Does Asm2Vec Reduce Drift on Malware Classification?

Does Asm2Vec Reduce Drift on Malware Classification?

Rafael Rocha[1], Stefano de Rosa[2], Paolo Castagno[2], Idilio Drago[2] e Lourenço Alves Pereira Junior[1]

[1]Instituto Tecnológico de Aeronáutica – ITA – Brazil

[2]Università di Torino – Turin – Italy

{rafaelror,ljr}@ita.br, {stefano.derosa,paolo.castagno,idilio.drago}@unito.it

Abstract. *Asm2Vec is an algorithm capable of learning representations for binary files using word embedding techniques. Researchers have employed this approach for binary analysis as well as malware classification. Malware classification is, however, known to be widely affected by drift, i.e., models built to identify a particular malware family become obsolete rapidly. We ask whether representation learning approaches such as Asm2Vec help reduce the impact of drift in malware classification. To answer this question, we design an experiment using two public malware datasets and train classic machine learning models with (i) static features extracted from malware headers and (ii) features obtained using Asm2Vec. Our results show that there is little difference in relation to the effect of drift and that the classifiers trained with Asm2Vec resources present worse classification performance. We provide initial insights into the effects of representation learning on the drift in malware classification.*

Resumo. *O Asm2Vec é um algoritmo capaz de aprender representações de arquivos binários com base em técnicas de embeddings de palavras. Pesquisadores têm utilizado essa técnica para análise de binários, bem como para classificação de malware. No entanto, a classificação de malware é conhecida por ser amplamente afetada por drifting, ou seja, modelos construídos para classificar malware tornam-se obsoletos com o passar do tempo. Portanto, investigamos neste artigo se as abordagens de aprendizado de representação, como Asm2Vec, ajudam a reduzir o impacto do drifting na classificação de malware. Para responder a essa pergunta, projetamos um experimento usando dois datasets públicos de malware e treinamos modelos clássicos de aprendizado de máquina com (i) features estáticas extraídas de cabeçalhos de malware e (ii) features obtidas usando Asm2Vec. Nossos resultados mostram que há pouca diferença em relação ao efeito de drift e que os classificadores treinados com os recursos do Asm2Vec apresentam desempenho de classificação pior. Como contribuição, fornecemos insights iniciais sobre os efeitos do aprendizado de representação em drifiting na classificação de malware.*

1. Introduction

Word embedding techniques such as Word2Vec [Mikolov et al. 2013a, Mikolov et al. 2013b] have been proven to extract rich features from natural language text. Similar approaches have been successfully applied in other scenarios,

such as *code embeddings* in which representations are learned from computer languages [Allamanis et al. 2018]. These approaches have found several applications in cybersecurity, including for assembly code clone searching [Ding et al. 2019], traffic analysis [Gioacchini et al. 2021, Le et al. 2022, Dietmüller et al. 2022, Houidi et al. 2022], and honeypot log interpretation [Boffa et al. 2022].

Asm2Vec [Ding et al. 2019], in particular, is a state-of-the-art approach for learning representations from binary files. Asm2Vec builds vectors that represent assembly instructions from a corpus of disassembled binaries. These vectors are then leveraged to obtain representations for the binaries and support a variety of downstream machine learning (ML) tasks. Notably, Asm2Vec has been proposed for function clone searching and has been used for malware analysis and classification as well [Chandak et al. 2021].

Malware classification is a challenging task since malware code is continuously mutated to evade detection by security systems. Techniques such as *packing* and *obfuscation* have been shown to render ineffective classifiers built with static features extracted from binaries [Aghakhani et al. 2020]. Equally, traditional ML approaches are not effective if models are not updated to cope with changes in the malware families.

This problem – called drift – is well-known by the ML community. Model drift refers to the situation in which the performance of an ML model deteriorates over time. Here we focus on *data drift*, i.e., the statistical properties of the dataset change over time, leading to a mismatch between the training and testing data. The security community facing malware classification has proposed multiple alternatives to improve the performance of ML-based malware classifiers in the presence of drift. Some authors propose algorithms to monitor the drift, applying different approaches to determine the right moment to retrain models [Jordaney et al. 2017, Yang et al. 2021b, Barbero et al. 2022]. Others suggest online training alternatives that update models continuously [Narayanan et al. 2017, Xu et al. 2019, Kan et al. 2021]. Finally, some works introduce models that, by using semantic information, are less sensitive to drift [Zhang et al. 2020]. There is, however, a lack of literature on the impact of representation learning approaches on the drift.

We provide a preliminary investigation on whether representation learning approaches, such as Asm2Vec (a recent and successful approach capable of extracting semantic information from assembly code), contribute to rendering malware classification less sensitive to drift. We perform experiments using two independent and long-term datasets of labeled malware samples. Each dataset contains 13 months of malware samples, which have been labeled by security experts and practitioners. We first isolate the most prevalent malware samples in the initial months of the datasets and train machine learning models with (i) static features extracted from the binaries and (ii) features obtained using Asm2Vec. We then freeze the models and study the model drift.

Our results show that Asm2Vec does not reduce drift effects, except for minor differences in some tested scenarios. Besides that, using Asm2Vec features produced worse classification performance in all tested scenarios. As an explanation for these results, we conjecture that Asm2Vec applied to full (large) binary samples provides too coarse-grained representations, as opposed to its original application in (small)

function cloning search. Overall, our results provide initial insights into the potential of representation learning in malware classification and call for further research to better understand its impact on the drift.

This paper is structured as follows: Section 2 provides background information on representation learning, malware classification, and drift. Section 3 describes our methodology and experimental setup, while ection 4 presents our results on the drift analysis of Asm2Vec. Finally, Section 5 indicates previous works about drift and Section 6 concludes the paper and outlines future work.

2. Background

2.1. Asm2Vec basics

Word embedding is a widely used technique in NLP tasks. Word2Vec [Mikolov et al. 2013a, Mikolov et al. 2013b], in particular, utilizes the co-occurrence of words in sentences to learn a high-dimensional latent space representing the words. Despite usually being constructed based on word co-occurrence, the obtained representations ultimately encode semantic and syntactic properties of words. Word2Vec vectors have served as input for multiple downstream ML algorithms in NLP tasks. Following the same concept, code embedding algorithms learn representations for computer languages. Asm2Vec [Ding et al. 2019] is a code embedding algorithm that focuses on assembly. It has been created to embed assembly code into a latent space, mapping assembly instructions and functions to vectors in the embedding space.

Asm2Vec learns vector representations using large corpora of assembly code. It adapts the PV-DM model, which in turn is an extension of the classic Continuous Bag-of-Words (CBOW) model. Asm2vec jointly learns vector representations for assembly mnemonics and parameters. Given the current assembly instruction in a function and its neighboring instructions, the algorithm relies on the context provided by the adjacent instructions to predict the current mnemonic and parameters.

Off-the-shelf disassemblers (e.g., IDA or RADARE2) can be used to extract assembly functions and control flow graphs from the binaries. The control flow graphs form the sequences of instructions passed as documents to Asm2Vec. Asm2Vec requires a key parameter m, the minimum number of assembly instructions to consider a found sequence to form a valid function.

Asm2Vec maximizes the log probability of observing a current token t_c (a mnemonic or the instruction parameters), given the current function f_s and a window of w neighboring instructions. The output vector for each instruction has dimension d and is built by concatenating vectors for the instruction mnemonic and the average of vectors for its parameters. The final representation for each function (and binary) is obtained by averaging vectors of the instructions composing the functions. For the sake of brevity, we refer readers to [Ding et al. 2019] for details of the Asm2Vec training procedure.

In [Ding et al. 2019], authors show that the obtained embeddings capture relationships between instructions and deliver good performance for assembly similarity/clone searching [Ding et al. 2019]. Equally, embeddings have been shown to provide good features for the classification of binary files [Chandak et al. 2021], without, however, any specific study focusing on drift.

2.2. Measuring drift

Previous studies have shown the robustness of classification models to concept drift using mainly two approaches: (i) monitoring classic performance metrics of classifiers over time or (ii) using ad-hoc metrics that directly quantify the drift.

Considering the first group of metrics, we use the overall accuracy and the F1-score of individual families to monitor drift on new batches of data that are processed without retraining the models. Given the context of malware classification (which involves a multi-class classification problem), we can define the overall accuracy as follows:

$$\textit{Acc} = \frac{\textit{Correct Predictions}}{\textit{All Predictions}}. \tag{1}$$

The latter metric is calculated from the precision and recall as

$$\textit{F1-score} = 2 \cdot \frac{\textit{Precision} \cdot \textit{Recall}}{\textit{Precision} + \textit{Recall}}. \tag{2}$$

Considering metrics proposed for quantifying drift directly, some authors [Pendlebury et al. 2019, Zhang et al. 2020] rely on a metric that quantifies the total drift over a fixed period of time, called *Area Under Time* (AUT). The AUT is calculated from the model performance in each time slot as follows:

$$AUT(f, N) = \frac{1}{N-1} \sum_{k=1}^{N-1} \frac{[f(x_{k+1}) + f(x_k)]}{2}$$

where f is a performance metric (e.g., accuracy or F1-score) and N is the number of test slots – the higher the metric, the better. We here use $AUT(Acc, 12)$, i.e., we calculate AUT taking accuracy as metric and summing up the drift observed in 12-time slots (i.e., $N = 12$ months).

However, the $AUT(Acc, 12)$ provides a view into drift based on absolute performance metrics. In some scenarios, this procedure may result in artifacts. For example, a model with high initial accuracy may show higher AUT (if its drift is very severe) than a poorer-performing model that suffers no drift whatsoever. We thus propose to observe $AUT(Acc, 12)$ together with the *Percentage of Performance Decay* ($PoPD$). The $PoPD$ of a model is calculated as follows:

$$PoPD = \frac{(AUT_c - AUT_r)}{AUT_c}$$

where AUT_c is the AUT assuming that the classifier performance remains constant over time (i.e., no drift is observed) and AUT_r is the actual AUT for the model. The lower the metric, the better – i.e., the classifier has not lost performance. Notice that $PoPD$ may become negative if performance improves over time. This may happen due to artifacts, e.g., if the popularity of classes changes.

3. Methodology

3.1. Dataset

We rely on the BODMAS [Yang et al. 2021a] and Malwarebazaar[1] datasets to explore Asm2Vec impact on drift. We use BODMAS for most experiments and Malwarebazaar

[1] https://bazaar.abuse.ch/

Table 1. Number of binaries for most popular malware families in the datasets. Only a portion of the files could be disassembled for BODMAS. The table reports also the number of packed files (BODMAS).

BODMAS			
Family	**Samples**	**Disassembled**	**Packed**
Sfone	4 729	3 275	1 897
Wacatac	4 694	4 608	2 094
Upatre	3 901	3 736	382
Wabot	3 673	3 546	0
Small	3 339	3 293	18
Total top-5	**20 336**	**18 458**	**14 067**
ganelp	2 232	2 232	10
dinwo	2 057	2 057	821
mira	1 960	1 960	240
Other Families	**6 249**	**6 249**	**1 071**
MALWAREBAZAAR			
Top-5 Families	**—**	**46 205**	**—**

to confirm results.

BODMAS is a curated dataset that contains 57,293 malware samples, all in PE32 format, collected between Aug 2019 and Sep 2020. Security experts have labeled the dataset with 581 different malware families. The dataset also contains 77,142 samples of benign binaries, which are ignored in this work. In addition to the (disarmed) malware binaries, BODMAS provides pre-extracted features obtained using LIEF.[2] For this, BODMAS adopts the same format of the EMBER dataset [Anderson and Roth 2018], a previous public dataset that exposes a total of 2,381 static features.

Malwarebazaar is an open collection of malware samples that are tagged by experts and practitioners. We download 13 months of PE files from the dataset (46 k samples), collected between Mar 2020 and Apr 2021. As for BODMAS, we extract the static features (EMBER) for Malwarebazaar samples.

To assess Asm2Vec's impact on the drift, we first focus on the most popular malware families. With this selection, we reduce oscillations in results due to unpopular classes, easing the observation of drift. For example, for BODMAS the top-5 families add up to 20,336 samples and represent 35% of the original dataset. We also aggregate a sample of the remaining families into an *Other* class when performing experiments in an *open-world* scenario.

Table 1 summarizes the datasets, showing the number of samples per class. We provide more details for the BODMAS dataset, as we rely on this dataset primarily for the experiments. For BODMAS we could not dissemble 5,698 binaries for the

[2]https://lief.quarkslab.com/

selected families.[3] When performing experiments with Asm2Vec, we drop these samples. Finally, we have confirmed that 11,065 samples from the BODMAS dataset are packed. ML models that use static features have difficulties in classify packed malware samples [Aghakhani et al. 2020]. We thus perform specific experiments with these samples to assess whether Asm2Vec brings any advantage to this scenario.

3.2. Malware embeddings and classifiers

We run all experiments on a Linux server running Ubuntu 22.04, with an Intel Xeon Gold 6140 at 2.30GHz with 72 virtual cores (including hyperthreading) and 320GB RAM.

We leverage a PyTorch implementation of Asm2Vec for learning the malware code embeddings.[4] We tested several Asm2Vec parameter setups when learning the embeddings, verifying the performance of downstream classifiers using our training dataset (described next). In particular, we explore the impact of (i) the minimum length of assembly functions m and (ii) the embedding dimension d since previous efforts [Ding et al. 2019, Marcelli et al. 2022] report results using different values for the parameters. All other parameters of ASM2VEC that were not explicitly tested were utilized in the default configuration of its implementation.

We use the two initial months of each dataset (e.g., August and September 2019 for BODMAS) as the training set and the remaining months as independent testing sets. For learning the classifiers, we rely on a simple (yet effective) shallow classification model. In particular, we use a Random Forest (RF) with the following parameters: number of estimators equal to 10; criterion as "entropy", and random states 42. We have also tested other configurations and conclusions that follow holds. We leave a similar analysis with more sophisticated ML models (e.g., deep neural networks) for future work.

We define multiple experiments to test combinations of features, malware samples, and parameters. In particular, we call *EMBER* all RF models built with static features. We generically call *Asm2Vec* (specifying the key parameters d and m when relevant) the RFs built with the embeddings generated with Asm2Vec.

We build different RF models using (i) all samples of the *top-5* malware families (BODMAS) – note that we ignore Asm2Vec in this case, as it cannot operate with the samples that are not disassembled; (ii) the *disassembled* samples of the top-5 families (for both BODMAS and Malwarebazaar); (iii) the previous case, with an additional class composed of other malware families, so to simulate an *open-world* scenario (BODMAS); (iv) only the *packed* malware samples (BODMAS).

4. Assessing Asm2Vec's impact on drift

We now present our evaluation of the impact of Asm2Vec on the drift. We first compare the classification performance of the RF models built with static features and the off-the-shelf setup of Asm2Vec when facing only the top malware families in BODMAS and MalwareBazaar datasets (Section 4.1). To deepen results, we use the BODMAS dataset to explore other Asm2Vec parameters (Section 4.2) and test different scenarios, such as in an open-world use case (Section 4.3) and with packed binaries (Section 4.4).

[3] Automated script using RADARE's `af` command could not find assembly functions in some binaries.

[4] `https://github.com/oalieno/asm2vec-pytorch`

Table 2. Summary of results: The closer AUT is to 1, the better the accuracy; the smaller $PoPD$, the smaller the drift. Results marked as "average" show the average when testing with dimensions 20, 50, 100, 200, and 400.

Setting	$AUT(Acc, 12)$	$PoPD$ (%)
BODMAS DATASET		
EMBER Top-5	0.874	8.86
EMBER Disas.	0.870	10.08
EMBER Disas. Open-World (OW)	0.678	12.07
EMBER Packed	0.748	4.22
EMBER Packed Open-World (OW)	0.376	39.01
Asm2Vec Disas. m=10 (average)	0.795	9.38
Asm2Vec Disas. m=35 (average)	0.794	6.02
Asm2Vec Disas. d=100, m=10 OW	0.618	13.00
Asm2Vec Packed d=100, m=10	0.694	6.92
Asm2Vec Packed d=100, m=10 OW	0.240	26.09
MALWAREBAZAAR DATASET		
EMBER	0.535	33.96
ASM2VEC d=100, m=10	0.307	59.32

Table 2 summarizes the results, showing the $AUT(Acc, 12)$ and $PoPD$ for various experiments. Each line reports the results for an experimental configuration, encoding the used features (EMBER or Asm2Vec), the samples used for learning and testing (top-5, disassembled, open-world, or packed), and the relevant algorithm parameters.

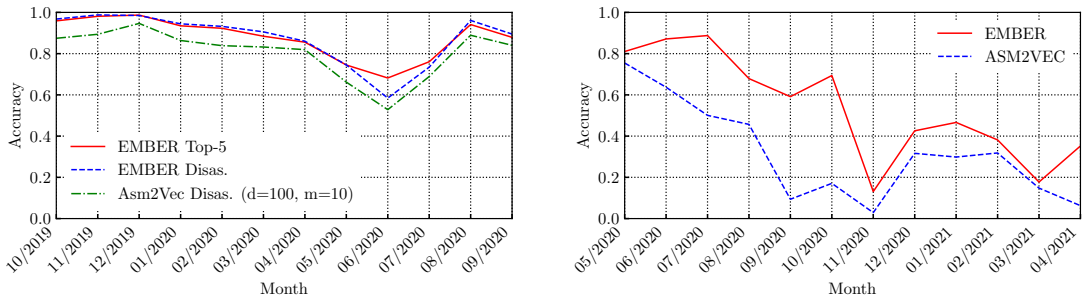
4.1. Asm2Vec vs EMBER features

4.1.1. BODMAS

Figure 1(a) shows the results for accuracy when using the BODMAS dataset over the test months for three different setups: i) EMBER top-5, our closed-world scenario with all samples of the top-5 malware families; ii) EMBER disassembled, thus restricting the analysis to samples that can be classified by Asm2Vec as well; iii) Asm2Vec with $d = 100$ and $m = 10$, which are the default parameters of the algorithm.

It is possible to notice a similar behavior in the three depicted series. Notice how the accuracy for RF models built with EMBER features drops sharply around June 2020 and then increases in the following months. This behavior is somehow expected and can be observed in the original BODMAS paper too, due to a sudden increase in the popularity of the Sfone malware family. This family is under-represented in the training samples, as it was not a widespread malware in the initial months of the dataset.

The accuracy for the RFs built with EMBER features varies little when discarding files that could not be disassembled. This suggests that the absence of these files has little



(a) BODMAS (b) MalwareBazaar

Figure 1. Accuracy for EMBER vs. Asm2Vec features.

impact on the accuracy result throughout the test period. From now on, we thus ignore the samples that could not be disassembled.

The accuracy obtained with Asm2Vec is lower than that obtained with EMBER features. As we can see in Table 2, the $AUT(Acc, 12)$ metric is reduced from 87% (EMBER disassembled) to 80% (Asm2Vec disassembled d=100, m=10). That is, the RFs built with the Asm2Vec features (in the algorithm's default parameters) deliver worse performance than the static features.

Overall, the figure suggests that all models suffer from some drift, with consistent fluctuations across all tested configurations. Table 2 shows that using Asm2Vec features results in slightly better $PoPD$ metrics, e.g., obtaining 8.32% for the Asm2Vec disassembled case (d=100, m=10) as opposed to 10% with the BODMAS disassembled. Yet, these differences seem to come from the lower performance of the RF built with the Asm2Vec features. In other words, Asm2Vec features result in a classifier performing worse than EMBER features. However, the accuracy of the Asm2Vec-based classifier is slightly more constant over time.

4.1.2. MalwareBazaar

To verify the consistency of the results, we repeat the tests using the MalwareBazaar dataset. Recall that we have only disassembled files for the top-5 malware families in this case. Results are in Table 2 and Figure 1(b).

Again, the EMBER features allow more accurate classifiers than the Asm2Vec ones, with $AUT(Acc, 12) = 0.535$ for the former and $AUT(Acc, 12) = 0.307$ for the latter. The $AUT(Acc, 12)$ and $PoPD$ metrics suggest strong drift in both cases. Numbers for Asm2Vec are clearly worse, questioning the appropriateness of these features in this downstream task.

These bad results are, however, explained not only by changes in the features of each malware family but also by changes in the proportion of samples observed throughout the months. Indeed, observing the distribution of the top-5 families in the Malwarebazaar dataset over time (Figure 2), we can see that the number of samples for each family is very different from month to month. Two of the 5 families (*dridex* and

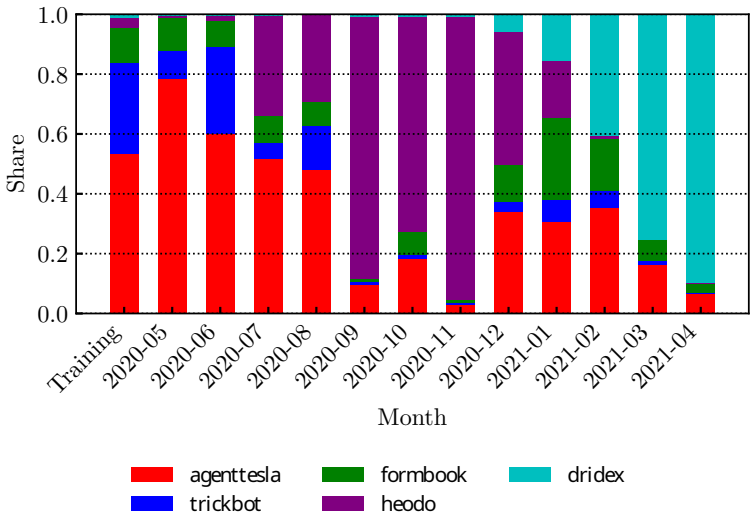


Figure 2. Distribution of the popularity of samples per month (top-5 families) in the MalwareBazaar dataset.

heodo) are indeed under-represented in the training set. Notice how their increase in popularity is coincident with the sharp drop in the performance of the classifiers.

To evaluate better the accuracy without that effect, Figure 3 shows the evolution of the F1-Score of the models for the *Agenttesla* family only. This is the most popular family in our training set for the Malwarebazaar dataset. The F1-Score trend indeed improves – notice for example how numbers are better around Dec 2020. Yet, both models still present strong drift – in this case due to changes in the samples of the family. Asm2Vec features are affected the most, with both $AUT(Acc, 12)$ and $PoPD$ pointing to stronger drift when classifying samples of this family (omitted for brevity).

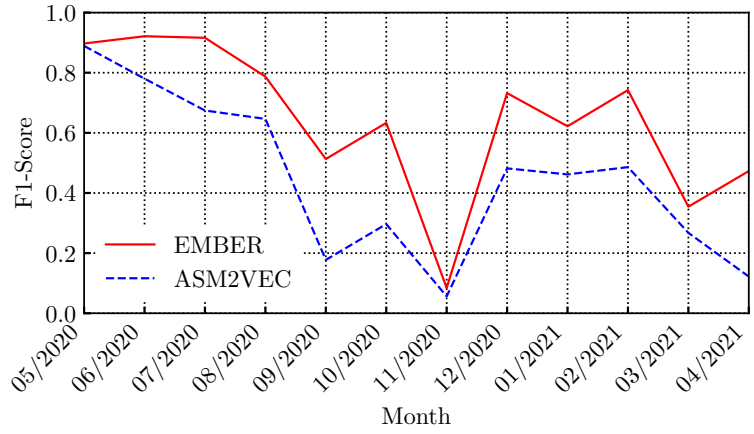


Figure 3. F1-Score trend for the *Agenttesla* family (Malwarebazaar).

4.2. Impact of Asm2Vec parameters

We next verify whether varying parameters of Asm2Vec could reduce the performance gap of its downstream RF model compared to the one built with the EMBER static

features. To check that, we vary the embedding dimension d and the minimum number of instructions m and record the obtained $AUT(Acc, 12)$ and $PoPD$.

Figure 4 summarizes results with different plots for each metric. The x-axis reports the tested embedding dimensions d, while the lines show results with different parameters m. Table 2 summarizes numeric results.

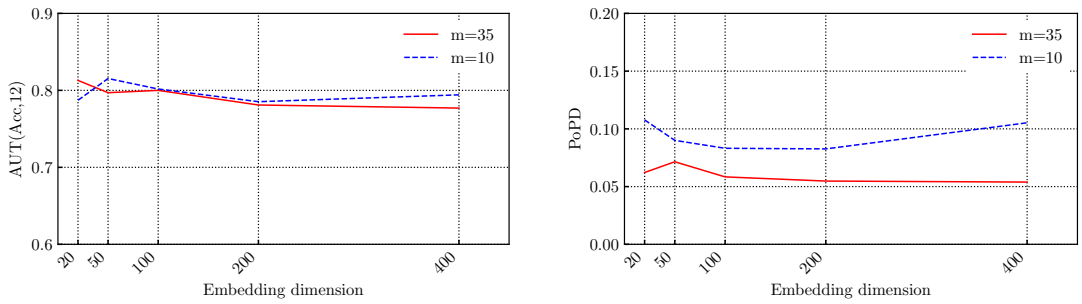
Figure 4(a) shows that the variation in the minimum number of instructions m does not cause significant changes in accuracy. From Figure 4(b), there is a small but consistent decrease in the $PoPD$ when increasing the value of m from 10 to 35. In other words, a greater m leads to a slight reduction in the drift effect. Quantitatively, we can observe from Table 2 that the average value (considering the 5 tested dimensions) for AUT is practically the same (0.795 for $m = 10$ and 0.794 for $m = 35$). Meanwhile, the average value for $PoPD$ is 3.36 percentage points lower for $m = 35$ (9.38 against 6.02).

Overall, the performance of the RF built with Asm2Vec features is only marginally affected by the algorithm parameters.

4.3. Open-world scenario

We now present experiments in an open-world scenario, i.e., by considering also samples of the other malware families. In this case, we add samples from three classes (*ganelp*, *dinwo* and *mira*) encompassing a total of 6 249 samples of malicious binaries. These samples are not present during training and are generically mixed in a class "Other" during testing. The multiple trees present in the RFs are used to decide whether the testing binaries are out-of-sample during classification time. This is a more realistic use case, which tests the robustness of the models when applied to real datasets that may contain unknown malware families.

Figure 5 depicts the accuracy of the models over time in this scenario. When comparing these numbers to those in Figure 1(a), we see that models built with both feature sets deliver an accuracy that is much lower than their respective figures in the previous scenario. As expected, the presence of multiple unknown malware families during testing complicates the classification, leading to a reduction of up to 20% in accuracy. We see that the Asm2Vec features result in a classifier that has lower accuracy also in the open-world case.



(a) $AUT(Acc, 12)$ (b) $PoPD$

Figure 4. Asm2Vec – AUT and $PoPD$ when varying the embedding dimension d and the minimum number of instruction m (BODMAS).

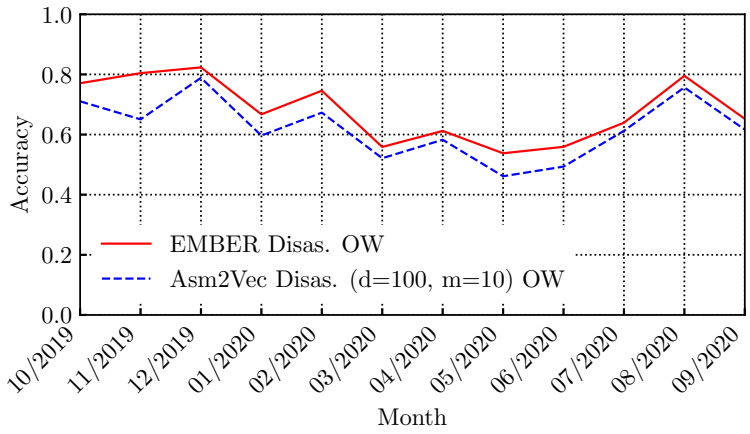


Figure 5. Accuracy by month in the open-world scenario.

From Table 2, we can see that the comparative results between feature extraction approaches (EMBER x Asm2Vec) are indeed very similar to the other tested cases (e.g., EMBER features have $AUT = 8.85\%$ higher than Asm2Vec). All in all, the previous conclusions hold also in this case: Asm2Vec leads to slightly lower performance than the EMBER features, without significantly improving the drift.

4.4. Asm2Vec and Packing

We carried out additional tests using only the packed malware samples, again using the BODMAS dataset. As there were no packed samples for two malware families in our top-5 (*Sfone* and *Wabot*), we replace them with the next two most popular ones for which we could identify packing (*Ganelp* and *Dinwod*), thus keeping 5 families in the dataset.

Figure 6 shows the accuracy obtained for EMBER and Asm2Vec features. Both approaches have a very similar performance in this experiment. Generally, accuracy is lower than in the previous experiments, as expected since the classification of packed malware is known to be harder for ML models. Asm2Vec and EMBER deliver a very similar profile in terms of drift, with no remarkable differences. We tested also a similar configuration in the open-world setting, and the results are indeed inline – These numbers are shown only in Table 2 for brevity.

Generally, we notice only minor differences in the AUT and $PoPD$ metrics for the two types of features in multiple settings. These results reinforce the conclusions of the previous section, about the equivalence of the features in terms of drift effects.

5. Related work

Previous works have proposed different approaches to address the drift effect by considering that the ML model can be updated through retraining with new (which means "drifted") malware objects. Hence, some studies focused on developing metrics that allow identifying the ideal moment in which retraining should be done [Jordaney et al. 2017, Yang et al. 2021b, Barbero et al. 2022]. Furthermore, other authors proposed online learning approaches, which allow the model to identify the concept drift in a malware object in real time and use it for retraining[Narayanan et al. 2017, Xu et al. 2019,

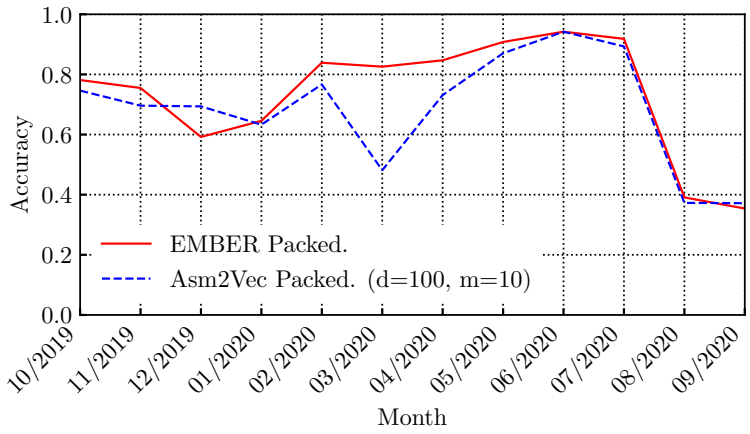


Figure 6. Accuracy by month for EMBER vs. Asm2Vec features - Only packed files.

Kan et al. 2021]. However, such approaches do not address the issue of designing a more concept-drift-robust model to concept drift.

On the other hand, other authors addressed the problem of developing a more robust model to concept drift by using semantic information (malware behavior), which is less less prone to change over time. [Onwuzurike et al. 2019] proposed using an abstraction of Android malware API calls as a feature, utilizing only information from its package and family. Similarly, [Zhang et al. 2020] used a structure called APIGraph, capable of abstracting Android API calls to semantic clusters of APIs. However, these works rely on crafted features (API calls) to generate more generalized abstractions.

This work stands out from previous studies by investigating if a representation learning technique capable of extracting semantic information from code can be useful for the development of more concept-drift-robust malware classification models.

6. Discussion and future work

We investigated whether using Asm2Vec to learn representations for binaries could make ML classification models more robust to concept drift. Using two datasets that include hundreds of thousands of malware samples collected during several months, we compared models trained with static features and the Asm2Vec vectors.

Our experiments showed that the Asm2Vec provides similar (or lower) classification performance as static features. Moreover, in almost all tested cases, Asm2Vec did not enhance the classification robustness against drift. Whereas our results do not allow us to pinpoint the causes for the lower performance of the Asm2Vec features, we conjecture that the features are (i) too coarse and (ii) impacted by noise.

Regarding the first point, recall that Asm2Vec obtains a single vector for whole binary, obtained by averaging the embedding vectors of each of its functions. We believe that this representation washes out knowledge obtained from the sequence of instructions, thus reducing the potential of the approach. Regarding the second point, the disassembled code of a binary usually includes thousands of functions, and most of those are irrelevant to characterize a binary (e.g., libraries or noncritical code paths). Asm2Vec has been

proposed as a means to easy assembly function clone identification, a scenario in which both problems are not critical. For full binary classification, our initial results show that the application of Asm2Vec as-is does not provide gains, thus calling for further research on the potential of representation learning for this scenario.

In future work, we will investigate other approaches (eventually not based on Asm2Vec) to learn representations for full binary files. Other techniques for code representation will be tested in the context of malware classification. Moreover, we also intend to explore other ML techniques besides RF, verifying whether they can extract better knowledge from the code representation vectors. Finally, an investigation of the malware drift behavior in other classification techniques (e.g., heuristics, behavior-base analysis, signature-based)

Acknowledgements

This work is partially supported by ITA's graduate program on operational applications PPGAO/ITA and FAPESP grant #2020/09850-0.

References

Aghakhani, H., Gritti, F., Mecca, F., Lindorfer, M., Ortolani, S., Balzarotti, D., Vigna, G., and Kruegel, C. (2020). When Malware is Packin' Heat; Limits of Machine Learning Classifiers Based on Static Analysis Features. In *Proceedings of the Network and Distributed System Security Symposium*, NDSS'20.

Allamanis, M., Brockschmidt, M., and Khademi, M. (2018). Learning to Represent Programs with Graphs. In *Proceedings of the 6th International Conference on Learning Representations*, ICLR'18.

Anderson, H. S. and Roth, P. (2018). Ember: An open dataset for training static pe malware machine learning models. arXiv preprint arXiv:1804.04637.

Barbero, F., Pendlebury, F., Pierazzi, F., and Cavallaro, L. (2022). Transcending TRANSCEND: Revisiting Malware Classification in the Presence of Concept Drift. In *Proceedings of the IEEE Symposium on Security and Privacy*, SP'22, pages 805–823.

Boffa, M., Milan, G., Vassio, L., Drago, I., Mellia, M., and Houidi, Z. B. (2022). Towards NLP-based Processing of Honeypot Logs. In *Proceedings of the IEEE European Symposium on Security and Privacy Workshops*, EuroS&PW'22, pages 314–321.

Chandak, A., Lee, W., and Stamp, M. (2021). A Comparison of Word2Vec, HMM2Vec, and PCA2Vec for Malware Classification. In *Malware Analysis Using Artificial Intelligence and Deep Learning*. Springer, Cham.

Dietmüller, A., Ray, S., Jacob, R., and Vanbever, L. (2022). A New Hope for Network Model Generalization. In *Proceedings of the 21st ACM Workshop on Hot Topics in Networks*, HotNets'22, pages 152–159.

Ding, S., Fung, B., and Charland, P. (2019). Asm2Vec: Boosting Static Representation Robustness for Binary Clone Search against Code Obfuscation and Compiler Optimization. In *Proceedings of the IEEE Symposium on Security and Privacy*, SP'19, pages 472–489.

Gioacchini, L., Vassio, L., Mellia, M., Drago, I., Houidi, Z., and Rossi, D. (2021). DarkVec: Automatic Analysis of Darknet Traffic with Word Embeddings. In *Proceedings of the 17th International Conference on emerging Networking EXperiments and Technologies*, CoNEXT '21, pages 76–89.

Houidi, Z., Azorin, R., Gallo, M., Finamore, A., and Rossi, D. (2022). Towards a Systematic Multi-Modal Representation Learning for Network Data. In *Proceedings of the 21st ACM Workshop on Hot Topics in Networks*, HotNets'22, pages 181–187.

Jordaney, R., Sharad, K., Dash, S., Wang, Z., Papini, D., Nouretdinov, I., and Cavallaro, L. (2017). Transcend: Detecting Concept Drift in Malware Classification Models. In *Proceedings of the 26th USENIX Security Symposium*, USENIX Security'17, pages 625–642.

Kan, Z., Pendlebury, F., Pierazzi, F., and Cavallaro, L. (2021). Investigating Labelless Drift Adaptation for Malware Detection. In *Proceedings of the 14th ACM Workshop on Artificial Intelligence and Security*, AISec '21, pages 123–134.

Le, F., Srivatsa, M., Ganti, R., and Sekar, V. (2022). Rethinking Data-driven Networking with Foundation Models: Challenges and Opportunities. In *Proceedings of the 21st ACM Workshop on Hot Topics in Networks*, HotNets'22, pages 188–197.

Marcelli, A., Graziano, M., Ugarte-Pedrero, X., Fratantonio, Y., Mansouri, M., and Balzarotti, D. (2022). How Machine Learning Is Solving the Binary Function Similarity Problem. In *Proceedings of the 31st USENIX Security Symposium*, USENIX Security'22, pages 2099–2116.

Mikolov, T., Chen, K., Corrado, G., and Dean, J. (2013a). Efficient Estimation of Word Representations in Vector Space. arXiv preprint arXiv:1301.3781.

Mikolov, T., Le, Q. V., and Sutskever, I. (2013b). Exploiting Similarities among Languages for Machine Translation. arXiv preprint arXiv:1309.4168.

Narayanan, A., Chandramohan, M., Chen, L., and Liu, Y. (2017). Context-Aware, Adaptive, and Scalable Android Malware Detection Through Online Learning. *IEEE Transactions on Emerging Topics in Computational Intelligence*, 1(3):157–175.

Onwuzurike, L., Mariconti, E., Andriotis, P., Cristofaro, E. D., Ross, G., and Stringhini, G. (2019). Mamadroid: Detecting android malware by building markov chains of behavioral models (extended version). *ACM Trans. Priv. Secur.*, 22(2).

Pendlebury, F., Pierazzi, F., Jordaney, R., Kinder, J., and Cavallaro, L. (2019). TESSERACT: Eliminating Experimental Bias in Malware Classification across Space and Time. In *Proceedings of the 28th USENIX Security Symposium*, USENIX Security'19, pages 729–746.

Xu, K., Li, Y., Deng, R., Chen, K., and Xu, J. (2019). DroidEvolver: Self-Evolving Android Malware Detection System. In *Proceedings of the IEEE European Symposium on Security and Privacy*, EuroS&P'19, pages 47–62.

Yang, L., Ciptadi, A., Laziuk, I., Ahmadzadeh, A., and Wang, G. (2021a). BODMAS: An Open Dataset for Learning based Temporal Analysis of PE Malware. In *Proceedings of the IEEE Security and Privacy Workshops*, pages 78–84.

Yang, L., Guo, W., Hao, Q., Ciptadi, A., Ahmadzadeh, A., Xing, X., and Wang, G. (2021b). CADE: Detecting and Explaining Concept Drift Samples for Security Applications. In *Proceedings of the 30th USENIX Security Symposium*, USENIX Security'21.

Zhang, X., Zhang, Y., Zhong, M., Ding, D., Cao, Y., Zhang, Y., Zhang, M., and Yang, M. (2020). Enhancing State-of-the-art Classifiers with API Semantics to Detect Evolved Android Malware. In *Proceedings of the ACM SIGSAC Conference on Computer and Communications Security*, CCS '20, pages 757–770.