MADCAT: Combating Malware Detection Under Concept Drift with Test-Time Adaptation

Eunjin Roh [1] **Yigitcan Kaya** [2] **Christopher Kruegel** [2] **Giovanni Vigna** [2] **Sanghyun Hong** [1]

Abstract

We present MADCAT (**MA**lware **D**etection under **C**oncept Drift through **A**daptation during **T**est-Time), a self-supervised approach designed to address the concept drift problem in malware detection. MADCAT employs an encoder-decoder architecture and works by test-time training of the encoder on a small, balanced subset of the test-time data using a self-supervised objective. During test-time training, the model learns features that are useful for detecting both previously seen (old) data and newly arriving samples. We demonstrate the effectiveness of MADCAT in continuous Android malware detection settings. MADCAT consistently outperforms baselines in detection performance at test-time. We also show the synergy between MADCAT and prior approaches in addressing concept drift in malware detection.

1 Introduction

Malware is *any* program that, when downloaded and executed on a victim's machine, exhibits malicious behavior—such as privilege escalation, data exfiltration, or backdoor injection (Sikorski & Honig, 2012; King et al., 2021; Severi et al., 2021). In recent years, the volume of malware threats has increased at an unprecedented scale, far exceeding what analysts and security experts can address manually. As a result, machine learning (ML) has emerged as a promising and scalable solution for combating this ever-evolving threat landscape (Sahin & Bahtiyar, 2021; Mohamad Arif et al., 2021; Chaulagain et al., 2020; Yuan et al., 2020; Nataraj et al., 2011; Arp et al., 2014). ML-based malware detectors are trained on large collections of malicious and benign samples, and then deployed in the wild to automatically determine whether a suspicious program is malware or not.

[1]Oregon State University, Corvallis, OR USA [2]University of California, Santa Barbara, CA USA. Correspondence to: Eunjin Roh <rohe@oregonstate.edu>.

Published at ICML 2025 Workshop on Test-Time Adaptation: Putting Updates to the Test! (PUT), Vancouver, Canada. PMLR 267, 2025. Copyright 2025 by the author(s).

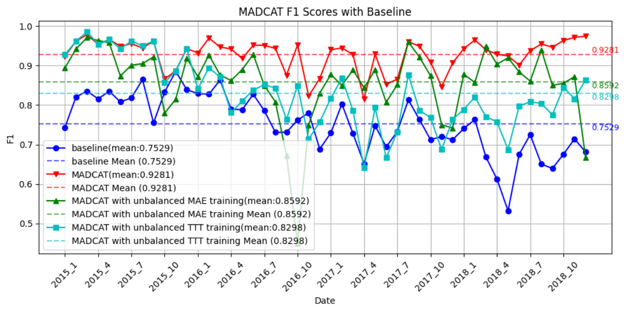

Figure 1: **MADCAT Performance.** MADCAT consistently outperforms the baselines across all evaluation periods, addressing the concept drift in ML-based malware detection.

However, this emerging paradigm has faced resistance due to a real-world challenge: *concept drift* (Schlimmer & Granger, 1986)—the phenomenon where detectors, once deployed in the wild, gradually become ineffective as the characteristics of malware evolve over time (Chen et al., 2023a;b). Most prior work addresses this issue with *supervised learning* techniques that allow models to learn *new* features useful for detecting emerging malware variants. This is typically achieved through re-training or fine-tuning detection models on newly collected and labeled datasets (Molina-Coronado et al., 2023; Chen et al., 2023a; Li et al., 2025). While shown to be effective, these approaches rely on the availability of high-quality labels for incoming future data—a requirement that is often difficult to satisfy in real-world scenarios (Wu et al., 2023; Zhu et al., 2020; Joyce et al., 2023).

Contributions. In this paper, we address the concept drift problem through an orthogonal approach: *self-supervised learning*. Instead of relying on detection models to extract useful features from high-quality labeled data, our method reduces the dependency on such supervision. Instead, it focuses on learning robust representations that preserve generalization performance over time (Hendrycks et al., 2019), even as malware evolves. Importantly, our approach is complementary to supervised learning techniques—meaning it can be combined with them to achieve synergistic benefits when a small set of labeled future data is available.

To demonstrate the potential, we present MADCAT, a self-supervised malware detector that performs test-time adapta-

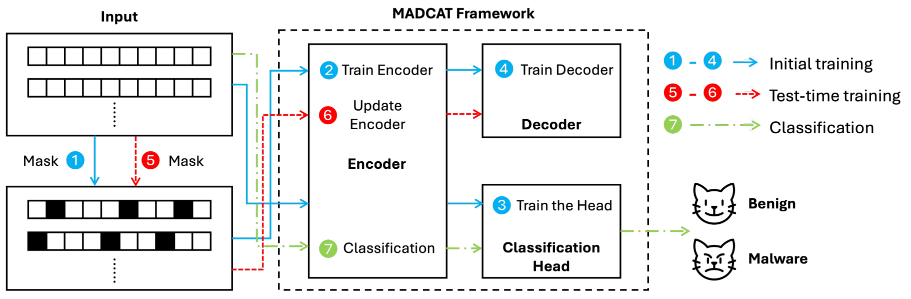

Figure 2: **MADCAT Workflow.**

tion on future malware data using the masked autoencoder (MAE) approach (Gandelsman et al., 2022). The MAE is first trained to reconstruct samples in which a random subset of the input has been masked. This form of self-supervision encourages the encoder to learn representations (or features) of both malware and benign samples that are robust to distributional shifts over time. At test-time, MADCAT performs fine-tuning of its encoder on future data.

We evaluate MADCAT on Android malware detection using a dataset consisting of malware and benign applications collected over a 7-year period. As summarized in Figure 1, our method maintains the performance over time and shows a better F1 score compared to the baseline approaches that rely on supervised learning. Our evaluation also highlights the MADCAT configurations that yield the best performance, such as balancing the test-time data used for fine-tuning the autoencoder. Moreover, we show that supervised learning can serve as a strong complement to our approach, showing performance synergy when combined with MADCAT. We hope our work draws the attention of the test-time adaptation community to malware detection and inspires future research that integrates self-supervision.

2 Background and Related Work

Concept drift refers to the gradual change in the statistical properties of data over time. It was first introduced by Schlimmer & Granger (1986), and has since been further studied in subsequent works aimed at identifying and mitigating its effects (Tsymbal, 2004; Gama et al., 2014; Moreno-Torres et al., 2012; Jordaney et al., 2017; Barbero et al., 2022; Chen et al., 2023b;a; Li et al., 2025). In malware detection, concept drift can significantly degrade model performance as their static structure—such as their functions and components—changes over time.

There has been progress in addressing concept drift in malware detection (Jordaney et al., 2017; Barbero et al., 2022). The most recent works employ *supervised learning*. Chen et al. (2023a) adopt active learning. They encourage the encoder to map similar samples to nearby embeddings and iteratively expand the training set by selecting the most uncertain samples. The detector is then retrained on this expanded dataset using ground-truth labels. Chen et al. (2023b) identified two types of concept drift—feature-space drift and data-space drift—and found that data-space drift has a substantial impact on detection performance. To address this, they employ online learning based on pseudo-labels. In contrast, MADCAT employs a self-supervised strategy that operates using pseudo-labels.

Test-time adaptation is the process of adjusting a pre-trained model to unlabeled target domain data at inference time. It has emerged as a promising solution to mitigate performance degradation caused by distributional shifts when the traditional model only focuses on a limited distribution for both training and testing (Liang et al., 2025).

Test-time training is a form of test-time adaptation that fine-tunes a model during inference. The concept dates back to Bottou & Vapnik (1992), but its first application to modern computer vision tasks was introduced by Sun et al. (2020), who proposed a self-supervised approach that updates the model using a single unlabeled test sample. Since then, several works have advanced test-time training techniques across visual and textual domains (Gandelsman et al., 2022; Wang et al., 2023; Hardt & Sun, 2024). However, few studies have explored test-time adaptation for addressing concept drift in malware detection.

Alam et al. (2024) is the most recent work to adopt the concept of test-time training for addressing concept drift in malware detection. They perform pseudo-labeling of unlabeled test samples and fine-tune the classifier using high-confidence pseudo-labeled data. However, as shown in §4.3, this approach remains ineffective under long-term concept drift (over a period of $\sim$2 years), and notably, the concept of self-supervision has not been leveraged.

MADCAT shows the effectiveness of using self-supervision in malware detection for addressing concept drift, while also forming a synergistic combination with pseudo-labeling.

3 MADCAT

Figure 2 shows the overall workflow of MADCAT. The detector has two components: MAE for self-supervised test-time training and a classification head for malware detection.

3.1 Initial (Training-time) Training

MADCAT first performs an initial training. During this phase, a portion of the input features is randomly masked based on a predefined masking ratio (0.0–0.9). The encoder is trained to capture meaningful representations from the partially masked input, while the decoder learns to reconstruct the masked features. Once the MAE is trained, the encoder is frozen. A separate classification head (detector) is then trained on the unmasked input data, using the representations produced by the frozen encoder.

3.2 Test-time Adaptation with Self-supervision

Once deployed, MADCAT performs test-time adaptation to combat concept drift. Because obtaining high-quality, human-annotated labels for newly emerging malware at test time is often impractical (Wu et al., 2023; Zhu et al., 2020; Joyce et al., 2023), we adopt a self-supervised test-time training approach proposed by He et al. (2022). At test time, each new input sample is partially and randomly masked and passed through the MAE. The encoder is then fine-tuned by minimizing the reconstruction loss. After this adaptation step, the updated encoder is paired with the pre-trained classification head to perform malware detection. This approach enables MADCAT to adapt to distributional shifts in the data without requiring labeled test-time samples.

3.3 Handling Class Imbalance with Pseudo-Labeling

Prior work used highly imbalanced datasets, with benign samples significantly outnumbering malware (Pendlebury et al., 2019). This can lead the MAE to learn representations of benign data, potentially limiting its ability to generalize well to malware over time. To address this class imbalance, we incorporate pseudo-labeling into MADCAT. These pseudo-labels are generated by the base detection model, and the encoder is updated during test-time on a rebalanced dataset prior to classification. This strategy helps the model maintain strong performance even under skewed data distributions, without relying on any human-provided labels.

4 Evaluation

4.1 Experimental Setup

Datasets. We use APIGraph (Zhang et al., 2020), which contains Drebin (Arp et al., 2014) features extracted from Android APKs collected during 7 years (2012–2018). Data from 2012–2014 is used for the initial training of the MAE and the classification head. We split the dataset into 80% for training and 20% for validation.

For test-time training, we use the data collected from 2015–2018. To analyze the performance over time, we divided this data by month. Each monthly dataset was further split into 70% for test-time training and 30% for validation. These splits are separate from the initial training set and relatively small due to the fine-grained temporal division.

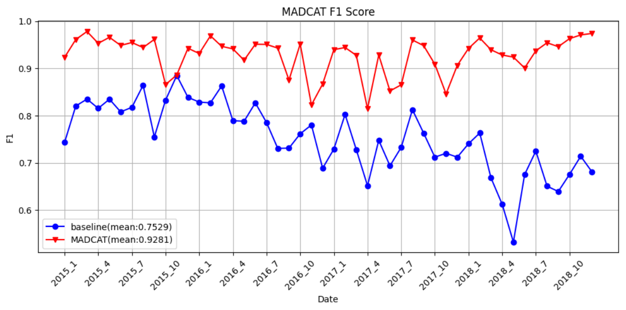

Figure 3: **MADCAT Performance.** It maintains consistent F1 scores, while the baseline shows gradual degradation.

Since the benign dataset is much larger than the malicious dataset (with a ratio of ∼9:1), we *randomly downsampled* the benign data to match the number of malicious ones.

Models. We utilize BinaryMLP, a classification model designed for malware detection in recent work (Chen et al., 2023a). We integrate the MAE (Gandelsman et al., 2022) into BinaryMLP. For training, we use a learning rate of 0.003 and 800 epochs for the initial training of the MAE-augmented BinaryMLP. Test-time training is performed for a single step per sample. We use cross-entropy loss for training both the MAE and the classification head.

Metrics. We employ two evaluation metrics: *F1 score* and *detection accuracy*. Our primary results are reported using F1 scores, with the mean F1 score across all monthly datasets provided in parentheses.

4.2 Effectiveness of MADCAT

Methodology. As a baseline, we trained the BinaryMLP model (Chen et al., 2023a) on the 2012–2014 data without the MAE module or test-time adaptation. This baseline model remains fixed after initial training and is not updated thereafter. The dataset for both initial training of MAE-augmented BinaryMLP and test-time training is balanced based on the ground truth labels. The default masking ratio of MAE is set to 0.3. All models are evaluated on the monthly-divided test dataset from 2015–2018.

Results. Figure 3 shows the performance of MADCAT over time. We report the F1 score for each month, and the detection accuracy over the entire period is shown in the legends. We first show that the baseline model exhibits a gradual decline in performance, indicating the presence of concept drift. In contrast, MADCAT maintains consistent performance, highlighting its robustness to concept drift in the data. Across all cases, MADCAT achieves higher F1 scores than the baseline, demonstrating its effectiveness for malware detection. We analyze the performance on benign and malicious data separately in Appendix B.1.

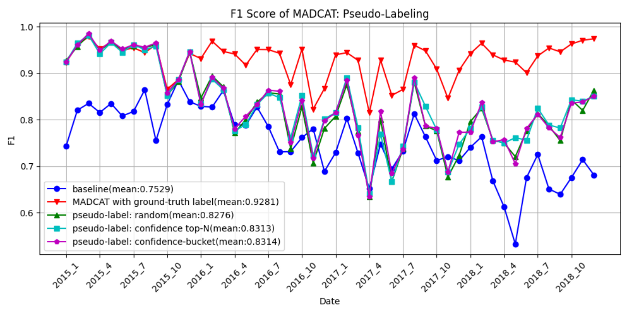

Figure 4: **MADCAT performance with pseudo-labeling.** All MADCATs achieve higher F1 scores vs. the baseline.

4.3 Synergy with Prior Approaches

Our previous results assume that ground-truth labels are available at test time to balance the dataset. While it may be feasible for a small subset of the test-time data, obtaining labels for the entire stream of future data is often impractical in real-world scenarios. Alam et al. (2024) address this issue using pseudo-labeling—where the base model is used to label incoming data, and it is continuously fine-tuned on the newly pseudo-labeled samples. We evaluate whether MADCAT can achieve synergy when combined with pseudo-labeling.

Methodology. Using the same experimental setup as before, we use the BinaryMLP model to generate pseudo-labels for each month's test-time data. However, when the pseudo-labeled data is highly imbalanced (e.g., dominated by one class) MADCAT's performance may degrade due to biased learning (Zheng et al., 2022). To address this, we additionally consider three label-balancing strategies:

- **Random:** Samples are randomly selected from each pseudo-labeled class to ensure an equal number of benign and malicious samples.
- **Confidence-based top-N:** All samples are sorted by the model's confidence scores within each pseudo-labeled class, and we select the top-N most confident samples.
- **Confidence-based bucket:** We divide the data into 10 buckets based on rounded confidence scores and select an equal number of samples from each bucket to maintain balance across varying confidence labels.

Results. Figure 4 shows our results. All MADCATs combined with pseudo-labeling achieve higher F1 scores compared to the baseline. However, the performance is slightly lower than that of MADCAT trained on a dataset balanced using ground-truth labels. This suggest that when the dataset collected for test-time training is unlabeled, pseudo-labeling using the base model can be an effective option.

4.4 Ablation Study

Dataset Balancing. We first evaluate the importance of dataset balancing. We test the individual impact of balancing during initial training and test-time training. For this analysis, we use the original unbalanced dataset, which has a benign-to-malicious sample ratio of 9:1. All other aspects of the experimental setup remain unchanged. The upper plot in Figure 5 shows the results of our analysis. We first observe that even when trained on an unbalanced dataset, MADCAT remains effective in addressing concept drift in malware detection. In both scenarios, MADCAT consistently outperforms the baseline on average. Among the two, balancing the dataset during initial training results in more stable performance over time.

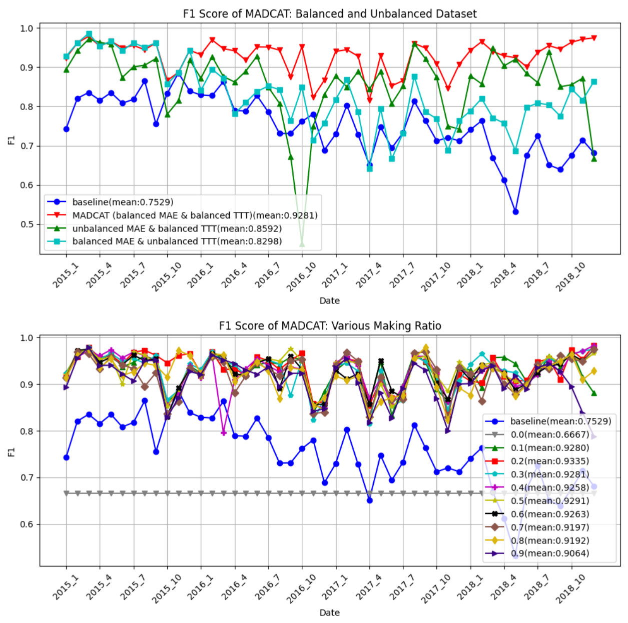

Figure 5: **Impact of database balancing (upper) and masking ratios (lower) on MADCAT performance.**

Masking Ratio for MAE Training. The lower plot in Figure 5 shows MADCAT's performance as the masking ratio for MAE training is varied from 0.0–0.9. We used the balanced dataset for all masking ratio setups. We first observe that without masking (ratio is 0.0), MAE fails to learn meaningful representations through the reconstruction process. In contrast, when masking is applied (ratios from 0.1–0.9), MADCAT consistently outperforms the baseline. Among these, masking ratios in the range of 0.1–0.6 yield the best results, achieving detection accuracy above 92%.

5 Conclusion

This work presents MADCAT, a novel approach that integrates self-supervised learning with test-time adaptation to maintain robust malware detection performance under concept drift. We demonstrate that MADCAT performs effectively in continuous Android malware detection. MADCAT requires only a small, balanced subset of data and does not rely on human-annotated labels for test-time adaptation.

Our approach is promising and opens up new avenues for future research: A more comprehensive evaluation of MADCAT—across diverse malware types (e.g., Windows PE files), various detection approaches, and alternative self-supervision techniques—will deepen our understanding of its effectiveness and generalizability.

Acknowledgment

E.R and S.H are partially supported by the Google Faculty Research Award 2023. Y.K is supported by the U.S. Intelligence Community Postdoctoral Fellowship. The findings and conclusions in this work are those of the author(s) and do not necessarily represent the views of the funding agency.

References

Alam, M. T., Fieblinger, R., Mahara, A., and Rastogi, N. Morph: Towards automated concept drift adaptation for malware detection, 2024. URL https://arxiv.org/abs/2401.12790. Accepted as a poster at NDSS 2024.

Arp, D., Spreitzenbarth, M., Hübner, M., Gascon, H., and Rieck, K. Drebin: Effective and explainable detection of android malware in your pocket. 02 2014. doi: 10.14722/ndss.2014.23247.

Barbero, F., Pendlebury, F., Pierazzi, F., and 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, 2022. doi: 10.1109/SP46214.2022.9833659.

Bottou, L. and Vapnik, V. Local Learning Algorithms. *Neural Computation*, 4(6):888–900, 1992.

Chaulagain, D., Poudel, P., Pathak, P., Roy, S., Caragea, D., Liu, G., and Ou, X. Hybrid analysis of android apps for security vetting using deep learning. In *2020 IEEE Conference on Communications and Network Security (CNS)*, pp. 1–9, 2020. doi: 10.1109/CNS48642.2020.9162341.

Chen, Y., Ding, Z., and Wagner, D. Continuous learning for android malware detection. In *Proceedings of the 32nd USENIX Conference on Security Symposium*, SEC '23, USA, 2023a. USENIX Association. ISBN 978-1-939133-37-3.

Chen, Z., Zhang, Z., Kan, Z., Yang, L., Cortellazzi, J., Pendlebury, F., Pierazzi, F., Cavallaro, L., and Wang, G. Is it overkill? analyzing feature-space concept drift in malware detectors. In *2023 IEEE Security and Privacy Workshops (SPW)*, pp. 21–28, 2023b. doi: 10.1109/SPW59333.2023.00007.

Gama, J. a., Žliobaitundefined, I., Bifet, A., Pechenizkiy, M., and Bouchachia, A. *ACM Comput. Surv.*, 46(4), March 2014. ISSN 0360-0300. doi: 10.1145/2523813. URL https://doi.org/10.1145/2523813.

Gandelsman, Y., Sun, Y., Chen, X., and Efros, A. A. Test-time training with masked autoencoders. In Oh, A. H., Agarwal, A., Belgrave, D., and Cho, K. (eds.), *Advances in Neural Information Processing Systems*, 2022. URL https://openreview.net/forum?id=SHMi1b7sjXk.

Hardt, M. and Sun, Y. Test-time training on nearest neighbors for large language models. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=CNL2bku4ra.

He, K., Chen, X., Xie, S., Li, Y., Dollár, P., and Girshick, R. Masked autoencoders are scalable vision learners. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 15979–15988, 2022. doi: 10.1109/CVPR52688.2022.01553.

Hendrycks, D., Mazeika, M., Kadavath, S., and Song, D. *Using self-supervised learning can improve model robustness and uncertainty*. Curran Associates Inc., Red Hook, NY, USA, 2019.

Jordaney, R., Sharad, K., Dash, S. K., Wang, Z., Papini, D., Nouretdinov, I., and Cavallaro, L. Transcend: detecting concept drift in malware classification models. In *Proceedings of the 26th USENIX Conference on Security Symposium*, SEC'17, pp. 625–642, USA, 2017. USENIX Association. ISBN 9781931971409.

Joyce, R. J., Amlani, D., Nicholas, C., and Raff, E. Motif: A malware reference dataset with ground truth family labels. *Comput. Secur.*, 124(C), January 2023. ISSN 0167-4048. doi: 10.1016/j.cose.2022.102921. URL https://doi.org/10.1016/j.cose.2022.102921.

King, J., Bendiab, G., Savage, N., and Shiaeles, S. Data exfiltration: Methods and detection countermeasures. In *2021 IEEE International Conference on Cyber Security and Resilience (CSR)*, pp. 442–447, 2021. doi: 10.1109/CSR51186.2021.9527962.

Li, A. S., Iyengar, A., Kundu, A., and Bertino, E. Revisiting concept drift in windows malware detection: Adaptation to real drifted malware with minimal samples. In *Proceedings 2025 Network and Distributed System Security Symposium*. In Network and Distributed System Security Symposium (NDSS) San Diego, CA, USA, 2025.

Liang, J., He, R., and Tan, T. A comprehensive survey on test-time adaptation under distribution shifts. *International Journal of Computer Vision*, 133(1): 31–64, 2025. ISSN 1573-1405. doi: 10.1007/

s11263-024-02181-w. URL https://doi.org/10.1007/s11263-024-02181-w.

Mohamad Arif, J., Ab Razak, M. F., Awang, S., Tuan Mat, S. R., Ismail, N. S. N., and Firdaus, A. A static analysis approach for android permission-based malware detection systems. *PLOS ONE*, 16(9):e0257968, 2021. doi: 10.1371/journal.pone.0257968.

Molina-Coronado, B., Mori, U., Mendiburu, A., and Miguel-Alonso, J. Efficient concept drift handling for batch android malware detection models. *Pervasive Mob. Comput.*, 96(C), December 2023. ISSN 1574-1192. doi: 10.1016/j.pmcj.2023.101849. URL https://doi.org/10.1016/j.pmcj.2023.101849.

Moreno-Torres, J. G., Raeder, T., Alaiz-Rodríguez, R., Chawla, N. V., and Herrera, F. A unifying view on dataset shift in classification. *Pattern Recognition*, 45(1):521–530, 2012. ISSN 0031-3203. doi: https://doi.org/10.1016/j.patcog.2011.06.019. URL https://www.sciencedirect.com/science/article/pii/S0031320311002901.

Nataraj, L., Karthikeyan, S., Jacob, G., and Manjunath, B. S. Malware images: visualization and automatic classification. In *Proceedings of the 8th International Symposium on Visualization for Cyber Security*, VizSec '11, New York, NY, USA, 2011. Association for Computing Machinery. ISBN 9781450306799. doi: 10.1145/2016904.2016908. URL https://doi.org/10.1145/2016904.2016908.

Pendlebury, F., Pierazzi, F., Jordaney, R., Kinder, J., and Cavallaro, L. {TESSERACT}: Eliminating experimental bias in malware classification across space and time. In *28th USENIX security symposium (USENIX Security 19)*, pp. 729–746, 2019.

Sahin, M. and Bahtiyar, S. A survey on malware detection with deep learning. In *13th International Conference on Security of Information and Networks*, SIN 2020, New York, NY, USA, 2021. Association for Computing Machinery. ISBN 9781450387514. doi: 10.1145/3433174.3433609. URL https://doi.org/10.1145/3433174.3433609.

Schlimmer, J. C. and Granger, R. H. Beyond incremental processing: tracking concept drift. In *Proceedings of the Fifth AAAI National Conference on Artificial Intelligence*, AAAI'86, pp. 502–507. AAAI Press, 1986.

Severi, G., Meyer, J., Coull, S., and Oprea, A. Explanation-Guided backdoor poisoning attacks against malware classifiers. In *30th USENIX Security Symposium (USENIX Security 21)*, pp. 1487–1504. USENIX Association, August 2021. ISBN 978-1-939133-24-3. URL https://www.usenix.org/conference/usenixsecurity21/presentation/severi.

Sikorski, M. and Honig, A. *Practical Malware Analysis: The Hands-On Guide to Dissecting Malicious Software*. No Starch Press, USA, 1st edition, 2012. ISBN 1593272901.

Sun, Y., Wang, X., Liu, Z., Miller, J., Efros, A. A., and Hardt, M. Test-time training with self-supervision for generalization under distribution shifts. In *Proceedings of the 37th International Conference on Machine Learning*, ICML'20. JMLR.org, 2020.

Tsymbal, A. The problem of concept drift: definitions and related work. 2004. URL https://api.semanticscholar.org/CorpusID:8335940.

Wang, R., Sun, Y., Tandon, A., Gandelsman, Y., Chen, X., Efros, A. A., and Wang, X. Test-time training on video streams. *JMLR*, 2023.

Wu, X., Guo, W., Yan, J., Coskun, B., and Xing, X. From grim reality to practical solution: Malware classification in real-world noise. In *2023 IEEE Symposium on Security and Privacy (SP)*, pp. 2602–2619, 2023. doi: 10.1109/SP46215.2023.10179453.

Yuan, B., Wang, J., Liu, D., Guo, W., Wu, P., and Bao, X. Byte-level malware classification based on markov images and deep learning. *Computers & Security*, 92:101740, 2020. ISSN 0167-4048. doi: https://doi.org/10.1016/j.cose.2020.101740. URL https://www.sciencedirect.com/science/article/pii/S0167404820300262.

Zhang, X., Zhang, Y., Zhong, M., Ding, D., Cao, Y., Zhang, Y., Zhang, M., and Yang, M. Enhancing state-of-the-art classifiers with api semantics to detect evolved android malware. In *Proceedings of the 2020 ACM SIGSAC conference on computer and communications security*, pp. 757–770, 2020.

Zheng, M., Wang, F., Hu, X., Miao, Y., Cao, H., and Tang, M. A method for analyzing the performance impact of imbalanced binary data on machine learning models. *Axioms*, 11(11):607, 2022. doi: 10.3390/axioms11110607. URL https://doi.org/10.3390/axioms11110607.

Zhu, S., Shi, J., Yang, L., Qin, B., Zhang, Z., Song, L., and Wang, G. Measuring and modeling the label dynamics of online anti-malware engines. In *Proceedings of the 29th USENIX Conference on Security Symposium*, SEC'20, USA, 2020. USENIX Association. ISBN 978-1-939133-17-5.

A Experimental Setup in Detail

Environment. All experiments were conducted using Python 3.8.5 and PyTorch 1.11.0 with CUDA 11.3 on a Rocky Linux 9.5. environment. We run our experiments on an internal cluster with an Intel(R) Xeon(R) Gold 6248R CPUs running at 3GHz with 48 cores and NVIDIA A40 GPUs with 48GB of VRAM. Our machine has 768GB of DDR4 RAM operating at 2933 MT/s.

Dataset. We used the APIGraph (Chen et al., 2023a) dataset throughout our evaluation. Each sample is represented by a binary feature vector of 1,159 dimensions, with 0 or 1 indicating the presence or absence of each feature. The dataset is labeled as either benign (0) or malicious (1). Table 1 shows the yearly distribution of samples.

Table 1: **Yearly distribution of the APIGraph dataset.** We use the same dataset as in Chen et al. (2023a).

Year	Malicious	Benign	Total
2012	3,061	27,472	30,533
2013	4,854	43,714	48,568
2014	5,809	52,676	58,485
2015	5,508	51,944	57,452
2016	5,324	50,712	56,036
2017	2,465	24,847	27,312
2018	3,783	38,146	41,929

B Additional Evaluation Restuls

B.1 MADCAT Accuracy per Class Labels

Figure 6 shows the class-wise accuracy of MADCAT on benign and malicious samples. Throughout the evaluation period, the accuracy on benign samples consistently exceeds that of the baseline. For malicious samples, MADCAT occasionally shows slightly lower accuracy than the baseline; however, its performance remains mostly comparable across all datasets.

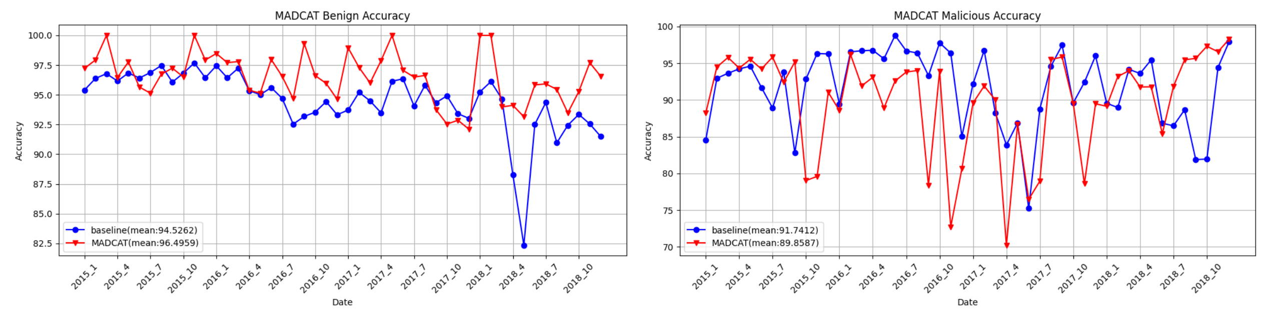

Figure 6: **MADCAT performance with baseline.** Benign accuracy consistently outperforms the baseline; malicious accuracy is sometimes lower but mostly comparable.

B.2 MADCAT Accuracy with Pseudo-Label

Figure 7 shows the class-wise accuracy of MADCAT when using pseudo-labeling for dataset balancing. Similar to the results with ground-truth-based balancing, pseudo-labeling improves the benign accuracy and yields comparable or slightly lower accuracy on malicious samples. Overall, pseudo-labeled MADCAT achieves results similar to those using ground-truth labels, indicating its robustness even under label-free conditions.

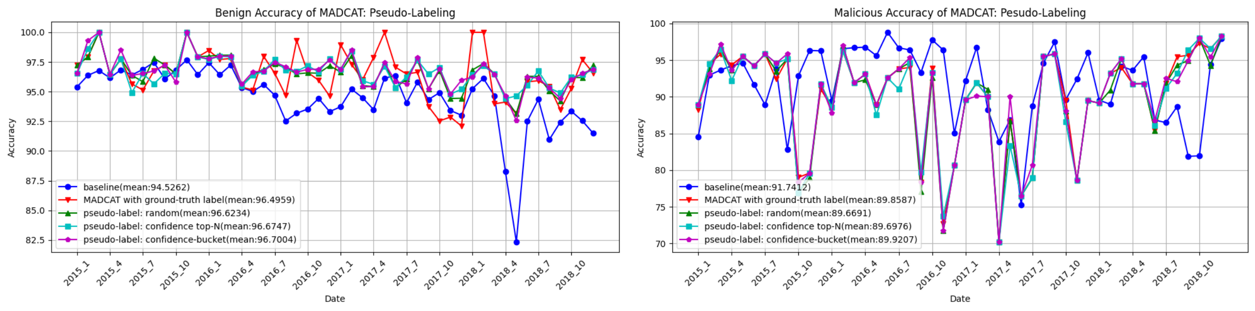

Figure 7: **Performance of MADCAT with different pseudo-labeling strategies.** All methods show similar or higher benign accuracy than the baseline; malicious accuracy is slightly lower but comparable.