Representation-Architecture Tradeoffs in Behavioral Malware Detection

Ameya Srivastava[0009-0002-7534-1734]*, Auritra Dey[0009-0006-6821-2183], Pavitra Kumar[0009-0006-8342-6155], and Devesh C Jinwala[0000-0003-4830-1702]

Department of Computer Science and Engineering
S.V. National Institute of Technology, Surat, India

Abstract. We present a controlled evaluation of representation-architectures for Windows PE API-based behavioral malware detection under a unified experimentation framework. We implement and compare sequential, graph, and attention-based models with their corresponding adversarially regularized variants using identical datasets, preprocessing pipelines, and evaluation protocols. Models are evaluated on both in-distribution and out-of-distribution (OOD) classification performance, robustness against adversarial evasion simulations, and computational efficiency. We find that sequential models remain efficient and achieve strong in-distribution performance, while OOD evaluation results show class-specific tradeoffs between malware recall and benign specificity across models. The effects of adversarial regularization are inconsistent and architecture-specific.

Keywords: behavioral malware detection · empirical evaluation · adversarial robustness · machine learning for cybersecurity

1 Introduction

One of the approaches used to develop behavioral malware detectors is to train machine learning (ML) models on artifacts extracted from the detonation of live malware in sandbox environments. Such artifacts typically include API calls, network analysis logs, and registry modification information [4]. Recent advances in ML have furthered this direction, allowing detectors to learn complex behavioral patterns from feature representations of the extracted data.

To improve detection performance, multiple combinations of behavioral artifact representations and their associated learning architectures, which we collectively refer to as representation-architectures, have been adopted. Underlying models differ both in design and complexity, and novel methods are generally evaluated through comparisons against models established in prior work, primarily using standard classification metrics like F1-score and accuracy. Only a limited subset of studies extend evaluation to alternate dimensions like efficiency or robustness.

Existing evaluation studies [9, 12] review the state-of-the-art, albeit primarily through predictive performance. Wider surveys utilize metrics reported in original works [3, 18] which individually use different datasets, preprocessing pipelines, and implementation

* Correspondence to srivastava.ameya@gmail.com

methods. Hence, the observed performance differences in such studies may reflect separate experimental design choices rather than the underlying method itself.

The aforementioned limitations motivate us to investigate and characterize the trade-offs of diverse representation-architectures for behavioral malware detection through a controlled evaluation on dimensions beyond standard classification performance.

2 Related Work

Behavioral malware detection has adopted a diverse range of representation-architectures for modeling API call sequences. Sequential approaches model the sequences directly using recurrent networks like Long Short-Term Memory (LSTM), simpler convolutional networks (CNN) [17], and attention-based transformers [21]. Alternative approaches encode execution behavior as graph representations, enabling graph models like deep-graph convolutional (DGCNN) [2] and graph-attention networks (GAT) [16].

Prior to our work, the TESSERACT [14] framework identified spatial and temporal sources of experimental bias to enable a more realistic evaluation of Android malware classifiers. Several surveys provide comprehensive overviews of ML-based malware analysis and detection methods [3, 18] but necessarily aggregate results reported across independent studies, and existing evaluation studies perform partially controlled comparisons across subsets of classifiers and datasets [9, 12], focusing primarily on standard classification performance.

In contrast, we perform a controlled evaluation of diverse representation-architectures and their corresponding adversarially regularized variants under a unified experimental framework. All models are trained and evaluated using identical data, preprocessing pipelines, and evaluation protocols. We extend evaluation beyond standard classification metrics to include robustness against adversarial evasion attack simulations, out-of-distribution generalization, and computational efficiency.

3 Evaluation Methodology

3.1 Datasets

All models are implemented and evaluated on samples from the Windows PE API-call sequences dataset introduced by Oliveira and Sassi (2019) [2], which contains 42,797 malware and 1,079 benign samples. Each sample in the dataset is identified by its MD5 hash and consists of the first 100 calls made by the parent process. All calls in the sample sequences are encoded into a vocabulary of 307 unique integers labeled `0-306`. We assemble an identically processed external corpus of 163 malware and 231 benign samples, not present in the aforementioned dataset, to evaluate models for out-of-distribution generalization. This corpus uses API call sequence data extracted from behavioral reports that were routinely downloaded and processed from a publicly accessible Cuckoo Sandbox deployment [7] throughout experimentation. Sample provenance was verified through behavioral analysis and detection reports by multiple security vendors via VirusTotal. The resultant corpus contains samples from 2009 to as recent as 2026, thereby providing a temporally diverse set for evaluation.

3.2 Threat Assumptions for Adversarial Evasion Simulations

We assume that the defender operates in a sandbox environment, detecting malware through ML-based classifiers prior to execution on a computer system, and that the adversary is aware of the detection pipeline and may engineer samples to evade detection through modifications that are designed to preserve malicious intent. Based on these assumptions, and consistent with prior work, we perform simulations of previously documented evasion tactics observed during malware analysis, designed to be compatible with each evaluated model. We simulate:

1. **Benign mimicry (B.M.)**, which replaces independent, non-critical calls with benign calls like `GetCursorPos` simulating user activity [15].
2. **Call reordering (C.O.)**, which swaps adjacent independent and non-critical API calls to modify local sequential patterns without altering malicious semantics [11].
3. **Stalling padding (S.P.)**, which prepends non-malicious or delay calls (such as `timeGetTime`) to stall actual malicious behavior execution and push relevant activity outside of the sandbox's observation time limit [1].

3.3 Implementation of Representation-Architectures

All models are implemented in PyTorch, trained using the Adam optimizer across five random seeds (10, 20, 30, 40, 50) and designed as competitive baselines. No model was disproportionately fine-tuned for the detection task. Models are purposefully trained on the naturally imbalanced dataset without resampling or class weighting. Architectures and hyperparameters were selected through manual experimentation, so results may be affected by selection bias. All implementation details are publicly available.[1]

Sequential Models utilize the sequences from the dataset directly using an initial embedding layer over the API call vocabulary: **(1) LSTM,** a two-layer stacked LSTM [6] that captures long-range temporal dependencies and is classified via the terminal hidden state; **(2) 1D-CNN,** a temporal convolutional network [10] that extracts local n-gram features before global adaptive max-pooling; and **(3) TFORM,** a two-layer Transformer encoder [19] that uses learnable positional embeddings and mean pooling to model contextual call interactions.

Graph Models use directed graphs where unique API calls from the sequences serve as nodes with one-hot identity features and consecutive calls define directed edges: **(1) DGCNN,** an isotropic two-layer graph convolutional network following Oliveira and Sassi [2] that uses unweighted edges to encode transition presence; and **(2) GAT,** an anisotropic two-layer graph attention network [20] with structurally masked attention over edge weights that encode transition frequencies.

To assess whether adversarial regularization provides consistent performance and robustness benefits, we follow the semi-supervised GAN (SGAN) framework [13] to implement adversarially regularized model variants. Each sequential and graph model serves as a discriminator with an additional logit designated for generated (fake) samples (discarded during inference) to implicitly regularize the supervised detection task through discrimination of real/generated samples. The respective generator projects

[1] Link: https://github.com/0ameyasr/DGvGAN

noise through a single-layer GRU [5] using Gumbel-Softmax [8] for differentiable sequence generation and uses feature-matching loss over the training distribution. The synthesized sequences are converted on-the-fly to the compatible graph formats when training graph discriminators.

4 Results and Discussion

4.1 Classification Performance Across Distribution Splits

Among sequential models, we find that the 1D-CNN achieves the strongest overall in-distribution (IID) discriminative performance, followed closely by its -SGAN variant. Graph models exhibit comparatively lower mean Macro-F1 scores (0.855) compared to sequential models (0.915) and higher false positive rates (FPR). While the LSTM-SGAN and graph-SGAN variants show improvements in each metric (IID), adversarial regularization produces no general benefits in detection performance.

Over out-of-distribution (OOD) samples, most baseline models exhibit higher malware recall than benign specificity. The same effect is particularly pronounced in graph-based networks, all of which have a mean AUC-ROC$_{\text{OOD}}$ less than 0.50. Sequential models and their respective -SGAN variants exhibit more varied metrics and generally higher variability across seeds. While the Transformer (TFORM) achieves the highest mean AUC-ROC$_{\text{OOD}}$, no representation-architecture is consistently superior across all metrics in both distribution splits. These results characterize performance on our corpus and should not be interpreted as a universal ranking of the evaluated methods.

Table 1. IID (over a hold-out evaluation split of 12,839 malware and 324 benign samples) and OOD (over an external corpus of 163 malware and 231 benign samples) performance aggregated across all 5 seeds, reported as mean±std. TPR and TNR respectively stand for true positive rate (malware recall) and true negative rate (benign specificity). Macro-F1 scores are excluded for brevity. For each model cell, the upper and lower entries correspond to the baseline and -SGAN variants, respectively.

Model	**AUC-ROC$_{\text{IID}}$**	**FPR$_{\text{IID}}$ (%)**	**AUC-ROC$_{\text{OOD}}$**	**TPR$_{\text{OOD}}$ (%)**	**TNR$_{\text{OOD}}$ (%)**
LSTM	0.9795 ± 0.0038	24.20 ± 1.34	0.5589 ± 0.0630	63.93 ± 12.55	43.38 ± 20.33
-SGAN	0.9816 ± 0.0046	23.27 ± 5.10	0.6519 ± 0.0624	49.45 ± 10.95	75.58 ± 3.69
1D-CNN	0.9938 ± 0.0024	14.88 ± 2.31	0.5267 ± 0.0416	62.70 ± 6.68	36.02 ± 11.72
-SGAN	0.9940 ± 0.0011	16.91 ± 1.63	0.5463 ± 0.0853	50.31 ± 12.56	58.53 ± 15.03
TFORM	0.9580 ± 0.0087	25.74 ± 3.64	0.7123 ± 0.0260	67.48 ± 4.00	69.00 ± 6.43
-SGAN	0.9662 ± 0.0081	29.94 ± 3.20	0.6288 ± 0.0605	67.73 ± 18.10	51.00 ± 27.78
DGCNN	0.9687 ± 0.0008	46.40 ± 1.52	0.4220 ± 0.0069	75.95 ± 1.18	19.74 ± 0.49
-SGAN	0.9830 ± 0.0009	36.42 ± 0.49	0.4208 ± 0.0035	77.42 ± 2.35	10.48 ± 1.95
GAT	0.9678 ± 0.0031	34.59 ± 1.52	0.4373 ± 0.0156	81.84 ± 7.24	7.97 ± 3.18
-SGAN	0.9680 ± 0.0040	33.95 ± 5.13	0.4481 ± 0.0240	89.33 ± 6.00	8.57 ± 5.28

4.2 Robustness to Simulated Evasions

We simulate evasion tactics by processing samples from our hold-out IID evaluation split, following the assumptions and methods described in Section 3.2. These evasion attacks are designed to be behavior-preserving and are applied to inputs before they reach the detector pipeline, rather than the models themselves. Each attack technique is processed at a particular perturbation rate, that is the fraction of non-critical, independent calls in the sequence affected by the simulation. It should be noted that the requested rates may not be entirely reflective of the processing carried out, since even at a requested rate of 20% for Call Reordering, almost 8% of samples remained unaffected. Behavior-critical call chains were identified through the *n*-gram analysis of all malware samples in the training dataset, and only calls identified to not be part of these chains were processed.

We find that all models show high malware recall over Call Reordering evasion samples, in the range of 99.54-99.90%, and little variation across increasing perturbation rates (5%, 10%, 15% and 20%). Several models remain unaffected even under aggressive Stalling Padding rates, but LSTM-SGAN and TFORM-SGAN show degradation with high variance, suggesting increased sensitivity to random seed initialization under these evasion conditions. Benign Mimicry produces a consistent decrease in malware recall for each model as the perturbation rate increases. There is also no consistent evidence of adversarial regularization improving robustness compared to baseline performance. This experiment measures malware recall under the simulated detector evasions, and not general robustness against adaptive attackers.

Table 2. Malware recall (%) on IID evaluation split (for reference) and processed evasion samples (12,839 malware samples) for Stalling Padding (S.P.) and Benign Mimicry (B.M.) aggregated across 5 model seeds, reported as mean ± std. Only standard deviations >1pp are reported. Malware recall values over Call Reordering (C.O.) evasion samples are excluded for brevity.

Attack	**Rate**	**1D-CNN -SGAN**	**LSTM -SGAN**	**TFORM -SGAN**	**DGCNN -SGAN**	**GAT -SGAN**
IID	-	99.88 99.90	99.73 99.83	99.63 99.78	99.80 99.84	99.59 99.63
S.P.	30	100.00 99.96	95.09 93.47 ± 1.78	99.98 80.88 ± 21.30	100.00 100.00	99.48 95.06 ± 1.17
	50	100.00 99.97	96.41 89.46 ± 4.96	100.00 67.95 ± 40.32	100.00 100.00	99.97 99.98
	70	100.00 99.96	99.85 89.82 ± 7.08	100.00 56.65 ± 51.34	100.00 100.00	100.00 100.00
B.M.	10	99.78 99.86	98.02 97.62	97.43 ± 2.14 96.73 ± 1.46	97.81 96.93	96.85 96.98 ± 1.53
	20	99.54 99.58	95.33 ± 1.52 95.33 ± 1.39	92.98 ± 6.83 89.34 ± 4.68	96.87 94.98	94.03 91.89 ± 4.20
	30	99.32 99.24	93.03 ± 3.21 93.62 ± 1.84	88.12 ± 11.38 79.20 ± 8.76	96.27 93.63	92.48 ± 1.33 85.94 ± 7.14
	40	99.09 98.66	91.13 ± 4.30 92.29 ± 1.98	85.29 ± 14.26 72.76 ± 10.46	96.14 93.14	92.34 ± 1.22 83.93 ± 8.58

4.3 Computational Efficiency

Computational cost varies across evaluated models and is not directly predicted by parameter count. The dense graph representation from our specific implementation requires 21.9 GB to store adjacency and feature matrices (required by the implemented graph networks), compared to approximately 17.1 MB for the sequential dataset ($\approx 1,280\times$ more storage). Sequential models are observed to be generally more computationally efficient compared to the graph models, despite having a larger number of parameters. While adversarial regularization substantially increases the time taken to train sequential and GAT model variants, its effect on inference time per sample is not as pronounced.

Table 3. Parameter count of implemented representation-architectures, with training time (reported per 100 epochs) and mean inference time per sample (including preprocessing and model loading). Models were trained on a T4 GPU, and inference was performed on a CPU with 3.30 GHz base speed, 6 cores and 8 GB RAM.

Model	**Parameters**	**Training Time**	**Inference Time**
LSTM / -SGAN	994,050 / 994,179	14.3 min / 1 hr 3 min	2.58 ms / 3.17 ms
1D-CNN / -SGAN	318,594 / 318,723	3.7 min / 31.2 min	0.78 ms / 0.72 ms
TFORM / -SGAN	333,826 / 333,955	10 min / 52 min	1.58 ms / 1.57 ms
DGCNN / -SGAN	12,618 / 31,654	1 hr 53 min / 2 hr 1 min	1.97 ms / 1.98 ms
GAT / -SGAN	137,730 / 177,027	3 hr 54 min / 9 hr 2 min	3.32 ms / 5.57 ms

5 Conclusion

Our evaluation reveals tradeoffs among representation-architectures rather than a universally superior model family. Sequential models are observed to achieve stronger in-distribution discrimination and generally lower computational costs, while OOD evaluation results show class-specific tradeoffs between malware recall and benign specificity across models. The robustness study reveals model-specific responses but does not establish a general measure of adversarial robustness to adaptive attacks, and may reflect the models' prior over the imbalanced dataset. The results and tradeoffs reported from our work should be interpreted within the context of our experiments rather than defining characteristics of the evaluated paradigms. Future work should examine different datasets, robustness against live or adaptive evasion tactics, and alternative behavioral representations like network telemetry or registry modification information.

Disclosure of Interests. The authors have no competing interests to declare that are relevant to the content of this article.

References

1. Afianian, A., Niksefat, S., Sadeghiyan, B., Baptiste, D.: Malware dynamic analysis evasion techniques: A survey. ACM Computing Surveys (CSUR) **52**(6), 1–28 (2019)

2. Angelo Schranko De Oliveira, R.J.S.: Behavioral malware detection using deep graph convolutional neural networks. International Journal of Computer Applications **174**(29), 1–8 (2021). https://doi.org/10.5120/ijca2021921218
3. Bensaoud, A., Kalita, J., Bensaoud, M.: A survey of malware detection using deep learning. Machine Learning with Applications **16**, 100546 (2024)
4. Cabau, G., Buhu, M., Oprisa, C.P.: Malware classification based on dynamic behavior. In: 2016 18th international symposium on symbolic and numeric algorithms for scientific computing (synasc). pp. 315–318. IEEE (2016)
5. Chung, J., Gulcehre, C., Cho, K., Bengio, Y.: Empirical evaluation of gated recurrent neural networks on sequence modeling. arXiv preprint arXiv:1412.3555 (2014)
6. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural computation **9**(8), 1735–1780 (1997)
7. https://sandbox.pikker.ee: Cuckoo sandbox, a public-domain sandbox instance; last accessed 9 August 2026
8. Jang, E., Gu, S., Poole, B.: Categorical reparameterization with gumbel-softmax. arXiv preprint arXiv:1611.01144 (2016)
9. Kakisim, A.G., Nar, M., Carkaci, N., Sogukpinar, I.: Analysis and evaluation of dynamic feature-based malware detection methods. In: International Conference on Security for Information Technology and Communications. pp. 247–258. Springer (2018)
10. Kim, Y.: Convolutional neural networks for sentence classification. In: Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP). pp. 1746–1751 (2014)
11. Kolbitsch, C., Comparetti, P.M., Kruegel, C., Kirda, E., Zhou, X.y., Wang, X., et al.: Effective and efficient malware detection at the end host. In: USENIX security symposium. vol. 4, pp. 351–366 (2009)
12. Narudin, F.A., Feizollah, A., Anuar, N.B., Gani, A.: Evaluation of machine learning classifiers for mobile malware detection. Soft Computing **20**(1), 343–357 (2016)
13. Odena, A.: Semi-supervised learning with generative adversarial networks. arXiv preprint arXiv:1606.01583 (2016)
14. Pendlebury, F., Pierazzi, F., Jordaney, R., Kinder, J., 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)
15. Rosenberg, I., Shabtai, A., Rokach, L., Elovici, Y.: Generic black-box end-to-end attack against state of the art api call based malware classifiers. In: International Symposium on Research in Attacks, Intrusions, and Defenses. pp. 490–510. Springer (2018)
16. Sarı, N.V., Acı, M., Acı, Ç.İ.: Windows malware detection via enhanced graph representations with node2vec and graph attention network. Applied Sciences **15**(9), 4775 (2025)
17. Schofield, M., Alicioglu, G., Binaco, R., Turner, P., Thatcher, C., Lam, A., Sun, B.: Convolutional neural network for malware classification based on api call sequence. In: Proceedings of the 8th International Conference on Artificial Intelligence and Applications (AIAP 2021), Zurich, Switzerland. pp. 23–24 (2021)
18. Sihwail, R., Omar, K., Ariffin, K.Z.: A survey on malware analysis techniques: Static, dynamic, hybrid and memory analysis. Int. J. Adv. Sci. Eng. Inf. Technol **8**(4-2), 1662–1671 (2018)
19. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł., Polosukhin, I.: Attention is all you need. Advances in neural information processing systems **30** (2017)
20. Veličković, P., Cucurull, G., Casanova, A., Romero, A., Lio, P., Bengio, Y.: Graph attention networks. arXiv preprint arXiv:1710.10903 (2017)
21. Youssef, N., Elbaraway, N., Elmaghraby, A.: Transformer-based api call sequence modeling for dynamic malware detection. In: SoutheastCon 2025. pp. 494–500. IEEE (2025)