Empirical Quantification of Spurious Correlations in Malware Detection

Bianca Perasso[1], Ludovico Lozza[1], Andrea Ponte[1,*], Luca Demetrio[1,*], Luca Oneto[1] and Fabio Roli[1,2]

[1]Università degli Studi di Genova, Italy
[2]Università degli Studi di Cagliari, Italy

Abstract

End-to-end deep learning exhibits unmatched performance for detecting malware, but such an achievement is reached by exploiting spurious correlations – features with high relevance at inference time, but known to be useless through domain knowledge. While previous work highlighted that deep networks mainly focus on metadata, none investigated the phenomenon further, without quantifying their impact on the decision. In this work, we deepen our understanding of how spurious correlation affects deep learning for malware detection by highlighting how much models rely on empty spaces left by the compiler, which diminishes the relevance of the compiled code. Through our seminal analysis on a small-scale balanced dataset, we introduce a ranking of two end-to-end models to better understand which is more suitable to be put in production.

Keywords

Malware Detection, Spurious Correlations, Deep Neural Networks

1. Introduction

Antivirus programs are now deeply integrated with machine learning components, extending the defensive capabilities of endpoints. In particular, one prominent approach is posed by end-to-end techniques, which directly digest the raw bytes of programs, thus learning an intermediate representation directly from data. This is opposed to regular feature extraction processes, that pre-process each sample to extract aggregated and compressed information, later used at training time. While being time-consuming [1], feature extraction in cybersecurity contexts are easily prone to pre-processing errors [2], since malicious actors always try to complicate their programs to impede the analysis. Hence, even if the requirement of available samples is higher, end-to-end models ditch this problem entirely, still exhibiting excellent performance in production [3, 4, 5, 6]. However, as a downside, these end-to-end models are completely opaque in terms of which are the relevant features used to compute predictions. In particular, previous work [7, 5, 8] debated whether Windows malware detectors implemented with deep neural networks are affected by *spurious correlations* – correlations between the predicted class and features that are known to be not relevant in terms of domain knowledge. While they all acknowledge an excessive focus on the metadata of input samples, none of the previous work either (i) tried to quantify how much these models rely on spurious correlations, and (ii) ignore other spurious correlations that might be learned by those models. Hence, in this preliminary analysis, we investigate and propose an empirical quantification of three spurious correlations that can be learned by Windows malware detectors, while also highlighting the relevance of features known to be important. Such is achieved by leveraging *integrated gradients* [9] (IG), a gradient-based method for computing the relevance of each input feature of deep learning models. Relying on domain knowledge, we select the locations inside samples where there might be spurious correlations, and we estimate their relevance through the

Ital-IA 2025: 5th National Conference on Artificial Intelligence, organized by CINI, June 23-24, 2025, Trieste, Italy
*Corresponding author.
bianca.perasso@gmail.com (B. Perasso); ludovico.lozza@hotmail.com (L. Lozza); andrea.ponte@edu.unige.it (A. Ponte); luca.demetrio@unige.it (L. Demetrio); luca.oneto@unige.it (L. Oneto); fabio.roli@unige.it (F. Roli)
0009-0001-5647-1077 (A. Ponte); 0000-0001-5104-1476 (L. Demetrio); 0000-0002-8445-395X (L. Oneto); 0000-0003-4103-9190 (F. Roli)

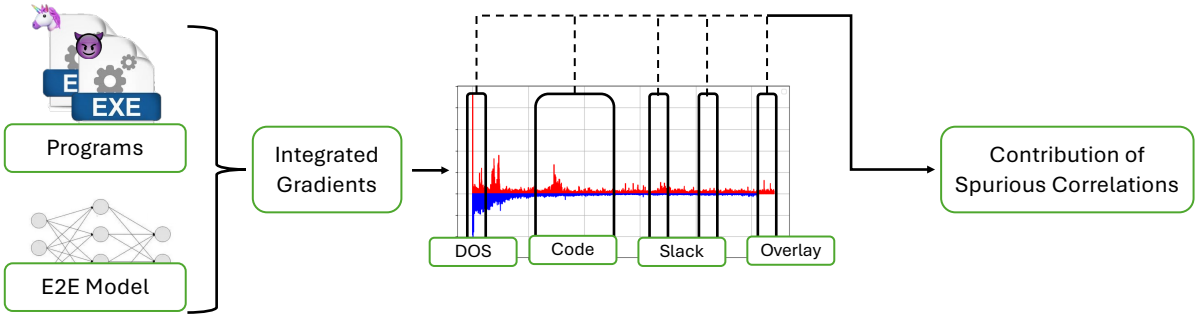

Figure 1: Recap of our analysis: given an end-to-end model and Windows programs, we compute Integrated Gradients, and we extract the attributed relevance of (i) the DOS header, (ii) the slack bytes, (iii) the overlay, and (iv) the ".text" section. These are aggregated to obtain a score, which hint how much the input model relies on spurious correlations when computing predictions.

ℓ_2 norm. Such aggregated relevance is then normalized to gain an understanding of their impact in the decision. Through an experimental analysis on a balanced small-scale dataset, we show how two state-of-the-art end-to-end deep networks are affected by spurious correlations, impacting the relevance attributed to the compiled code which should be, in theory, the most relevant portion of any executable.

2. Background and Related Work

Before describing our research work, we briefly present the key concepts to fully understand our manuscript. In the following, we explain the type of data we deal with, how a malicious sample can be detected, and how this decision can be explained.

Windows PE File Format. In this work, we focus on malware detection for Windows operating systems. Each Windows program is stored in Portable Executable (PE) format[1]. The PE format is composed of several parts (Figure 2), containing everything the OS needs to store, load, and execute the program: (**A**) the DOS header and stub, which compose a valid DOS program, are components kept only for backward compatibility purposes, and they are not informative about the program structure; (**B**) the PE header is the real header of the PE format, and, together with the (**C**) Optional Header, contains essential information for storing and loading the program; the Section Table (**D**) and (**E**) the Sections, that instructs the OS where to find the real content of the program, such as *.text*, being the compiled code of the program (for all the others, we refer to the documentation of the format). While being necessary for each program to comply with this structure, the format itself permits modifications. These are often named code caves, and together with backward compatibility parts (**A**), leave an open field to malware developers [10, 11]. For instance, due to alignment requirements, the compiler allocates more space than needed to store sections: these uninitialized bytes between sections are called *slack bytes*, represented in gray in Figure 2 (**E**). Moreover, the PE format does not forbid the appending of bytes at the end of the last section (*overlay space*, depicted in **F**).

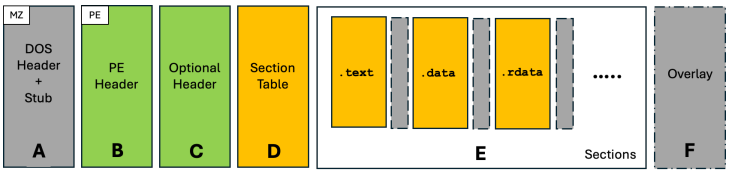

Figure 2: Representation of the Windows PE File Format.

End-to-end Malware Detectors. In recent years, malware detection methods have fully embraced the AI paradigm, training Machine Learning (ML) and Deep Learning (DL) models on past data [6]. Models can learn from different kinds of malware analysis, which can extrapolate knowledge from the sole program structure (static analysis) or the behavior of malicious samples, previously executed in isolated

[1] https://learn.microsoft.com/en-us/windows/win32/debug/pe-format

environments (dynamic analysis). In this work, we leverage end-to-end static detectors, which do not rely on feature extraction, which can be faulty [2], but can learn directly from raw bytes, as proposed by Raff et al. [3], Coull et al. [5] and Krčál et al [4]. Generally, these DL architectures treat each PE byte as an input for the network, which is represented as a vector using an embedding layer. This approach produces matrices in which each row corresponds to a byte, represented as a vector generated by the embedding layer. After embedding, the input matrices pass through a series of convolutional and pooling layers, followed by fully connected layers that generate the final output probability.

Integrated Gradients. Explaining predictions of deep neural networks is a thorny challenge, due to the black-box nature of these complex models. In their work, Sundararajan et al. [9] propose *Integrated Gradients*, an attribution method that compute the relevance of each input feature in computing the prediction of input samples. These contributions are computed with respect to a *baseline*, representing a null signal from which the relevant features "emerge". To calculate them, the method accumulates the gradients of the model to interpret with respect to the input along a straight-line path from the baseline to the actual input, and then it averages all them to find the relevance of each feature.

Related Work on Spurious Correlations in Malware Detection. Recent work rise the debate the presence of *spurious correlation* in malware detection [7, 5, 8, 12]. While these work analyses peculiar behaviors of malware detectors, supporting both the presence of useless features that might cause drop in robustness [7, 8], or deep networks being able to learn relevant locations inside headers [5], none of them proposes an empirical way to quantify these awkward phenomena.

3. Measuring Spurious Correlations for Malware Detectors

We now illustrate how we quantify the relevance attributed by end-to-end Windows malware detectors to known spurious correlations, and we report our methodology in Algorithm 1. First, we leverage domain knowledge on Windows PE file format to isolate parts of each executable file that are known to be semantically irrelevant for malware detection. While there might be plenty, in this paper we focus on three specific parts that should not be relevant at inference time: the *DOS header*, the *slack bytes*, and the *overlay space*. Then, given an end-to-end Windows malware detector, for each sample (line 2) we compute feature attributions using Integrated Gradients (IG) [9] (line 3). If the attributions computed by IG on the selected regions are different from zero, it means that the model is considering those to compute predictions with other features as well, thus confirming the presence of spurious correlations. These attribution, divided in the selected regions through domain knowledge, are characterized by their ℓ_2 norm (line 5 – line 8), and normalized accordingly with the total ℓ_2 norm of the attributions. In this way, we obtain a score for each spurious correlation that has an higher bound on the total norm of the attribution: if this ratio leans towards 1, it mean that all the prediction relies on such portion of the executable. We also select the *.text section* to show how much relevance is given to the most important part of an executable, expecting it to be higher than the other numbers. Finally, we compute a score based on the previous steps, by subtracting from the average relevance of the compiled code, the average relevance of the spurious correlations (line 10). If this metric is positive, the analyzed model is attributing most of relevance to the really important features. If it is zero or negative, it means that the analyzed model is giving more relevance to spurious correlations, harming its reliability.

Algorithm 1: Impact of Spurious Correlation in Windows malware detectors

Data: f, a malware detector; $\boldsymbol{X}$, a set of Windows program
Result: $\boldsymbol{r}_f$, reliance on spurious correlation

1 $\boldsymbol{r_{dos}}, \boldsymbol{r_{slack}}, \boldsymbol{r_{ovelay}}, \boldsymbol{r_{.text}} \leftarrow 0$
2 **for** $\boldsymbol{x} \in \boldsymbol{X}$ **do**
3 $\quad \boldsymbol{ig} \leftarrow IG(f, \boldsymbol{x})$
4 $\quad \eta \leftarrow 1/(|\boldsymbol{X}|||\boldsymbol{ig}||_2^2)$
5 $\quad \boldsymbol{r_{dos}} \leftarrow \boldsymbol{r_{dos}} + \eta||\text{select\_dos}(\boldsymbol{ig})||_2^2$
6 $\quad \boldsymbol{r_{slack}} \leftarrow \boldsymbol{r_{slack}} + \eta||\text{select\_slack}(\boldsymbol{ig})||_2^2$
7 $\quad \boldsymbol{r_{ov}} \leftarrow \boldsymbol{r_{ov}} + \eta||\text{select\_overlay}(\boldsymbol{ig})||_2^2$
8 $\quad \boldsymbol{r_{.text}} \leftarrow \boldsymbol{r_{.text}} + \eta||\text{select\_text}(\boldsymbol{ig})||_2^2$
9 **end**
10 **return** $\boldsymbol{r_{.text}} - (\boldsymbol{r_{dos}} + \boldsymbol{r_{slack}} + \boldsymbol{r_{ovelay}})$

Model	Data	DOS	Slack	.text	Overlay	Aggregate Score
MalConv	goodware	0.0493	0.2567	0.5388	0	0.2939
	malware	0.0705	0.0911	0.5111	0	
BBDNN	goodware	0.0228	0.4082	0.4802	0	0.2502
	malware	0.0273	0.1117	0.5529	0	

Table 1
Magnitude of integrated gradients applied to both malware and goodware samples, characterized by the output of Algorithm 1. We also report the relevance of the four analyzed PE regions (DOS, Slack, .text, and Overlay) to clarify the contribution of each of them.

4. Experimental Analysis

Experimental Setup. We detail here how we have setup our analysis, describing which data and deep neural networks we have used.

Dataset. We use a dataset composed of 210 malware samples and 210 goodware samples. We take malware samples from the Speakeasy testset [13], sampling 30 PEs for each family (*Backdoor*, *Coinminer*, *Dropper*, *Keylogger*, *Ransomware*, *RAT* and *Trojan*). We take benign samples from a fresh installation of Windows 11, inside *sys32* folder.

Models. In our experimental work, we leverage two CNN architectures, mentioned in Sect. 2: the first is MalConv, proposed by Raff et al. [3], the second is BBDNN, proposed by Coull et al. [5]. MalConv embeds input sequences (truncated to 1 MB) with an 8-dimensional embedding space, then it implements a single gated convolutional layer [14] with global max-pooling, followed by a single fully-connected layer and softmax. BBDNN is quite similar to MalConv, but it has a larger embedding space (10 dimensions), and forwards input sequences to five alternating convolutional and pooling layers and finally to a fully connected layer with a sigmoid function. BBDNN results in a deeper architecture, and it pays this price by truncating input byte sequences to 100 KB to have reasonable training times. Both models are trained on the EMBER dataset [1], and we use the pretrained version included into *maltorch*[2] library.

Setup of Integrated Gradients. The method requires two hyper-parameters: the baseline from which compute the attributions, and the size of the interpolation. Regarding the baseline, we adopt the same setting provided by previous work [7], and we set the baseline as the empty file, thus filled with the special character 256. Thereafter, we set the interpolation size to evaluate 50 gradients.

Setup of Algorithm 1. Since many malware samples are obfuscated, thus they have renamed or removed the `.text` section, we use as code the first section that is marked as executable to conclude our analysis. In this way, we ensure that each sample is correctly treated as specified in Algorithm 1.

Experimental Results. We now present the results of our analysis, by computing the relevance of the four analyzed PE regions using Integrated Gradients (IG) method, applied to both MalConv and BBDNN models. For each model, we report the average values computed for the four analyzed regions of PE files, both for goodware and for malware, and all results are presented in Table 1. Even in this simplified setting, we can observe that: (i) on average, both models rely more on the bytes found in the executable section, which is evidenced by the positive aggregated scores, nevertheless these are diminished by the presence of the spurious correlations; (ii) there is a notable distinction between goodware and malware samples, particularly the values in the Slack regions are higher for the goodware samples with respect to the malware ones, suggesting that the models might focus too much on unused space; (iii) comparing the two models, their aggregated scores appear largely similar on average, however in this analysis we do not account the different models' input spaces; (iv) for both models and across both datasets, the relevance attributed to overlay region is always zero, which may indicate that the models are not attributing relevance to those bytes. However, it is important to notice that the majority of the samples is longer than the actual file sizes: there are 283 files larger than 100 KB (BBDNN's input space) and 100 files larger than 1 MB (MalConv's input space).

[2] https://github.com/zangobot/maltorch

These trends are summarized by the average response of Integrated Gradients in Figure 3, where it is clear that plenty of relevance is attributed to bytes in executable sections, but there are peaks in other regions as well. Also, this aggregated view highlight that it is very likely that many other spurious correlations have been learned by these models which are located in other places other than the ones we have analyzed.

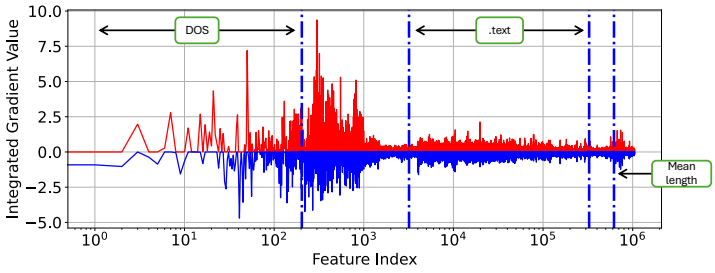

Figure 3: Average attribution of Integrated Gradients computed on MalConv. Red / Blue bars symbolize attribution towards the malicious / benign class.

5. Conclusions

Limitations and Future Work. Our preliminary analysis is timely, but it has some limitations. First, we tested our methodology with a small-scale dataset using only two state-of-the-art end-to-end neural network. Since this analysis depends on both data and trained models, results might change accordingly. Nevertheless, the dataset we have used is balanced, both in terms of ratio between goodware and malware, but also in terms of the malware families we have considered. Secondly, our results do not consider the length of the input window of each network. Hence, larger sections, which are kept without being cut, contribute more to the total score computed for the model. However, with our methodology we are characterizing how much the spurious correlations contribute to the norm of the calculated attribution, showing that they absorb a consistent fraction of it. Thus, as future work, we will consider larger dataset like analyzing the whole Speakeasy dataset [13], and other state-of-the-art data sources, also including other end-to-end neural networks for malware detection [4]. Lastly, while we only covered spurious correlations on static analysis, we will extend our work to also cover spurious correlation in *dynamic analysis*, which track the execution of programs and train models on summary reports of their activity.

Final Remarks. In this preliminary analysis, we proposed a methodology for quantifying the impact of spurious correlations in end-to-end Windows malware detectors. While most of the prediction is focused inside the section containing code, a consistent amount of the attribution is also given to spurious correlation. Through our analysis, we introduce a way to characterize such reliance to spurious correlations in malware detectors, paving the road towards novel techniques that can, in the near future, serve as the first benchmark of these technologies, providing a more reliable way to deploy those.

Acknowledgments

The authors acknowledge the help of Daniel Gibert for releasing trained models in *maltorch* library. This work was partially supported by projects SERICS (PE00000014), FAIR (PE00000013) under the NRRP MUR program funded by the EU - NGEU, and FISA-2023-00128 funded by the MUR program “Fondo italiano per le scienze applicate”.

Declaration on Generative AI

The authors declare that no generative AI tools were used to write the manuscript.

References

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

[2] A. Ponte, D. Trizna, L. Demetrio, B. Biggio, I. T. Ogbu, F. Roli, Slifer: Investigating performance and robustness of malware detection pipelines, Computers & Security 150 (2025) 104264.

[3] E. Raff, J. Barker, J. Sylvester, R. Brandon, B. Catanzaro, C. K. Nicholas, Malware detection by eating a whole EXE, in: Workshops at the 32 AAAI Conf. on Art. Int., 2018.

[4] M. Krčál, O. Švec, M. Bálek, O. Jašek, Deep convolutional malware classifiers can learn from raw executables and labels only, in: ICLR Workshop, 2018.

[5] S. E. Coull, C. Gardner, Activation analysis of a byte-based deep neural network for malware classification, in: 2019 IEEE Sec. and Priv. Workshops (SPW), IEEE, 2019, pp. 21–27.

[6] D. Gibert, C. Mateu, J. Planes, The rise of machine learning for detection and classification of malware: Research developments, trends and challenges, Journal of Network and Computer Applications 153 (2020) 102526.

[7] L. Demetrio, B. Biggio, G. Lagorio, F. Roli, A. Armando, Explaining vulnerabilities of deep learning to adversarial malware binaries, in: ITASEC, 2019.

[8] S. Bose, T. Barao, X. Liu, Explaining ai for malware detection: Analysis of mechanisms of malconv, in: 2020 international joint conference on neural networks (IJCNN), IEEE, 2020, pp. 1–8.

[9] M. Sundararajan, A. Taly, Q. Yan, Axiomatic attribution for deep networks, in: Int. Conf. on Machine Learning (ICML), 2017, pp. 3319–3328.

[10] H. S. Anderson, A. Kharkar, B. Filar, P. Roth, Evading machine learning malware detection, Black Hat 2017 (2017) 1–6.

[11] L. Demetrio, B. Biggio, G. Lagorio, F. Roli, A. Armando, Functionality-preserving black-box optimization of adversarial windows malware, IEEE Transactions on Information Forensics and Security 16 (2021) 3469–3478.

[12] D. Arp, E. Quiring, F. Pendlebury, A. Warnecke, F. Pierazzi, C. Wressnegger, L. Cavallaro, K. Rieck, Dos and don'ts of machine learning in computer security, in: 31st USENIX Security Symposium (USENIX Security 22), 2022, pp. 3971–3988.

[13] D. Trizna, Quo vadis: hybrid machine learning meta-model based on contextual and behavioral malware representations, in: Proceedings of the 15th ACM Workshop on Artificial Intelligence and Security, 2022, pp. 127–136.

[14] Y. N. Dauphin, A. Fan, M. Auli, D. Grangier, Language modeling with gated convolutional networks, in: International conference on machine learning, PMLR, 2017, pp. 933–941.