Malware Classification using Transfer Learning through the GPT-2 model

Matheus Vanzan, Julio Cesar Duarte

Instituto Militar de Engenharia (IME)
Praça Gen. Tibúrcio, 80 – 22.290-270 – Urca, Rio de Janeiro - RJ – Brazil

{vanzan.matheus,duarte}@ime.eb.br

Abstract. *Malware detection and classification pose critical challenges in the field of cybersecurity. In recent years, deep learning techniques have made remarkable progress in addressing the classification problem, outperforming traditional methods. Moreover, Natural Language Processing has proven successful in extending its applications beyond natural language texts across numerous semantic domains. This research work focuses on presenting a proposal that extends the Transfer Learning from OpenAI's GPT-2 model to identify different malware families, without prior knowledge of their behaviors. The achieved results are highly promising, with an exceptional accuracy rate of 99.72%, close to state-of-the-art results reported for the problem.*

1. Introduction

Despite ongoing advancements in information security research, malicious files remain a constant threat in the digital environment. Moreover, the annual creation of malware files demonstrates an exponential growth pattern. By the end of 2022, the global detection of new malware files had already surpassed one billion programs [AV-Test 2023]. Figure 1 depicts the cumulative malware detection worldwide from January 2008 to February 2023.

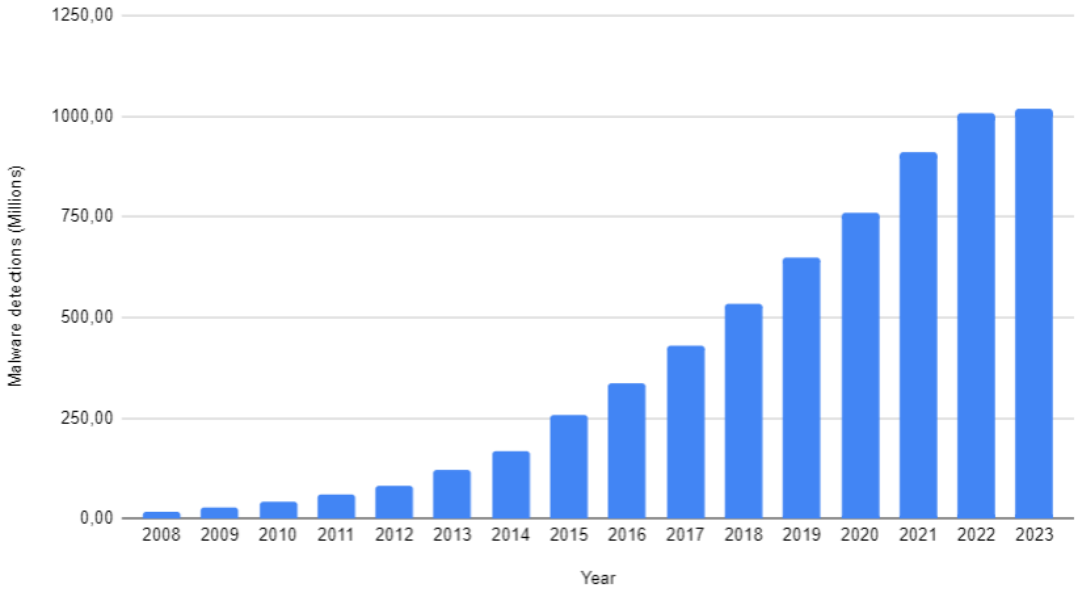

Figure 1. Malware detection by year. Source: [AV-Test 2023].

In this current landscape, tasks such as malware classification are of utmost importance for protecting systems and devices against potential threats. Traditionally, one of the most common techniques for malware classification involves the utilization of digital signatures, matching a set of attributes from a given sample with a previously known database. However, code obfuscation techniques can modify software signatures while maintaining its behavior. As an alternative approach, Deep Learning techniques have been employed as a viable solution for malware identification and classification by automatically learning relevant attributes [Zhang 2000].

As text processing techniques have been refined, their applications have expanded into various semantic fields. By treating raw data collections as text, it is possible to obtain more useful contextual information by extracting attributes that would otherwise be difficult to select manually [Damodaran et al. 2022].

In addition to malware detection, it is crucial to understand the types of malware and how those impact a device. In an era marked by stringent data protection laws, concerns regarding data extraction attacks have become imperative in any network, escalating the need for the classification of threats.

Considering this new scenario, this research work proposes a novel method for malware classification using transfer learning from the Transformers architecture and the GPT-2 model without prior knowledge of malware behavior. Through the use of this new approach, experiments are performed on a widely utilized dataset, presenting exceptional results, closely aligned with the state-of-the-art reported for the task. Although a pure Deep Learning approach may be insufficient to determine the behavior of file samples in a real-world scenario, it can be of utmost value when combined with other methods in order to create complete anti-malware solutions.

The structure of this paper is organized as follows: Section 2 contains a brief literature review to provide a foundation for the subsequent sections. Next, Section 3 presents related work that approaches malware classification. Section 4 describes the proposed method, while Section 5 presents the experiments conducted and the analysis of their results. Finally, section 6 provides a conclusion and suggestions for future work.

2. Basic Concepts

To better understand the subsequent sections of this paper, it is important to define some key concepts and provide corresponding definitions. This section aims to cover the following ideas: malware, the BIG 2015 dataset, the Transformers architecture, and the GPT-2 model.

2.1. Malware

Derived from **mal**icious soft**ware**, malware can be intuitively defined as software that conducts malicious attacks on other software systems, with the term *malicious* referring to any behavior outside the normal intended scope. However, identifying malware poses a significant challenge due to the inherent difficulty of defining the intended behavior of individual software and subsequently detecting anomalies [Kramer and Bradfield 2010].

A malware file is capable of infecting devices and is designed to cause harm to these devices, networks, or their users in various ways. Depending on the malware type

and its goal, the damage caused can be manifested differently for the end user, ranging from relatively mild to potentially catastrophic [Chang et al. 2013]. Regardless of the specific method, malware is developed to exploit devices to the detriment of the user and for the benefit of its creator or deployer [Kramer and Bradfield 2010].

2.2. BIG 2015 Malware Dataset

The BIG 2015 dataset, consisting of a large collection of malware samples, was originally presented in the Microsoft Malware Classification Challenge [Ronen et al. 2018]. This dataset has seen extensive usage and citation across several works following the main competition, greatly enabling the comparison of different approaches to malware classification. Furthermore, its open accessibility significantly enhances the potential for replicating results, unlike other private datasets. The dataset consists of 10,868 ASM files (text files) generated using the IDA Pro software (Interactive Disassembler) [Eagle 2011], classified into nine distinct malware families, as outlined in Table 1.

Table 1. BIG 2015 malware families.

Id	Family name	Number of files
1	Ramnit	1541
2	Lollipop	2478
3	Kelihos_ver3	2942
4	Vundo	475
5	Simda	42
6	Tracur	751
7	Kelihos_ver1	398
8	Obfuscator.ACY	1228
9	Gatak	1013

The IDA Pro software was responsible for reverse engineering the compilation process and generating output in a high-level language [Eagle 2011]. A snippet of an ASM file from the dataset is presented in Figure 2. Each IDA output file in the dataset consists of six types of segments (CODE, DATA, CONST, BSS, STACK, XTRN) that serves to separate different instruction sets. The code segments specifically contain the most relevant instructions for malware analysis and classification.

2.3. The Transformers Architecture

Natural Language Processing (NLP) is an Artificial Intelligence branch that focuses on the automatic understanding of texts written in natural language. One of its goals is to extract meaning from text, including keywords, context, or the sentiment expressed within it, among other tasks. Furthermore, NLP can analyze textual representations of other structures, such as source code files.

Before the introduction of the Transformers Architecture, models applying NLP were mostly based on Recurrent Neural Networks (RNN), such as Long Short-Term Memory (LSTM) [Gers 1999] and Gated Recurrent Units (GRU) [Wang et al. 2018]. These models typically relied on sequential token processing, which posed a challenge

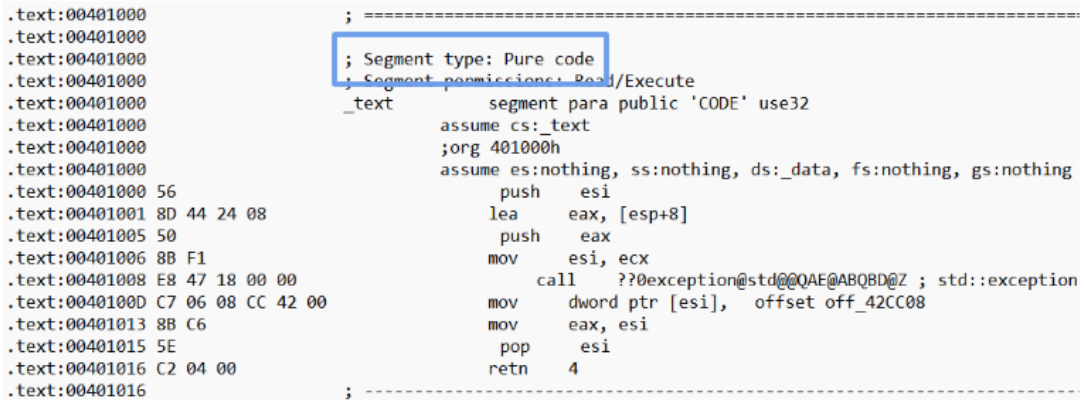

Figure 2. BIG 2015 ASM file excerpt.

to parallelism in their training, becoming critical as the number of training samples grew [Yang et al. 2020].

The Transformers Architecture [Vaswani et al. 2017] addresses this issue by eliminating the requirement for recurrent or convolutional layers and introducing the concept of **Attention Mechanisms**, which assign weights to each token in an input sequence. Unlike traditional models that iteratively update the network's internal state during token processing, the transformers model processes tokens in parallel, considerably reducing processing time for datasets. This increase in parallel processing capabilities allows transformers models to efficiently handle larger volumes of data, making them more adequate for tasks that involve extensive processing, such as malware classification.

2.4. The GPT-2 Model

While textual data is abundant in today's world, the same cannot be said for structured or labeled data. Getting this labeled data can be challenging, especially when it comes to specific domains, as a significant portion of the available data lacks any form of classification or metadata that could aid in tasks such as classification or semantic extraction.

The Generative Pre-Trained Transformer (GPT) model, which was introduced by OpenAI in 2018 [Radford et al. 2018], presented an innovative approach in which unsupervised pre-training could be performed on a text corpus without any additional data, apart from the text itself. Specifically, the GPT was trained to complete text through an automatic process of generating inputs and labels from the original text. Once a pre-trained model is obtained, it can be fine-tuned through supervised learning for a specific task.

Semi-supervised and data-agnostic learning models have been previously introduced to enhance performance in text sequencing tasks [Dai and Le 2015]. However, the uniqueness of GPT lies in using the Transformers architecture for training, which yielded superior results by establishing a new state of the art.

In 2019, a newer version of the model was released to the community, called GPT-2, with up to 1.5 billion parameters in its largest version [Radford et al. 2019]. This version of the GPT was trained using the WebText dataset, curated by the authors using a compilation of publicly available web pages, resulting in approximately 40 GB of textual content. However, the contents of the dataset were not made publicly available.

It is important to highlight, within the context of this paper, that a portion of the training data for the GPT-2 model originated from code repositories such as GitHub, Stack Exchange, and Stack Overflow; evidenced from the list of the top 1,000 domains[1] used in creating the dataset. Furthermore, it is also worth mentioning that even newer versions namely GPT-3 and GPT-4 have been released, however, these models have not been made available as open-source models but rather as closed applications as OpenAI has emphasized the need for careful consideration of ethical and safety concerns associated with language models.

3. Related Work

It is noteworthy that since the conclusion of the Microsoft Malware Classification Challenge in April 2015 [Ronen et al. 2018], the competition and the BIG 2015 dataset have been referenced in more than 50 works in the field of Cybersecurity [Ronen et al. 2018]. These references highlight the widespread utilization of the dataset as a resource for further progress in the field. In light of this, six relevant studies were chosen based on their utilization of the BIG 2015 dataset and their application of transfer learning techniques.

During the original competition in 2015, the winning team developed a complex approach using multiple manually extracted features, including opcode n-grams, segment counts, pixel intensity of ASM files, 4-gram bytes, single-byte frequencies, function names, and assembly resources to classify malware samples [Kaggle Team 2022]. The generation of these features required 200 GB of disk space, and the model took 72 hours for a training session, achieving an accuracy rate of up to 99.87%. Even without using transfer learning, the results achieved by the winning team remained state-of-the-art for a considerable period.

In 2018, [Kim et al. 2017] proposed a model based on Generative Adversarial Networks (GAN) using images generated from malware files. The model generates new data samples that resemble the training set, intending to distinguish between original and artificially generated images. Among the advantages of this approach, it is worth emphasizing that it does not rely on any specific domain knowledge of the problem. The authors conducted several experiments and achieved an accuracy rate of up to 96.39%.

[Cakir and Dogdu 2018] applied a word2vec technique to encode opcodes, generating syntactic and semantic relationships between words in the ASM files of the dataset. Using the generated vectors, they applied the Gradient Boosting Machine (GBM) algorithm and achieved accuracy rates ranging from 94% to 96%.

In a similar manner to [Kim et al. 2017], a Convolutional Neural Network (CNN) approach was proposed by [Kalash et al. 2018] using the VGG16 [Simonyan and Zisserman 2015] model to classify images. Notably, their results outperformed most feature-based approaches, achieving an accuracy rate of up to 99.97%.

In 2020, [de Albuquerque et al. 2020] proposed an opcode analysis from the malware dataset, also employing word2vec for encoding. A predictive LSTM structure was used to forecast the sequence of opcodes in the malware samples. The proposed method achieved an accuracy rate of up to 92%.

[1] https://github.com/openai/gpt-2/blob/master/domains.txt

Despite not using the BIG 2015 dataset, [Şahin 2021] proposed a method for malware detection based on GPT-2 transfer learning. The assembler code was obtained from *.text* sections of malware samples and the model was fine-tuned for malware detection in two steps. In the first step, the complete unlabeled dataset was submitted to the GPT-2 model generating a new pre-trained model. The dataset was then split into train and test subsets, and submitted with its corresponding labels into the custom pre-trained model. The proposed method achieved an accuracy rate of up to 85.4%.

During the search for related works, it has been discovered that both transfer learning and feature selection approaches have demonstrated the ability to achieve state-of-the-art results in several domains. However, most transfer learning solutions do not require domain knowledge. This ability to classify samples without prior knowledge of malware file structure is crucial for a model capable of generalizing new, unseen files. Furthermore, it became apparent that there is a limited number of publications exploring the application of transfer learning and the GPT model for malware classification. This scarcity reinforces the necessity for additional research in this particular field.

Considering this perspective, the proposed method outlined in this paper focuses on a solution that takes advantage of transfer learning and the inherent capabilities of the GPT-2 model by using only the source code of malware samples. Distinguishing itself from other approaches, it does not rely on any prior knowledge of the executable sample's behavior, nor does it employ any manual feature engineering.

4. Proposed Method

In this research work, a novel method is introduced for fine-tuning the GPT-2 model for the task of malware classification. The method is comprised of six phases, illustrated in Figure 3, covering all the required steps from dataset preprocessing to sample evaluation. The dashed lines involving phases indicate the possibility of a combined execution.

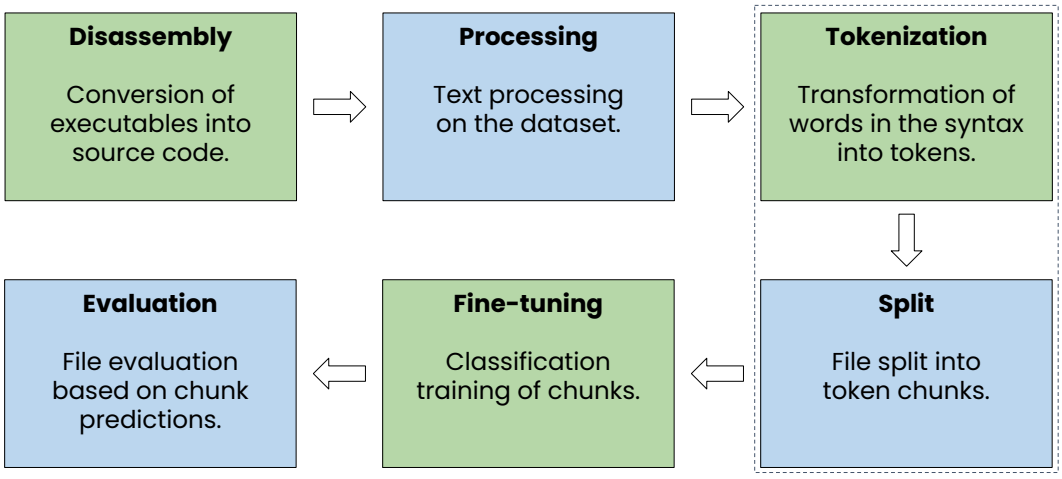

Figure 3. Proposed method execution flow.

Disassembly The method begins with disassembling executable files into high-level Assembly source code. This process entails the conversion of each malware sample from an

executable file format to a text file format, containing its assembler source code. It is worth mentioning that some datasets may already provide disassembled files, eliminating the need for this phase.

Processing Following the disassembly phase, text processing routines are employed to eliminate redundant words from the dataset, filtering out specific terms that hold no significance for the subsequent steps. Spaces, punctuation marks, special characters, and non-ASCII characters are systematically removed as well. Lastly, the resulting text is filtered, restricting words to only the assembler instruction set (i.e., opcodes and registers) and their adjacent words.

Tokenization and Split To streamline the process and improve efficiency, the Tokenization and Split phases can be combined into a single procedure. Here, each malware sample is transformed from its assembler syntax into tokens using the library's default tokenizer *GPT2Tokenizer* [Hugging Face 2021]. Figure 4 depicts an assembler code excerpt after the tokenization process, illustrating the tokenizer's ability to handle other forms of text besides natural language. Following the tokenization, the set of obtained tokens is split into multiple chunks, where each chunk contains an equal number of tokens. The split is performed to conform to the maximum token constraint imposed by the model for each sample. Before saving a newly created chunk sample, a truncation limitation can be imposed, reducing the overall size of the dataset. As a result, the new dataset consists of independent chunk files, with each chunk labeled according to its originating file class.

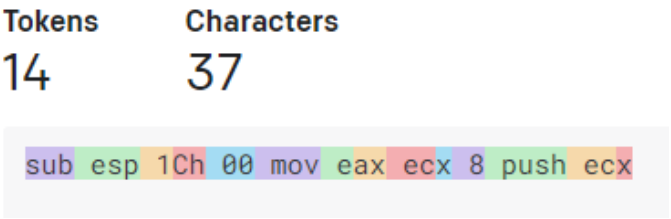

Figure 4. Tokens created from a code excerpt.

Fine-tuning After the creation of the chunk dataset, the fine-tuning phase begins, employing each chunk sample for classification training purposes. In this phase, a classification layer is added on top of the original generative model and the training process determines the internal weights of the classification layer, similar to conventional neural network training techniques. As a result of the fine-tuning phase, a set of predictions is generated, one for each chunk sample. Figure 5 presents a general diagram of the training strategy employed for the dataset, with chunk colors indicating sample predictions for each chunk after training, and file colors indicating the final prediction.

Evaluation After obtaining a set of chunks and their corresponding predictions, chunks are correlated back to their original malware samples. As a result, for each original malware sample, a collection of prediction classes is obtained based on the predictions made

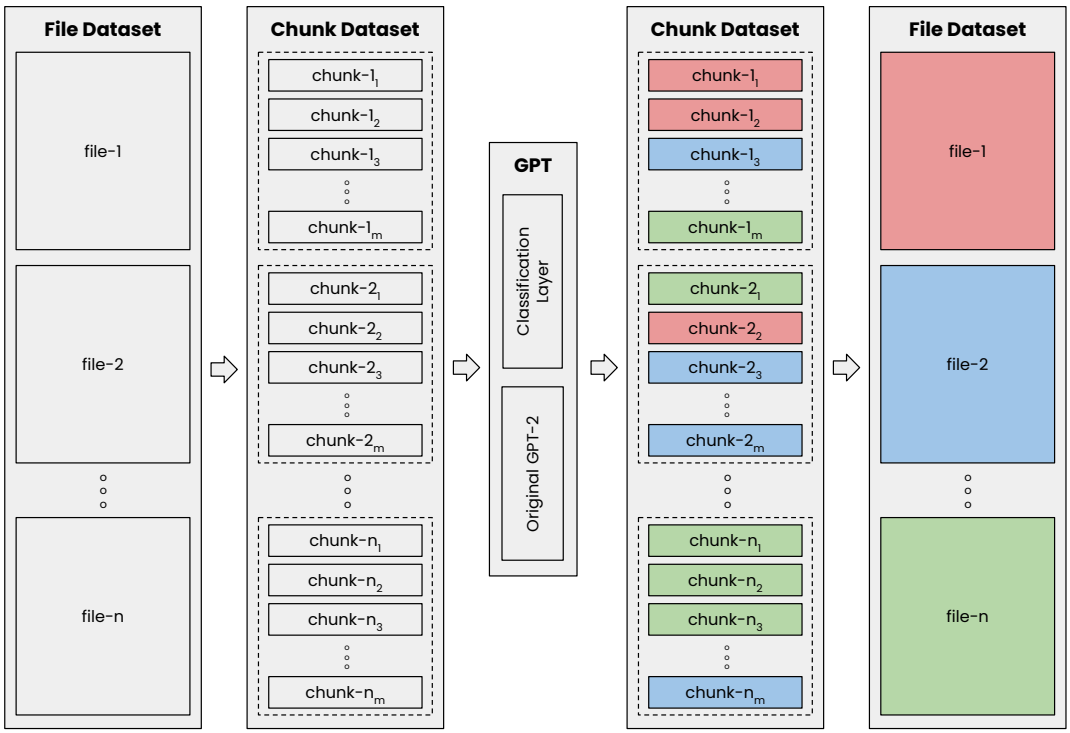

Figure 5. General diagram of the proposed method for fine-tuning.

for its associated chunks. Each chunk's prediction represents a potential classification for the original malware sample. To determine the final prediction for each malware sample, a decision is made by considering the collective predictions from its internal chunks. The class that appears most frequently among the predictions is then selected as the final prediction. By considering the majority vote among the internal chunks, this approach provides a consolidated prediction for each malware sample.

5. Experiments

To evaluate the proposal presented in Section 4, with the main objective of fine-tuning the GPT-2 model for malware classification, several testing experiments were conducted, varying the dataset truncation values and the number of epochs. Hardware limitations and execution times were primary factors for the training strategies. Finally, two experiments yielded significantly good results using the 124 million parameters version of the model.

5.1. Hardware and Software

The experiments were conducted in the free Google Colab environment [Bisong 2019], with a Tesla T4 16 GB GPU. The project's code was developed using Python 3.8 [Van Rossum and Drake 2009], and the Transformers Library [Wolf et al. 2020], version 2.2.2 and it is publicly available in a GitHub repository [Vanzan 2023].

5.2. Dataset Processing

In the **first experiment**, the dataset was split into three partitions for training, evaluation, and testing, comprising 80%, 10%, and 10% of the total number of files, respectively. The division was performed randomly, maintaining a stratified distribution of classes in the

dataset. Similarly, a **second experiment** was performed using a 10-fold cross-validation, with the first partition matching the split used in the first experiment.

Due to extensive training time, some hyperparameters were fixed, as testing all combinations of the hyperparameters would not be feasible. Initially, the **chunk size** was set to **32** tokens, and the number of **batches** was set to **160**. These fixed hyperparameters were chosen to balance training efficiency and model performance. Additionally, a **token limit** of **102,400** was set as a truncation point for the files to ensure that the GPU could handle the dataset load into memory. In previous experiments, a higher truncation point resulted in memory overflow.

After the complete processing of the dataset, its total size was reduced to 2.68 GB. Figure 6 presents a histogram of token volume per file (left), and the same volume in a logarithmic scale, with the truncation limit highlighted in red (right). This shows that less than 15% of the dataset was affected by truncation.

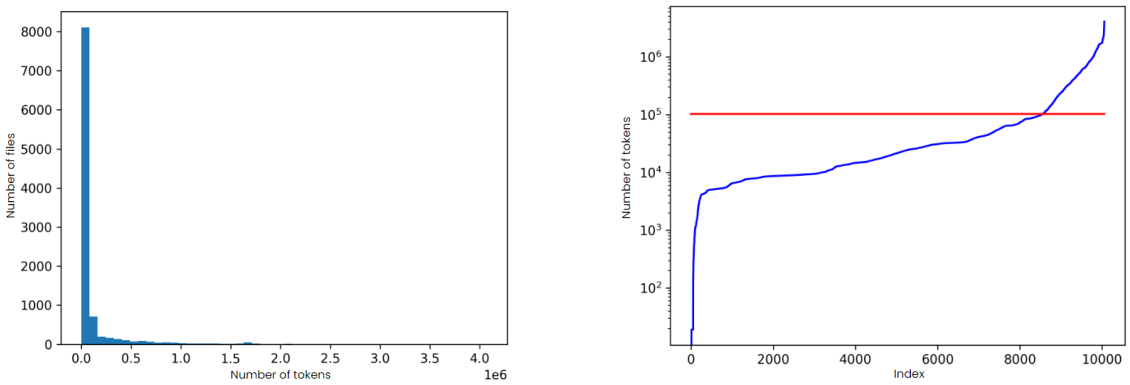

Figure 6. Dataset chunk volume (left) and dataset truncation (right).

5.3. Performance Metrics

The main performance metric taken into consideration is **accuracy**, computing **True Positives** over all samples. Considering the inherent class imbalance in the dataset, the **F1-score** for each malware class was also evaluated to assess the impact of the imbalance on **False Positives** and **False Negatives**.

5.4. Results

For the **first experiment**, the number of epochs varied from one to five. Table 2 presents the obtained results, including accuracy rates and training time. Additionally, each test phase of the experiment lasted for one hour, regardless of the number of training epochs. As observed, training time increased linearly with the number of epochs, providing a good predictability of the time required for longer training sessions. This information can be valuable for planning and scheduling training sessions in future experiments.

The best results were achieved with five epochs, reaching an accuracy rate of 99.72%. This indicates that, within the tested range, increasing the number of epochs improved the performance without causing overfitting. Figure 7 supports this reasoning by illustrating the training loss curve. It can be observed that the curve has not yet

Table 2. Experiment 1 - Accuracy and training time per epoch.

Epochs	Accuracy	Time (h)
1	99.26	17
2	99.35	34
3	99.45	52
4	99.63	67
5	99.72	85

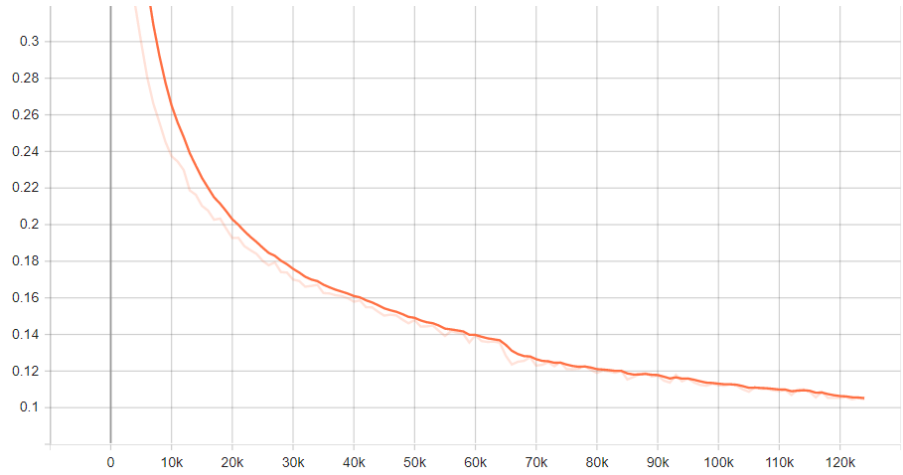

Figure 7. Experiment 1 - Loss curve for five epochs.

reached a strictly asymptotic behavior, which suggests that additional training iterations can potentially lead to even better results.

As mentioned in 5.3, the F1-score measures are presented in Table 3. Upon observation, it is evident that increasing the number of epochs has a positive impact on certain classes, notably class 5 (Simda), while potentially causing detrimental effects on others, such as class 6 (Tracur). This suggests that the model's training process has a varying impact on different malware classes, emphasizing the importance of class-specific performance metrics. By evaluating the performance of each class, a more comprehensive understanding of the model's capabilities and limitations can be acquired. This allows deep analysis of model performance across distinct malware classes, enabling focused refinements and optimizations.

For the **second experiment**, the training sessions were performed using a single epoch, enabling the adoption of a K-fold cross-validation with ten folds. Table 4 presents the accuracy rates for each fold. The average training time was similar to that observed in the first experiment for a single epoch.

The experiment yielded a minimum accuracy rate of 97.95% and a maximum of 99.26%. This demonstrates that even with the variation in the random distribution of the cross-validation, the results consistently remain above 97%. The adoption of the 10-Fold cross-validation helped provide a more robust evaluation of the model's performance by considering multiple splits. This approach helps evaluate the generalization capabilities

Table 3. Experiment 1 - F1 (%) per class per epoch.

Epochs	F1-1	F1-2	F1-3	F1-4	F1-5	F1-6	F1-7	F1-8	F1-9
1	98.72	99.39	100.00	100.00	85.71	100.00	97.37	97.93	99.51
2	99.04	99.39	100.00	100.00	85.71	100.00	97.37	98.35	99.51
3	99.04	99.39	100.00	100.00	85.71	100.00	97.37	98.77	100.00
4	99.35	99.60	100.00	100.00	100.00	100.00	98.70	98.77	100.00
5	99.68	99.60	100.00	100.00	100.00	99.34	100.00	99.17	100.00

Table 4. Experiment 2 - 10-fold accuracy.

Fold	1	2	3	4	5	6	7	8	9	10
Acc	99.26	97.95	99.07	98.15	98.70	98.06	98.98	98.80	98.34	98.89
Avg	98.62									

of the model across different data partitions and provides a more comprehensive understanding of its effectiveness in real-world scenarios.

Similar to the previous experiment, the F1-scores were also evaluated, as shown in Table 5. On average, the F1 scores presented a minimum value of 84.21%. These scores provide a more detailed understanding of the model's performance on individual malware classes, highlighting its ability to correctly classify instances from each class.

Table 5. Experiment 2 - F1 (%) per class per fold.

Fold	F1-1	F1-2	F1-3	F1-4	F1-5	F1-6	F1-7	F1-8	F1-9
1	98.72	99.39	100.00	100.00	85.71	100.00	97.37	97.93	99.51
2	96.23	99.21	100.00	96.23	90.91	95.95	96.70	94.57	99.04
3	96.77	99.80	100.00	100.00	85.71	100.00	100.00	97.91	98.51
4	95.33	99.60	100.00	96.91	100.00	99.33	96.00	93.56	99.50
5	97.12	99.39	99.83	96.91	100.00	98.65	98.70	97.07	99.00
6	95.62	100.00	99.83	94.38	75.00	97.99	100.00	92.89	100.00
7	98.39	99.40	99.83	98.95	66.67	97.99	97.44	99.17	98.49
8	96.23	99.80	100.00	98.95	66.67	100.00	98.70	97.48	98.49
9	97.14	99.39	99.49	94.85	85.71	98.01	98.70	95.32	100.00
10	98.06	99.59	100.00	98.92	85.71	99.33	94.59	96.77	99.50
Avg	**96.96**	**99.56**	**99.90**	**97.61**	**84.21**	**98.73**	**97.82**	**96.27**	**99.20**

The experiments' results provide strong evidence for the robustness and effectiveness of the proposed solution in classifying malware instances, even across different dataset splits in the cross-validation process. The consistently high accuracy rates of up to 99.72% in the first experiment and relatively high F1 scores indicate its generalization capacity. Although the second experiment produced slightly lower results, its execution was vital to assess the consistency of the solution across different dataset splits within a

reasonable time frame.

By conducting the 10-fold cross-validation, the potential impact of random factors was minimized, allowing for a more reliable assessment of the model's performance. Additionally, it is worth noting that, unlike other approaches, the proposed method does not rely on feature selection or any other form of previous behavior knowledge to achieve its results. Instead, it exclusively utilizes the sample source code for the classification task, as proposed by this work.

6. Conclusion

In the current landscape, the propagation of new digital threats is continuously increasing, emphasizing the necessity for novel detection methods, specifically, methods that do not rely on specific knowledge of malware behavior. This research work presents a new method for malware classification that uses Transfer Learning from the OpenAI GPT-2 model to accurately identify several types of malware families. The study yielded highly satisfactory results, which were consistent with the findings of previous research conducted in this domain.

To validate the proposed method, several experiments were conducted, resulting in two final experiments that yielded significant results, as described in Section 5. In the first experiment, the effectiveness of the method was assessed by varying the number of epochs, resulting in an accuracy rate of 99.72%. The second experiment, nonetheless, utilized the 10-fold cross-validation technique to assess the impact of dataset distribution across different folds, resulting in an average accuracy rate of 98.62%. It is worth noting that the fine-tuning process was successfully executed using readily available and completely cost-free hardware resources, showcasing the possibilities of using more complex hardware. Despite being limited by these constraints, the GPT-2 model presented its ability to comprehend the textual content of malware sample source code effectively.

Additionally, there are still opportunities for further improvement in this research. Future studies could include experiments using the complete BIG 2015 dataset, eliminating truncation limitations, along with other datasets, and removing any possible information bias of the current dataset regarding malware families. These additional experiments would offer valuable insights into the proposed method, enabling a more comprehensive evaluation of its generalization capabilities. Alternatively, conducting experiments with narrowed truncation limitations, which focus only on the first tokens of each file, would allow an evaluation of the minimum token volume necessary for satisfactory prediction results. This approach would provide faster results and enable a broader range of hyperparameter exploration. These future directions hold the potential to significantly contribute to the ongoing research in the field of malware classification. These efforts not only enhance the understanding of the proposed method but also expand its practical applicability, possibly extending its potential utilization to other domains.

Acknowledgments

This material is based upon work supported by the Air Force Office of Scientific Research under award number FA9550-22-1-0475. In addition, this work was partially supported by national funds through FINEP, Financiadora de Estudos e Projetos, and FAPEB, Fundação de Apoio à Pesquisa, Desenvolvimento e Inovação do Exército Brasileiro, under

project "*Sistema de Sistemas de Comando e Controle*" with reference nº 2904/20 under contract nº 01.20.0272.00.

References

AV-Test (2023). Av-test statistics: Malware. https://www.av-test.org/en/statistics/malware/. Accessed on May 12, 2023.

Bisong, E. (2019). Google colaboratory.

Cakir, B. and Dogdu, E. (2018). Malware classification using deep learning methods. In Proceedings of the ACMSE 2018 Conference, pages 1–5.

Chang, J., Venkatasubramanian, K. K., West, A. G., and Lee, I. (2013). Analyzing and defending against web-based malware. ACM Computing Surveys (CSUR), 45(4):1–35.

Dai, A. M. and Le, Q. V. (2015). Semi-supervised sequence learning. Advances in neural information processing systems, 28.

Damodaran, A., Troia, F. D., Corrado, V. A., Austin, T. H., and Stamp, M. (2022). A comparison of static, dynamic, and hybrid analysis for malware detection. Journal of Computer Virology and Hacking Techniques.

de Albuquerque, D. G., Vieira, L. d. Q., Sant'Ana, R., and Duarte, J. C. (2020). Análise de comportamento de malware utilizando redes neurais recorrentes - uma abordagem por intermédio da previsão de opcodes. Revista Militar de Ciência e Tecnologia, 37(3).

Eagle, C. (2011). The IDA Pro Book. William Pollock, San Francisco, second edition edition.

Gers, F. A. (1999). Learning to forget: Continual prediction with lstm. In 9th International Conference on Artificial Neural Networks: ICANN '99. IET.

Hugging Face (2021). Hugging face transformers: Gpt-2 documentation. https://huggingface.co/docs/transformers/model_doc/gpt2. Accessed on March 20, 2023.

Kaggle Team (2022). Microsoft malware winners' interview: 1st place, "no to overfitting!". https://medium.com/kaggle-blog. Accessed on June 20, 2022.

Kalash, M., Rochan, M., Mohammed, N., Bruce, N. D., Wang, Y., and Iqbal, F. (2018). Malware classification with deep convolutional neural networks. In 2018 9th IFIP international conference on new technologies, mobility and security (NTMS), pages 1–5. IEEE.

Kim, J.-Y., Bu, S.-J., and Cho, S.-B. (2017). Malware detection using deep transferred generative adversarial networks. In International Conference on Neural Information Processing.

Kramer, S. and Bradfield, J. C. (2010). A general definition of malware. Communications of the ACM.

Radford, A., Narasimhan, K., Salimans, T., Sutskever, I., et al. (2018). Improving language understanding by generative pre-training.

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., Sutskever, I., et al. (2019). Language models are unsupervised multitask learners. OpenAI blog, 1(8):9.

Ronen, R., Radu, M., Feuerstein, C., Yom-Tov, E., and Ahmadi, M. (2018). Microsoft malware classification challenge. http://arxiv.org/abs/1802.10135/.

Şahin, N. (2021). Malware detection using transformers-based model gpt-2. Master's thesis, Middle East Technical University.

Simonyan, K. and Zisserman, A. (2015). Very deep convolutional networks for large-scale image recognition.

Van Rossum, G. and Drake, F. L. (2009). Python 3 Reference Manual. CreateSpace, Scotts Valley, CA.

Vanzan, M. (2023). GPT-2 Malware Classification Github Repository. https://github.com/matheusvanzan/gpt-2-malware-classification.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. (2017). Attention is all you need. Advances in neural information processing systems, 30.

Wang, N., Wang, J., and Zhang, X. (2018). Ynu-hpcc at semeval-2018 task 2: Multi-ensemble bi-gru model with attention mechanism for multilingual emoji prediction. In Proceedings of The 12th International Workshop on Semantic Evaluation, pages 459–465.

Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., Cistac, P., Rault, T., Louf, R., Funtowicz, M., Davison, J., Shleifer, S., von Platen, P., Ma, C., Jernite, Y., Plu, J., Xu, C., Scao, T. L., Gugger, S., Drame, M., Lhoest, Q., and Rush, A. M. (2020). Transformers: State-of-the-art natural language processing. In Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations, pages 38–45, Online. Association for Computational Linguistics.

Yang, S., Yu, X., and Zhou, Y. (2020). LSTM and GRU neural network performance comparison study: Taking Yelp review dataset as an example. In 2020 International workshop on electronic communication and artificial intelligence (IWECAI), pages 98–101. IEEE.

Zhang, G. P. (2000). Neural networks for classification: a survey. IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews), 30(4):451–462.