Article

Rapidrift: Elementary Techniques to Improve Machine Learning-Based Malware Detection

Abishek Manikandaraja *, Peter Aaby * and Nikolaos Pitropakis *

School of Computing, Engineering & the Built Environment, Edinburgh Napier University, Edinburgh EH10 5DT, UK
* Correspondence: 40450966@live.napier.ac.uk (A.M.); p.aaby@napier.ac.uk (P.A.); n.pitropakis@napier.ac.uk (N.P.)

Abstract: Artificial intelligence and machine learning have become a necessary part of modern living along with the increased adoption of new computational devices. Because machine learning and artificial intelligence can detect malware better than traditional signature detection, the development of new and novel malware aiming to bypass detection has caused a challenge where models may experience concept drift. However, as new malware samples appear, the detection performance drops. Our work aims to discuss the performance degradation of machine learning-based malware detectors with time, also called concept drift. To achieve this goal, we develop a Python-based framework, namely Rapidrift, capable of analysing the concept drift at a more granular level. We also created two new malware datasets, TRITIUM and INFRENO, from different sources and threat profiles to conduct a deeper analysis of the concept drift problem. To test the effectiveness of Rapidrift, various fundamental methods that could reduce the effects of concept drift were experimentally explored.

Keywords: malware; machine learning; PE malware; Rapidrift; inferno; tritium; windows malware; malware dataset

check for updates

Citation: Manikandaraja, A.; Aaby, P.; Pitropakis, N. Rapidrift: Elementary Techniques to Improve Machine Learning-Based Malware Detection. *Computers* **2023**, *12*, 195. https://doi.org/10.3390/computers12100195

Academic Editor: Paolo Bellavista

Received: 6 August 2023
Revised: 11 September 2023
Accepted: 19 September 2023
Published: 28 September 2023

Copyright: © 2023 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

The increased adoption of Internet of Things (IoT) devices and the introduction of artificial intelligence (AI) to people's everyday lives has created an unbreakable bond. Machine learning (ML) solutions are continuously being integrated into devices and software solutions, making them more popular than they have ever been in the past. However, this increased attention has also attracted malicious parties who take advantage of the existing threat landscape to serve their cause. Computational infrastructures around the globe are susceptible to attacks. Recently, a cyber-attack on a major Yorkshire Coast Company has been thwarted by officers from North Yorkshire Police [1], while a cyber-attack has hit twelve Norwegian government ministries [2].

Malicious applications or malware have become popular with new technological artefacts, frameworks, and applications. Malware is a class of programs developed and propagated to gain illicit access to systems, exfiltrate information, and perform other tasks without the user's consent. The delivery of payloads could occur through various techniques such as VBA, APK files, PE files, HTA files, PDF, and others. Moreover, some malware uses uncommon file types to attack [3].

The increasing variety of methods to deliver malicious payloads makes it harder to prepare datasets for training machine learning/deep learning models. The difference in feature representation is the reason behind such limitations. For instance, features used in Windows malware cannot be applied to Android or PDF-based malware. Such inconsistencies require the development of different datasets for each use case.

In the malware threat landscape, it is common for classifiers to degrade with time as new malware samples emerge. This phenomenon is called concept drift and is a commonly observed problem in malware detection. The development of malware that takes advantage of new weaknesses or targets a particular demography of users demands updating the model with new data, retraining the model, or training new models with the latest malware samples. Since 2017, more benchmark malware datasets have been released to aid machine learning-based malware detection research. These datasets were evaluated using different methods and constraints.

Presumably, each dataset contains malware from various threat actors from different timelines (with some common malware families too). One of the critical issues in this domain is that data distribution varies rapidly, and there needs to be a standardised method to tackle or retain acceptable detection rates with time. Additionally, various types of biases affect the quality of experiments and produce impressive results in unrealistic configurations. Current research provides broader solutions, such as those [4], to improve detection against multiple malware types not restricted to Windows PE files. However, testing these solutions without extensive experimentation for each malware domain is complex due to different feature sets being used for every file type. Some feature sets might better represent the maliciousness of the file better than the other. The diverse variety of malware combined with its rapidly evolving nature is a challenge for machine learning-based malware detection.

In the scope of our work, fundamental methods used in machine learning solutions to improve classification quality are suggested and evaluated. We focus on Windows-based PE malware and how traditional threat intelligence could enhance machine learning approaches.

The contributions of our work can be summarised as follows:

1. We develop a Python-based framework to rapidly evaluate and build malware classifiers and analyse concept drift at a more granular level, namely Rapidrift.
2. We introduce two new novel malware datasets, TRITIUM, which is based on real-world data and INFERNO, which was developed with popular cybersecurity tools that allow the creation/modification of malicious payloads.
3. We experimentally evaluate the effectiveness of our methods using the framework and discuss the outcomes.

The rest of the paper is structured as follows. Section 2 briefly describes the related work in machine learning and malware detection, while Section 3 details our methodological approach. Section 4 explains the dataset creation while presenting our experimental design. Section 5 discusses the outcomes of our experiments, while Section 6 concludes while providing some pointers for future work.

2. Related Work

Various research has addressed the application of machine learning for malware classification problems. Multiple components constitute a machine learning solution, each tuned to achieve a specific objective. These components are discussed in the context of malware classification.

2.1. Classification Objectives

For a given machine learning-based malware classifier, there are primarily two objectives: binary classification and multi-class classification. Binary classification is the ability of the classifier to predict if a given binary is malware or benign. On the other hand, multi-class classification aims to classify malware into various categories, such as worms, ransomware, trojan, and others. However, the classification objective of this study would be binary classification.

In a multi-class approach, many classes would directly contribute to increased misclassification. Ref. [5] supports this with a slightly higher accuracy score for binary classification than multi-class. Moreover, the Blue Hexagon Open Dataset for Malware Analysis (BODMAS) [6] discusses the same problem and how it magnifies when introducing novel samples. The same study observed performance degradation with unseen malware families in an “open world” scenario.

2.2. Static and Dynamic Features

Malware features can be classified into static and dynamic features. Static features can be extracted without executing the PE file, while dynamic features can only be retrieved by observing the behaviour of the PE after execution. A few examples of both feature types are shown in Table 1.

Table 1. Static and dynamic features.

Static Features	Dynamic Features
Histograms	API calls
Byte Entropy	Registry modifications
Strings	DNS queries
File size	URLs accessed
PE Headers	Files modified
DLL Imports	

From a machine learning perspective, it would be ideal to have easily extractable feature vectors and, at the same time, to better represent the malware sample through them. A study [7] concluded that the accuracy of static malware analysis (99.36%) is slightly higher than that of a dynamic malware analysis (94.64%). However, a specific combination of both feature sets might improve accuracy. Extracting dynamic features is expensive in terms of time and computing resources [6]. In addition, the behaviour of malware, i.e., dynamic features, cannot be documented entirely for several reasons and be extracted by following standardised and straightforward methods. Malicious programs have evolved to limit their activity when they detect the presence of sandbox or virtual machines [7]. Ref. [8] discusses how malware does not consistently exhibit malicious behaviour. A common observation here is the unpredictable behaviour of malware that might make dynamic analysis methods less practical to be used as a feature set.

For the same reason, most existing PE datasets, including BODMAS, could be limited to static features. Unlike the Microsoft malware classification challenge and EMBER [9] datasets, BODMAS contains the original PE files and feature vectors. The availability of the original PE files could be helpful in further research that might involve newer feature selection/extraction methods or for possible breakthroughs in dynamic analysis.

2.3. Feature Selection and Extraction

Engineering a feature vector to represent and better differentiate between a malicious and a benign binary is a different problem. A few studies [10–13] have addressed this by evaluating other feature selection methods for malware. However, it would be convenient to follow the same feature vector throughout different datasets for a fair comparison of results. In this study, the feature vector used in the EMBER dataset was used consistently for all the experiments. The study associated with EMBER has provided the code for feature extraction from raw PE binaries. Other datasets, such as SOREL and BODMAS, released after EMBER, follow the same feature format. The new datasets introduced in this study were also consistent with the EMBER feature vector for fair experimentation.

2.4. Dataset Comparison

The quality of the dataset plays a crucial role in feature selection. A good feature set would improve accuracy and reduce learning time [14]. Four commonly used datasets for PE-based malware detection purposes in chronological order are the Microsoft malware classification challenge [15], EMBER [9], SOREL [16] and BODMAS [6]. It is important to note that some research articles refer to the Microsoft malware classification dataset as the BIG-15 dataset. Of all the datasets shown in Table 2, EMBER has 300,000 unlabelled samples.

Table 2. Comparison of PE-based malware datasets used in machine learning.

Dataset	Time-Range	Total Sample Count	Malicious Count	Benign Count
BIG-15	2015	20,000	20,000	20,000
EMBER	2017–2018	1,100,000	400,000	400,000
SOREL-20M	2018–2020	19,389,877	9,470,626	9,919,251
BODMAS	2019–2020	134,435	57,293	77,142

BODMAS is relatively more recent than other datasets and has been used in this study to maintain relevance to the evolving nature of malware. BODMAS is a milestone in releasing Windows malware datasets containing accurate family information about each sample. SOREL-20M would be a good candidate for evaluation. However, the dimensions of the dataset and its respective metadata are enormous (close to 20 million samples). The dataset size makes it challenging to train, test, and derive conclusions. An alternative would be to utilise the pre-trained models available for evaluation.

2.5. Existing ML-Based Malware Classifiers

Many studies have previously developed various malware classifiers using machine learning. Each has experimented with multiple combinations of malware datasets, machine learning algorithms, and feature types, where most have an AUC score/accuracy rating above 90%. However, certain experimental conditions lead to such results.

Factors such as smaller datasets, testing within the same dataset, and temporal and spatial biases might not precisely represent an "open world" scenario that a classifier might be subjected to in the future. A brief evaluation of current research about malware detection and its performance and experimental conditions are summarised as follows.

All the studies in Table 3 have achieved an excellent AUC or accuracy rating, but multiple caveats for each study have caused these results. These factors usually are older dataset usage, relatively smaller sample size, evaluation metric, publicly inaccessible datasets, and a closed-world approach, as discussed in the BODMAS dataset taken during the testing phase.

Table 3. Existing studies on machine learning-based malware detection.

Study	Dataset	Samples	Feature Type(s)	Classifier Used	Metric	Value
[10]	EMBER	400,000	Static	LightGBM	Accuracy	92.7%
[6]	BODMAS	57,293	Static	LightGBM	Accuracy	98%
[13]	BIG-15	20,000	Static	XGBoost	Accuracy	99.77%
[12]	BIG-15	20,000	Static	CNN	Accuracy	97.6%
[7]	N/A	2200	Static and dynamic	GBDT	AUC	99.36%
[17]	N/A	7863	Static	SVM	AUC	98.19%
[5]	N/A	984	Static and dynamic	SVM	Accuracy	96.1%

The rapidly evolving nature of malware demands the consistent development of newer datasets. Several studies use the BIG-15 dataset as a benchmark, resulting in higher accuracy scores. These results might be less relevant to the current malware threat landscape. Smaller sample sizes could reduce the quality of the resulting malware classifiers. Except for EMBER and BODMAS datasets, all others have a sample count of less than 20,000. Some of these studies are relatively older, and only certain conclusions might be applicable at this point.

Only two studies in Table 3 use AUC as their evaluation metric, but they have a smaller sample size and a publicly inaccessible dataset. Refs. [5,17] have kept the dataset private and have a sample count of less than 10,000. Commonly used classifiers against publicly well-known datasets such as BIG-15, BODMAS, and EMBER are variants of gradient boosting algorithms such as XGBoost/LightGBM. In summary, using static features and gradient-boosting algorithms are a common theme in the previously discussed studies.

2.6. Concept Drift and Performance Degradation

As newer malware strains and families appear, classifiers trained on older data tend to misclassify more recent malware. This is referred to as concept drift and is usually observed with time. Methods to counter such aberration in classification were initially discussed in BODMAS.

Performance degradation is an effect of concept drift and is generally a function of time. A recent study [18] suggests different transfer learning methods to counter performance degradation with time. However, the dataset used is relatively minor, and there is an emphasis on dynamic features. As discussed previously, the reliability and computational expense of extracting dynamic features are usually higher than static features. A brief comparison of various studies that discuss about concept drift and malware dataset evaluation are presented in Table 4.

Table 4. Comparison of Rapidrift with other studies and datasets.

Study	Training Set Count >50,000	Family Analysis	Concept Drift Analysis	Utilisation of Multiple PE Malware Datasets	Testing with Newer PE Malware Datasets	Ease of Implementation	Standardised and Improved Dataset Format
[9]	✔					N/A	
[6]	✔		✔	✔		N/A	
[16]	✔					N/A	
[15]						N/A	
[4]	✔		✔				
[19]			✔				
RAPIDRIFT	✔	✔	✔	✔	✔	✔	✔

(✔—addressed in the study, N/A—Not applicable).

More extensive research on concept drift in the context of static malware analysis was studied by TRANSCEND [4]. The critical factor that is being addressed here is improving prediction confidence. They suggest a rejection-based framework to identify low-confidence predictions and further evaluate those samples using other means, such as sandbox testing. Concept drift is prominent across datasets, according to the results observed in BODMAS.

Another study called CADE [19] has performed better than TRANSCEND in novel malware detection. CADE and TRANSCEND evaluate their solutions on multiple use cases, such as Android malware, PE-based malware, and malicious logs from network intrusion detection systems. Although, each of these cases utilise a single dataset for training and testing, which might not effectively measure the usability of such solutions in an "open world" scenario.

3. Methodology

In this section, as a core part of our methodological approach, we introduce Rapidrift. This Python-based framework allows for easy manipulation of datasets based on malware's family, time of initial discovery and SHA digest values. The framework also serves to train and evaluate machine learning models using LightGBM. A new dataset format was introduced, combining malware features and associated metadata into a single data frame. All these factors help add more context to the research and conduct granular analysis of a classifier's capabilities. Figure 1 is a visual representation of the improved dataset format that contains both malware features and its associated metadata.

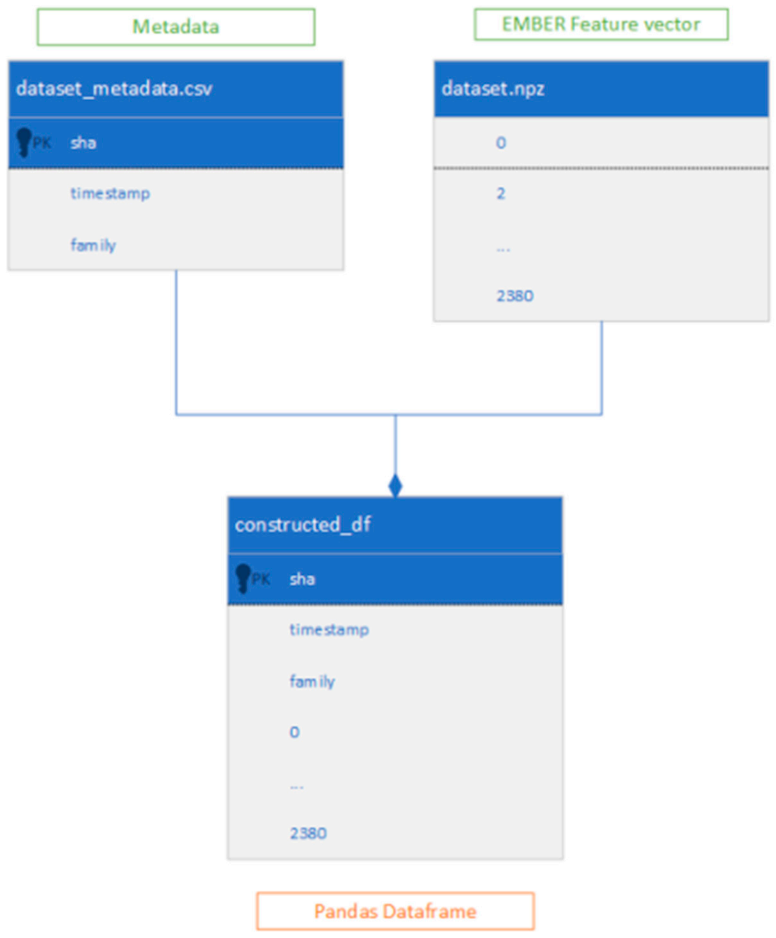

Figure 1. Rapidrift dataset format.

The substantiation for creating such a framework is to make malware-based machine learning more accessible to new researchers. Temporal and spatial bias were eliminated carefully using the framework while conducting experiments. The framework is built with a variety of functions to create a dataset from raw PE files, manipulate the dataset, filter based on parameters, and evaluate the models. A class object (superframe) containing the dataset is loaded into memory and can be accessed via multiple functions to avoid reading from the disk every time. However, the limitation here is that huge datasets like SOREL cannot be directly used with this framework. A few of the functions available in the framework are described in Table 5.

Table 5. Example functions available in rapidrift framework.

Function	Arguments	Description
family_filter()	Malware family name	Returns samples belonging to a certain malware family
time_filter()	Start_time, end_time	Returns samples within the given time frame
family_rank()	None	Returns the malware family rank based on frequency of occurrence
df_to_trainset()	Training and testing dataframes	Returns the X and Y arrays for training the model using methods other than GBDT
model_trainer()	Training dataframe and model path	Creates a GBDT model and saves it to a file

The experimentation procedures were executed on a computational platform equipped with an Intel i7 12th generation central processing unit and 64 gigabytes of random-access memory (RAM). Python version 3.6.8 was selected due to its compatibility with the requisite dependencies of all employed libraries throughout the experiments.

3.1. Analysis and Mitigation of Concept Drift

A set of common issues that arise in machine learning-based malware detectors could be due to the following:

- The time in which the malware sample was first observed;
- The emergence of newer malware families.
- Lack of sufficient samples in each family.
- Variance in the malware features that belong to a specific family;
- Highly customised malware that does not share features of any previously seen malware families.

In summary, the two key reasons that potentially cause concept drift are as follows:

Case 1: Imbalanced learning of malware leading to biased predictions for only certain malware;

Case 2: Emergence of new families/malware that shares lesser features with previously used training data.

Hence, it is crucial to address both scenarios to reduce concept drift's impact on malware detection effectively. BODMAS discussed initial attempts to combat concept drift under certain experimental conditions. Two different classifiers were used to measure its performance against BODMAS data monthly. The first classifier is the EMBER baseline, whose decision threshold was chosen at 0.1% FPR during the validation phase. The other classifier was incrementally retrained with one month of BODMAS data with the EMBER dataset and tested in the subsequent month; however, the decision threshold was modified to retain 0.1% FPR.

According to BODMAS, incremental retraining effectively reduces concept drift's effects. But, adding new training data from another dataset to build a classifier requires three tasks:

- Labelling data for retraining (open-world scenario);
- Retraining the model with new data;
- Changing the decision threshold.

Both labelling data and retraining the model with new data require computational resources. However, labelling malware could be beneficial in developing Indicators of Compromise (IOC) for threat-hunting purposes. The labelling of unknown samples is a vital part of retraining. The final classification is based on the decision threshold that decides whether a given sample is malicious. A comparison between the performance of EMBER and BODMAS is shown in Figure 2.

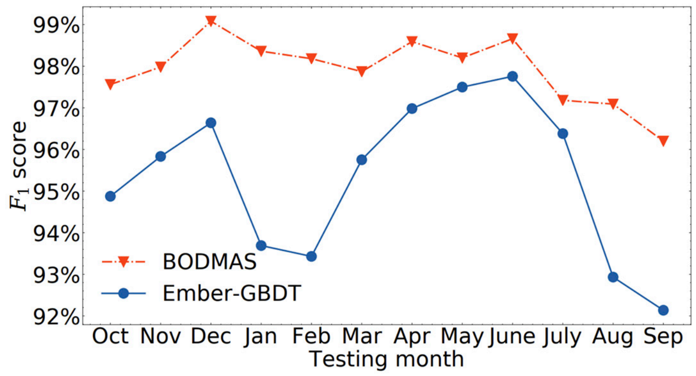

Figure 2. Comparison of BODMAS against EMBER Source: [6].

3.2. Confusion Matrix in the Context of Malware Detection

In any classification problem, the results of a model belong to four key categories that fall within the confusion matrix. In the context of the malware classification problem, these groups are clarified below:

a. **True Positive (TP):** a malware sample correctly classified as malware;
b. **False Positive (FP):** a benign sample misclassified as malware;
c. **True Negative (TN):** a benign sample correctly classified as not malware;
d. **False Negative (FN):** a malware sample misclassified as benign.

False positives determine the usability of the classifier, as an extremely high count of false positives is undesirable. False negatives usually comprise evasive or novel malware that tends to bypass the classifier.

3.3. Moving Decision Thresholds

The decision boundary decides the maliciousness of a given PE sample. Hence, developing a strategy to determine the optimal threshold to detect malware is essential. The baseline EMBER classifier was evaluated against the BODMAS dataset with two different decision thresholds to demonstrate this experimentally. The thresholds have been changed based on the results from the previous month (using it as validation data). However, this time interval could be changed to any number of months depending on the cost of retraining. The cost of retraining smaller datasets is relatively lower; hence, it is affordable and facilitates the periodic movement of decision boundaries.

As the intention here is to address concept drift, the threshold is based on the maximum possible difference between the TPR and FPR instead of minimising FPR to a constant. However, both methods are evaluated to compare the classifier's performance in both classes. Results from two different threshold selection strategies are shown in Table 6.

Table 6. Results with different decision boundaries.

Decision Threshold Strategy	Mean F1 Score	Mean FN	Mean FP
0.1% FPR	0.961596 ± 0.04	305 ± 331	6 ± 8
argmax (TPR-FPR)	0.992728 ± 0.002	34 ± 26	29 ± 27

The results suggest that using FPR purely to select decision thresholds results in a higher false-negative rate contributing to more malware evading the classifier. However, a disproportionate amount of malware can be successfully captured when traded off with slightly more false positives. These results explain the sensitivity of the classifier's behaviour when the decision threshold is altered. It is essential to emphasise selecting the correct decision threshold, as retraining could prove to be a computationally expensive process depending on the size of the dataset.

3.4. Incremental Retraining

BODMAS was cumulatively retrained with various proportions of new data from the previous month. From the previous month, 10%, 25%, and 50% of new data were tested against the next month. The range of months tested was 2019-10 to 2020-09. Around 12,000 samples from previous months were used as a base for retraining. The results shown in Figure 3 and Table 7 show a steady improvement in the classifiers' performance with more training data.

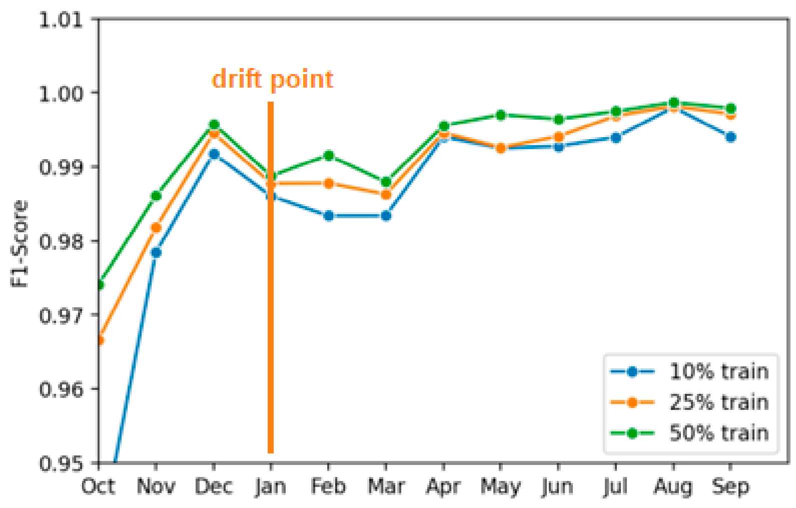

Figure 3. F1 scores—incremental retraining.

Table 7. Results of retraining with different training proportions.

Random Sampling Rate	Mean F1 Score	Mean FN	Mean FP
10% train	0.985589 ± 0.023	89 ± 143	32 ± 37
25% train	0.989863 ± 0.008	60 ± 81	25 ± 31
50% train	0.992285 ± 0.008	43 ± 65	21 ± 26

3.5. Probability Calibration

Considering the effect of selecting decision thresholds periodically on the classifier's performance, it is crucial to ensure the reliability of such values. Probability calibration methods attempt to match the predicted probabilities of a classifier with the true values. An ideally calibrated model would have comparable predicted probabilities with reality. By default, the predicted probabilities of LightGBM are discrete, where almost every sample has a unique probability, hence resulting in a probability distribution that is highly extensive and uninterpretable. Calibrating the classifier's results would result in a more constrained distribution with only specific possible probabilities. There are a variety of techniques used for probability calibration tasks. However, the scope of this study is restricted to Platt, Isotonic, and Spline. The results of probability calibration are shown in Figure 4 and Table 8.

Table 8. Results of different probability calibration methods.

Calibration Method	F1 Score	Log Loss
Uncalibrated	0.99602	0.022363
Platt	0.99633	0.020509
Isotonic	0.99712	0.009490
Spline	0.99701	0.012282

By calibrating the probabilities, it can be observed that all three methods have improved F1 scores and reduced log loss values. This suggests that probability calibration could help improve the classifier's performance. Unlike retraining, calibration does not require heavy use of computational resources but improves the overall quality of predictions.

In summary, as demonstrated in Figure 5, these components combined would increase the quality of malware detection by reducing the false negatives significantly at the cost of slightly more false positives. Almost all functions in the figure have been included as a part of the Rapidrift framework.

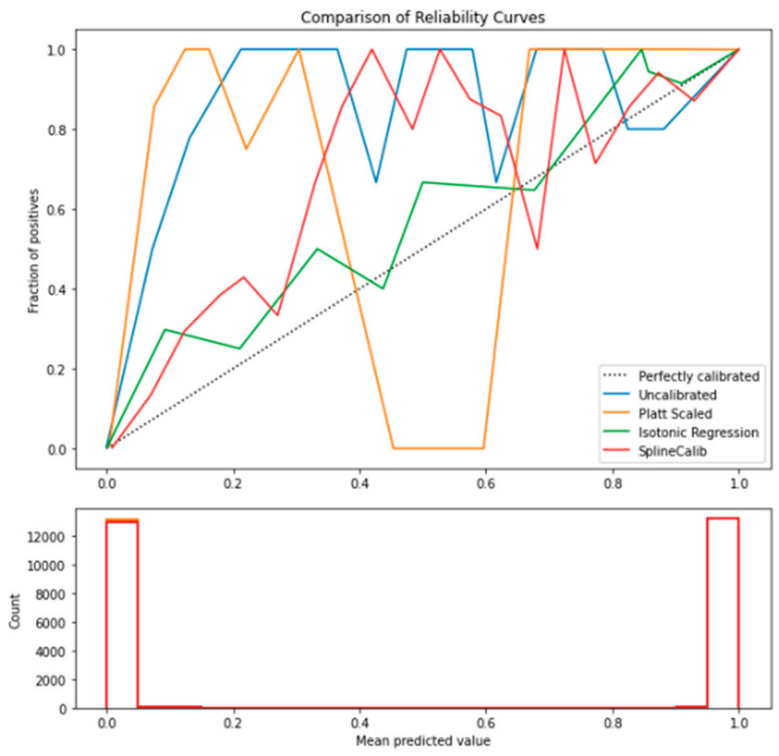

Figure 4. Comparison of probability calibration methods.

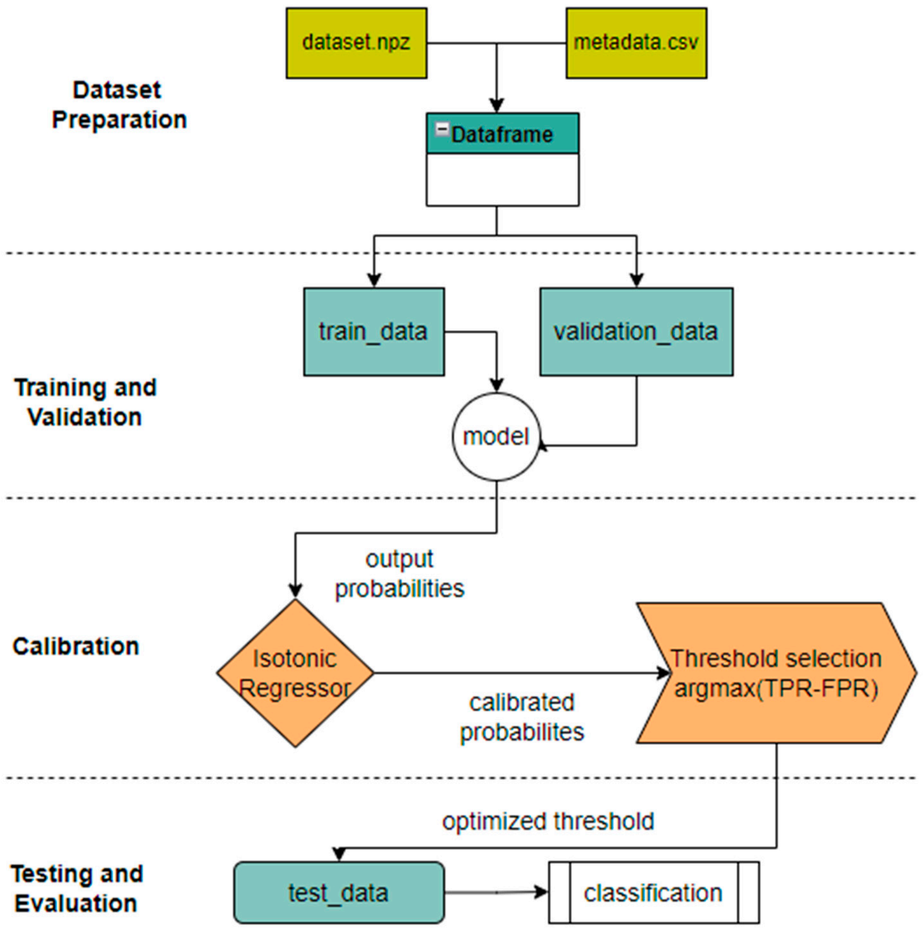

Figure 5. Flowchart for improved malware detection.

4. Introduction to New Datasets—TRITIUM and INFERNO

Two new datasets are introduced to evaluate concept drift in current malware and the performance of classifiers built on relatively older datasets. These two datasets (refer to Supplementary Materials) have different threat profiles as the malware samples have been aggregated from various sources. The TRITIUM dataset contains 23,256 samples, of which 12,471 are malicious and 10,785 are benign. This dataset constitutes 22 unique malware families to facilitate the research of concept drift within a given malware group. The dataset was sourced from MalwareBazaar, an open-source threat intelligence provider.

From an adversarial standpoint, a malware classifier might not be able to detect customised or targeted attacks effectively. A new dataset called INFERNO was developed to verify the classifier's applicability in such scenarios. This specific category of malware is usually created with tools that allow the creation/modification of malicious payloads, such as Metasploit, CobaltStrike, Empire, Jlaive, UACBypass, and other offensive tools and obfuscators. Red teamers and adversaries commonly use these tools to bypass endpoint protection and execute malicious code. Both datasets were evaluated using the previously discussed classifiers EMBER, SOREL, and BODMAS.

4.1. Testing with Dataset Combinations

One of the previous experiments has proven the effectiveness of incremental retraining. In this experiment, two different datasets were combined to identify if they outperformed the classifiers trained separately by each of those datasets. Two classifiers trained with BODMAS and EMBER independently and another classifier trained with a combination of both datasets were evaluated against TRITIUM. The results are shown in Table 9 and Figure 6.

Table 9. Classifier performance with combined datasets.

Classifier	Mean F1 Score	Mean FN	Mean FP
EMBER	0.987457 ± 0.008	10 ± 8	11 ± 18
BODMAS	0.973429 ± 0.01	31 ± 29	20 ± 50
EMBER and BODMAS	0.984426 ± 0.009	13 ± 12	10 ± 19

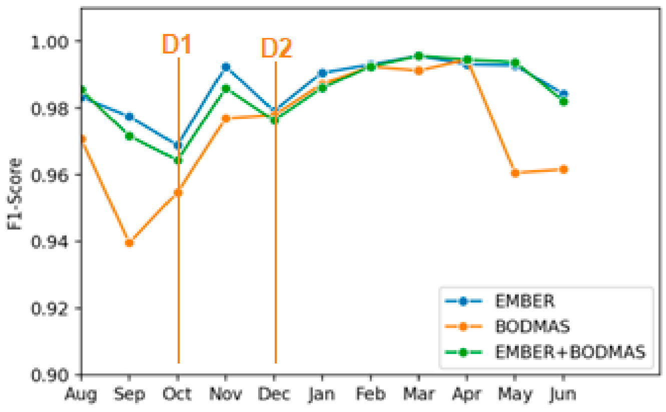

Figure 6. Performance of classifiers against TRITIUM (drift points—D1 and D2).

Surprisingly, EMBER performs better in calibrated and uncalibrated scenarios (uncalibrated results omitted for brevity) against the TRITIUM dataset. On the other hand, BODMAS contains more recent data but underperforms. Combined, BODMAS and EMBER result in the least false-positive rates. Another observation is that calibrated probabilities slightly increase the F1 scores in all three cases, highlighting the importance of probability calibration.

4.2. Testing with Novel Families from TRITIUM

As TRITIUM is relatively new, it contains novel malware families. The novelty makes it an ideal candidate for concept drift evaluation, as no other datasets except BODMAS contain accurate family information. BODMAS observes that most malware misclassifications are attributed to new and unseen families. Three classifiers trained using BODMAS, SOREL, and EMBER were tested against a novel version of the TRITIUM dataset.

The Rapidrift framework was used to filter the TRITIUM dataset to isolate malware samples (around 7000) that have not appeared in BODMAS before. The exact process could not be repeated with SOREL and EMBER as they lack accurate malware family information (e.g., WannaCry, Emotet, and similar). This experiment intends to gauge the performance of the malware classifier when left untrained to the point where the current threats are entirely new. This should provide a realistic insight into the classifier's detection capabilities in an "open world" scenario. The results were evaluated based on the decision threshold set at 0.1% FPR, similar to the work demonstrated by BODMAS. Table 10 shows the performance of three different classifiers against the novel version of the TRITIUM dataset.

Table 10. Classifier performance against a novel version of TRITIUM.

Classifier	F1 Score	AUC Score	False Negatives
EMBER	0.90423	0.91250	1260
SOREL	0.81317	0.84241	2275
BODMAS	0.79891	0.83240	2420

The results are counterintuitive as the classifier trained with more samples or from the most recent malware would be expected to perform well. But EMBER is better than the other two individual classifiers regarding all three metrics. However, as this is a scenario where classifiers are only exposed to novel malware, further analysis was conducted. Certain malware families were found to have a lower number of false negatives when tested against SOREL and BODMAS, although their overall performance was not as good as EMBER.

It was found that "Lokibot info stealer" was one of the highly misclassified malware families for SOREL and EMBER classifiers. But, BODMAS has less than 200 false negatives for the same malware. On the other hand, EMBER produced less than 100 false negatives for the "guloader" malware, while SOREL and BODMAS misclassified more than 350 samples belonging to the same family. This suggests the classifier's overall performance may not be sufficient to judge its capabilities amongst certain malware families. By extension, no specific classifier can effectively capture all malware variants, irrespective of the merits of the training dataset used.

Hence, the probability scores from older classifiers should not be considered obsolete, and the results from the most recent models cannot be trusted entirely. On a similar note, the classifier's quality cannot be determined purely by the volume of the training dataset. SOREL was trained on 20 million samples but could not outperform the other two classifiers. This might suggest that ensemble learning could produce quality predictions as it encompasses the capabilities of multiple classifiers.

As the Guloader family was the most misclassified malware when tested against BODMAS, incremental retraining for that specific family was conducted. This experiment could indicate the number of samples required to effectively capture all the malware from one family by retraining. There are 511 Guloader samples available in the TRITIUM dataset. From Table 11 it is inferred that even fewer malware samples added in the retraining process can create a significant impact on being able to detect more malware from the same family.

Table 11. Retraining with single malware family.

Number of Guloader Samples Added to BODMAS	False Negatives
0	486
1	351
2	291
3	143
10	0

There was a significant improvement in the results when BODMAS was retrained with this new malware family. Ten random samples were sufficient to detect this malware family fully. More tests were conducted by training BODMAS with one random Guloader sample every time, and different results were observed. In some cases, more samples were needed to detect this malware family fully, but specific scenarios required less than ten samples to capture the same successfully.

These observations suggest the importance of sample quality that must be considered during training or retraining machine learning models. A specific dataset containing too many redundant samples could affect the quality of the resulting classifier. Fuzzy hashes and import hashing are used in threat intelligence to detect patterns in malware, and the same technique was applied to the Guloader family that was used to train and its related testing set. About 25% of the tested samples shared similar Imphash and ssdeep values. An alternative strategy for retraining/training is to ensure that the new samples do not contain similar fuzzy hash or Imphash values. Such an approach might improve the quality of training data and any future malware dataset created. Robust statistical measures such as non-conformity used by TRANSCEND could be used as a retraining strategy despite its focus on detecting drifting samples. Although it is computationally intensive, the scalability of such methods has yet to be discussed in previous research.

These methods could be easily incorporated into the training phase by adding fuzzy hash values to the metadata of malware samples. Later, the samples containing similar hashes could be grouped into clusters to only select *n* samples from each cluster for training. Such techniques not only improve the classifier's quality but also reduce the volume of the dataset. EMBER's superior performance against the novel dataset could be due to the quality of the training data. Additionally, if the testing dataset contains many similar samples, it would overestimate the performance of the classifier's true capabilities.

4.3. Testing against the INFERNO Dataset

Compared to other datasets discussed previously, INFERNO uses a different source of malware. INFERNO was tested against BODMAS, SOREL, and EMBER. The family information for the results was not reported as they are discrete for every sample (filenames were used). The reported result is calculated based on the decision threshold at 0.1% FPR, corresponding to one false positive. Table 12 shows the performance of three different classifiers against the INFERNO dataset.

Table 12. Classifier performance against INFERNO.

Classifier	F1 Score	AUC Score	False Negatives
EMBER	0.95889	0.96049	112
SOREL	0.95814	0.95979	114
BODMAS	0.90513	0.91329	247

The results are similar to the previous experiment, where the overall performance of EMBER is better than the other two classifiers. BODMAS generated twice the number of false negatives compared to EMBER and SOREL.

5. Discussion

A general limitation for any machine learning-based malware classification solution would be to surpass existing malware detection approaches. It is crucial to understand that machine learning is a computationally intensive process. Hence, it is expected to improve the quality of binary malware classification, such as detecting novel malware as opposed to traditional methods.

Imphash and fuzzy hashing algorithms are commonly used in threat intelligence [20] to capture slightly modified or similar malware. Another study [21] demonstrates how combining fuzzy hashing methods can achieve F1 scores of up to 94% with limited resources (the study used 6 GB RAM and four CPUs). Although the F1 score is lower than the results achieved in this study, it is quite significant considering the relatively low computational intensity and time taken.

Furthermore, refs. [4,22] suggest using a nonconformal evaluation (*p*-value as a similarity metric) to detect drifting malware and flag low-confidence predictions. However, the approximate time taken for their conformal evaluator is 46.1 CPU hours. This suggests that data comparison, resource, and time constraints are fundamental problems in malware classification, especially in novelty detection. In brief, the computational cost of retraining models or creating new models should be less than that of implementing other solutions such as CADE and TRANSCEND. However, the comparison between the cost of retraining and their proposed solutions are yet to be evaluated. On the other hand, the solution(s) suggested here can easily be implemented during the training process itself.

Current research focuses on obtaining and training using malware samples with almost no pre-processing. For instance, the UCSB-20 dataset contains packed malware samples only in its dataset and would not perform well when tested against generic malware and was also experimentally proven by BODMAS. A better strategy would be to group potentially packed samples using some form of automation and expose them to a classifier trained using UCSB-20 or a similar dataset. PE architecture could be an essential subdomain to research, considering the wide range of programming languages that malware authors use. Languages such as C/C++ that are low-level might have different feature vector values compared with NET, which requires a framework to run successfully.

All three commonly used benchmark malware datasets, EMBER, SOREL and BODMAS, use the same hyperparameters to build their models. The dimensions of the dataset vary drastically and using the same hyperparameters without any tuning might cause a loss in performance. Ref. [23] evaluated the EMBER dataset and another custom-built dataset by tuning hyperparameters for LightGBM. The same study discusses how it is computationally intensive to obtain a well-tuned hyperparameter combination. Hence, a lack of hyperparameter tuning could affect the quality of classifiers. However, with retraining practices, it would be hard to regularly tune the hyperparameters every time to produce optimum performance.

Another limitation of this study would be to use certain feature subsets in the training phase and assess its performance. When there are lesser number of features that effectively determine a malware sample, it could potentially reduce the time to retrain or build new models.

6. Conclusions

Various methods to improve the relevance of classifiers with newer malware were discussed. Optimising the decision threshold to consider false-positive and true-positive rates has improved the detection quality. Probability calibration using an Isotonic regressor also demonstrated better results without significant computational overhead. The overall performance of EMBER surpasses other models, such as SOREL and BODMAS, against the new TRITIUM dataset. But, in some instances, the underperforming classifiers could detect specific malware families better than EMBER.

Two new datasets with different threat profiles were created to test the resilience of classifiers trained using existing datasets, and their results were critically analysed.

A Python utility was developed to assist researchers in quickly evaluating and training malware classifiers using various constraints. Adopting the dataset and metadata format suggested by BODMAS and this study would help add more context to malware detection. The proposed methods also help elongate the relevance of classifiers with time.

Previous experiments revealed that the classifier's quality can only partially be decided on the timeline and size of the dataset. Each classifier could be capable of detecting certain types of malware. Hence, discarding a classifier based on age or size might cause more long-term misclassifications. In such situations, using stacked ensemble methods could improve performance. On the other hand, transfer learning overcomes the limitations of incremental retraining, where the models cannot be updated with new data periodically.

Supplementary Materials: The following supporting information can be downloaded as follows: INFERNO dataset—https://github.com/4dsec/inferno, TRITIUM dataset—https://github.com/4dsec/tritium.

Author Contributions: Conceptualization, A.M., P.A. and N.P.; Methodology, A.M., P.A. and N.P.; Validation, A.M.; Writing—original draft, A.M., P.A. and N.P.; Writing—review & editing, A.M., P.A. and N.P.; Supervision, N.P. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Pells, M. *Cyberattack on Yorkshire Coast Firm*; Yorkshire Coast News: North Yorkshire, UK, 2023.
2. *Norway Government Ministries Hit by Cyber-Attack*; Reuters: London, UK, 2023.
3. Jeong, Y.-S.; Woo, J.; Kang, A.R. Malware Detection on Byte Streams of Hangul Word Processor Files. *Appl. Sci.* **2019**, *9*, 5178. [CrossRef]
4. Barbero, F.; Pendlebury, F.; Pierazzi, F.; Cavallaro, L. Transcending TRANSCEND: Revisiting Malware Classification in the Presence of Concept Drift. 22 October 2020. Available online: http://arxiv.org/abs/2010.03856 (accessed on 25 June 2022).
5. Shhadat, I.; Bataineh, B.; Hayajneh, A.; Al-Sharif, Z.A. The Use of Machine Learning Techniques to Advance the Detection and Classification of Unknown Malware. *Procedia Comput. Sci.* **2020**, *170*, 917–922. [CrossRef]
6. Yang, L.; Ciptadi, A.; Laziuk, I.; Ahmadzadeh, A.; Wang, G. BODMAS: An Open Dataset for Learning based Temporal Analysis of PE Malware. In Proceedings of the 2021 IEEE Security and Privacy Workshops (SPW), Francisco, CA, USA, 27 May 2021; IEEE: Piscataway, NJ, USA, 2021; pp. 78–84. [CrossRef]
7. Ijaz, M.; Durad, M.H.; Ismail, M. Static and Dynamic Malware Analysis Using Machine Learning. In Proceedings of the 2019 16th International Bhurban Conference on Applied Sciences and Technology, IBCAST, Islamabad, Pakistan, 8–12 January 2019; Institute of Electrical and Electronics Engineers Inc.: Piscataway, NJ, USA, 2019; pp. 687–691. [CrossRef]
8. Palahan, S.; Babić, D.; Chaudhuri, S.; Kifer, D. Extraction of statistically significant malware behaviors. In Proceedings of the ACM International Conference Proceeding Series, New Orleans, LA, USA, 9–13 December 2013; pp. 69–78. [CrossRef]
9. Anderson, H.S.; Roth, P. EMBER: An Open Dataset for Training Static PE Malware Machine Learning Models, April 2018. Available online: http://arxiv.org/abs/1804.04637 (accessed on 25 April 2022).
10. Oyama, Y.; Miyashita, T.; Kokubo, H. Identifying useful features for malware detection in the ember dataset. In Proceedings of the 2019 7th International Symposium on Computing and Networking Workshops, CANDARW, Nagasaki, Japan, 26-29 November 2019; Institute of Electrical and Electronics Engineers Inc.: Piscataway, NJ, USA, 2019; pp. 360–366. [CrossRef]
11. Zhang, Z.; Qi, P.; Wang, W. Dynamic Malware Analysis with Feature Engineering and Feature Learning, Jul. 2019. Available online: http://arxiv.org/abs/1907.07352 (accessed on 25 May 2022).
12. Rafique, M.F.; Ali, M.; Qureshi, A.S.; Khan, A.; Mirza, A.M. Malware Classification Using Deep Learning Based Feature Extraction and Wrapper Based Feature Selection Technique, October 2019. Available online: http://arxiv.org/abs/1910.10958 (accessed on 25 April 2022).
13. Ahmadi, M.; Ulyanov, D.; Semenov, S.; Trofimov, M.; Giacinto, G. Novel Feature Extraction, Selection and Fusion for Effective Malware Family Classification, November 2015. Available online: http://arxiv.org/abs/1511.04317 (accessed on 25 April 2022).
14. Cai, J.; Luo, J.; Wang, S.; Yang, S. Feature selection in machine learning: A new perspective. *Neurocomputing* **2018**, *300*, 70–79. [CrossRef]

15. Ronen, R.; Radu, M.; Feuerstein, C.; Yom-Tov, E.; Ahmadi, M. Microsoft Malware Classification Challenge. In Proceedings of the CODASPY 2016—Proceedings of the 6th ACM Conference on Data and Application Security and Privacy, New Orleans, LA, USA, 9–11 March 2016; pp. 183–194. February 2018. Available online: http://arxiv.org/abs/1802.10135 (accessed on 1 May 2022).
16. Harang, R.; Rudd, E.M. SOREL-20M: A Large Scale Benchmark Dataset for Malicious PE Detection. *arXiv Preprint* **2020**, arXiv:2012.07634.
17. Wang, T.-Y.; Wu, C.-H.; Hsieh, C.-C. Detecting unknown malicious executables using portable executable headers. In Proceedings of the NCM 2009—5th International Joint Conference on INC, IMS, and IDC, Seoul, Republic of Korea, 25–27 August 2009; pp. 278–284. [CrossRef]
18. García, D.E.; DeCastro-García, N.; Castañeda, A.L.M. An effectiveness analysis of transfer learning for the concept drift problem in malware detection. *Expert Syst. Appl.* **2023**, *212*, 118724. [CrossRef]
19. Yang, L.; Guo, W.; Hao, Q.; Xing, X.; Wang, G. CADE: Detecting and Explaining Concept Drift Samples for Security Applications. 2022. Available online: https://www.usenix.org/conference/usenixsecurity21/presentation/yang-limin (accessed on 25 June 2022).
20. Naik, N.; Jenkins, P.; Savage, N.; Yang, L.; Boongoen, T.; Iam-On, N. Fuzzy-import hashing: A static analysis technique for malware detection. *Forensic Sci. Int. Digit. Investig.* **2021**, *37*, 301139. [CrossRef]
21. Shiel, I.; O'Shaughnessy, S. Improving file-level fuzzy hashes for malware variant classification. *Digit. Investig.* **2019**, *28*, S88–S94. [CrossRef]
22. Pendlebury, F.; Pierazzi, F.; Jordaney, R.; Kinder, J.; Cavallaro, L. TESSERACT: Eliminating Experimental Bias in Malware Classification across Space and Time, July 2018. Available online: http://arxiv.org/abs/1807.07838 (accessed on 25 June 2022).
23. Kundu, P.P.; Anatharaman, L.; Truong-Huu, T. An empirical evaluation of automated machine learning techniques for malware detection. In Proceedings of the IWSPA 2021—Proceedings of the 2021 ACM Workshop on Security and Privacy Analytics, Virtual Event USA, 28 April 2021; Association for Computing Machinery, Inc.: New York, NY, USA, 2021; pp. 75–81. [CrossRef]

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.