Short Paper: Empirical Analysis of Evasion and Poisoning Against Malware Data Drift Detection

Mingyue Yang[0009−0002−7783−1091], David Lie[0000−0002−2000−6827], and Nicolas Papernot[0000−0001−5078−7233]

University of Toronto and Vector Institute
myshirley.yang95@gmail.com,{david.lie,nicolas.papernot}@utoronto.ca

Abstract. As concept drift due to malware evolution presents challenges for malware classification, machine learning-based data drift detection tools are developed to mitigate this problem. These data drift detector tools are designed for a different purpose and built with different techniques compared to malware classifiers. Although evasion and poisoning attacks against machine learning-based malware classifiers can cause misclassification of malware samples, it is not clear how these attacks work against data drift detectors and malware classifiers in combination. This work investigates the effect of evasion and poisoning attacks on the data drift detector along with the malware classifier. We demonstrate how unique characteristics of data drift detectors cause attacks against malware classifiers to work differently against them.

1 Introduction

Machine learning models can detect whether a piece of software is malware or not [2, 3, 12]. Machine learning-based malware classifiers have better generalizability than signature-based malware detection approaches and can catch previously unseen patterns. However, as both malware and benign software (benignware) evolve over time, machine learning-based malware classifiers can soon become outdated and no longer perform well on new inputs. To mitigate this problem, machine learning-based data drift detectors [4, 9, 19] have been proposed to find drifting malware samples (i.e. samples that drift away from the data distribution of the training set). These data drift detectors can be used to reject the predictions of samples that are likely misclassified [4] or to continuously train the malware classifier using detected drifting samples to overcome concept drift [6]. As these malware data drift detectors have different goals than malware classification, they have different loss terms, model architectures, and behaviours than the malware classifiers they work with [4, 9, 19].

As machine learning models are susceptible to adversarial attacks, many existing works study evasion and poisoning attacks against machine learning malware classifiers [5, 7, 11, 13, 17]. However, a successful attack for such system should work against ***both*** the malware classifier ***and*** the ***data drift detector***.

As it is unclear how malware data drift detectors, also built with machine learning models but have different designs and purposes, behave under evasion and poisoning attacks, our paper makes the following contributions:

1) We conduct evasion and poisoning attacks against the data drift detector along with the malware classifier. We study how factors such as the size of the perturbations and the number of poisoning samples affect data drift detectors with different designs.

2) We demonstrate properties of data drift detectors that cause the attacks to work differently against them. We illustrate cases for which attacks that work for the malware classifier do not work well for data drift detectors.

2 Background: Data Drift Detector

We attack two recent data drift detectors designed for malware as follows.

CADE [19]: The CADE data drift detector [19] trains a contrastive autoencoder (CAE) model that converts data from the input space to a lower dimensional embedding space. CAE has a ***contrastive loss*** term that pushes samples from different classes away from each other and keeps samples from the same class closer together in the embedding space. Similar to other autoencoders, CAE also has a ***reconstruction loss*** term that minimizes the difference between the reconstructed input from its decoder and the raw input to its encoder, so the CAE embedding space accurately represents details in the input.

To determine whether a test sample is a drifting sample of a given class, CADE computes an ***anomaly score*** based on the distance between the sample and the centroid of that class in its CAE embedding space. The ***OOD (out-of-distribution) score*** of a sample is the minimum of its anomaly scores for all classes. A sample with a high OOD score is a drifting sample for all classes.

Transcend and Transcendent [4]: Transcend [9] uses p-values from conformal prediction to determine whether a sample is a drifting sample of a class i. The p-value of a sample s for a given class i is the proportion of training samples in class i that are at least as dissimilar to other samples in class i as the given sample, with the similarity measured using ***a similarity metric***.

Transcend [4,6] builds a linear SVM model upon the embedding space of the CAE, and the distance to the hyperplane of the SVM decision boundary is used as the ***similarity metric***. For samples not predicted as class i by the SVM (i.e. samples on the other side of the SVM hyperplane), their distances to the hyperplane are negated, so these samples have negative distance measures.

Transcend [9] then computes a ***credibility score***, a ***confidence score***, and a ***cred+conf score*** using the p-value of a given sample. For a predicted class i, the ***credibility score*** of a sample is simply its p-value for class i. A high credibility score means this sample is more similar to other samples in class i. The ***confidence score*** for class i, however, is calculated as 1 minus the highest p-value for classes other than the class i. A high confidence score means this sample is not similar to classes other than class i. To represent the driftedness of a sample for class i, a ***cred+conf score*** is calculated by multiplying the credibility score by the confidence score. A non-drifting sample of class i would have a high cred+conf score (both high credibility score and confidence score), meaning it is similar to class i while not similar to other classes.

Our work uses Transcendent [4], an improved version of Transcend [9], which approximates and sometimes surpasses the performance of the initial Transcend work while requiring less computational overhead.

3 Threat Model

Datasets: We use the Androzoo [1] and Bodmas [18] datasets with malware and benignware spanning over several months. For Androzoo, all features are either 0 or 1, indicating whether a component is present in the sample. For Bodmas, different features have different ranges of numerical values. Features of both datasets are obtained from simple static analysis without running the samples.

Victim Models and Drifting Sample Selection: The victim starts with an initial malware classifier and a data drift detector trained using the ***initial training set*** in Table 1 with data balancing by randomly oversampling malware samples. The victim uses MLP (Multi-Layer Perceptron) for malware classification, and either CADE [19] or Transcendent [4] for data drift detection (See Section 2). The malware classifier predicts whether new samples from the ***future selection set*** (See Table 1) are malware or not.

Table 1: Bodmas and Androzoo Datasets, Without Data Balancing

	Bodmas		**Androzoo**	
	Time Range	**# of Malware : Benignware**	**Time Range**	**# of Malware : Benignware**
Initial Training Set	2007-01 to 2019-09	4741 : 17565	2019-01 to 2019-12	4542 : 40947
Future Selection Set	2019-11	2494 : 3718	2020-02	406 : 3697

As the malware classifier can make wrong predictions when samples change over time, the data drift detector filters drifting samples in the ***future selection set*** for human inspection and correction. Since it is costly for human to analyze samples, the victim can only manually investigate ***k samples*** from the ***future selection set*** (See Table 1). We assume the victim manually examines the k most drifted samples picked by the initial data drift detectors [6], as they are mostly likely wrongly predicted by the malware classifier. All malware among the k inspected samples are blocked. This means for a successful attack, a malware must be ***BOTH predicted as benign by the malware classifier AND not among the top drifted samples selected by the data drift detector***.

Our ***future selection set*** contains all samples two months after initial training (Table 1). For the poisoning attack, we pick this month (2019-11/2020-02) instead of simply the next month after initial training (2019-10/2020-01), as we assume it needs time from the next month to retrain the models. For comparable evalution results across all our evasion and poisoning attack experiments, we use the same future selection set for the evasion attack.

Retrain/Poison Victim Models: As time passes, the victim needs to update the malware classifier and data drift detector to combat concept drift. The new training set to retrain the malware classifier and data drift detector is formed by inserting the most drifted samples encountered by the existing data drift detector into the balanced *initial training set* [6]. To only study the effect of poisoning and rule out other factors, we assume only the most drifted poisoning samples

from the attacker are inserted to form the new training set for experimental purposes. For simplicity, we assume no further data balancing is done on the inserted samples and these updated models are trained from scratch.

We assume the victim conducts a rough manual triage on every inserted sample and correctly assigns every true label [17]. An attacker cannot insert any sample into the training set with a wrong label, and thus has to perform ***clean-label poisoning***. However, we assume the attacker can insert poisoning samples derived from benignware with benign functionality and labels [13], as the victim includes recent samples from third-party intelligence platforms for training. All users, including the attacker, can upload files to these platforms, and the victim cannot carefully examine every incoming training sample to notice subtle changes, due to the sheer volume of incoming data.

Attacker Knowledge: We assume the attacker does not have full access to the victim models. However, as information for well-known malware families and benign software is semipublic, the attacker randomly knows 10% of benignware and 10% of malware from the *initial training set* of the victim model (See Table 1). The attacker builds ***surrogate models*** according to their limited knowledge, and then attack the surrogate models they have full control over, hoping these attacks can transfer to the victim model.

Feature Restrictions to Keep Malware Functionality: For a practical attack, the feature values of a modifed sample cannot affect its functionality or file format. For example, adding unused space at the end of a Portable Executable (PE) file does not change its functionality, but changes its features and thus predictions from machine learning models. We assume the attacker can increase feature values but cannot decrease them, as it is easy to increase these features (make file larger, add extra API calls, etc.) but cannot decrease them without breaking file functionality. For Bodmas [18], only 17 out of 2,381 features are feasible for modification without affecting the functionality of the underlying binary file [13] or breaking the file format. We assume there is no such restriction for Androzoo [1], and all of its 16,978 features can be incremented to 1.

Perturbation Limit: To preserve malicious functionality and avoid being conspicuous, the attacker needs resources to tweak files, so their corresponding features are changed. With limited resources, the universal perturbation added to malware/benignware cannot be arbitrarily large.

With Bodmas [18], different features have different value ranges. All features in the initial training set are scaled between 0 and 1 with a min-max scaler. We set limits on the magnitude of the perturbation after scaling for its 17 modifiable features [13]. Modified feature values beyond the 0-to-1 limit are clipped.

With Androzoo [1], all features are either 0 or 1. As there are a large number of features (16,978 features) and the feature space is sparse, for better attack performance, we find significant features in Androzoo using Recursive Feature Elimination (RFE) with logistic regression [13] and data only known by the attacker rather than the entire training set. We ensure only the selected significant features are modified. Depending on the number of modifiable features, we use the 200 and 1000 most significant features for a smaller and larger feature limit.

Attacker's Goals: The attacker aims to attack both malware classification and data drift detection.

1) A malware sample crafted by the attacker should be classified as benignware by the malware classifier.

2) The malware sample should not be among the top-k drifting samples picked by the data drift detector for human inspection.

The attacker aims to compute a ***universal perturbation***, which can be generally applied to all malware samples to cause misclassification (See Section 4.2).

4 Attack Methods

4.1 Different Surrogates & Loss for Perturbation

We evaluate perturbations generated from ***either*** a surrogate MLP model ***or*** a surrogate CAE model, and apply the perturbations to malware samples.

Option 1: With a ***surrogate MLP*** model, we investigate whether a perturbation u that moves malware samples across the decision boundary can simply make them a non-drifted member of the benign class.

$$\arg\min_{u} cross\_entropy(MLP(X_{malware} + u), y_{benign}) \tag{1}$$

Option 2: With a ***surrogate CAE*** model, the perturbation u moves samples towards the benign centroid in the CAE embedding space:

$$\arg\min_{u} MSE(CAE\_encoder(X_{malware} + u), benign\_centroid) \tag{2}$$

To compute the perturbations, we use the rectified Adam optimizer and apply the *perturbation limit* from Section 3. To allow comparison, ***the same perturbation*** is used for both evasion and poisoning.

4.2 Evasion and Poisoning Attacks

Scenario 1) Evasion: For evasion, our work focuses on ***universal perturbation***, as it is similar to the backdoor trigger in the sense that they are both generated from a large set of samples owned by the attacker and can be applied to any malware sample to cause misclassification. This is different from a per-sample perturbation that aims to allow misclassification when added to only one specific sample but not other samples. While we also performed evaluation on per-sample perturbation, it has similar results as the universal perturbation.

Scenario 2) Poisoning: We conduct ***clean-label poisoning***, in which the attackers cannot change the correct labels for the injected training data (restriction from Section 3). The attack is also ***backdoor poisoning***, during which the attacker generates a ***backdoor trigger*** (same as the universal perturbation for evasion) and adds it to benign samples to form poisoning samples. The poisoned victim model should still have a generally good performance, but should misclassify the attacker's samples only when the backdoor trigger is added.

5 Evaluation

We evaluate the attacks in Section 4 with attacker capabilities, restrictions, and data in Section 3. For generalizable attacks, when calculating the attack success rate (ASR) for testing, we apply the universal perturbation/backdoor trigger to ***all malware samples*** in the *initial training set* (Table 1) instead of only the malware samples the attacker knows. To eliminate randomness, we perform the attack for 30 runs and average the results across all runs.

5.1 Evaluation Metrics

We compute an overall attack success rate $ASR_{overall}$ as our general evaluation metric, which is the proportion of malware samples that achieve attack success against ***both malware classification and data drift detection***.

To attack against ***malware classification***, a crafted malware sample m is successful if it is misclassified as benignware. To evaluate the success of this subtask, we compute $ASR_{malware\_classification}$, which is simply the proportion of malware samples successfully misclassified as benignware.

To attack against ***data drift detection***, a successful malware sample should not be among the K most drifted samples picked by the data drift detector. Most of our experiments use K=500 for comparable results.

As discussed in Section 2, CADE picks top-drifting samples with the ***highest*** OOD scores for all classes. Transcendent picks top-drifting samples with the ***lowest*** cred+conf score of the predicted class rather than the benign class.

5.2 Evasion

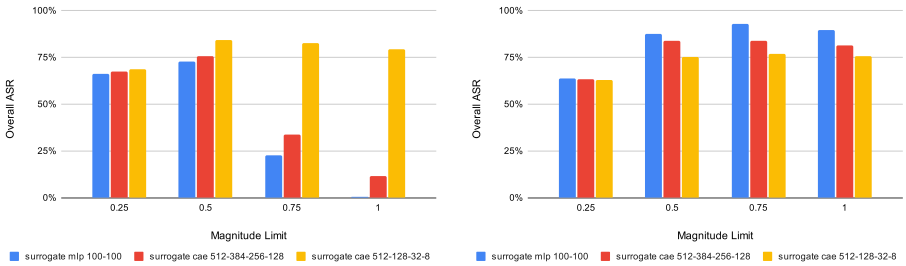
Evading CADE + Malware Classification: Unlike evasion against malware classification only (See Appendix), when evading both the CADE data drift detector and malware classifier in combination, the overall ASR can ***decrease*** as an already large perturbation continues to increase, as shown in Figure 1a.

This is because the large perturbations result in large features in crafted samples. The CAE models treat these large features as unseen patterns different from other samples in the training set and push these samples with large features away from the benign centroid. CADE with the underlying CAE models thus can mark these samples as drifting, resulting in a decreasing overall ASR.

As a large perturbation keeps increasing, the ASR from the ***surrogate MLP*** decreases faster than the ASR from the ***surrogate CAE*** models. This is because the cross-entropy loss for surrogate MLP does not take into account the distance towards the benign centroid in CAE, and therefore, perturbations generated from surrogate MLP prefer a region with higher confidence to be benign for MLP, but this region can be further away from the benign centroid of the CAE.

Evading Transcendent + Malware Classification: Unlike CADE with CAE, for Transcendent + CAE, large perturbations generated with the cross-entropy loss and the ***surrogate MLP*** can achieve higher overall ASRs than the ones

with ***surrogate CAE*** models, as demonstrated in Figure 1b. This is because Transcendent builds an SVM on the CAE embedding space and uses the distance to the SVM decision boundary to detect drifting samples. A drifting sample in CADE far away from the benign centroid can also have a large distance to the decision boundary and thus be non-drifting in Transcendent.



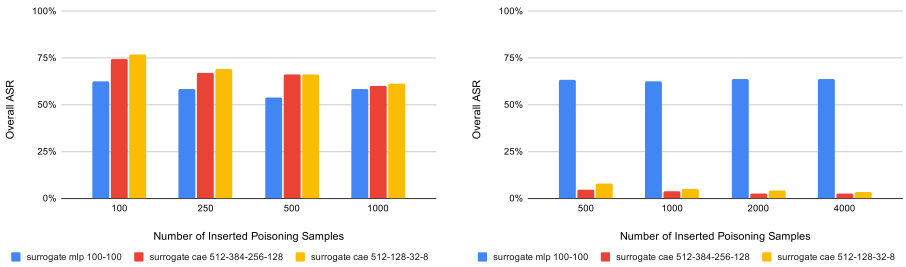
(a) Evasion Against Malware Classification + CADE

(b) Evasion Against Malware Classification + Transcendent

Fig. 1: $ASR_{overall}$, Evasion, Victim CAE 512-128-32-8, Bodmas, Top K = 500

5.3 Clean-Label Backdoor Poisoning

Results from poisoning data drift detectors are also different than poisoning the malware classifier only. For both datasets, Figure 2 shows there can be a ***decrease*** in the overall ASR with more poisoning samples when poisoning CADE. See Appendix for similar effects when poisoning Transcendent + CAE.



(a) Top K = 500, Bodmas, Magnitude Limit = 1

(b) Top K = 500, Androzoo, 500 Features (Sig Feat 1000)

Fig. 2: $ASR_{overall}$, Backdoor Poisoning Against Malware Classification and Data Drift Detection, Poisoned CADE 512-128-32-8

This is due to the reconstruction loss term of the victim CAE models. As more benign poisoning samples are inserted, the CAE models need to differentiate between the different poisoning samples in the embedding space to reconstruct them accurately. The limited embedding space close to the benign centroid is

not enough to reconstruct different poisoning samples properly. The victim CAE models thus learn to put samples with the backdoor trigger into areas further away from the benign centroid, resulting in a decreasing overall ASR.

This effect is especially obvious for the Androzoo dataset with more samples and input features, as the small capacity of the embedding space is less likely to be enough for the rich information in its input space. In such case as illustrated by Figure 2b, poisoning with ***surrogate CAEs*** has much lower ASRs than ***surrogate MLPs***, as the loss terms for CAE deliberately move poisoning samples towards the already crowded space around the benign centroid.

6 Related Work

While existing works evade malware classifiers with manually crafted features extracted by static analysis [5, 14] or raw binaries [7, 10, 11], and poison malware classifiers with static features [13, 17] using backdoor triggers, as of our knowledge, no existing work attacks malware data drift detectors as we do.

Similar to our experiment settings, [13, 15, 17] assume the attacker does not have full knowledge of the victim model and trains surrogate models with the same features as the victim model but a smaller training set and/or different model architecture. To preserve malware functionality during attacks, existing works also modify static features [7, 13], add bytecode gadgets from benign Android apps [17], and manipulate binary instructions [11]. Similar to our experiment results, [7, 10, 11, 13, 17] can also achieve high attack success rates for both evasion and poisoning. [8, 16] investigate the underlying reasons for such success.

7 Conclusion

Data drift detectors with underlying CAE models can have complex interactions with attack parameters. Larger perturbations for evasion and more poisoning samples, which are generally more effective against MLP malware classifiers, can inhibit attack success against CAE-based data drift detectors. The attacker's surrogate model and loss term need to be carefully chosen to align with the properties of the CAE embedding space under different scenarios.

8 Acknowledgements

Funding for this work was provided in part by NSERC Discovery Grant RGPIN-2026-07548 and NSERC-CSE Research Communities Grant ALLRP 588144-23. David Lie is supported by a Tier 1 Canada Research Chair CRC-2019-00242. Mingyue Yang has received support through Ontario Graduate Scholarships and Edward S. Rogers Sr. Graduate Scholarships. Researchers funded through the NSERC-CSE Research Communities Grants do not represent the Communications Security Establishment Canada or the Government of Canada. Any research, opinions or positions they produce as part of this initiative do not represent the official views of the Government of Canada.

References

1. Allix, K., Bissyandé, T.F., Klein, J., Le Traon, Y.: Androzoo: Collecting millions of android apps for the research community. In: Proceedings of the 13th international conference on mining software repositories. pp. 468–471 (2016)
2. Anderson, H.S., Roth, P.: Ember: an open dataset for training static pe malware machine learning models. arXiv preprint arXiv:1804.04637 (2018)
3. Arp, D., Spreitzenbarth, M., Hubner, M., Gascon, H., Rieck, K., Siemens, C.: Drebin: Effective and explainable detection of android malware in your pocket. In: Ndss. vol. 14, pp. 23–26 (2014)
4. Barbero, F., Pendlebury, F., Pierazzi, F., Cavallaro, L.: Transcending transcend: Revisiting malware classification in the presence of concept drift. In: 2022 IEEE Symposium on Security and Privacy (SP). pp. 805–823. IEEE (2022)
5. Boutsikas, J., Eren, M.E., Varga, C., Raff, E., Matuszek, C., Nicholas, C.: Evading malware classifiers via monte carlo mutant feature discovery. arXiv preprint arXiv:2106.07860 (2021)
6. Chen, Y., Ding, Z., Wagner, D.: Continuous learning for android malware detection. In: 32nd USENIX Security Symposium (USENIX Security 23). pp. 1127–1144 (2023)
7. Demetrio, L., Biggio, B., Lagorio, G., Armando, A., Roli, F.: Adversarial exemples: Functionality-preserving optimization of adversarial windows malware. In: ICML 2021 Workshop on Adversarial Machine Learning (2021)
8. Demetrio, L., Biggio, B., Lagorio, G., Roli, F., Armando, A.: Explaining vulnerabilities of deep learning to adversarial malware binaries. arXiv preprint arXiv:1901.03583 (2019)
9. Jordaney, R., Sharad, K., Dash, S.K., Wang, Z., Papini, D., Nouretdinov, I., Cavallaro, L.: Transcend: Detecting concept drift in malware classification models. In: 26th USENIX security symposium (USENIX security 17). pp. 625–642 (2017)
10. Kreuk, F., Barak, A., Aviv-Reuven, S., Baruch, M., Pinkas, B., Keshet, J.: Adversarial examples on discrete sequences for beating whole-binary malware detection. arXiv preprint arXiv:1802.04528 pp. 490–510 (2018)
11. Lucas, K., Sharif, M., Bauer, L., Reiter, M.K., Shintre, S.: Malware makeover: Breaking ml-based static analysis by modifying executable bytes. In: Proceedings of the 2021 ACM Asia Conference on Computer and Communications Security. pp. 744–758 (2021)
12. Raff, E., Barker, J., Sylvester, J., Brandon, R., Catanzaro, B., Nicholas, C.K.: Malware detection by eating a whole exe. In: Workshops at the thirty-second AAAI conference on artificial intelligence (2018)
13. Severi, G., Meyer, J., Coull, S., Oprea, A.: {Explanation-Guided} backdoor poisoning attacks against malware classifiers. In: 30th USENIX security symposium (USENIX security 21). pp. 1487–1504 (2021)
14. Song, W., Li, X., Afroz, S., Garg, D., Kuznetsov, D., Yin, H.: Automatic generation of adversarial examples for interpreting malware classifiers. arXiv preprint arXiv:2003.03100 (2020)
15. Suciu, O., Coull, S.E., Johns, J.: Exploring adversarial examples in malware detection. In: 2019 IEEE Security and Privacy Workshops (SPW). pp. 8–14. IEEE (2019)
16. Suciu, O., Marginean, R., Kaya, Y., Daume III, H., Dumitras, T.: When does machine learning {FAIL}? generalized transferability for evasion and poisoning attacks. In: 27th USENIX Security Symposium (USENIX Security 18). pp. 1299–1316 (2018)

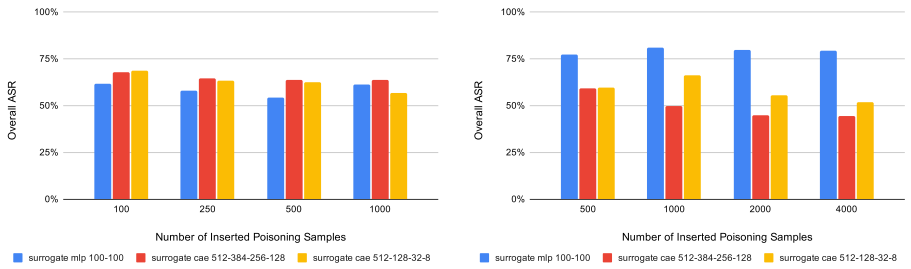
17. Yang, L., Chen, Z., Cortellazzi, J., Pendlebury, F., Tu, K., Pierazzi, F., Cavallaro, L., Wang, G.: Jigsaw puzzle: Selective backdoor attack to subvert malware classifiers. In: 2023 IEEE Symposium on Security and Privacy (SP). pp. 719–736. IEEE (2023)
18. Yang, L., Ciptadi, A., Laziuk, I., Ahmadzadeh, A., Wang, G.: Bodmas: An open dataset for learning based temporal analysis of pe malware. In: 2021 IEEE Security and Privacy Workshops (SPW). pp. 78–84. IEEE (2021)
19. Yang, L., Guo, W., Hao, Q., Ciptadi, A., Ahmadzadeh, A., Xing, X., Wang, G.: {CADE}: Detecting and explaining concept drift samples for security applications. In: 30th USENIX Security Symposium (USENIX Security 21). pp. 2327–2344 (2021)

A Appendix A

Note that a full version of this paper with all graphs and data is available at `https://arxiv.org/pdf/2608.03642` on arXiv.

Table 2: Evasion with Universal Perturbation Against MLP Malware Classification

Dataset	**Perturbation Limit**	**Mean $ASR_{malware\_classification}$ (Std)**		
		Surrogate Model Structure		
		MLP 100-100	**CAE 512-384-256-128**	**CAE 512-128-32-8**
Bodmas	0.25	77.7% (3.2%)	79.1% (5.1%)	78.5% (6.3%)
	0.5	97.6% (1.9%)	97.4% (3.8%)	92.5% (6.0%)
	0.75	100% (0)	99.985% (0.07%)	95.4% (5.8%)
	1	100% (0)	99.997% (0.02%)	95.6% (5.6%)
Androzoo	50 (Sig Feat 200)	72.1% (2.8%)	71.2% (2.3%)	72.2% (2.7%)
	100 (Sig Feat 200)	74.3% (2.9%)	71.6% (2.5%)	73.6% (2.7%)
	200 (Sig Feat 1000)	98.3% (0.7%)	95.0% (1.8%)	95.3% (2.0%)
	500 (Sig Feat 1000)	98.9% (0.4%)	94.9% (1.9%)	95.4% (1.9%)



(a) Top K = 500, Bodmas, Magnitude Limit = 1

(b) ***Top K = 200***, Androzoo, 500 Features (Sig Feat 1000)

Fig. 3: $ASR_{overall}$, Backdoor Poisoning Against Malware Classification and Data Drift Detection, Poisoned Transcendent CAE 512-128-32-8