by Building Markov Chains of Behavioural Models*.
[ESORICS17] Grosse et al., *Adversarial Examples for Malware Detection*.

TESSERACT Evaluations

Experimental Constraints

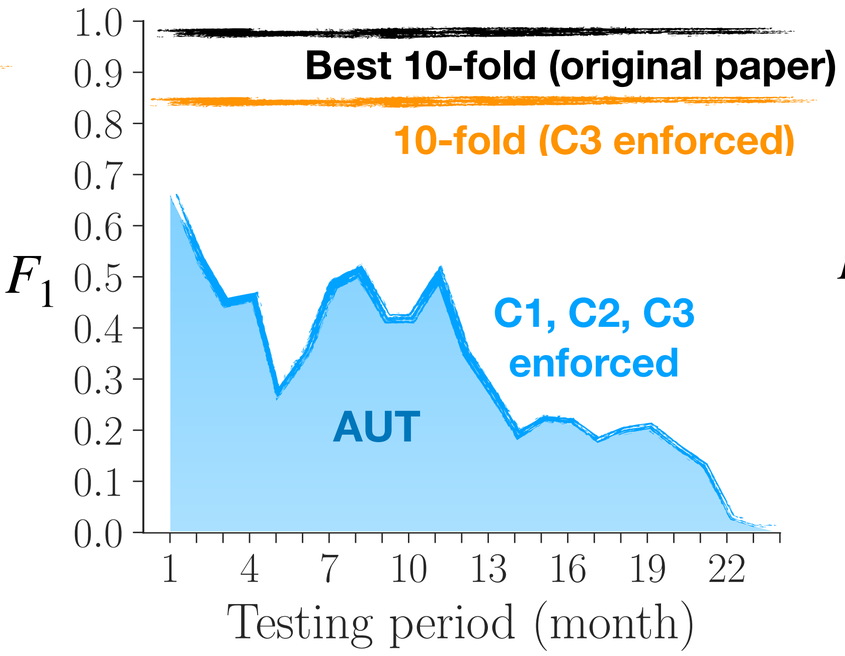
- **C1** Temporal training consistency
- **C2** {good|mal}ware temporal consistency
- **C3** Realistic testing classes ratio

NDSS14



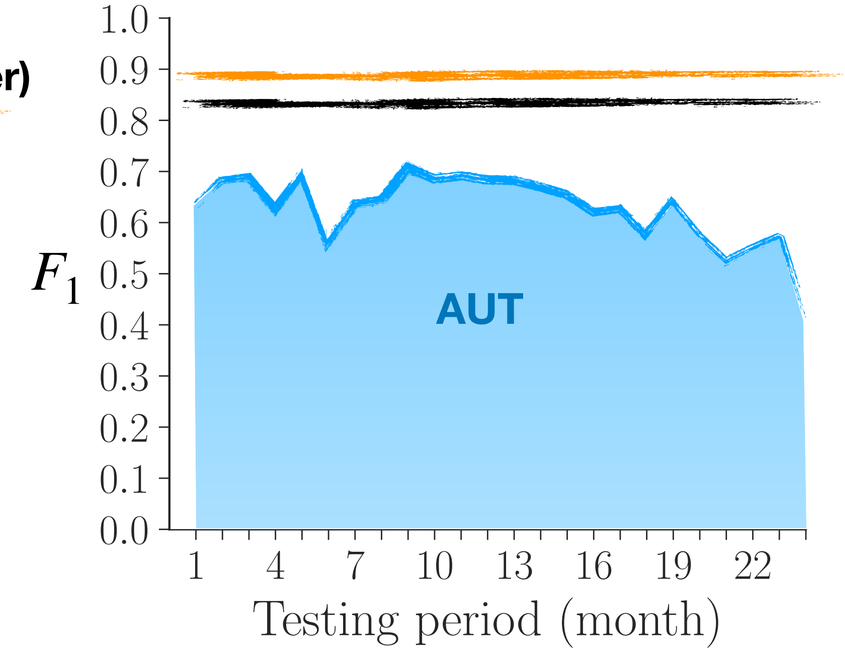
$AUT(F_1, 24m) = 0.58$

NDSS17



$AUT(F_1, 24m) = 0.32$

ESORICS17

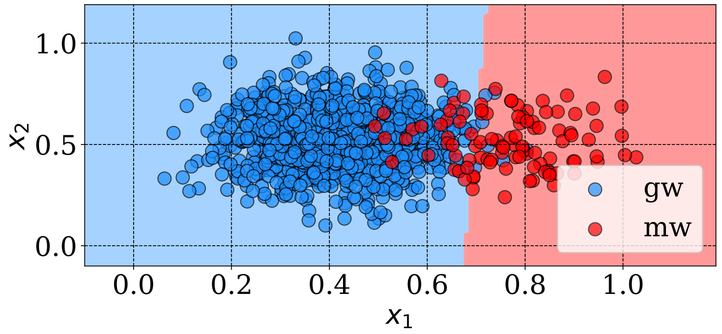


$AUT(F_1, 24m) = 0.64$

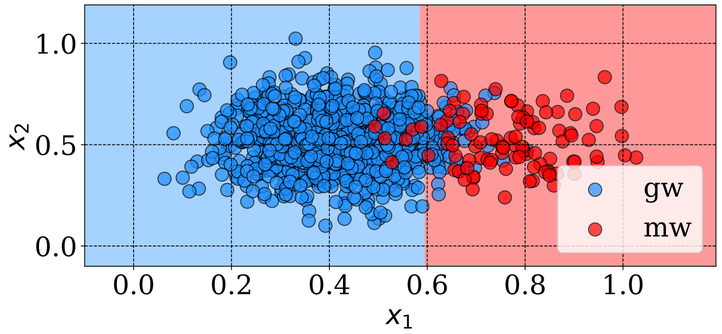
Training Class Distribution

Example:

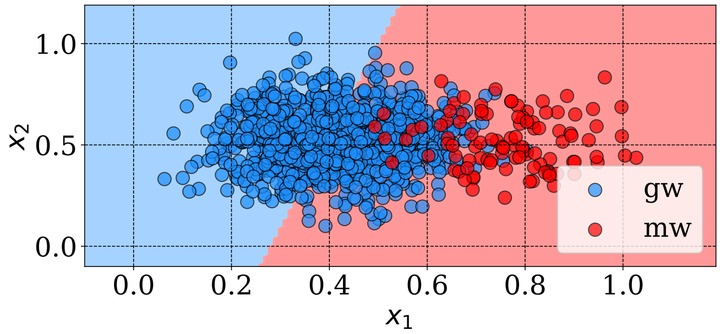
- 2 features
- Test points are fixed
- Training class distribution changes



Training with 10% malware



Training with 50% malware



Training with 90% malware

… more in the paper.

TESSERACT: Actionable Points

Realistic Evaluations

- Unveils performance in realistic deployment
- Removes space-time experimental bias
- **Practitioners:** Choose Best Solution
- **Researchers**: Evaluate New Solutions

Tunable Parameters (e.g., time window, granularity)

Performance-Cost Trade Offs

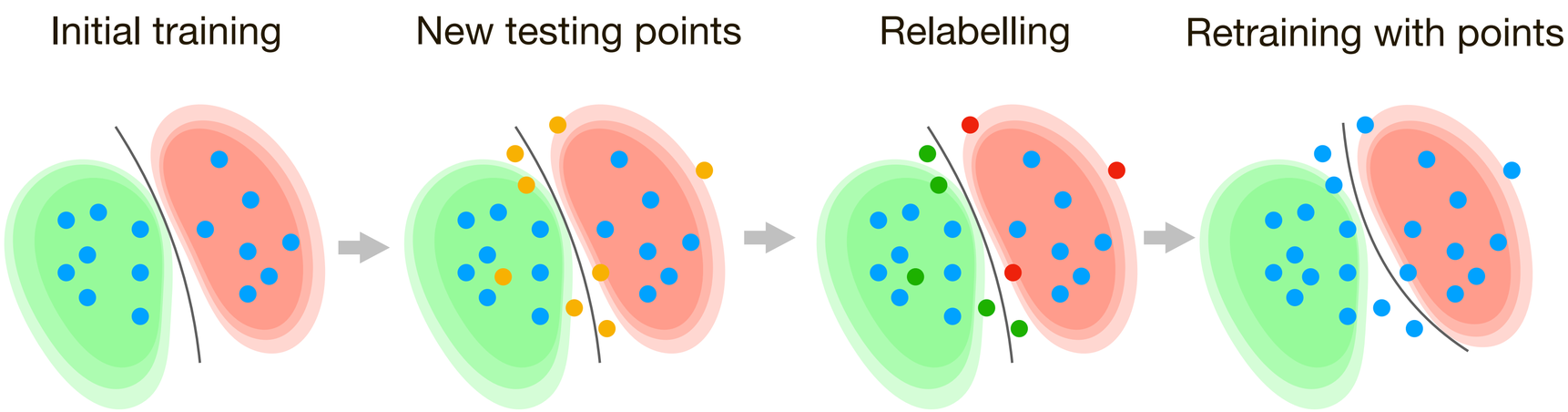
- **Detection Performance** (e.g., AUT F1)
- **Labeling Cost** for retraining (e.g., manpower)
- **Quarantine Cost** for rejection (e.g., low-confidence decisions)

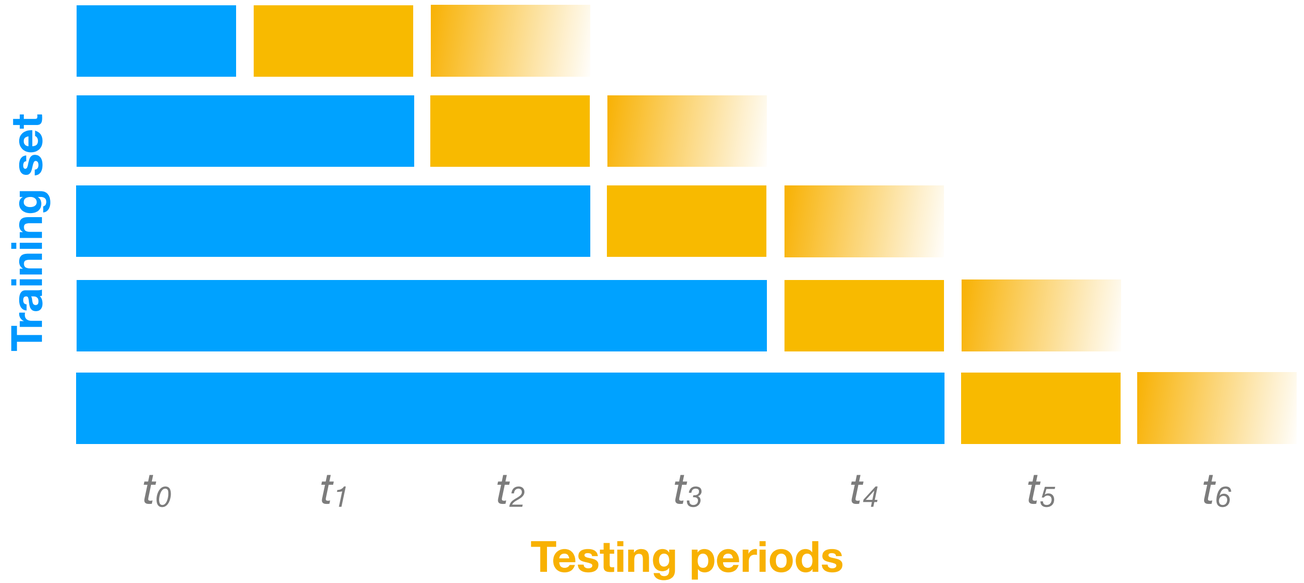
More in Paper

Incremental Retraining Active Learning Rejection*

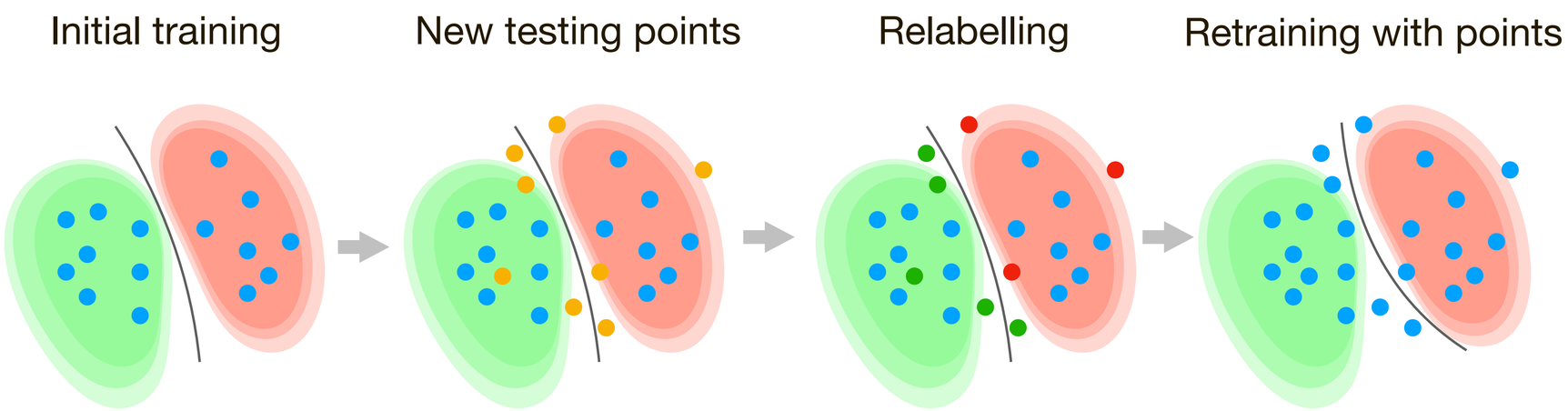
* R. Jordaney, K. Sharad, S. Dash, Z. Wang, D. Papini, I. Nouretdinov, and L. Cavallaro. ***Transcend: Detecting Concept Drift in Malware Classification Models.*** In USENIX Security Symposium, 2017

Incremental Retraining





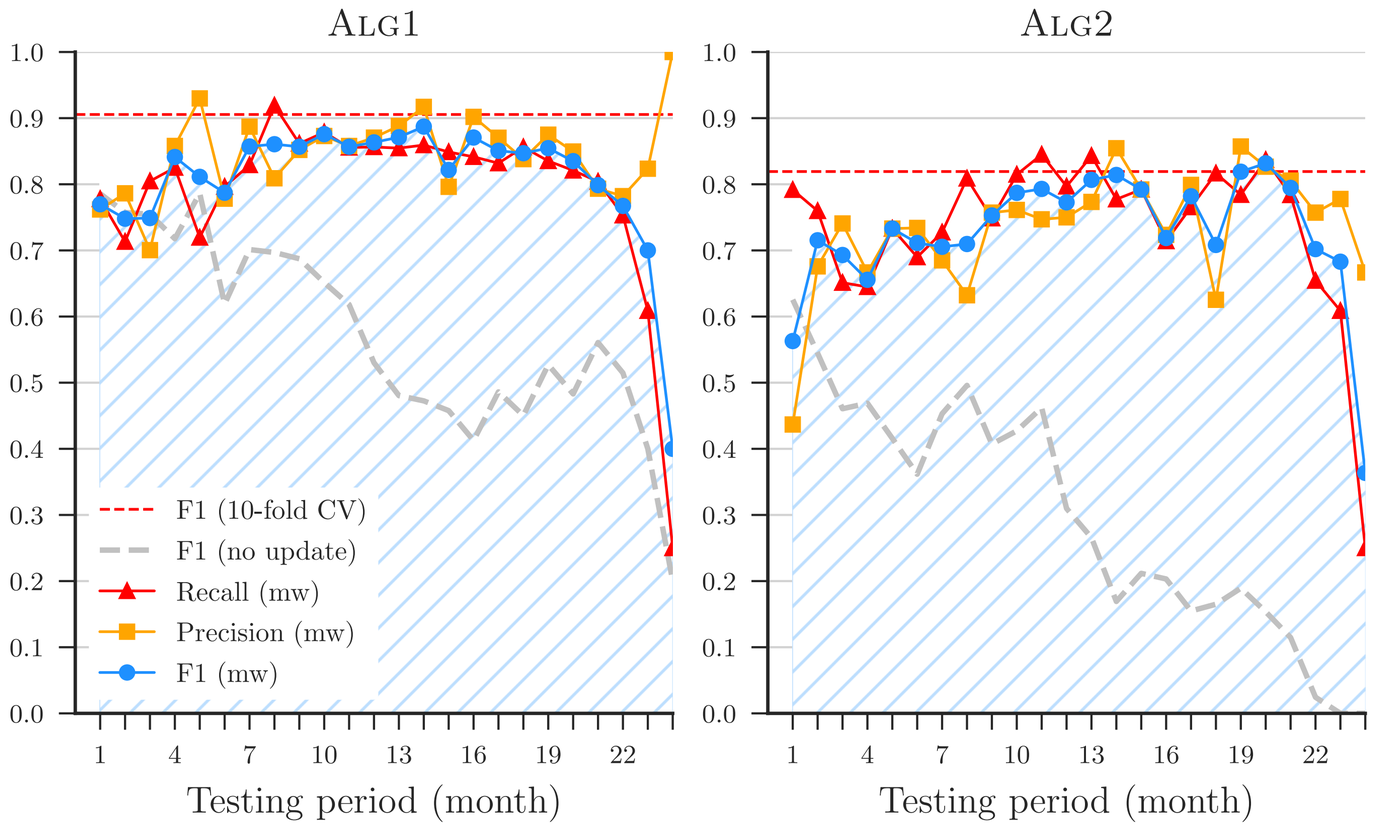
Incremental Retraining



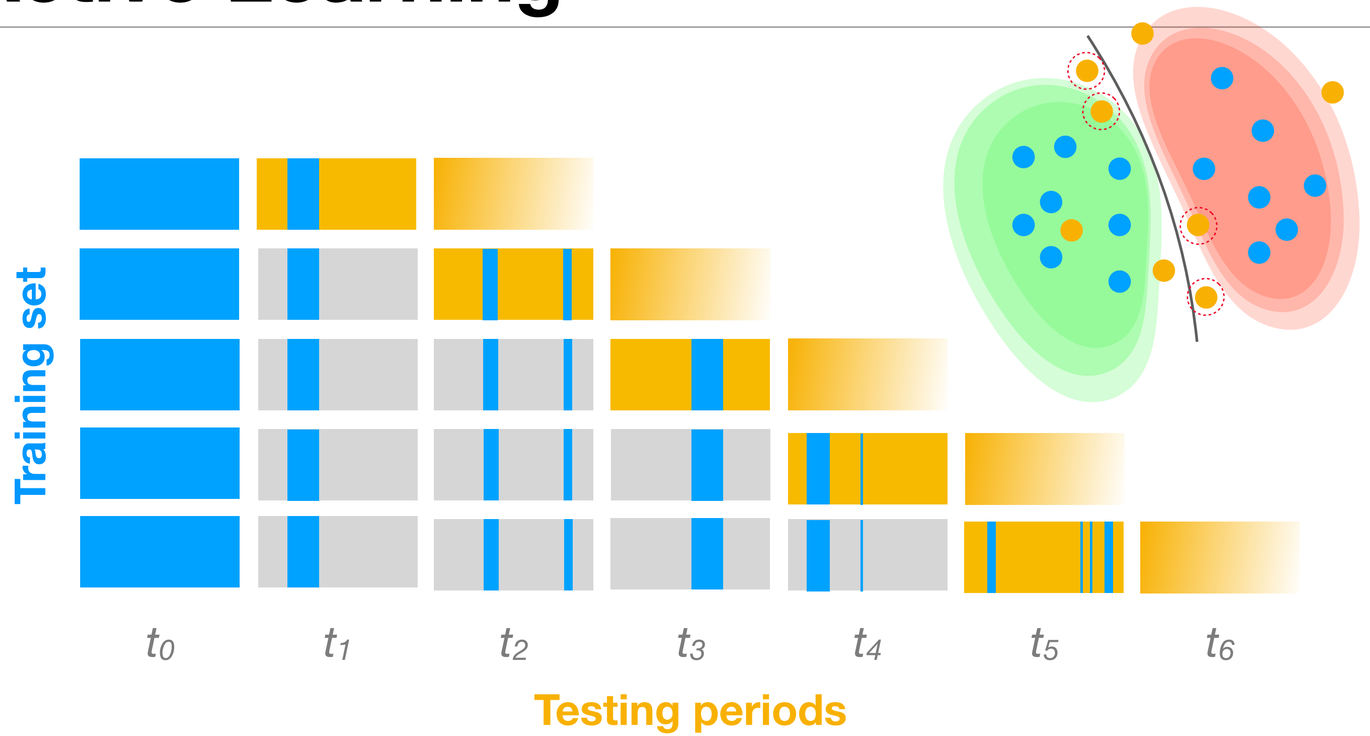
Relabelling Cost

- Getting the true label of an observation is not free!
- Classifying malware manually is specialized, time-consuming work

Incremental Retraining



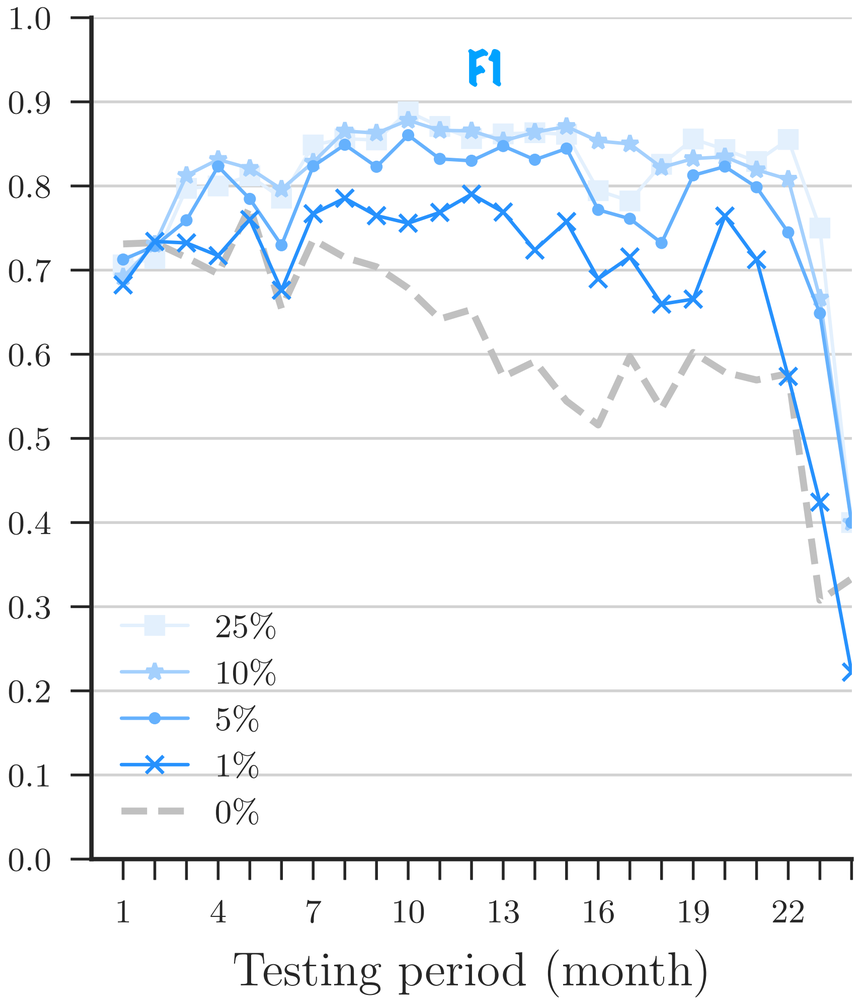
Active Learning



- Only the *n* most relevant objects are used for retraining

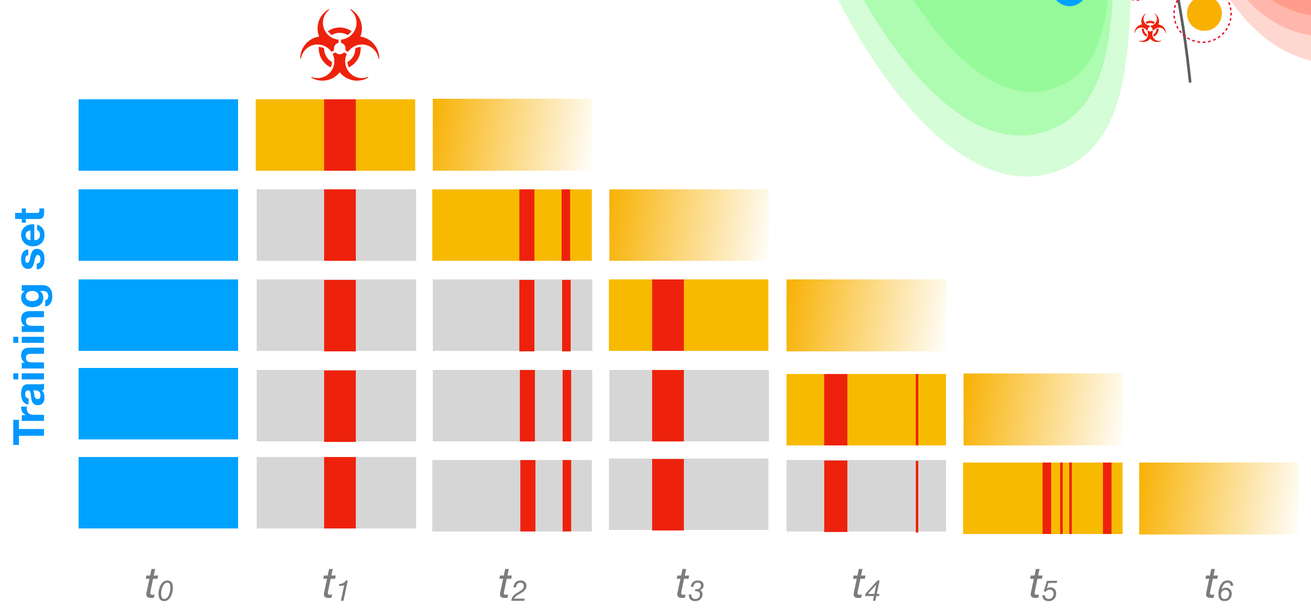
Performance-Cost Trade-Off

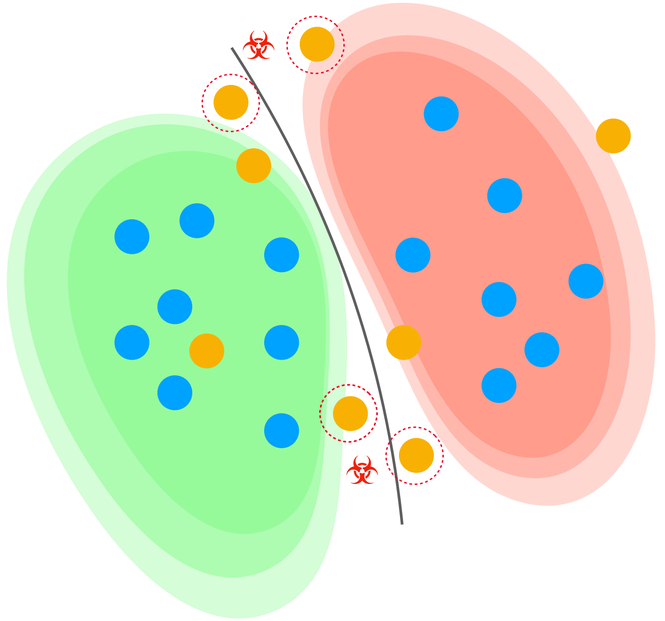
- **Detection Performance** (e.g., AUT F1)
- **Labeling Cost** for retraining (e.g., manpower)
- **Quarantine Cost** for rejection (e.g., low-confidence decisions)



Classification with Rejection

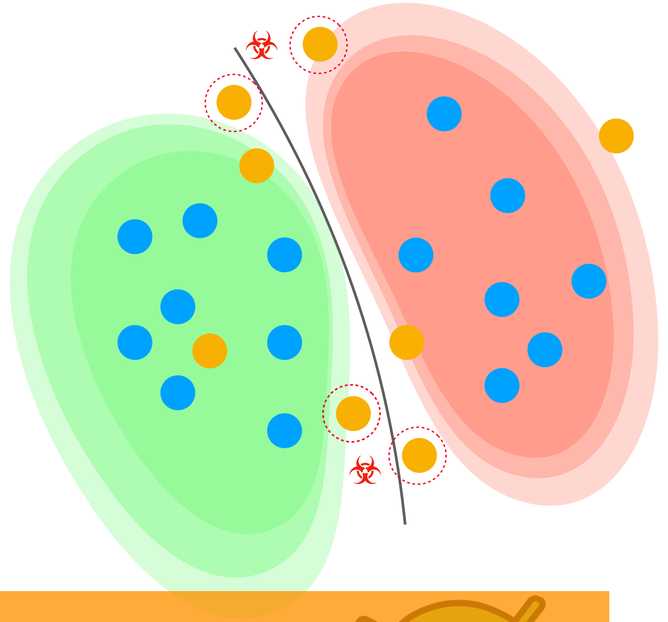
- Low confidence predictions are **rejected**
- Instead of being classified, rejected test objects are sent to **quarantine**
- A rising rejection rate is a sign of concept drift





Classification with Rejection

- Low confidence predictions are **rejected**
- Instead of being classified, rejected test objects are sent to **quarantine**
- A rising rejection rate is a sign of concept drift



Quarantine Cost

- Rejection has no relabelling cost
- However quarantined objects must be dealt with at a later stage of the pipeline

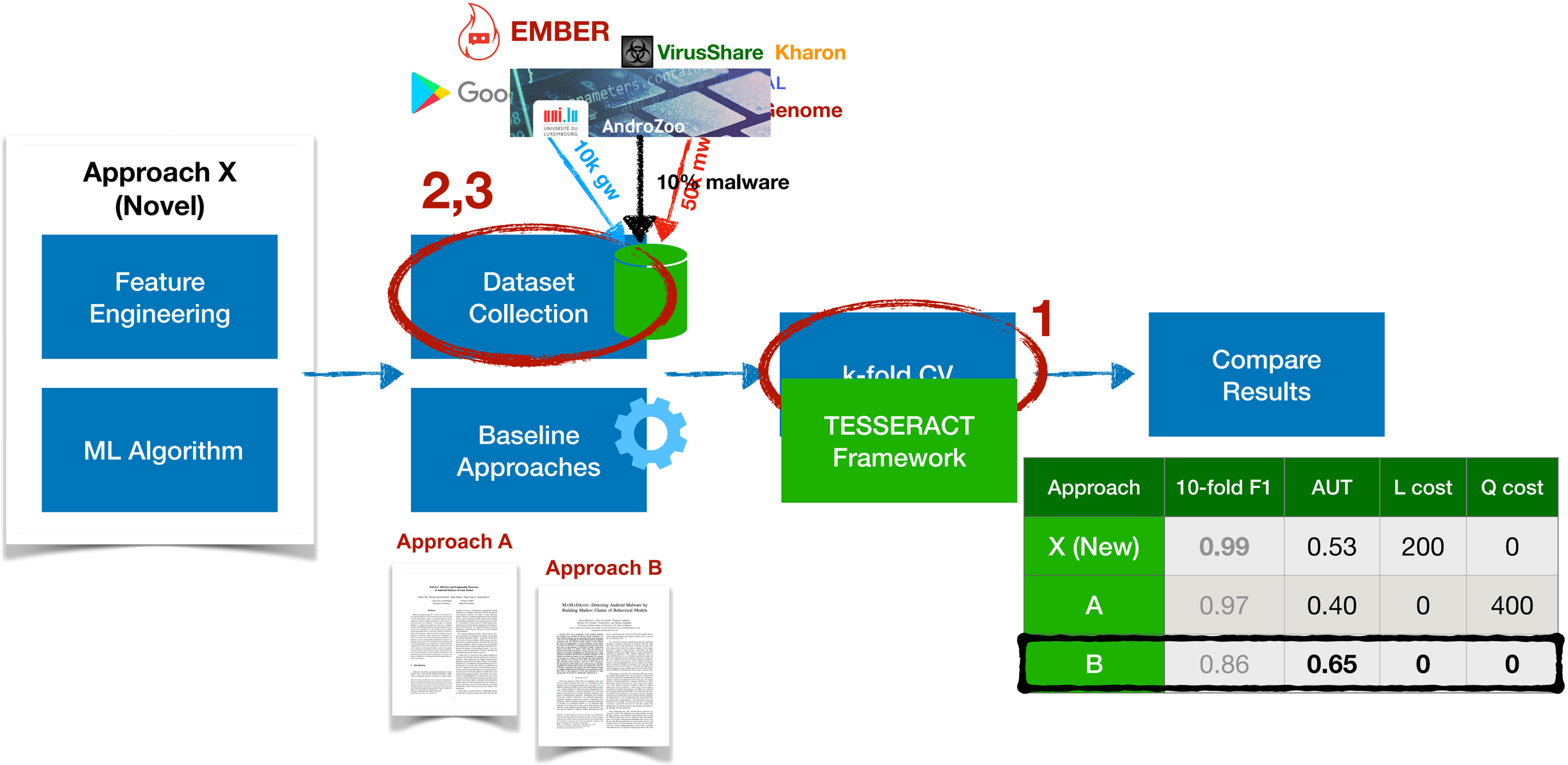
Classification with Rejection

Transcend[4] [USENIX 2017], Transcendent [arXiv 2020]

- Transcend is a framework for handling concept drift in detection
- It uses a form of conformal prediction to identify evolving malware
- It can be used effectively to perform classification with rejection

[4] R. Jordaney, K. Sharad, S. Dash, Z. Wang, D. Papini, I. Nouretdinov, and L. Cavallaro *Transcend: Detecting Concept Drift in Malware Classification Models*. In Proceedings of Usenix Security Symposium, 2017

ML for Malware Detection

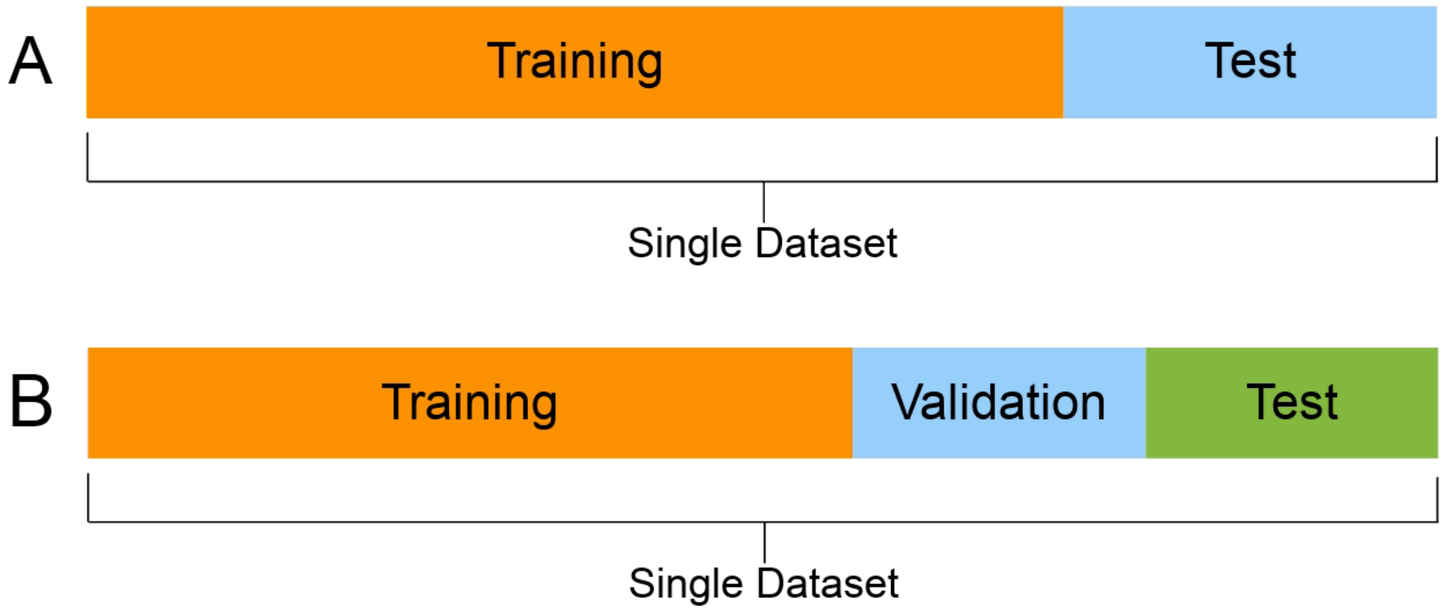


Approach	10-fold F1	AUT	L cost	Q cost
X (New)	0.99	0.53	200	0
A	0.97	0.40	0	400
B	0.86	**0.65**	**0**	**0**

Other Sources of Experimental Bias

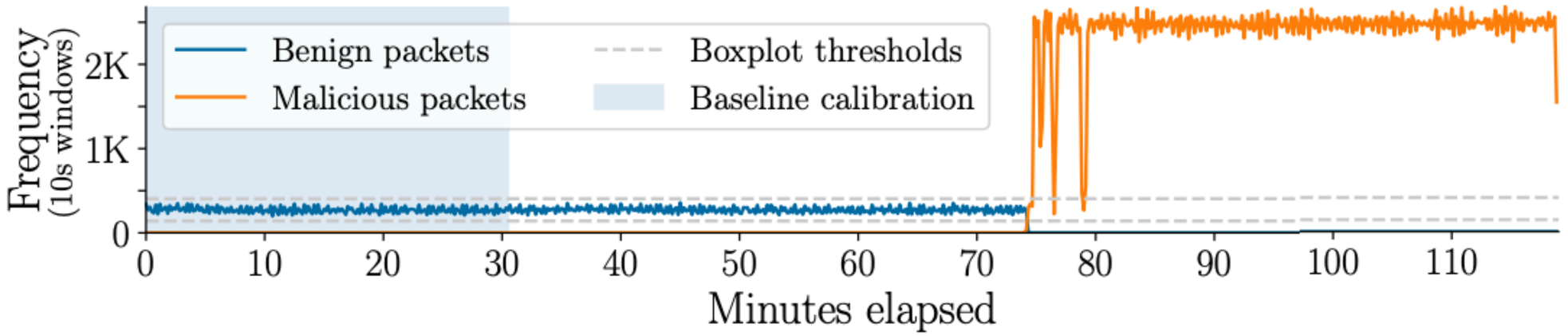
Data Snooping and Biased Parameter Selection

- Always keep a **separate test set** only for very final tests.
- Any parameter tuning should always be on **validation set**
- “Cleaning” the dataset (e.g., removing edge cases) is **cherrypicking**



Inappropriate Baselines

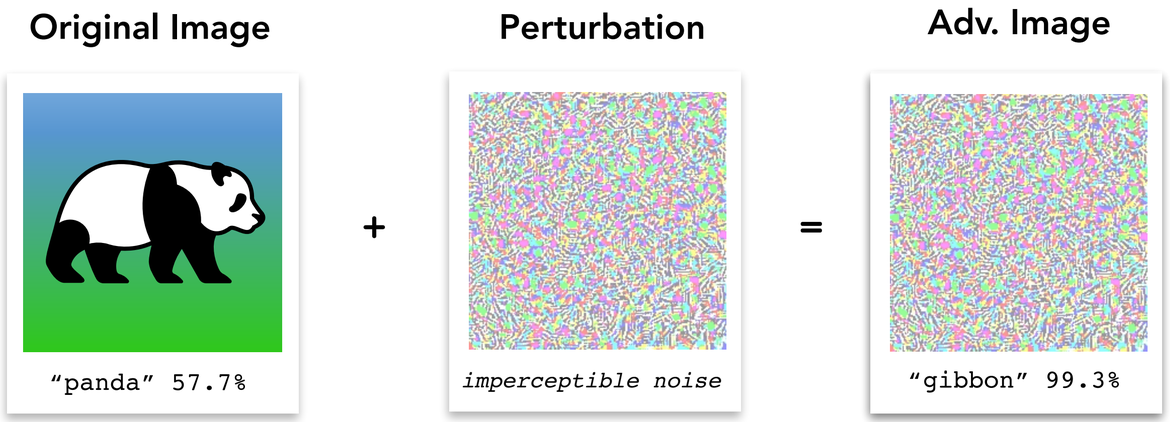
- Always compare your approach against a simple baseline



	AUC	TPR (FPR at 0.001)	TPR (FPR at 0.000)
Kitsune [NDSS'18]	0.968	0.882	0.873
Simple Baseline	0.998	0.996	0.996

Adversarial Behaviors

Feature-Space Attacks



x $\quad$ δ $\quad$ $x + \delta$

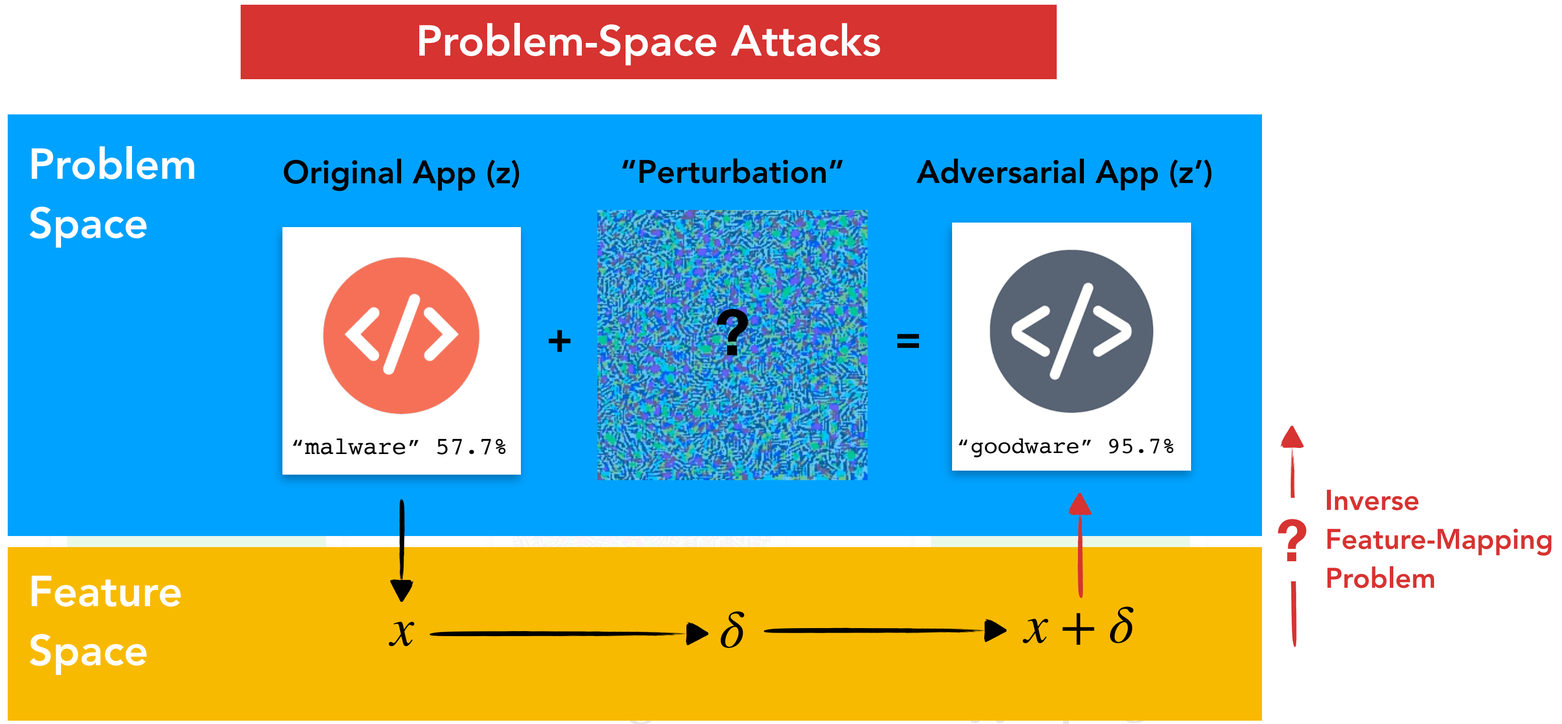
Optimization

$$\text{minimize}_{\delta} \; ||\delta||_p + c \cdot f(x+\delta)$$

Pixel Perturbations

Loss of Target Class

Adversarial Behaviors



- **Problem-Space Adversarial ML Attacks**
  - Novel Formalization
  - Novel end-to-end Adversarial Malware
- **Project website:**
  - https://s2lab.kcl.ac.uk/projects/intriguing/

Problem-space attacks research is just beginning!

Problem-Space Constraints

- Available Transformations
- Preserved Semantics
- Plausibility
- Robustness to Preprocessing

Search Strategy

- Gradient-driven
- Problem-driven
- Hybrid

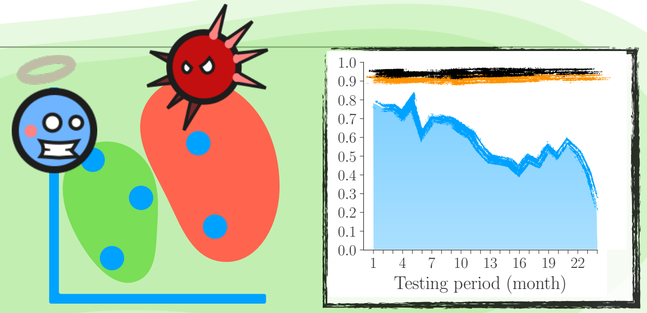
Pierazzi et al, **Intriguing Properties of Adversarial ML Attacks in the Problem Space**, IEEE S&P, 2020

Conclusions

Conclusions

Experimental Constraints

C1	Temporal training consistency
C2	{good|mal}ware temporal consistency
C3	Realistic testing classes ratio

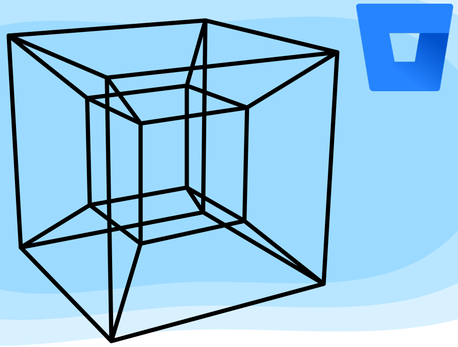


- Sources of experimental bias in the software domain

- TESSERACT Framework: **Sound time-aware evaluations**
- Problem-space adversarial attacks formalization
- TRANSCEND(ENT): Classification with rejection strategies

- **Open-source** code, dataset, features

https://s2lab.kcl.ac.uk/projects/tesseract/
https://s2lab.kcl.ac.uk/projects/intriguing/
https://s2lab.kcl.ac.uk/projects/transcend/



Our open-source libraries

Some institutions who got access:

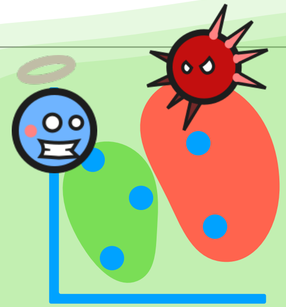
Conclusions

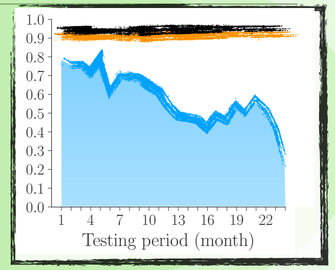
SYSTEMS SECURITY RESEARCH LAB

KING'S College LONDON

Experimental Constraints

C1	Temporal training consistency
C2	{good|mal}ware temporal consistency
C3	Realistic testing classes ratio



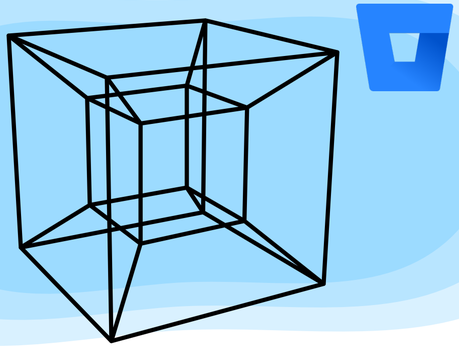


- Sources of experimental bias in the software domain

- TESSERACT Framework: **Sound time-aware evaluations**
- Problem-space adversarial attacks formalization
- TRANSCEND(ENT): Classification with rejection strategies

- **Open-source** code, dataset, features

https://s2lab.kcl.ac.uk/projects/tesseract/
https://s2lab.kcl.ac.uk/projects/intriguing/
https://s2lab.kcl.ac.uk/projects/transcend/



Thanks!

SYSTEMS SECURITY RESEARCH LAB

KING'S College LONDON

ROYAL HOLLOWAY UNIVERSITY OF LONDON

CODE RI Research Institute Cyber Defence
Bundeswehr University Munich

On removing experimental bias in ML-based malware detection

Fabio Pierazzi
Assistant Professor at King's College London
<fabio.pierazzi@kcl.ac.uk> — @fbpierazzi
https://fabio.pierazzi.com

Mar 23, 2021
Software Engineering Forschungsmethodentraining
Humboldt-Universität zu Berlin, DE