CADE: Detecting and Explaining Concept Drift Samples for Security Applications

USENIX Security 2021

Limin Yang[1], Wenbo Guo[2], Qingying Hao[1],

Arridhana Ciptadi[3], Ali Ahmadzadeh[3], Xinyu Xing[2] , Gang Wang[1]

[1]University of Illinois at Urbana-Champaign [2]Penn State [3]Blue Hexagon Inc.

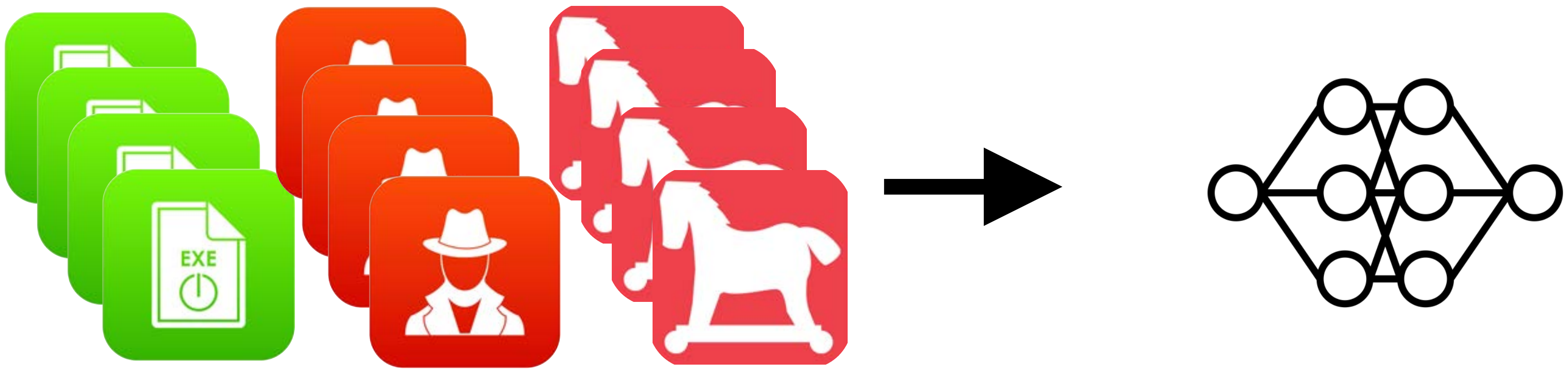
ILLINOIS Computer Science

PennState College of Information Sciences and Technology

BLUEHEXAGON

A Multi-class Malware Classification Model

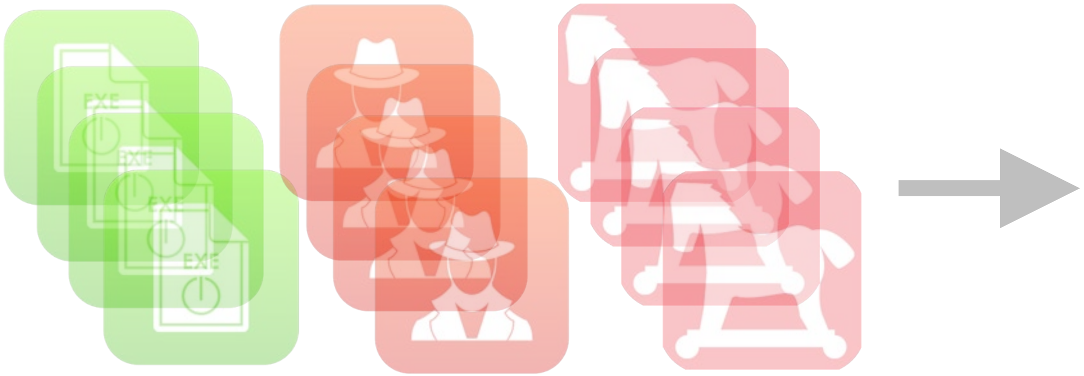
1. Train



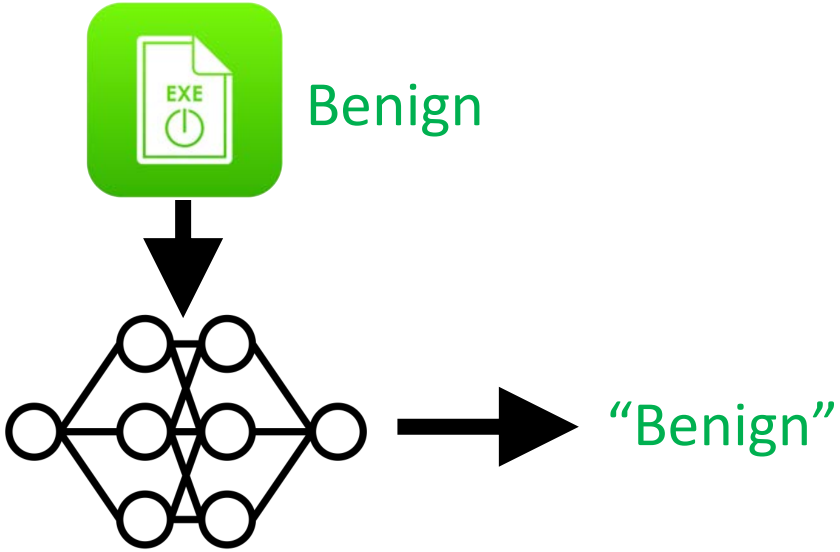
Benign   Malware A   Malware B

A Multi-class Malware Classification Model

1. Train



2. Predict



A Multi-class Malware Classification Model

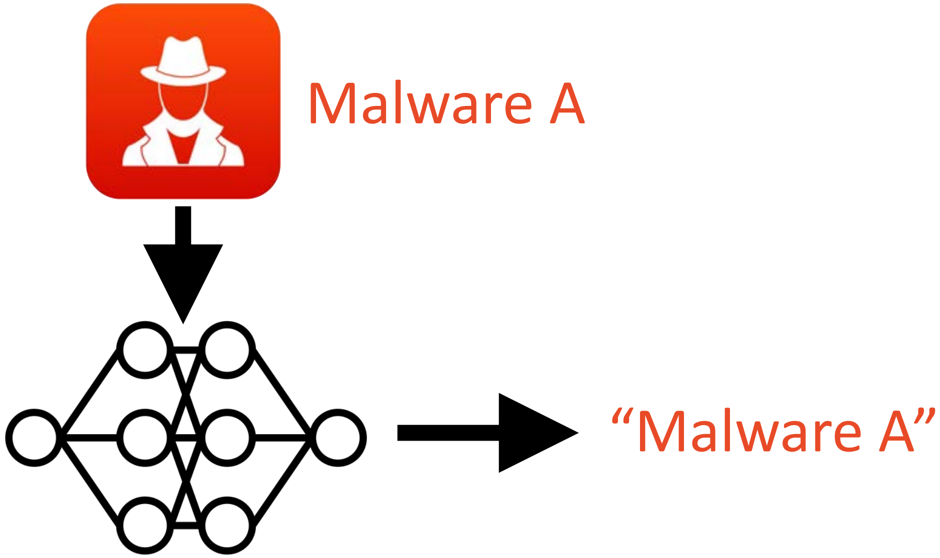
1. Train

Benign

Malware A

Malware B

2. Predict



A Multi-class Malware Classification Model

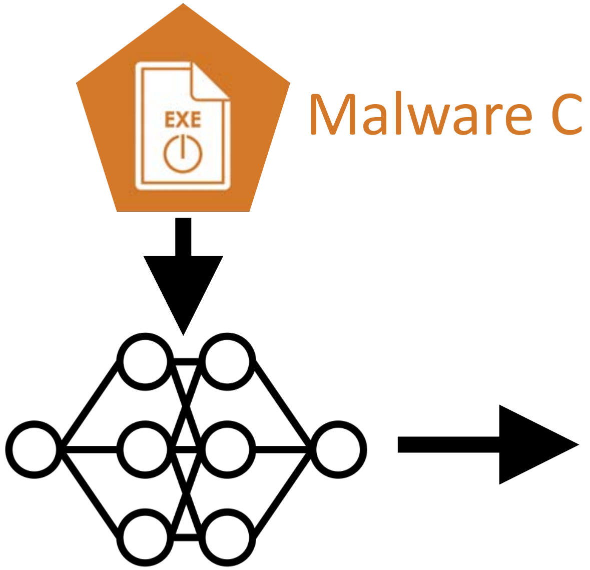
1. Train

Benign

Malware A

Malware B

2. Predict

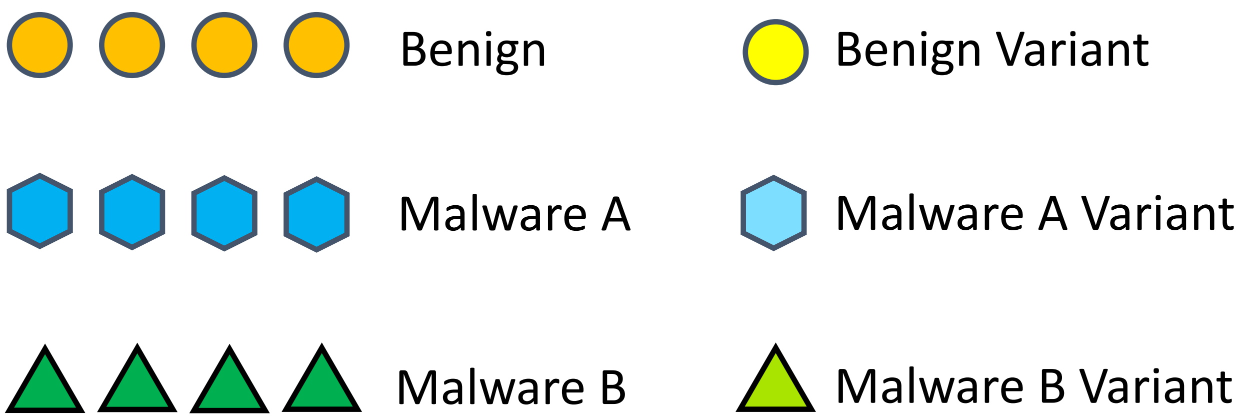


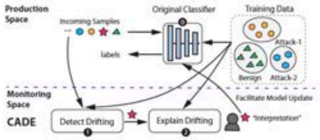
“Benign”?
“Malware A”?
“Malware B”?

Concept Drift!
(Unseen family)

Another Type of Drift: In-class Evolution

Train **Predict**





CADE: Detecting and Explaining Concept Drift Samples for Security Applications

ARTIFACT EVALUATED usenix ASSOCIATION PASSED

Limin Yang*, Wenbo Guo†, Qingying Hao*, Arridhana Ciptadi‡
Ali Ahmadzadeh‡, Xinyu Xing†, Gang Wang*
*University of Illinois at Urbana-Champaign †The Pennsylvania State University ‡Blue Hexagon
liminy2@illinois.edu, wzg13@ist.psu.edu, qhao2@illinois.edu, {arri, ali}@bluehexagon.ai, xxing@ist.psu.edu, gangw@illinois.edu

Abstract

Concept drift poses a critical challenge to deploy machine learning models to solve practical security problems. Due to the dynamic behavior changes of attackers (and/or the benign counterparts), the testing data distribution is often shifting from the original training data over time, causing major failures to the deployed model.

To combat concept drift, we present a novel system CADE aiming to 1) *detect* drifting samples that deviate from existing classes, and 2) *provide explanations* to reason the detected drift. Unlike traditional approaches (that require a large number of new labels to determine concept drift statistically), we aim to identify individual drifting samples as they arrive. Recognizing the challenges introduced by the high-dimensional outlier space, we propose to map the data samples into a low-dimensional space and automatically learn a distance function to measure the dissimilarity between samples. Using contrastive learning, we can take full advantage of existing labels in the training dataset to learn how to compare and contrast pairs of samples. To reason the meaning of the detected drift, we develop a distance-based explanation method. We show that explaining "distance" is much more effective than traditional methods that focus on explaining a "decision boundary" in this problem context. We evaluate CADE with two case studies: Android malware classification and network intrusion detection. We further work with a security company to test CADE on its malware database. Our results show that CADE can effectively detect drifting samples and provide semantically meaningful explanations.

Figure 1: Drifting sample detection and explanation.

environments in which the models are deployed are usually dynamically changing over time. Such changes may include both organic behavior changes of benign players and malicious mutations and adaptations of attackers. As a result, the testing data distribution is shifting from the original training data, which can cause serious failures to the models [23].

To address concept drift, most learning-based models require periodical re-training [36, 39, 52]. However, retraining often needs labeling a large number of new samples (expensive). More importantly, it is also difficult to determine *when* the model should be retrained. Delayed retraining can leave the outdated model vulnerable to new attacks.

We envision that combating concept drift requires establishing a monitoring system to examine the relationship between the incoming data streams and the training data (and/or the current classifier). The high-level idea is illustrated in Figure 1. While the original classifier is working in the *pro-*

Why Concept Drift Matters?

- New attacks (zero-day) are NOT trivial

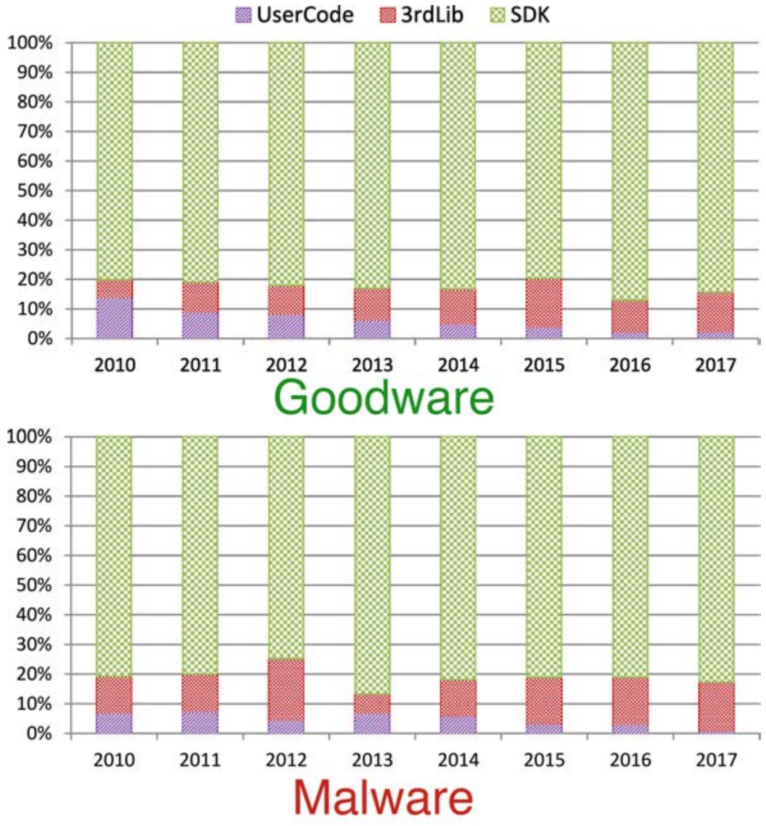
FireEye Annual Report 2020

"1.1 million malware samples per day"

"41% malware families never seen before"

Why Concept Drift Matters?

- New attacks (zero-day) are NOT trivial
- Both malware and goodware evolve over time

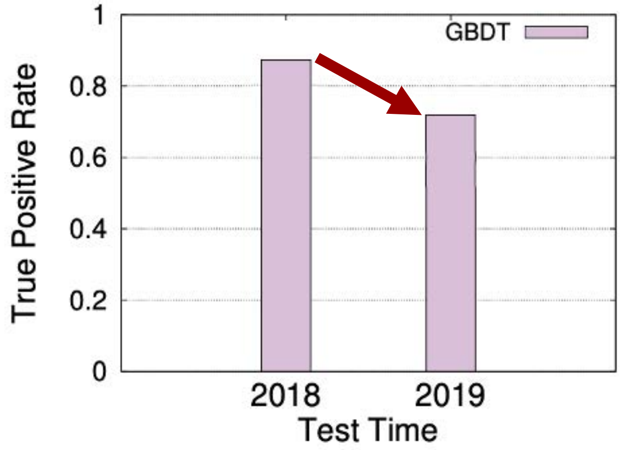


Functionality scope distribution of Android goodware and malware [1].

[1] A study of run-time behavioral evolution of benign versus malicious apps in android, Information and Software Technology, 2020.

Why Concept Drift Matters?

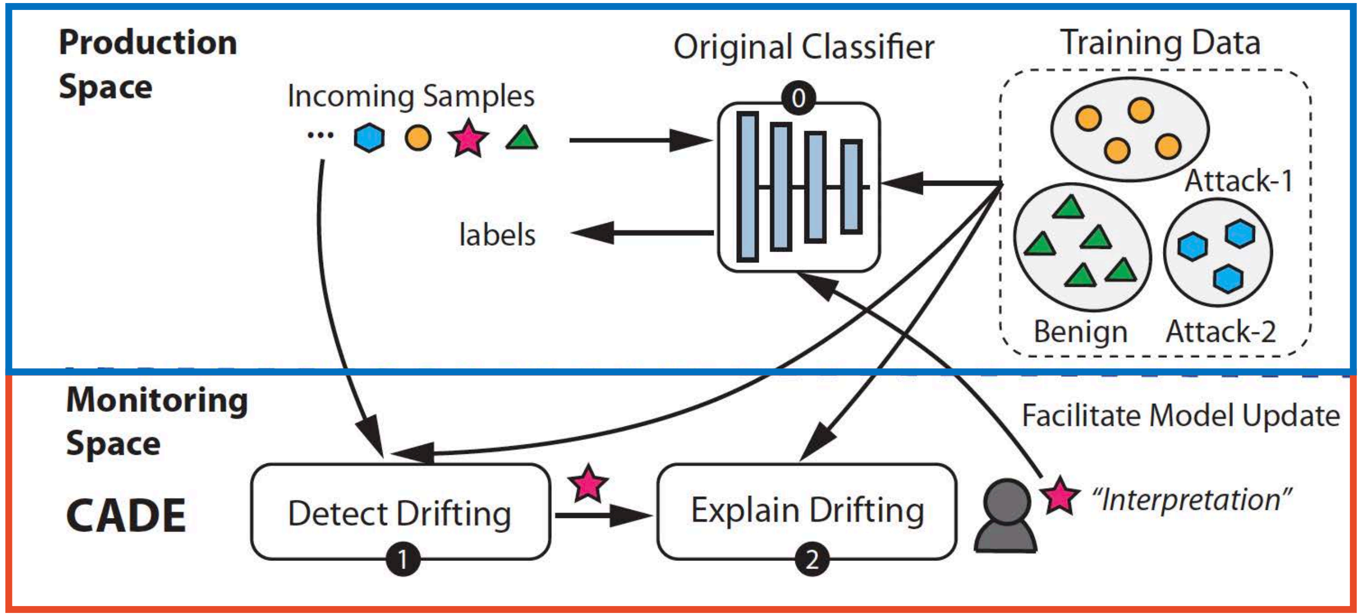
- New attacks (zero-day) are NOT trivial
- Both malware and goodware evolve over time
- ML models' decision boundaries shift



GBDT malware classifier trained on Ember-2018; Tested a year later using malware samples from Blue Hexagon Inc. [DLS'21]

* GBDT: Gradient Boosted Decision Tree

When NOT to Predict?



Goals

❶ Detect drifting samples

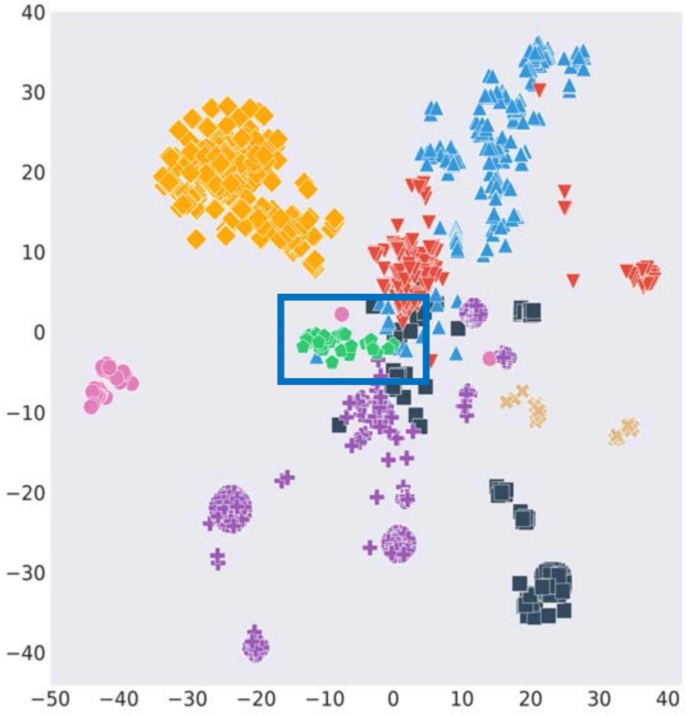
❷ Find a small subset of important features that explain why the drifting sample is different from training data

Existing Drifting Detection Solutions

- Solutions from ML community
  - JSTOR'54, SBIA'04, SDM'07, ICML'14, KDD'16, CIKM'19, etc.
  - Most require data labeling on the testing set → *costly for security domain*
- Solutions from security community
  - Transcend [USENIX Sec'17]
  - *Highly dependent on a good definition of "dissimilarity", scalability issues*

Why It’s Hard to Define a “Good” Distance Function?

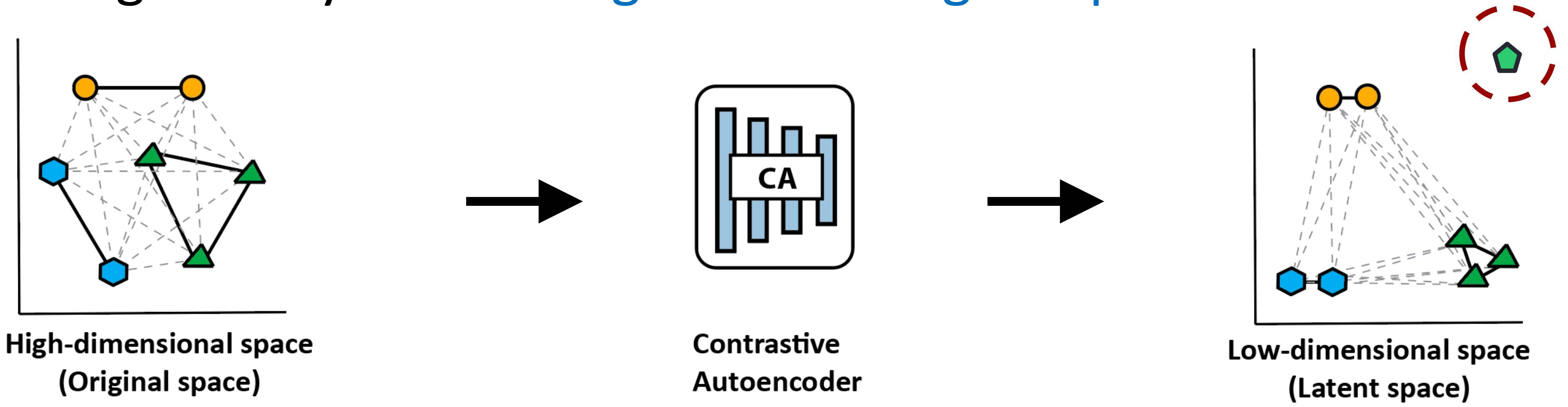
- Distance loses effectiveness in high-dimensional data
  - This sample feature space has 1,368 dimensions
- Drifting samples are not labeled, hard to differentiate from normal samples



T-SNE plot for the original space of an Android malware dataset (Unseen family:)

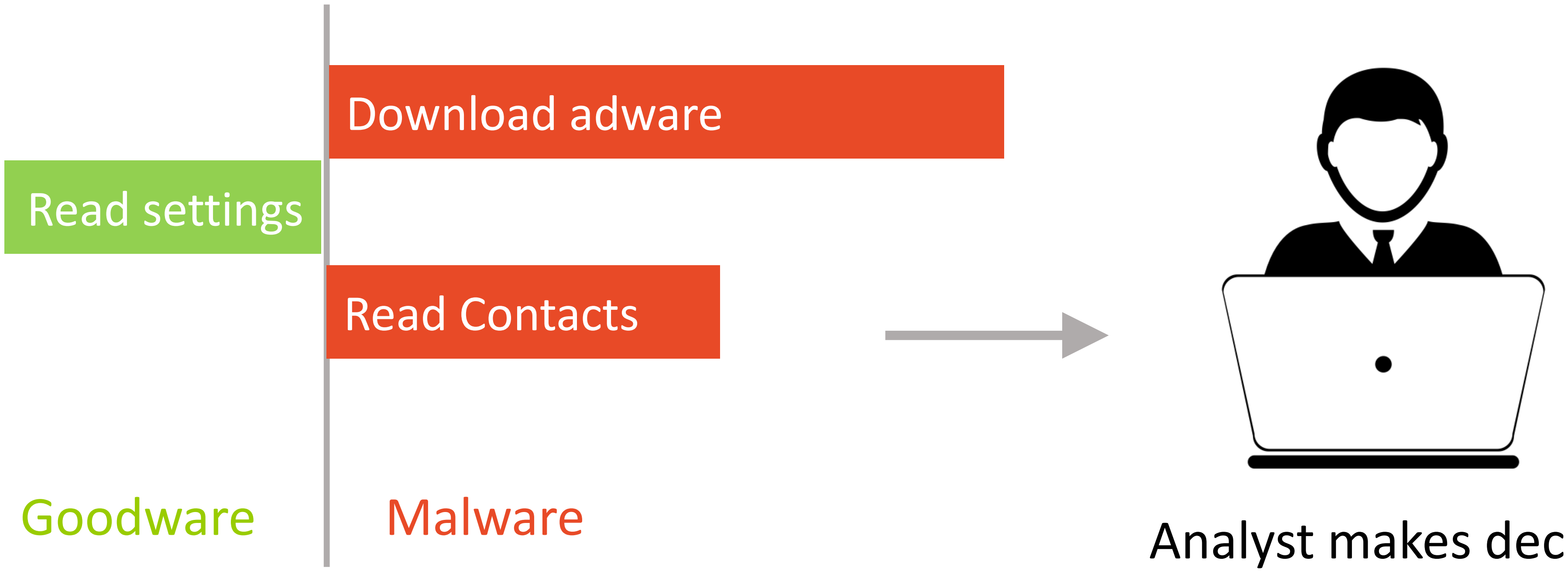
Self-supervision: Contrastive Learning

- No knowledge about future drifting samples → self-supervision
- Use contrastive learning to learn a compressed representation of the training data by contrasting with existing samples



- A sample is far away from ANY existing families' centroids, it's a potential drifting sample; rank for investigation

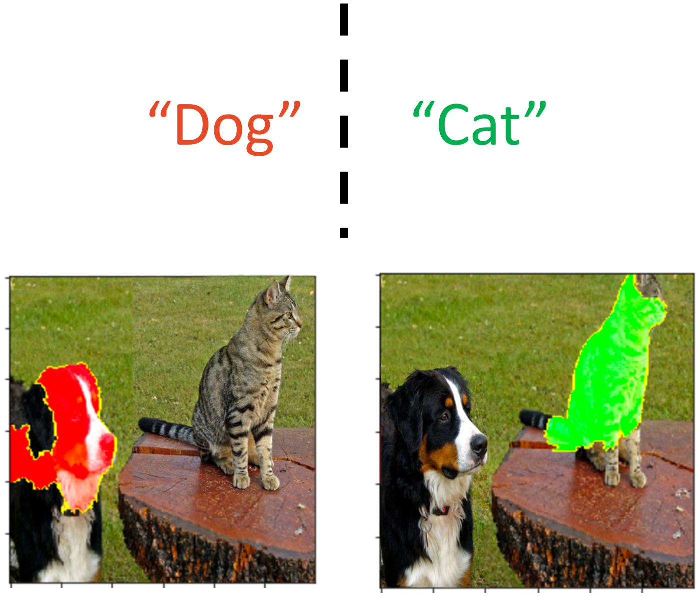
How to explain these drifting samples?



Analyst makes decision

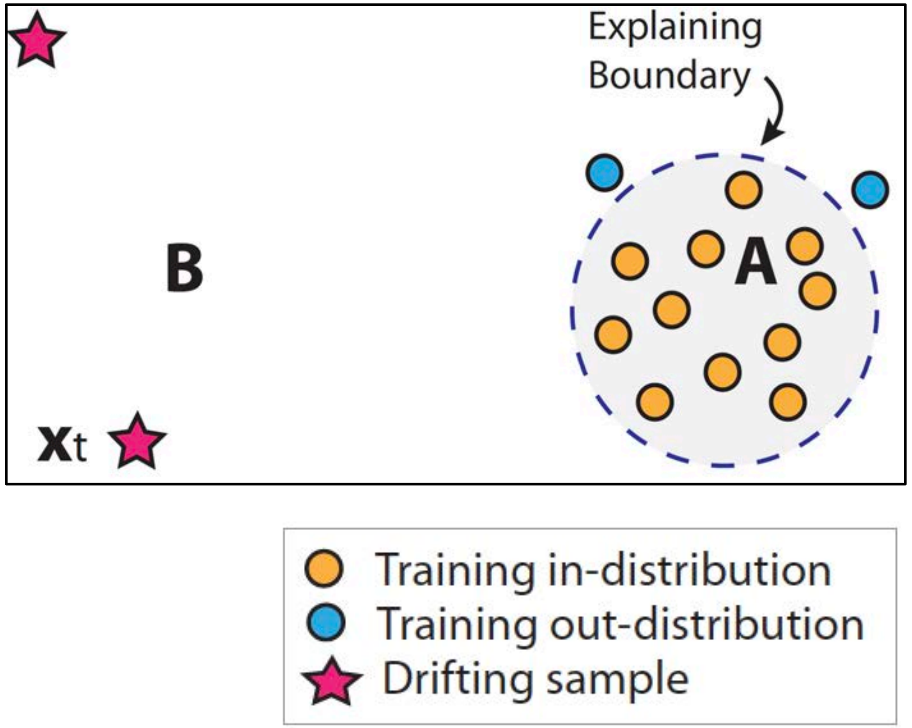
A Rich set of Explanation Methods

- Identify a small set of important features that make the drifting sample an outlier
- Naïve idea: boundary-based explanation
  - Approximate the decision boundary between *drift* and *in-distribution*
  - Explaining a supervised learning model
    - LIME [KDD'16], SHAP[NeurIPS'17], LEMNA [CCS'18], Perturbation [ICCV'17]
  - Using "crossing the boundary" as a signal to derive important features



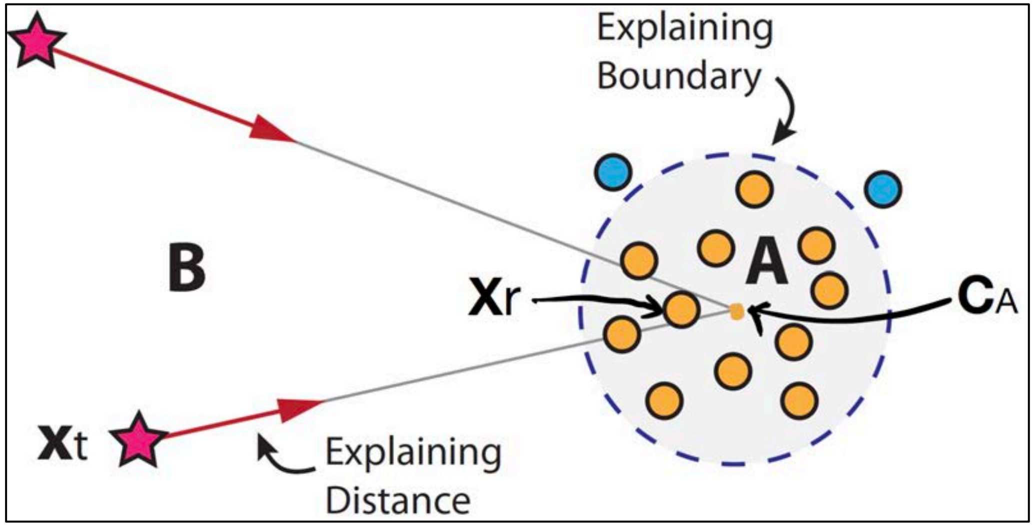
Problems with Boundary-based Explanation

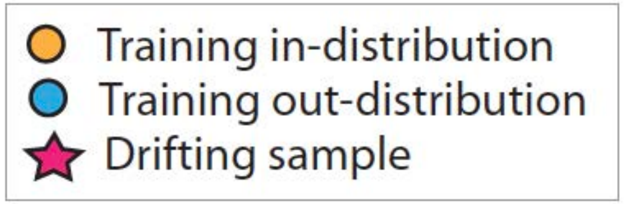
- Difficult to approximate the boundary
  - Drifting samples are limited
- Difficult to drag a drifting sample to cross the boundary
  - Drifting samples are far away from the boundary in the sparse area



Our Method: Distance-based Explanation

- Perturb the original features and observe the distance changes in latent space
- Perturbation strategy
  - Replace $\mathbf{x}_t$'s feature value with those of a reference sample $\mathbf{x}_r$
  - $\mathbf{x}_r$ is closest to the centroid of nearest family
- Optimization goal
  - Minimize the distance between $\mathbf{x}_t$ and $\mathbf{C}_A$
  - Use elastic-net regularization to minimize the number of selected important features





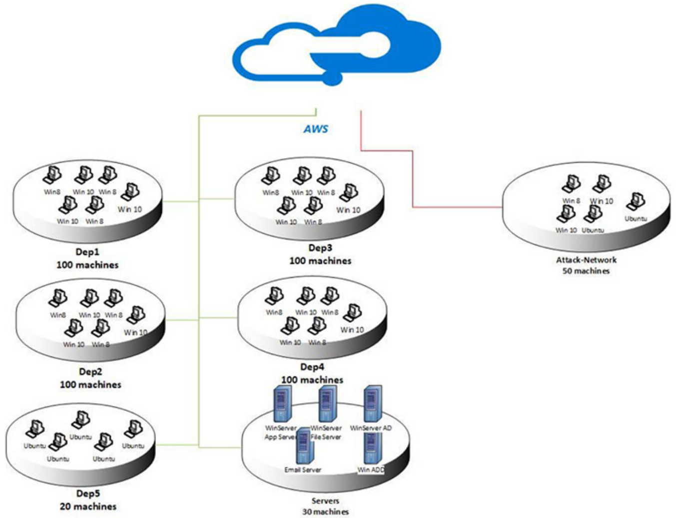
Evaluation: Datasets

Drebin [NDSS'14]

- Top 8 malware families
- 3,317 malware samples
- Training set: 80% of 7 families
- Testing set: 20% of 7 families + unseen family

IDS2018 [ICISSP'18]

- Benign + 3 types of network intrusion
- 130,702 network flows
- Training set: 80% of 3 families
- Testing set: 20% of 3 families + unseen family



Drift Detection Results

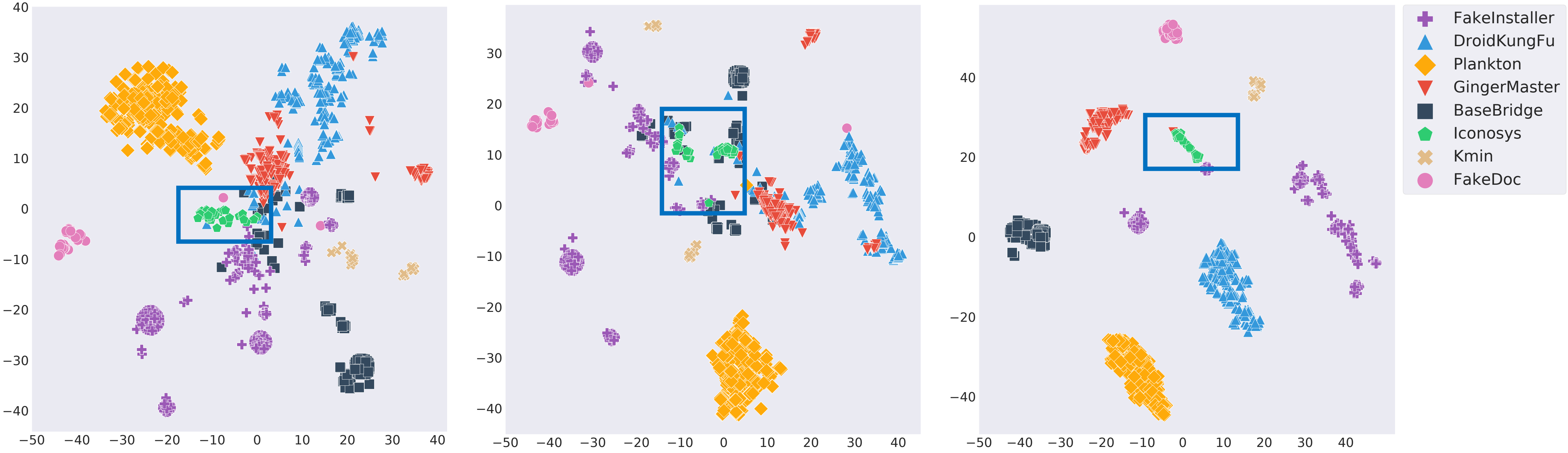
Iteratively choose a family as the unseen family and report the average results here.

Method	Drebin (Avg±Std)		IDS2018 (Avg±Std)	
	F_1	Norm. Effort	F_1	Norm. Effort
Vanilla AE	0.72±0.15	1.48±0.31	0.74±0.12	1.74±0.40
Transcend	0.80±0.12	1.29±0.45	0.65±0.46	1.45±0.57
CADE	**0.96±0.03**	**1.00±0.09**	**0.96±0.06**	**0.95±0.07**

Real-world test: evaluate on Blue Hexagon PE malware dataset, still effective!

* Vanilla AE: Standard Autoencoder without contrastive learning.

Why CADE Works?



Original Space | Latent Space (Vanilla AE) | Latent Space (CADE)

T-SNE visualization for Drebin dataset (Unseen family: Iconosys)

Drift Explanation: Case Study

Drifting sample family: **FakeDoc**; closest family: **GingerMaster**

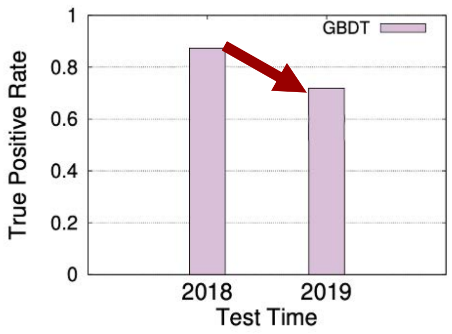
- Key difference: FakeDoc usually subscribes to premium services via SMS.

[api_call::android/telephony/SmsManager;->sendTextMessage] , [call::readSMS] , [permission::android.permission.DISABLE_KEYGUARD] , [permission::android.permission.RECEIVE_SMS] , [permission::android.permission.SEND_SMS] , [permission::android.permission.WRITE_SMS] , [real_permission::android.permission.SEND_SMS] , [permission::android.permission.READ_SMS] , [feature::android.hardware.telephony] , [permission::android.permission.READ_CONTACTS] , [real_permission::android.permission.READ_CONTACTS] , [api_call::android/location/LocationManager;->isProviderEnabled], [api_call::android/accounts/AccountManager;->getAccounts], [intent::android.intent.category.HOME], [feature::android.hardware.location.network], [real_permission::android.permission.RESTART_PACKAGES] , [real_permission::android.permission.WRITE_SETTINGS] , [api_call::android/net/ConnectivityManager;->getAllNetworkInfo], [api_call::android/net/wifi/WifiManager;->setWifiEnabled], [api_call::org/apache/http/impl/client/DefaultHttpClient], [url::https://ws.tapjoyads.com/] , [url::https://ws.tapjoyads.com/set_publisher_user_id?] , [permission::android.permission.CHANGE_WIFI_STATE], [real_permission::android.permission.ACCESS_WIFI_STATE], [real_permission::android.permission.BLUETOOTH], [real_permission::android.permission.BLUETOOTH_ADMIN], [call::setWifiEnabled].

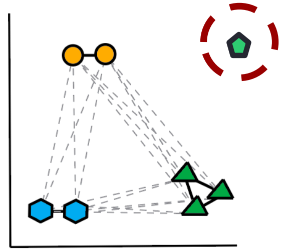
Important features selected by CADE (avg # of selected features is 45 out of 1000+)

Takeaways

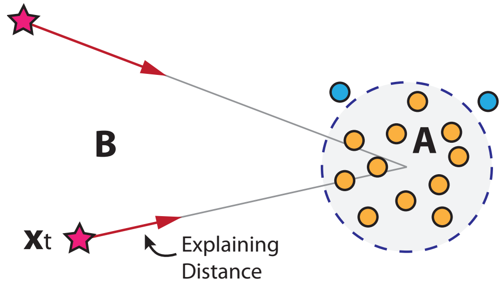
- Concept drift is a critical problem for ML/Security applications



- Contrastive Autoencoder is effective to detect concept drift



- Distance-based explanation is more suitable for explaining drifting samples



Thank you!

Homepage

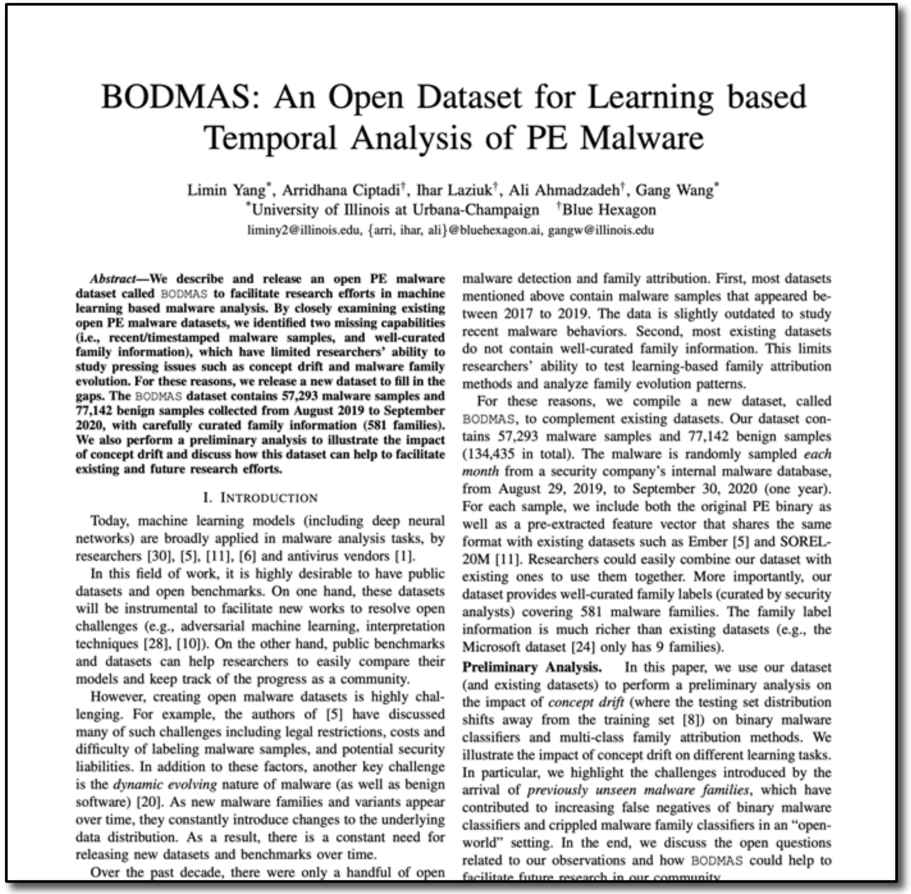
https://liminyang.web.Illinois.edu

Code, features, and supplemental materials available

https://github.com/whyisyoung/CADE

A new PE malware dataset [DLS'21]

https://whyisyoung.github.io/BODMAS/



BODMAS: An Open Dataset for Learning based Temporal Analysis of PE Malware

Limin Yang*, Arridhana Ciptadi†, Ihar Laziuk†, Ali Ahmadzadeh†, Gang Wang*
*University of Illinois at Urbana-Champaign †Blue Hexagon
liminy2@illinois.edu, {arri, ihar, ali}@bluehexagon.ai, gangw@illinois.edu

***Abstract*—We describe and release an open PE malware dataset called BODMAS to facilitate research efforts in machine learning based malware analysis. By closely examining existing open PE malware datasets, we identified two missing capabilities (i.e., recent/timestamped malware samples, and well-curated family information), which have limited researchers' ability to study pressing issues such as concept drift and malware family evolution. For these reasons, we release a new dataset to fill in the gaps. The BODMAS dataset contains 57,293 malware samples and 77,142 benign samples collected from August 2019 to September 2020, with carefully curated family information (581 families). We also perform a preliminary analysis to illustrate the impact of concept drift and discuss how this dataset can help to facilitate existing and future research efforts.**

I. Introduction

Today, machine learning models (including deep neural networks) are broadly applied in malware analysis tasks, by researchers [30], [5], [11], [6] and antivirus vendors [1].

In this field of work, it is highly desirable to have public datasets and open benchmarks. On one hand, these datasets will be instrumental to facilitate new works to resolve open challenges (e.g., adversarial machine learning, interpretation techniques [28], [10]). On the other hand, public benchmarks and datasets can help researchers to easily compare their models and keep track of the progress as a community.

However, creating open malware datasets is highly challenging. For example, the authors of [5] have discussed many of such challenges including legal restrictions, costs and difficulty of labeling malware samples, and potential security liabilities. In addition to these factors, another key challenge is the *dynamic evolving* nature of malware (as well as benign software) [20]. As new malware families and variants appear over time, they constantly introduce changes to the underlying data distribution. As a result, there is a constant need for releasing new datasets and benchmarks over time.

Over the past decade, there were only a handful of open

malware detection and family attribution. First, most datasets mentioned above contain malware samples that appeared between 2017 to 2019. The data is slightly outdated to study recent malware behaviors. Second, most existing datasets do not contain well-curated family information. This limits researchers' ability to test learning-based family attribution methods and analyze family evolution patterns.

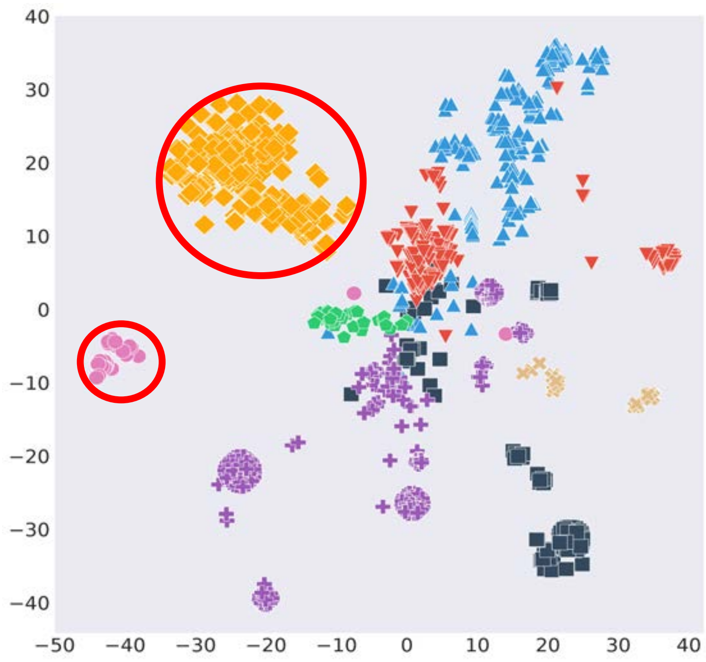
For these reasons, we compile a new dataset, called BODMAS, to complement existing datasets. Our dataset contains 57,293 malware samples and 77,142 benign samples (134,435 in total). The malware is randomly sampled *each month* from a security company's internal malware database, from August 29, 2019, to September 30, 2020 (one year). For each sample, we include both the original PE binary as well as a pre-extracted feature vector that shares the same format with existing datasets such as Ember [5] and SOREL-20M [11]. Researchers could easily combine our dataset with existing ones to use them together. More importantly, our dataset provides well-curated family labels (curated by security analysts) covering 581 malware families. The family label information is much richer than existing datasets (e.g., the Microsoft dataset [24] only has 9 families).

Preliminary Analysis. In this paper, we use our dataset (and existing datasets) to perform a preliminary analysis on the impact of *concept drift* (where the testing set distribution shifts away from the training set [8]) on binary malware classifiers and multi-class family attribution methods. We illustrate the impact of concept drift on different learning tasks. In particular, we highlight the challenges introduced by the arrival of *previously unseen malware families*, which have contributed to increasing false negatives of binary malware classifiers and crippled malware family classifiers in an "open-world" setting. In the end, we discuss the open questions related to our observations and how BODMAS could help to facilitate future research in our community.

Backup Slides

Drift Detection in the Latent Space

- Drift detection: if a sample is far away from ANY existing families' centroid, it's a potential drifting sample
- But different families' tightness vary, how to set distance thresholds?
- MAD (Median Absolute Deviation) [1]
  - Median of the median distance to the centroid
  - $MAD = b * median(|X_i - median(X)|)$
  - X is a set of distances to the centroid
  - Any new data outside $median(X) \pm A * MAD$ → outlier
- Rank drifting samples for investigation



[1] Detecting outliers: Do not use standard deviation around the mean, use absolute deviation around the median, Journal of Experimental Social Psychology, 2013

Drift Detection: Evaluation Metrics

- $\text{Precision} = \frac{\text{detected unseen family samples}}{\text{inspected samples}}$
- $\text{Recall} = \frac{\text{detected unseen family samples}}{\text{total \# of unseen family samples}}$
- $F_1 = \frac{2 * \text{Precision} * \text{Recall}}{\text{Precision} + \text{Recall}}$
- Normalized Inspection effort = $\frac{\text{inspected samples}}{\text{total \# of unseen family samples}}$

Training set: A (200), B(200), C(200)
Testing set: A(50), B(50), C(50), **D(10)**

ID	Family	Precision	Recall
1	D	1/1 = 1	1/10 = 0.1
2	D	2/2 = 1	2/10 = 0.2
3	C	2/3 = 0.67	2/10 = 0.2
4	D	3/4 = 0.75	3/10 = 0.3
5	D	4/5 = 0.8	4/10 = 0.4
	…….		

A ranked list of detected samples

Real-world Test on Blue Hexagon PE Malware

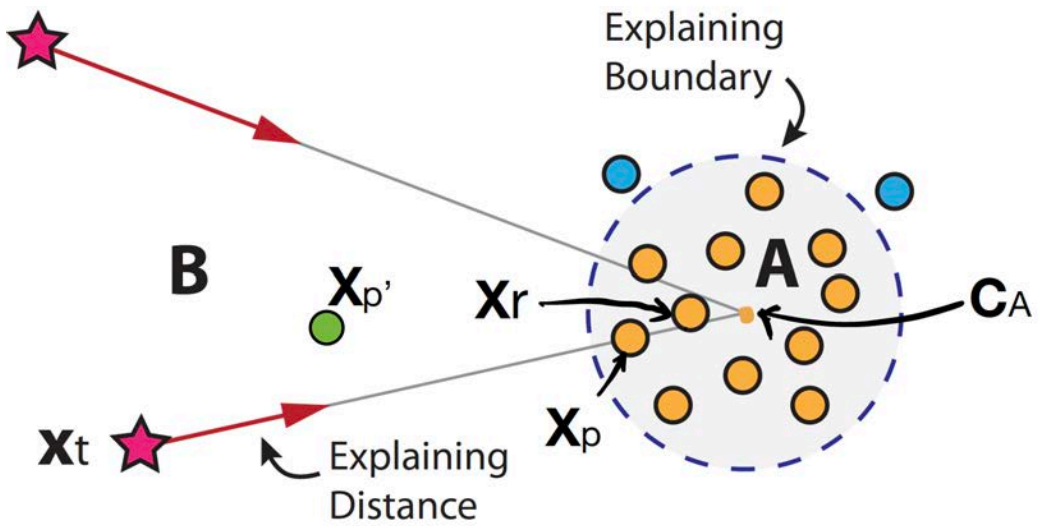
- 20,613 Windows PE malware, 395 families, Sept. 2019 – Feb. 2020
- 2,381 features, training with top N families Sept. 2019 – Jan. 2020
- Testing set: Feb. 2020
- CADE is still effective!

N (training families)	F_1	Norm. Effort	Detected Unseen Families
5	0.97	1.02	161/165
10	0.95	0.98	153/160
15	0.87	0.84	140/155

BLUEHEXAGON

Drift Explanation: Evaluation Metrics and Results

- Metric: the latent distance between a perturbed sample and its closest centroid.
- Using CADE to select important features, $\mathbf{x}_t \rightarrow \mathbf{x}_p$, while baseline methods may $\rightarrow \mathbf{x}_p{}'$, which is still far away from $\mathbf{C}_A$.



Method	Drebin-FakeDoc Avg±Std	IDS2018-Infiltration Avg±Std
Original distance	5.363±0.568	11.715±2.321
Random	5.422±1.773	11.546±3.169
Boundary-based	3.960±2.963	6.184±3.359
COIN [IJCAI'18]	6.219±3.962	8.921±2.234
CADE	0.065±0.035	2.349±3.238