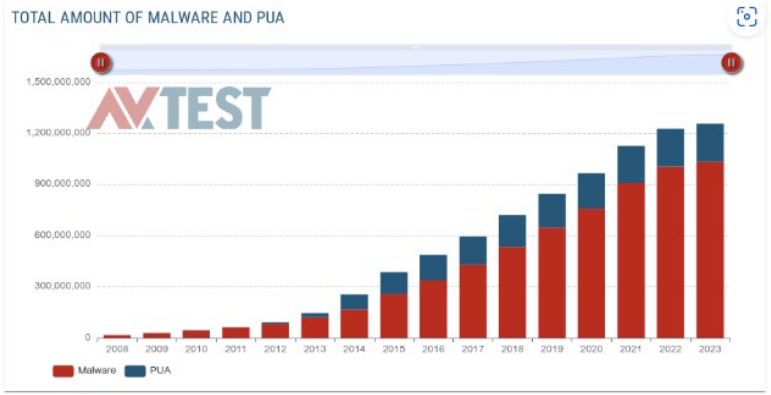
Density Boosts Everything : A One-stop Strategy for Improving Performance, Robustness, and Sustainability of Malware Detectors

Jianwen Tian[1] , Wei Kong[2], **Debin Gao[3]** , Tong Wang[1], Taotao Gu[1],
Kefan Qiu[4], Zhi Wang[5], Xiaohui Kuang[1]

Institute of Systems Engineering, Academy of Military Sciences, China [1]
Zhejiang Sci-Tech University[2]
Singapore Management University [3]
Beijing Institute of Technology [4]
Nankai University [5]

ML-based Malware detectors

- ML-based malware detectors become popular
- ML-based detectors are extremely vulnerable to various problems (e.g., adversarial attacks, concept drift).
- Defenses usually target specific problems, and improvement in one aspect leads to negative impact on others.
- There is no universal solution.



Source: av-atlas.org

CROWDSTRIKE | BLOG

Featured Recent Videos

Why Machine Learning Is a Critical Defense Against Malware

July 17, 2019 Jackie Castelli Endpoint & Cloud Security

cynet

XDR PLATFORM SERVICE WHY CYNET PARTNERS

Related Content

Malware Protection

4 Malware Detection Techniques and Their Use in EPP and EDR

Avira | OEM

Solutions Technology Partnerships Resources Blog Portal Contact

Machine Learning

Machine learning is how Avira scales the detection and classification of malware. It is one of the powerful techniques we use to protect our technology partners and their customers from threats.

Download Whitepaper

Our contributions

- Elevating the sparsity problem and associating sparsity with the key issues in malware detection tasks.

- Proposing subspace compression and density boosting robust training to solve sparsity problem.

- Practical solution for malware detection by integrating our strategies with other defenses.

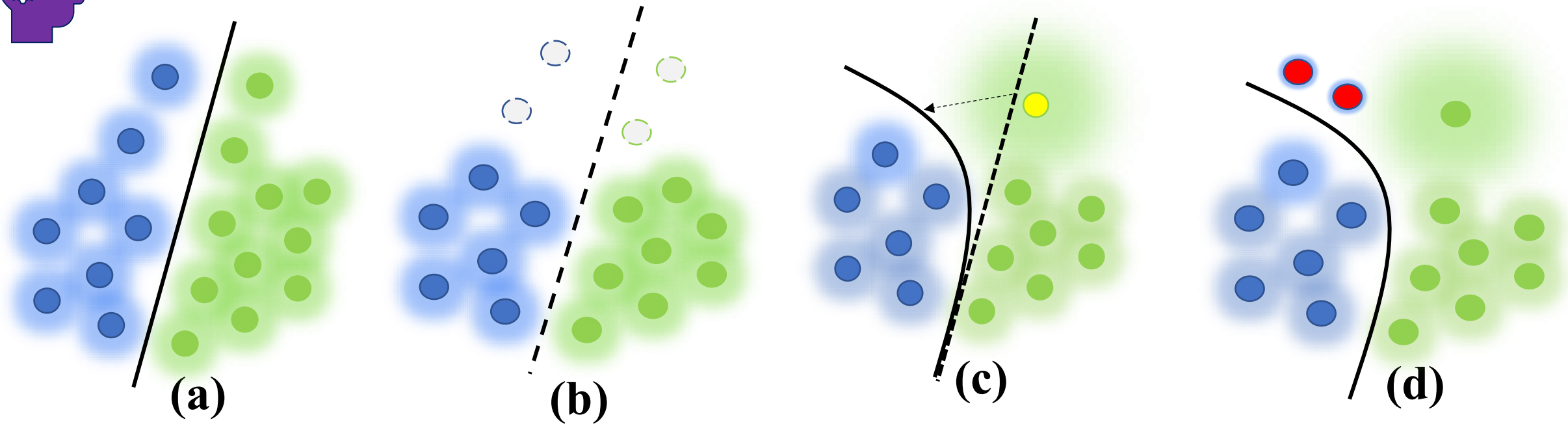
Key issues for malware detection

- **Performance:** malwares should be detected as accurately as possible.

- **Robustness:** the detector should have resistance to adversarial attacks (such as backdoor and evasion).

- **Sustainability:** the detector should maintain stable performance in concept drift scenarios as much as possible.

Motivation – association with all issues



(a) Actual distribution of the two-class data.

(b) Training data sampled from the overall distribution.

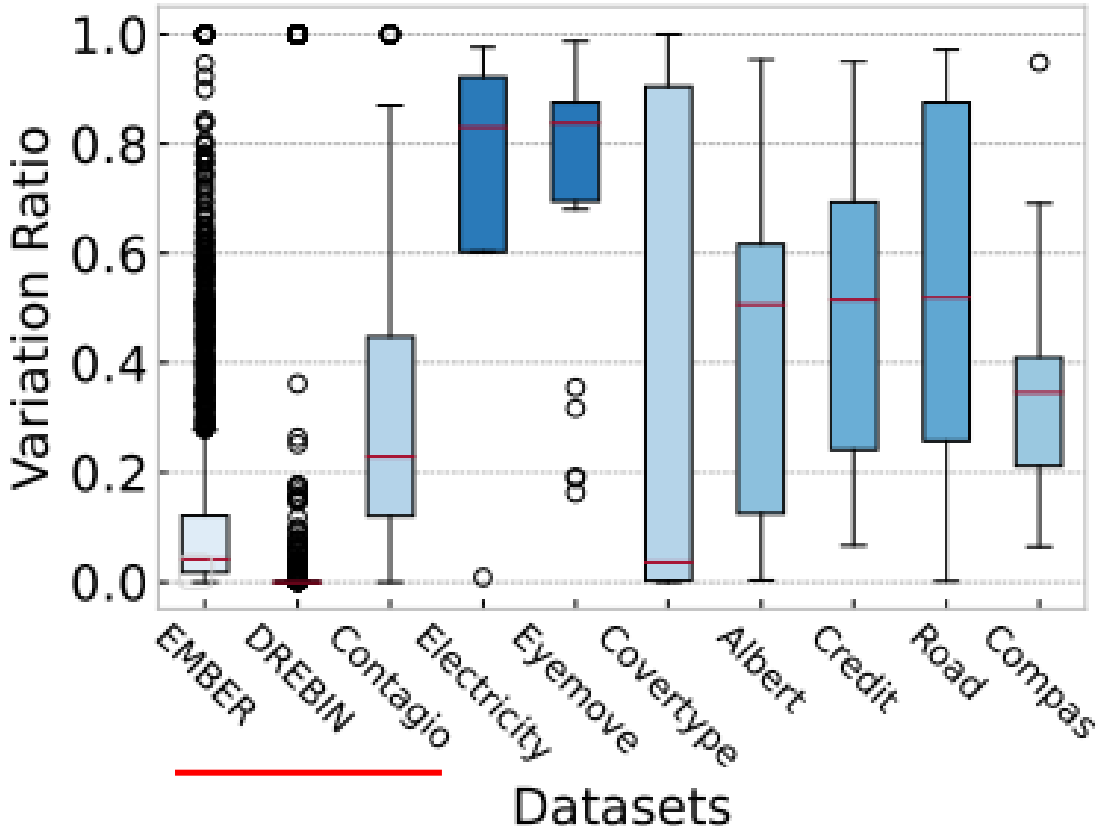
(c) A sample warp the decision boundary.

(d) testing samples are wrongly classified.

Definition: **Sparsity means some feature values or sub-regions occur rarely in a dataset**.

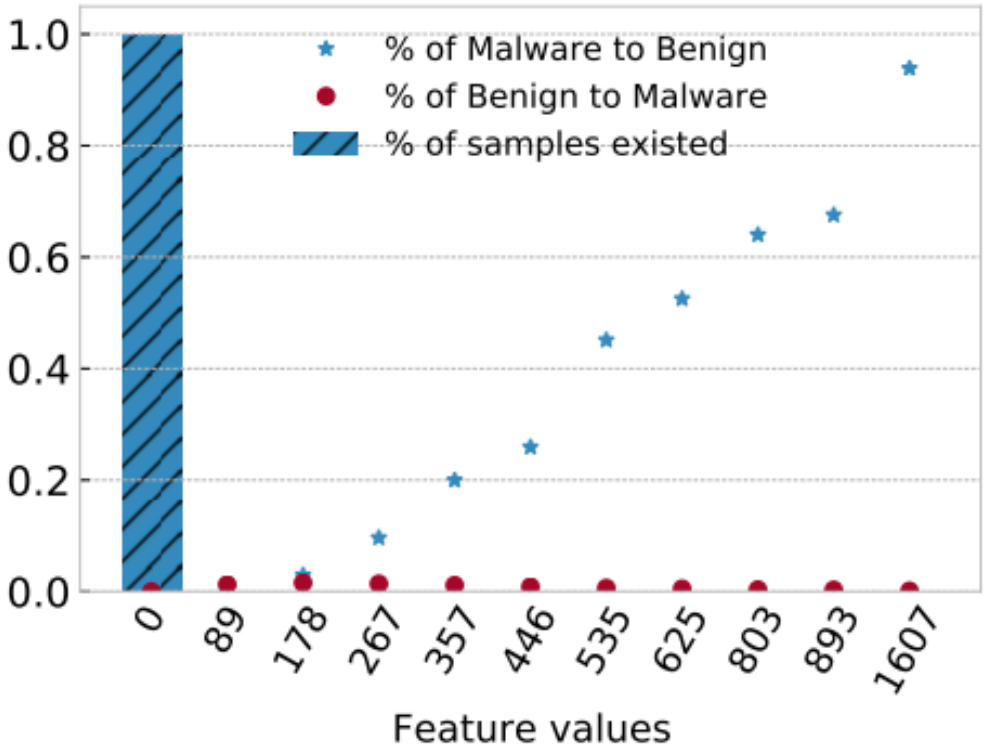
In this case, a model may assign large weights to these sparse values to lower training loss, with no immediate performance impact.

Motivation – why is mitigating sparsity necessary for malware datasets



- Malware dataset tends to have more severe sparsity.

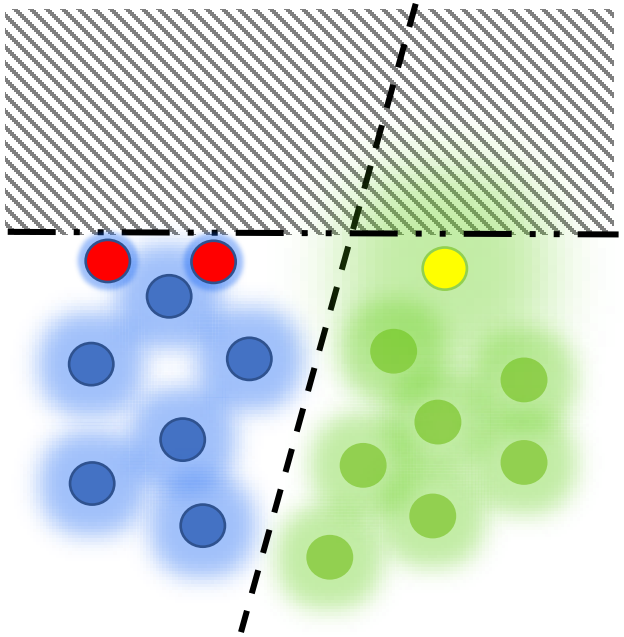
Motivation – how it affects detectors



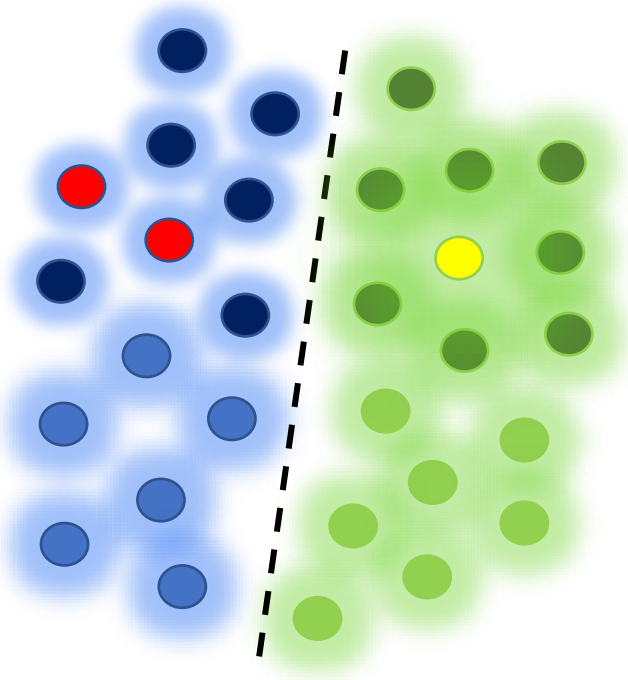
The feature's value distribution
of registry_count in EMBER(PE).

- **Sparse features and values are always assigned with large weights.**

Intuitively solving sparsity problems

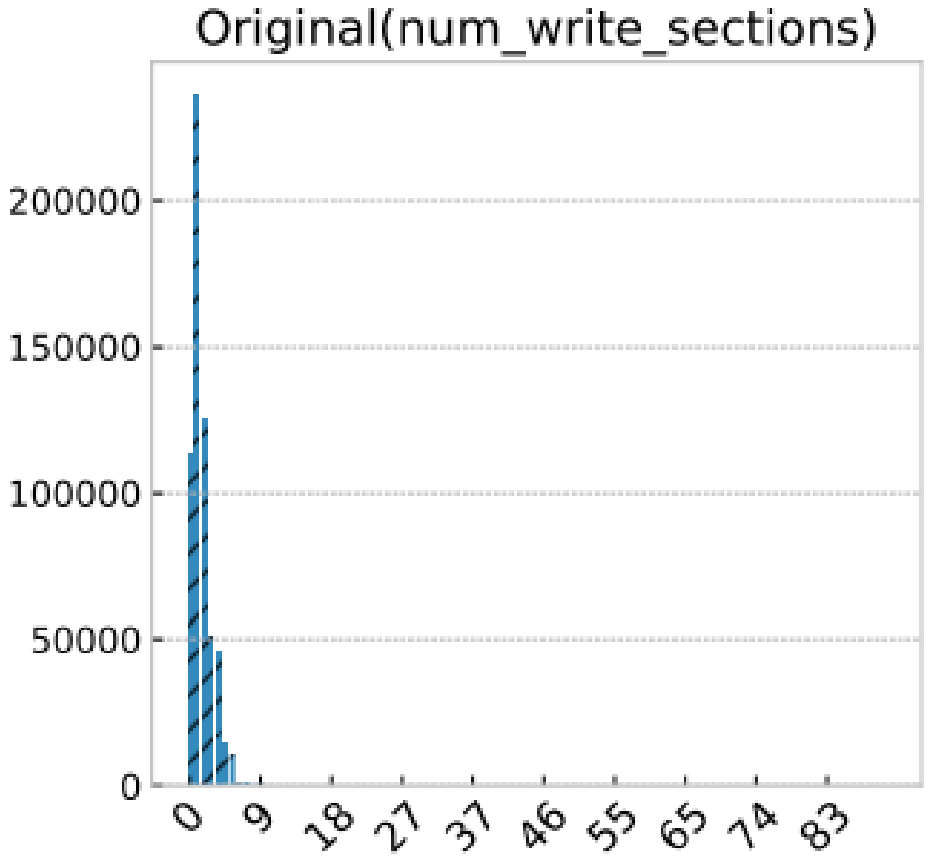


Compressing the sparse regions

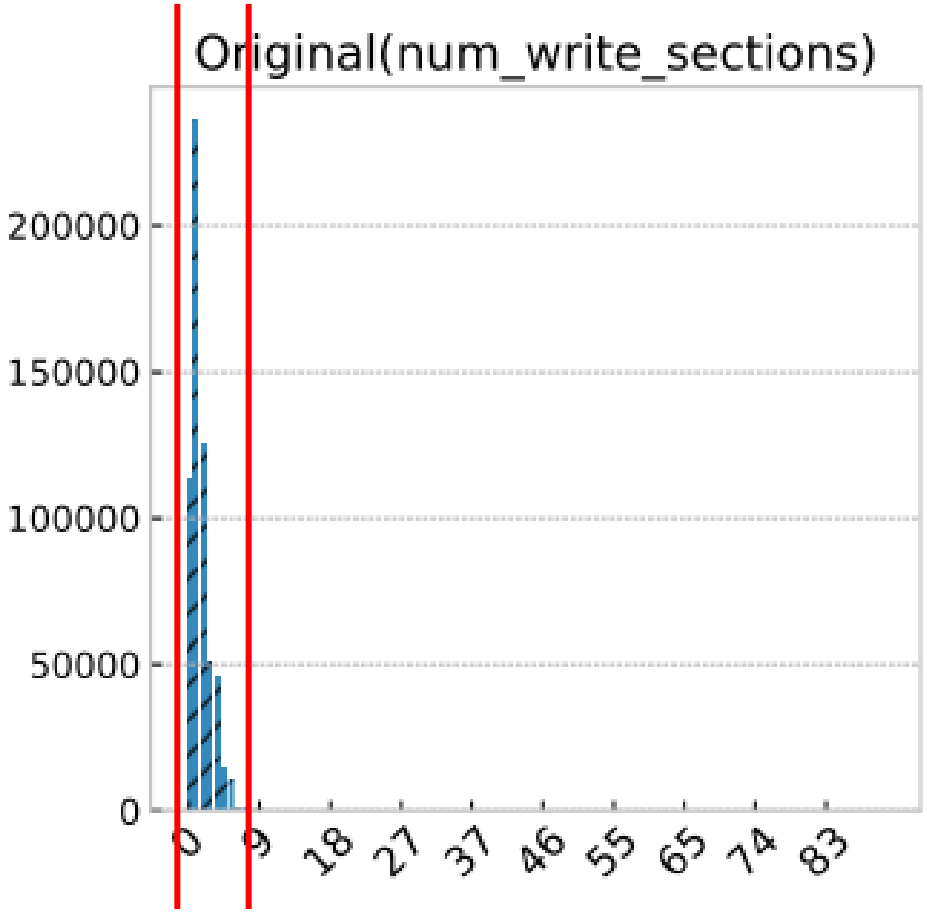


Filling the sparse regions

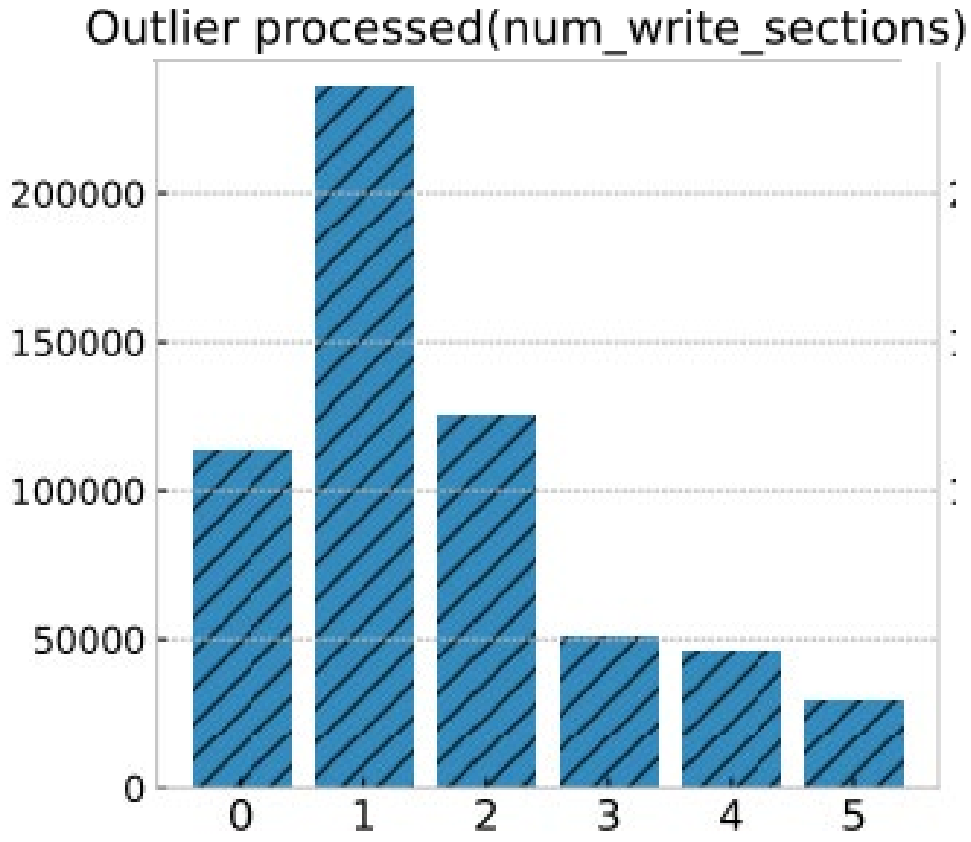
Subspace Compression with Bundling (SCB)



Subspace Compression with Bundling (SCB)

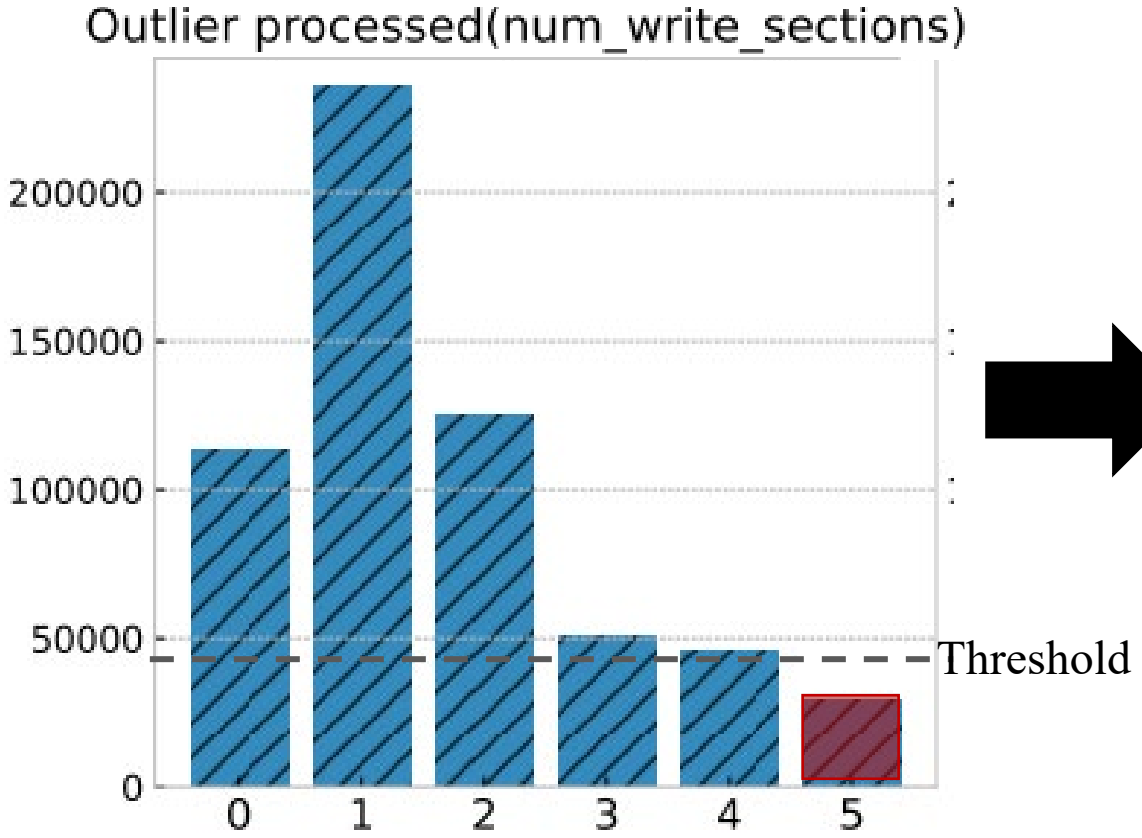


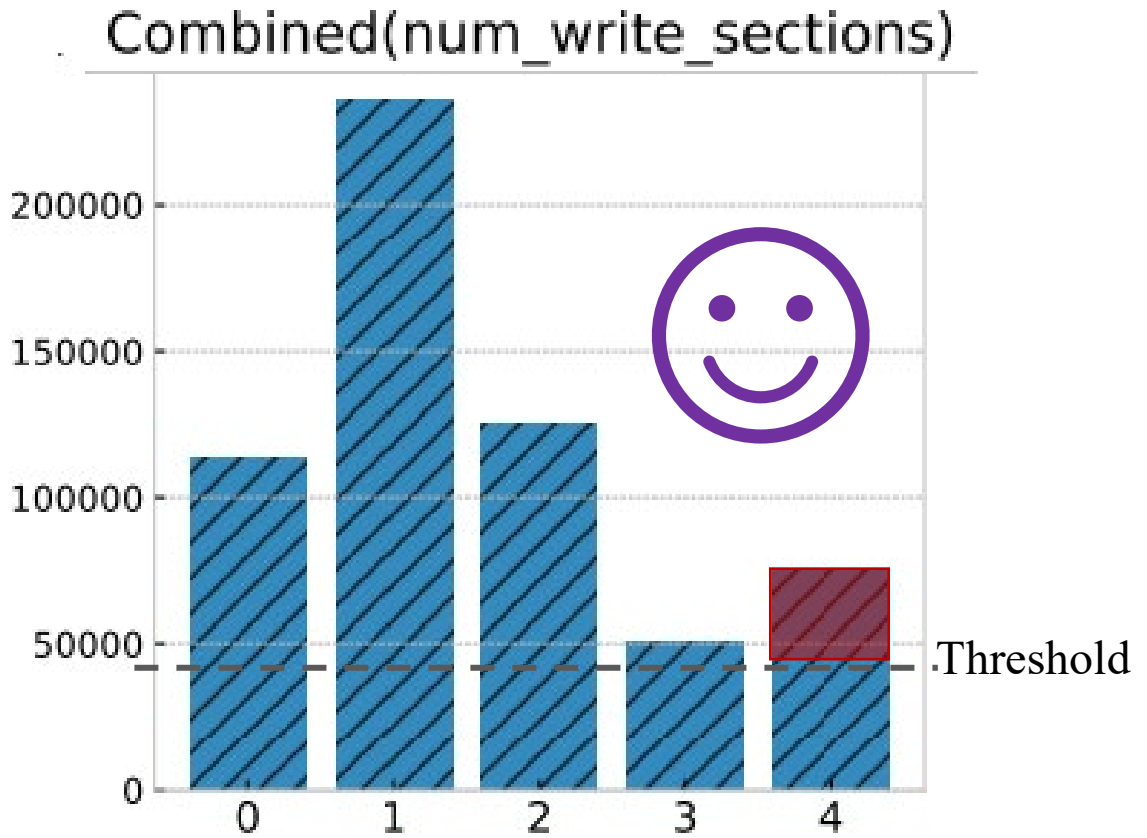
[Q1−3×IQR, Q3+3×IQR]



1. **Process outliers**

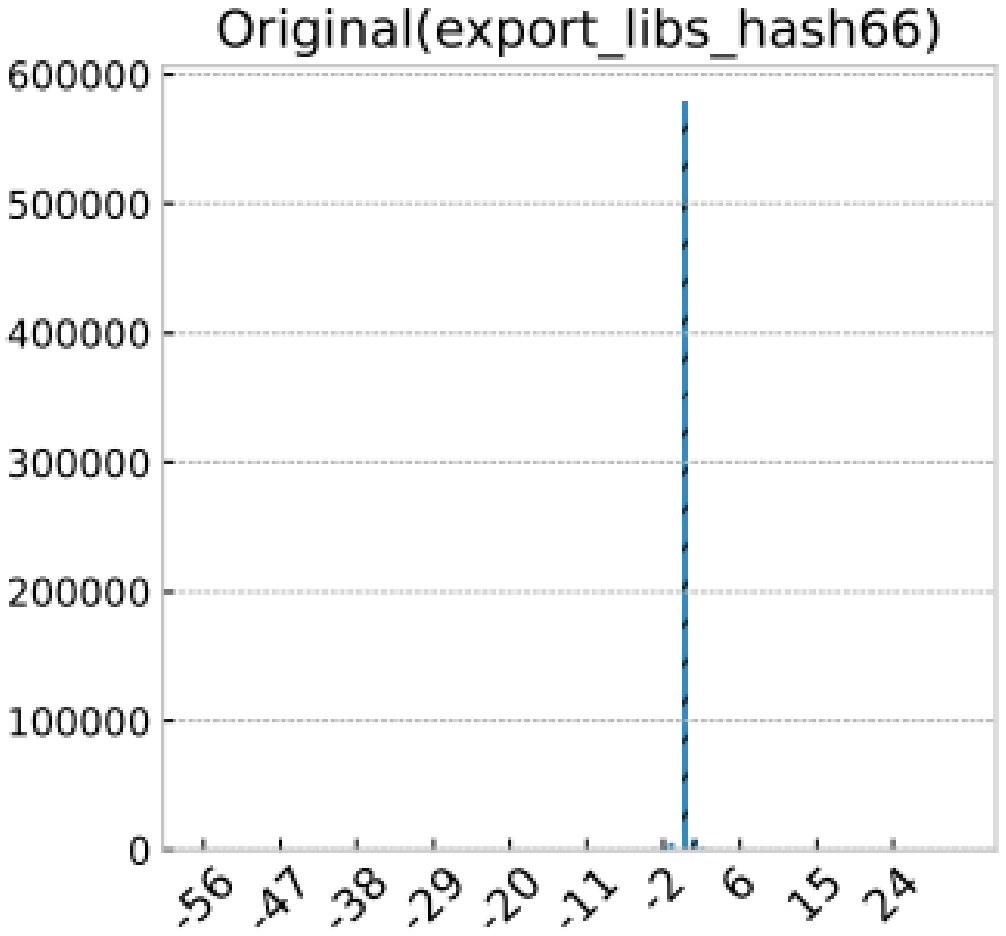
Subspace Compression with Bundling (SCB)





2. Combine sparse regions

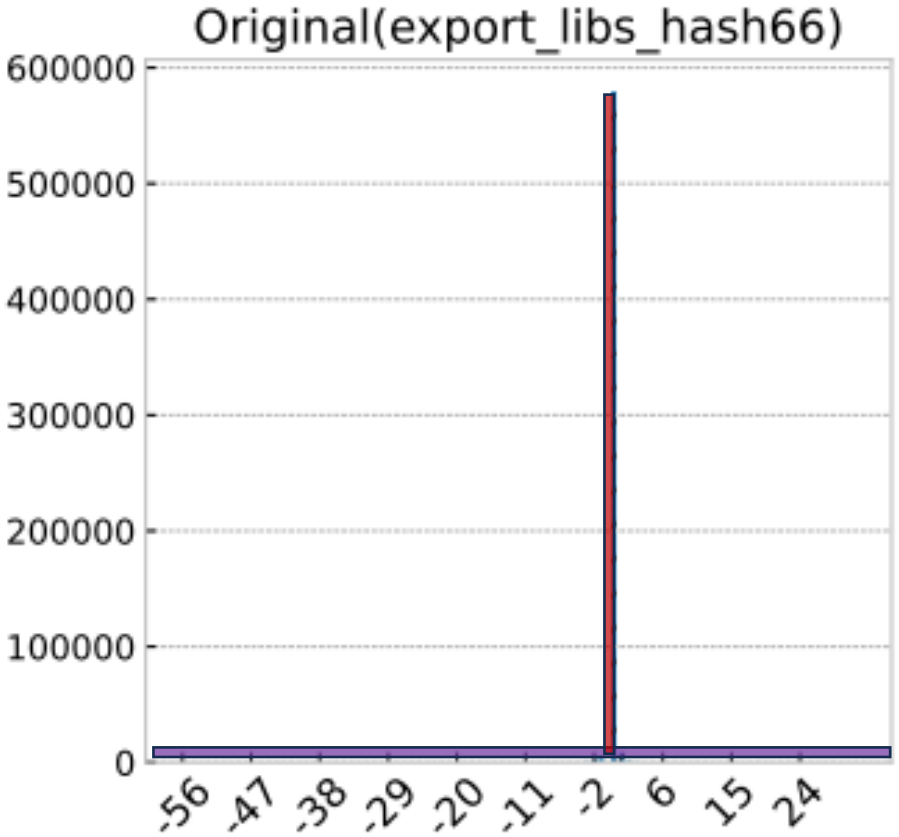
Subspace Compression with Bundling (SCB)



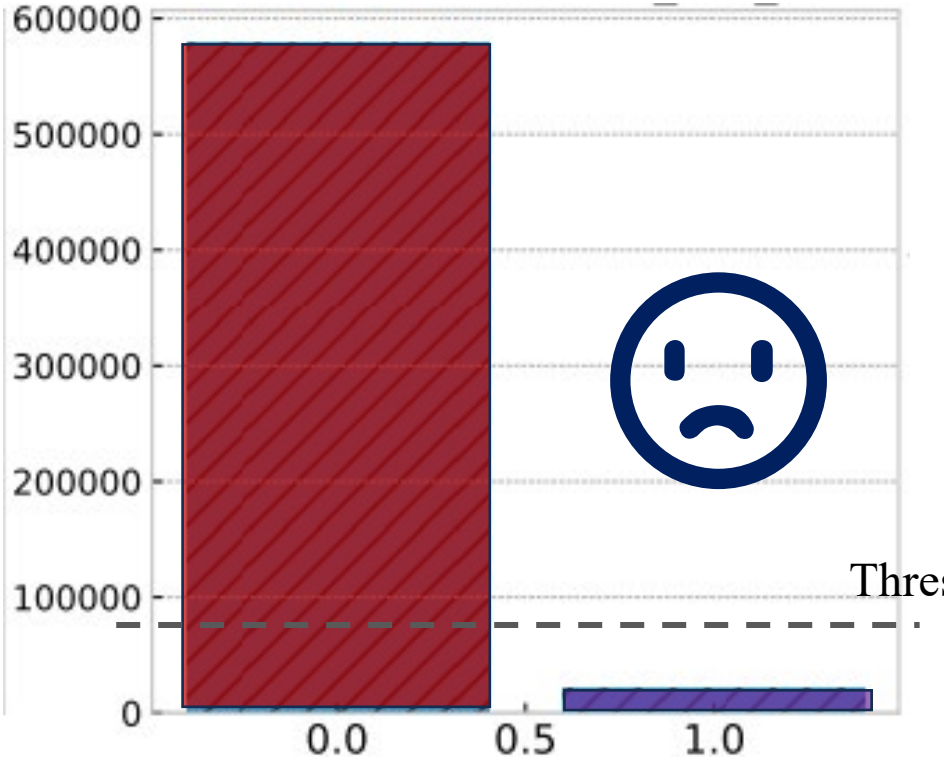
Q1−3×IQR==Q3+3×IQR

If cut it, only one single value left

Subspace Compression with Bundling (SCB)



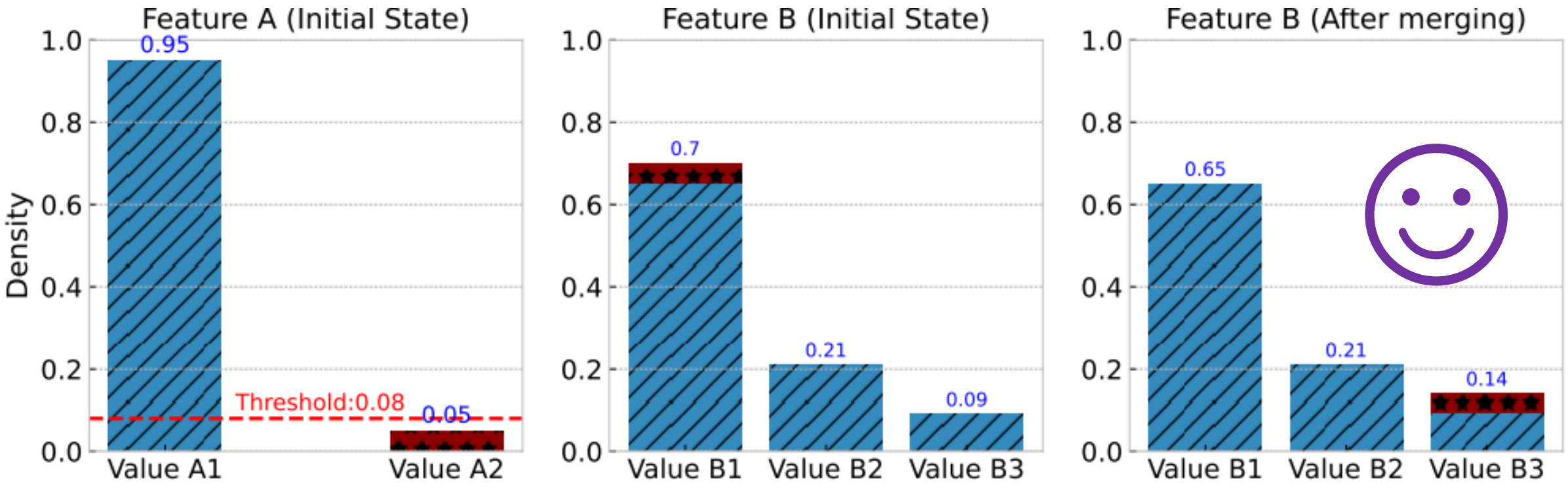
$Q1-3\times IQR == Q3+3\times IQR$



Cannot combing anymore!

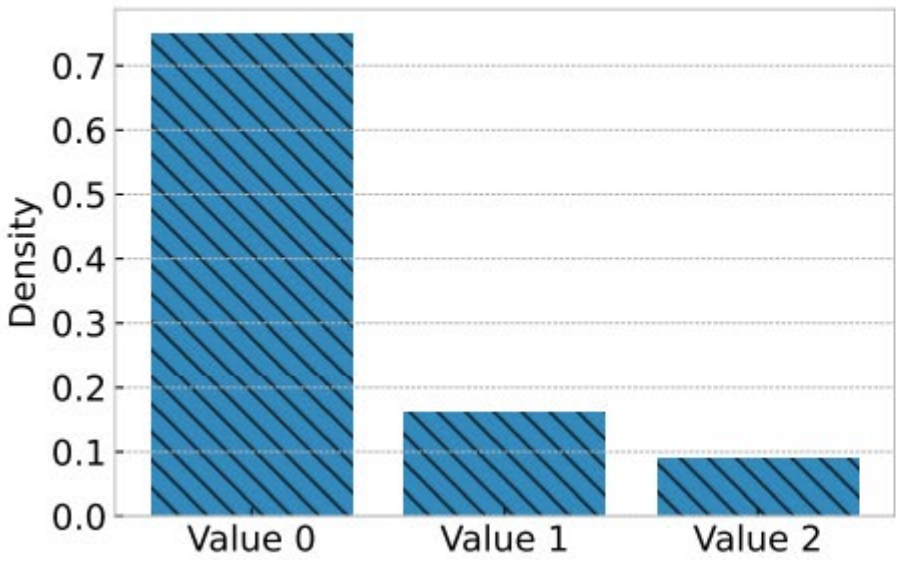
Subspace Compression with Bundling (SCB)

Bundling values between different features

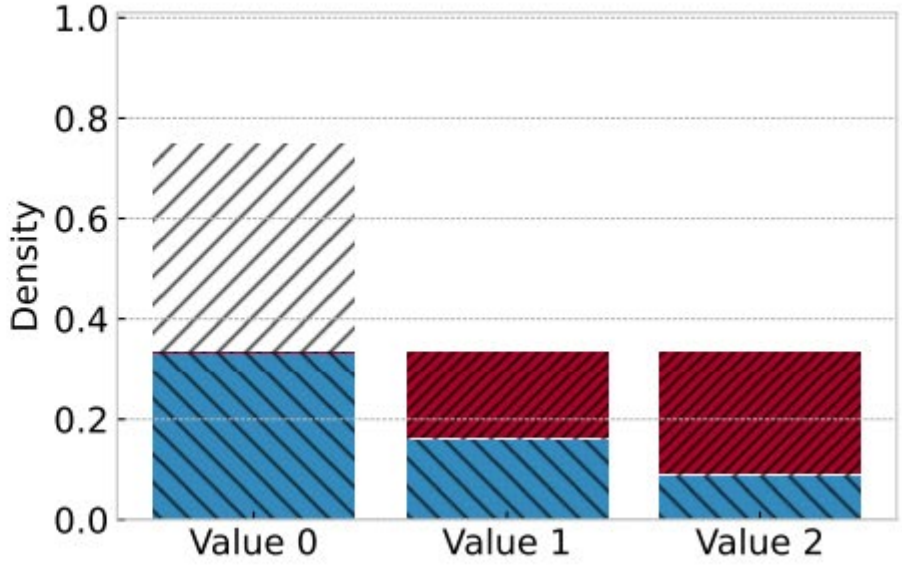


1. Value A2 always comes with Value B1.
2. Replace Value B1 that shows up with Value A2 with Value B3.
3. Remove Feature A.

Density boosting



Before



After

$$\min_{\{\theta\}} \mathbb{E}_{(x,y)\in\widehat{D}} \left[L(x, y, \theta) + L(db(x), y, \theta) \right]$$

db(x): replacing dense values with sparse ones

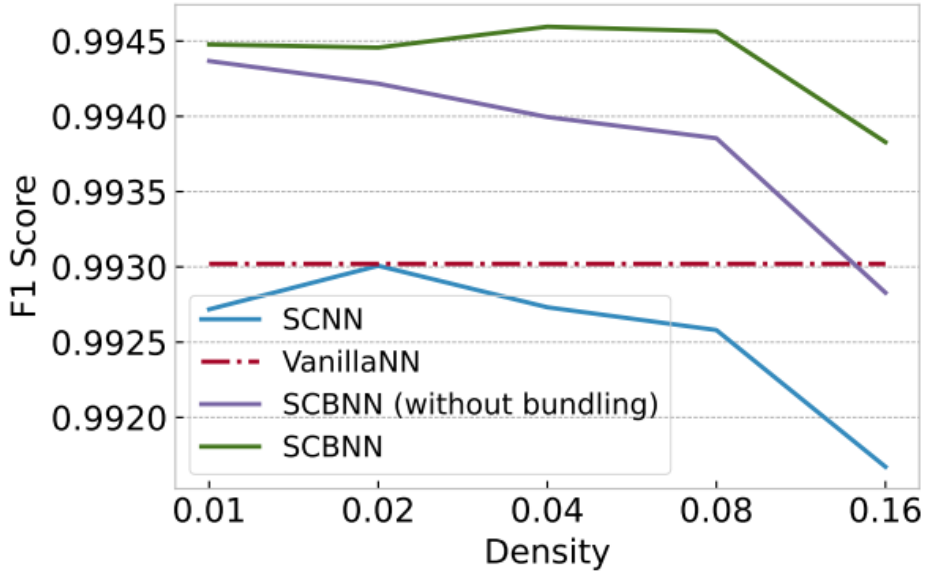
Datasets

Dataset	Size	Type	period
EMBER (Anderson et al. 2018)	800K samples	Windows PE	2017-01~2017-12
SOREL-20M (Harang et al. 2020)	12.6M samples	Windows PE	2018-01~2019-04 (being used in our experiments)
DREBIN-2019 (Federico et al.)	232K samples	Android APK	2014-01~2018-12
Contagio[1]	20K samples	PDF	2013

1. https://contagiodump.blogspot.com/2013/03/16800-clean-and-11960-malicious-files.htm

Performance and Sustainability

- Improved performance with SCB ONLY
- Further improvement with Density Boosting combined.
- Improved Sustainability (higher AUT)
- Outperform others

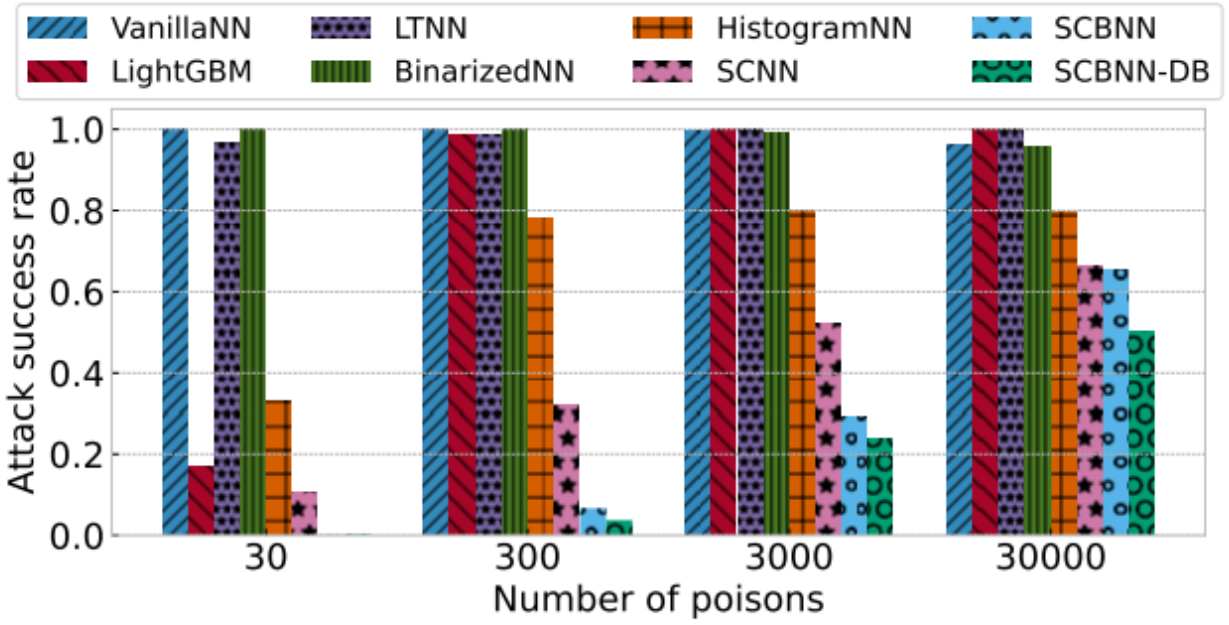


Model	F1 score	FP rate	FN rate	AUT (F1,16m) on SOREL
VanillaNN	0.99302	0.00442	0.00958	0.92850
LTNN	0.99311	0.00400	0.00977	0.93312
BinarizedNN	0.98942	0.00776	0.01339	0.91887
HistogramNN	0.99390	0.00323	0.00852	0.94148
SCNN	0.99225	0.00397	0.01148	0.93387
LightGBM	0.99470	0.00258	0.00799	0.94651
SCBNN	0.99456	0.00363	0.00721	0.94444
SCBNN-DB	**0.99488**	0.00381	0.00642	**0.95135**

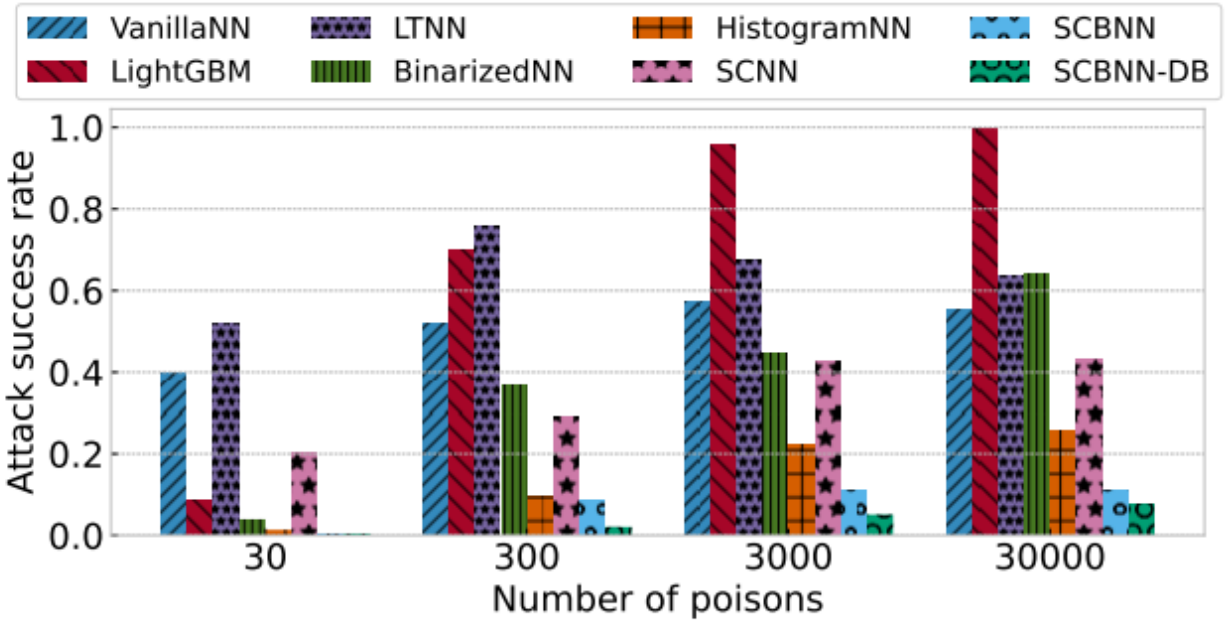
Backdoor attacks

Evaluation of backdoor attacks on different models (practical 16-feature triggers):

- Largely improved robustness against backdoor attacks (VR and EG)
- Still outperforms others



(a) VR-based Backdoor Attacks



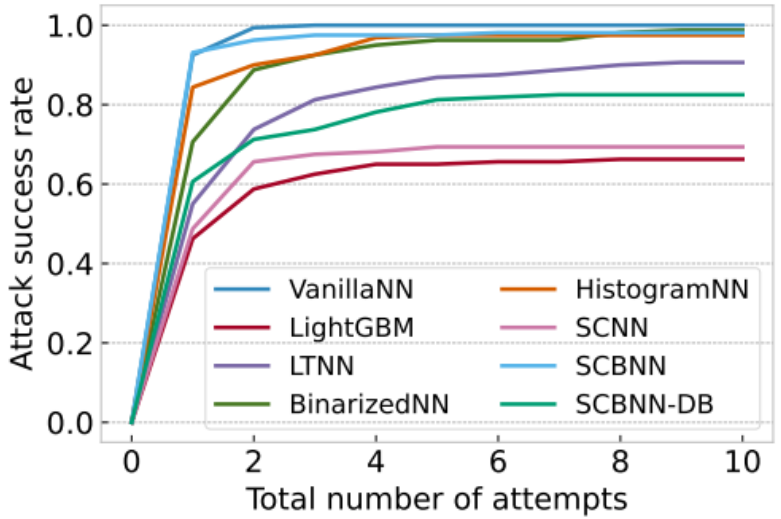
(b) EG-based Backdoor Attacks

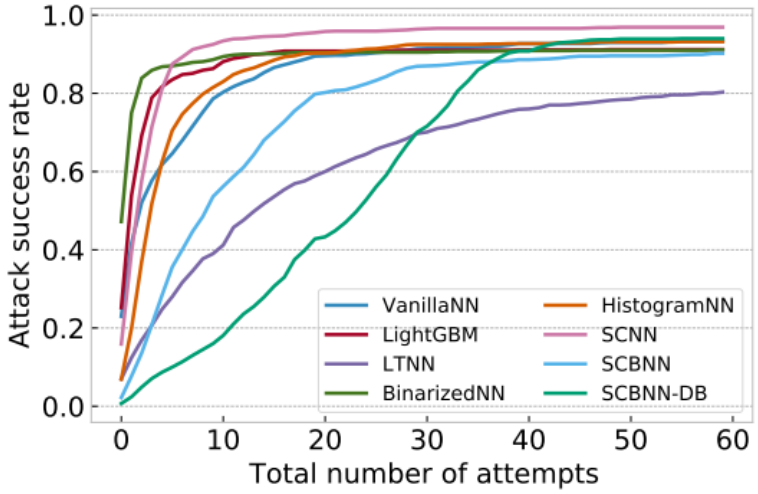
Evasion attack

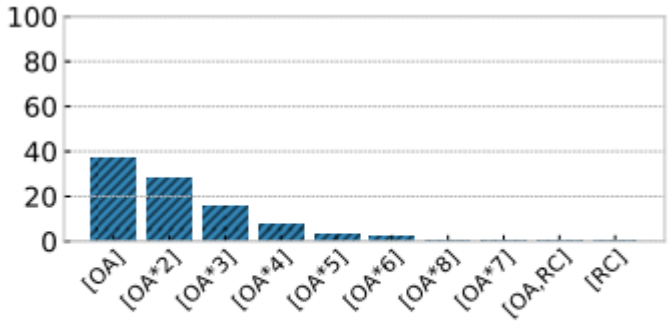
(Gamma and MAB-malware)

Evaluating query-based evasion attacks :

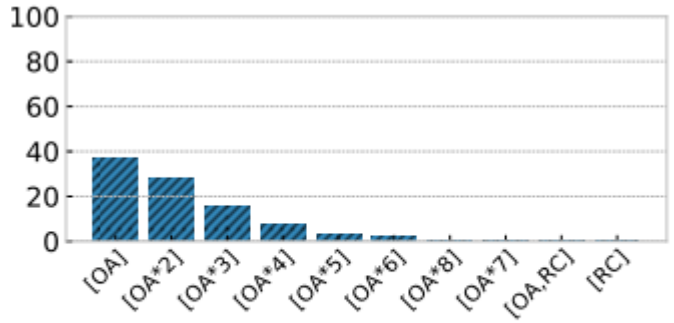
- Marginal defenses against query-based evasions by mitigating sparsity solely
- **Increased attackers' query budgets and required perturbation.**







(h) SCBNN-DB



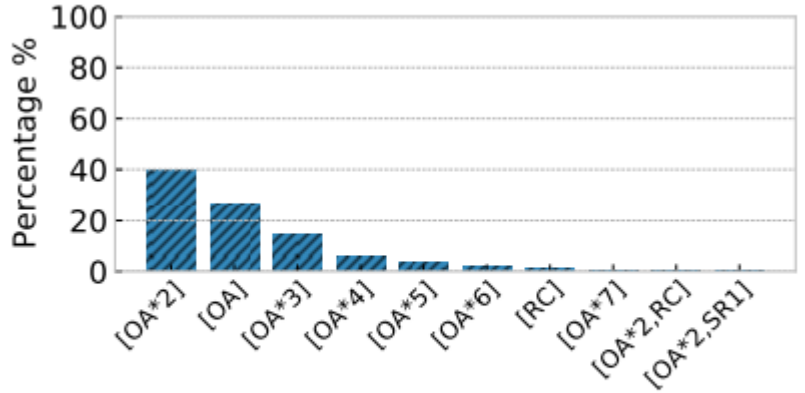
(h) SCBNN-DB

MAB-malware attacks at 6,000 queries

Model	Attack success rate
VanillaNN	1.0
LightGBM	0.9588
LTNN	0.8454
BinarizedNN	0.8980
HistogramNN	0.9375
SCNN	0.8854
SCBNN	0.8351
SCBNN-DB	**0.3718**

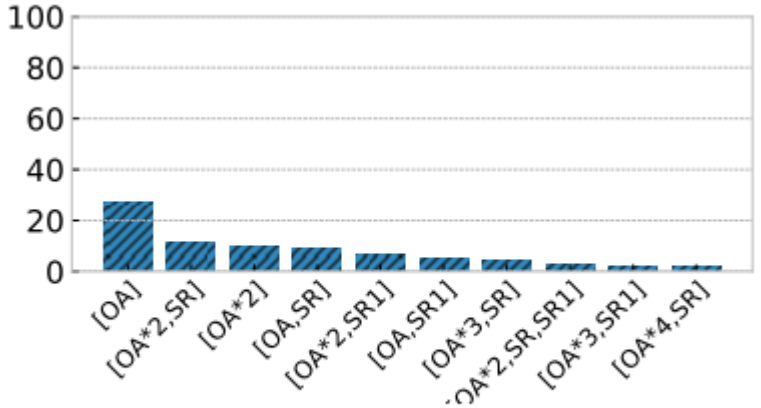
Combination with other defenses

Metric	VanillaNN+PAD	SCBNN+PAD	SCBNN-DB+PAD
F1 score	0.97136	0.99240	**0.99362**
Rejected ratio	0.05	0.0235	0.0305
ASR on VRB	0.99021	0.13640	**0.01883**
ASR on EGB	0.63251	0.05210	**0.00875**
ASR on GAMMA	0.868	0.563	**0.256**
ASR on MAB	0.968	0.873	**0.806**



(b) SCBNN-DB+PAD

- Complementary to other defenses
- Maintain higher performance.
- Better robustness against backdoors and GAMMA evasion.
- Slightly improved robustness against MAB evasion, but it largely increased perturbation size and query budget.



(a) SCBNN+PAD

Discussion

- Largely improved performance and sustainability on Android dataset
- Largely improved robustness on PDF dataset.
- Still works well with PAD on Android and PDF datasets
- Every step of processing is verified functional for the SCB.
- Directly removing sparse features instead of mitigating them can cause large performance loss.
- SCB is also complementary to advanced API features like API-Graph (improved the AUT to 60.156% from 51.549%).

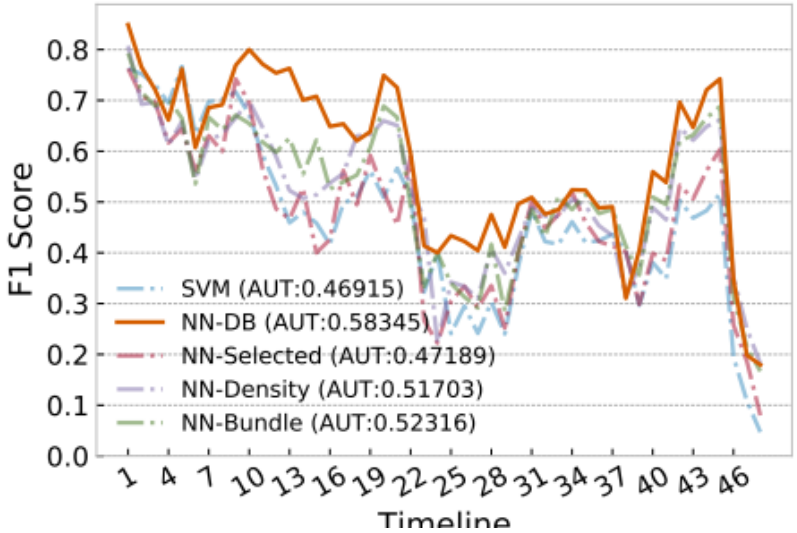


TABLE IV: Attack success rate on PDF datasets.

	F1 score	VRB	EGB	Mimicry ×1	Mimicry ×10	Mimicry ×30
Random Forest	0.99863	0.99370	0.99201	0.743	1.0	1.0
NN	0.99852	0.99119	0.98690	0.575	0.995	1.0
LTNN	0.99874	0.74498	0.18952	0.475	0.910	0.970
BinarizedNN	0.99849	0.32736	0.56511	0.230	0.765	0.935
HistogramNN	0.99892	0.40286	0.25122	0.320	0.950	0.990
SCNN	0.99849	0.40023	0.14560	0.205	0.76	0.9750
SCBNN	0.99897	0.22615	0.14880	0.281	0.862	0.977
SCBNN-DB	0.99882	0.01061	0.00068	0.334	0.893	1.0
SCBNN+PAD	0.99904	0.02105	0.02280	0.02	**0.053**	**0.075**
SCBNN-DB+PAD	**0.99939**	**0.00520**	**0.0**	**0.01**	0.06	0.085

TABLE VI: Performance under Sparse feature elimination.

	feture kept	F1 score	AUT(F1,16m)
Original	2,381	0.99302	0.92850
VR≥0.01	287	0.99041	0.92235
VR≥0.1	64	0.98547	0.90645
SCB (8% density)	1,239	0.99456	0.94444

Conclusion

- The sparsity problem runs through most of malware datasets, especially datasets with tabular features. (*Tabular dataset is usually based on expert experience, which are dedicated to highlight the abnormal situation, which brings many sparsity problem in the dataset.)
- Mitigating sparsity is crucial for the success of ML-based malware detection.
- Subspace Compression with Bundling (SCB) and Density boosting are shown to be effective in improving performance, robustness, and sustainability.

Thank you!

https://github.com/IanWE/Density-Boosts-Robustness-Code