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

New (unseen) class

Correct

“Evolution” of existing class
ML for Malware Detection

Dataset Collection

Baseline Approaches

k-fold CV

Compare Results

Three major unrealistic assumptions

Approach X (Novel)

Feature Engineering

ML Algorithm

Approach A

Approach B

ML Best Practices

<table>
<thead>
<tr>
<th>Approach</th>
<th>k-fold F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>X (New)</td>
<td>0.99</td>
</tr>
<tr>
<td>A</td>
<td>0.97</td>
</tr>
<tr>
<td>B</td>
<td>0.86</td>
</tr>
</tbody>
</table>

VirusShare, Kharon, VIRUSTOTAL, DREBIN, MalGenome

F. Pierazzi - On removing experimental bias in ML-based malware detection
Sources of Experimental Bias (1/3)

Temporal Inconsistency in Train/Test Sets

Kevin Allix et al. [ESSoS 2015]

Are Your Training Datasets Yet Relevant? 
An Investigation into the Importance of Timeline 
In Machine Learning-based Malware Detection

Kevin Allix, Tegawendé B. Biyani, Jacques Klein, and Yves Le Traon
SST - University of Luxembourg

Brad Miller et al. [DIMVA 2016]

Reviewer Integration and Performance 
Measurement for Malware Detection

Brad Miller, Alex Kantchelian, Michael Carl Tschetschel, Sadia Afroz, Bekha Bachchan, Rizae Fainalbhooy, Ling Huang, Vaishal Shankar, Tony Wu, George Yin, Anthony D. Joseph, and J. D. Tygar

1 Google Inc: bradmiller@google.com
2 UC Berkeley: avaint, rizaeafroz, vaishal, tony.wu, ad.tygar}@cs.berkeley.edu
3 International Computer Science Institute: victoia@icsi.berkeley.edu
4 Netflix: victoia@netflix.com
5 DataViper: ling.huang@davatographer.com
6 Pinterest: george@pinterest.com

Violations: use future knowledge in training

New Type of Malware
(different distribution)
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

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

Violations produce unrealistic results

\[
P_{mw} = \frac{TP}{TP + FP} \quad \text{Decrease}
\]

\[
R_{mw} = \frac{TP}{TP + FN}
\]

Realistic %mw (Android)

Higher % of malware in testing
**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

**Training Set**

- GW
- GW
- GW
- GW

**Test Set**

- MW
- MW
- MW
- MW

**Time**
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)

Benchmark Algorithms

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


F. Pierazzi - On removing experimental bias in ML-based malware detection
**TESSERACT Evaluations**

**Experimental Constraints**
- **C1** Temporal training consistency
- **C2** Good/malware temporal consistency
- **C3** Realistic testing classes ratio

### NDSS14

- **$AUT(F_1,24m) = 0.58$**

### NDSS17

- **$AUT(F_1,24m) = 0.32$**

### ESORICS17

- **$AUT(F_1,24m) = 0.64$**

---

F. Pierazzi - On removing experimental bias in ML-based malware detection
Training Class Distribution

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

... 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

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)

Incremental Retraining  Active Learning  Rejection*

Incremental Retraining

- Initial training
- New testing points
- Relabelling
- Retraining with points
Getting the true label of an observation is not free!

Classifying malware manually is specialized, time-consuming work
Incremental Retraining

**Alg1**

- F1 (10-fold CV)
- F1 (no update)
- Recall (mw)
- Precision (mw)
- F1 (mw)

**Alg2**

- F1 (10-fold CV)
- F1 (no update)
- Recall (mw)
- Precision (mw)
- F1 (mw)
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)
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
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\cite{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

ML for Malware Detection

Approach X (Novel)
- Feature Engineering
- ML Algorithm

Overview:
1. Dataset Collection
2. Baseline Approaches
3. TESSERACT Framework
4. Comparison

Approach A
- 10-fold F1
- AUT
- L cost
- Q cost

Approach B
- 10-fold F1
- AUT
- L cost
- Q cost

Comparison Table:

<table>
<thead>
<tr>
<th>Approach</th>
<th>10-fold F1</th>
<th>AUT</th>
<th>L cost</th>
<th>Q cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>X (New)</td>
<td>0.99</td>
<td>0.53</td>
<td>200</td>
<td>0</td>
</tr>
<tr>
<td>A</td>
<td>0.97</td>
<td>0.40</td>
<td>0</td>
<td>400</td>
</tr>
<tr>
<td>B</td>
<td>0.86</td>
<td>0.65</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
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

<table>
<thead>
<tr>
<th></th>
<th>AUC</th>
<th>TPR (FPR at 0.001)</th>
<th>TPR (FPR at 0.000)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Kitsune [NDSS’18]</td>
<td>0.968</td>
<td>0.882</td>
<td>0.873</td>
</tr>
<tr>
<td>Simple Baseline</td>
<td>0.998</td>
<td>0.996</td>
<td>0.996</td>
</tr>
</tbody>
</table>
Adversarial Behaviors

Feature-Space Attacks

Original Image

Perturbation

Adv. Image

\[ x + \delta \]

Optimization

\[ \text{minimize}_{\delta} \left( \frac{1}{p} \| \delta \|_p + c \cdot f(x + \delta) \right) \]

Pixel Perturbations

Loss of Target Class

"panda" 57.7% + imperceptible noise = "gibbon" 99.3%
Adversarial Behaviors

Problem-Space Attacks

Problem Space

Original App (z)  \( \Rightarrow \)  "malware" 57.7%

Perturbation

x + \( \delta \)

Adversarial App (z')  \( \Rightarrow \)  "goodware" 95.7%

Feature Space

Optimization

\[ \min_{\delta} \left\| x \right\|_p + c \cdot f(x + \delta) \]

Constraints

- Is it realistic/plausible?
- Does it crash?
- Can it be detected by signatures?
- Does it preserve malicious functionality?
- … are there “general” constraints?
• 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
Conclusions
Conclusions

- 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

- 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!
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