Cross-Regional Malware Detection via Model Distilling and Federated Learning

Marcus Botacin
botacin@tamu.edu
Texas A&M University
USA

Heitor Gomes
heitor.gomes@vuw.ac.nz
Victoria University of Wellington
NZ

ABSTRACT

Machine Learning (ML) is a key part of modern malware detection pipelines, but its application is not straightforward. It involves multiple practical challenges that are frequently unaddressed by the literature works. A key challenge is the heterogeneity of scenarios. Antivirus (AV) companies for instance operate under different performance constraints in the backend and in the endpoint, and with a diversity of datasets according to the country they operate in. In this paper, we evaluate the impact of these heterogeneous aspects by developing a classification pipeline for 3 datasets of 10K malware samples each collected by an AV company in the USA, Brazil, and Japan in the same period. We characterize the different requirements for these datasets and we show that a different number of features is required to reach the optimal detection rate in each scenario. We show that a global model combining the three datasets increases the detection of the three individual datasets. We propose using Federated Learning (FL) to build the global model and a distilling process to generate the local versions. We order the samples temporally to show that although retraining on concept drift detection helps recover the detection rate, only a FL approach can increase the detection rate.

CCS CONCEPTS

• **Software and its engineering** → Software creation and management; • **Security and privacy** → **Malware and its mitigation**; Human and societal aspects of security and privacy; **Intrusion/anomaly detection and malware mitigation**; • **Computing methodologies** → *Machine learning*.

KEYWORDS

malware, federated learning, model distilling

ACM Reference Format:
Marcus Botacin and Heitor Gomes. 2024. Cross-Regional Malware Detection via Model Distilling and Federated Learning. In *The 27th International Symposium on Research in Attacks, Intrusions and Defenses (RAID 2024), September 30–October 02, 2024, Padua, Italy*. ACM, New York, NY, USA, 17 pages. https://doi.org/10.1145/3678890.3678893

This work is licensed under a Creative Commons Attribution International 4.0 License.

RAID 2024, September 30–October 02, 2024, Padua, Italy
© 2024 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-0959-3/24/09
https://doi.org/10.1145/3678890.3678893

1 INTRODUCTION

Malware attacks are on the rise, and so is the use of Machine Learning (ML) to counter them [1]. Thus, developing ML pipelines to detect malware becomes key, as they are the basis of many security solutions, such as AntiViruses (AVs) [10]. However, these pipelines must be realistic [14, 31] to actually protect the users. This work is an effort to evaluate the realistic constraints of ML AVs.

Previous pipelines published in the literature presented two main assumptions that do not hold in reality: (1) the bigger the model, the better [9]. They assume that models can grow indefinitely to achieve greater accuracy, with no performance constraint. In practice, security solutions should be fast to not disrupt the regular system operation; and (2) One size fits all [7]. They assume that threats are global, with no localized threats to be handled particularly. This is not the case for AV companies that operate in multiple different countries all around the world.

We demonstrate how these assumptions fail by simulating an AV company operating in different regions of the world and trying to develop models that achieve the best performance in them, both in accuracy as well as in execution time and storage requirements. To that, we partnered with an AV company that provided us with 30K malware samples (10K for each country) collected in the same period (2017) in the United States (US), Brazil (BR), and Japan (JP). We used this data to explore the best strategies to build detection models and we hope this information might inform future developments in the field.

We first demonstrate that the size of the best feature set for each scenario varies, such that adopting a uniform, large model imposes unnecessary extra performance costs. Further, we also demonstrate that the knowledge learned by classifiers in each scenario is actually different, such that a global model does not naturally exist, except by intentional construction.

Our goal to move forward is to answer the question What is the best way to build detection models for heterogeneous threat scenarios like this? To do so, we evaluate how to achieve two competing goals: (i) making models smaller to achieve execution performance requirements; and (2) making models larger to create a global knowledge that generalizes. To conciliate that, we propose splitting the scenarios in two: (1) a local model that is responsible for achieving high execution performance in the specific country; and (2) a large global model responsible for transferring knowledge between multiple countries.

We propose AV companies use Federated Learning (FL) [25] to build the global model. Unlike previous proposals, we do not propose that AV users (clients) run training routines on their machines [25, 28]. Instead, we propose the AV subsidiaries in each country train local models that are at the same time deployed in the clients for that region and also sent to an AV centralized server to be consolidated in the global model. The information from the other countries is sent back to the subsidiaries to enrich their models. This strategy ensures the transfer of knowledge between the countries and it does not present risks of poisoning [37], unlike previous approaches, as all AV subsidiaries are trusted entities.

We also propose that the model from each subsidiary should be distilled [23] into a small model to be deployed in the endpoints (client machines) to achieve high execution performance. We rely upon the observation that each scenario requires a different feature set size and propose that models of different sizes should be derived from the main model. To accomplish that, we propose a modified version of the Random Forest (RF) algorithm that relies on a heterogeneous [2] set of trees of varied sizes.

We evaluate our propositions via a series of experiments and take a step further towards realistic evaluation by deriving from the ML model detection rules such as the ones used in commercial AVs (YARA [38]). We show that the differences in the features used by the ML models cause significant differences in the matching performance of the derived rules, motivating the presented investigation on the fine-tuning of the detection models.

In summary, our findings are:

- Different datasets require different numbers of features to be used on an ideal model able to achieve the maximum detection rate (e.g., 270 for US and 800 for JP).
- Making models bigger (e.g., by increasing the number of random forest trees) does not significantly increases detection rates for individual datasets (e.g., 0.5% gain for US), which supports the benefit of distilling smaller models.
- Making models bigger indeed helps models trained in one dataset to generalize better to detect the malware samples from others (e.g., from 60% to 95% in the US model), thus supporting the benefits of having global models.
- A time-series evaluation of the sample emergence reveals that although retraining on traditional concept drift detection allows recovering the original detection rate of individual models, their detection rate is only increased when data from the global model is used to complement them.

This paper's contributions are as follows:

- We shed light on the challenges of operating detection pipelines for heterogeneous scenarios.
- We demonstrate **how** datasets from different regions have different requirements in terms of the number of features required to achieve a target detection rate.
- We propose a combination of federated learning and model distilling with a heterogeneous number of features to adapt to the distinct scenario's requirements.

2 CHALLENGES & ARCHITECTURAL DESIGN

The first problem we address is the runtime matching performance of the ML models used by AV companies. AVs operate in a 2-level architecture, with components placed both in the end-user machines (endpoints) and on the AV cloud servers. The hard-to-classify samples not detected at the endpoint are sent for cloud inspection. Since these environments have different performance capabilities, they cannot run the same ML model, as proposed by many literature works. The endpoint's matching rules should be a simpler version of the cloud model. We propose that the endpoint should run a distilled version of the cloud model, as illustrated by Figure 1. In this architecture, the endpoint model is distilled to be compact (and thus faster), using a minimum number of features.

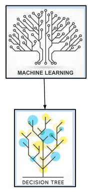

Figure 1: Single Model Distillation.

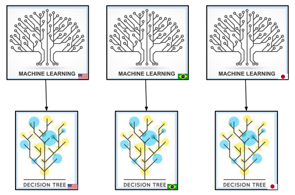

Figure 2: Multiple Regional Model Distillation.

The second problem we tackle is the accuracy of heterogeneous scenarios. Since AV companies operate in multiple countries, a single cloud model also does not apply to all of them. Thus, the previous architecture must be replicated for each country, as shown in Figure 2. With the parallel distillation of different endpoint models, each local model is allowed to have a different feature set.

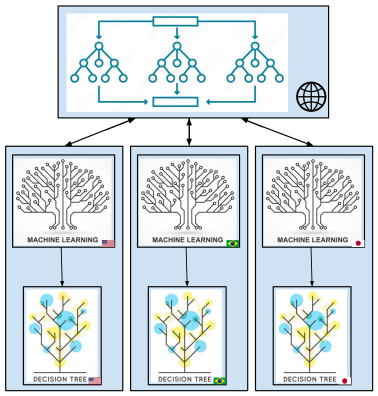

Figure 3: Regional Model Distillation from Global.

These two problems interconnect because data coming from one model might help increase the detection rate of another model. The rationale for that is that although the scenarios have particularities, some attackers might migrate their strategies from one scenario to another. Therefore, the models' knowledge should be shared. The best way to do that is to once again adopt a 2-level approach, but now by considering the 2-level architecture from Figure 2 as the country-level client and then build a world-level model (server) on top of that, as shown in Figure 3. We propose that Federated Learning (FL) is the appropriate technique for this scenario.

Whereas required to share knowledge, the models should still be optimized for their local scenarios. Thus, a key challenge of this architecture is to allow the easy distilling of models with different settings. We overcome this challenge by proposing a modification in the Random Forest (RF) algorithm to make it handle heterogeneous trees, as shown in Figure 4. In the new version, each tree in the ensemble has a different number of features, such that the distilling process becomes a matter of collecting an increasing number of

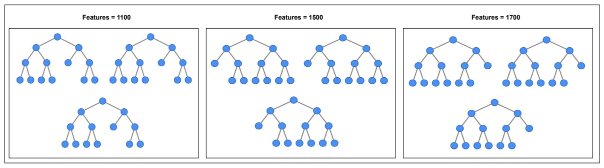

Figure 4: RF's ensemble of different features set sizes.

trees of increasing feature set size until reaching the target accuracy rate. After the regional and local models are derived from the global knowledge, the model is converted into rules by the AVs. The rules are distributed to the clients to be matched in the endpoints. This work will discuss in the following the best strategies to implement and deploy this architecture.

3 METHODOLOGY

Classification Model. This work's approach is to start from where the others stopped. Thus, we build on top of previous constructions. We adopted a previously-published, open-source PE-based classifier [13] to handle our dataset of Windows malware. This model is composed of categorical features extracted from the PE header and textual features derived from function imports embedded via TF-IDF. We modified the model to consider a minimum number of features to minimize the impact of model size. We implemented a feature selection mechanism (`SelectKBest` [35] with F-score) to consider the most representative features for this model (Sec. 4).

Implementation. We took the original `scikit-learn` [34] implementation of the model as a reference. We kept the original feature extractor implemented in `Python` and integrated to it the classifier implementation from the `MOA` [5] framework for increased analysis throughput. We adopted the `AdaptiveRandomForest` [30] classifier for all experiments due to its ability to integrate new training data without the need for retraining the entire model. Whereas this classifier was originally designed to be used in stream learning, we here benefit from this classifier to build the federated learning component. The proposed algorithm modifications were performed by training each tree separately as a new decision tree and then merging them into a new ensemble.

Security Evaluation Metrics. Our goal in this work is not to present a new ML model with incrementally higher accuracy. Model-specific accuracy improvements have already been presented by multiple related works. In turn, we assume that the base model has a high base accuracy. Our research interest is to evaluate how to make this accuracy sustainable over time and how to generalize it for multiple scenarios. Therefore, whenever we train a new ML model, we target the 99% accuracy level for the binary classification problem (malware vs goodware). Since our datasets are fully balanced, accuracy correctly describes the results.

Target Performance. In this work, we target not only to sustain a high detection rate but also to cause the minor performance impact possible. We evaluate performance impact via the size of the model, as the bigger the model, the greater the storage requirements it imposes and the longer it takes to traverse it. More specifically, we aim to select the minimum number of features and the minimum number of RF trees in the ensemble that allows us to still achieve the previously specified accuracy rate. We measured model size as the total number of tree nodes in the RF ensemble. It is key to highlight that there is a difference between counting the number of tree nodes and the actual amount of memory allocated to the tree. Many libraries, such as `scikit-learn`, allocate memory in batches, such that the actual storage does not grow linearly with the tree size. We opted to measure the number of tree nodes to remain agnostic to the memory allocator.

Results correction. By construction, the RF algorithm presents different results at different training runs. This characteristic provides it good generalization ability, but it might also bias the results if we report only an eventually over-positive, single case. To mitigate this possibility, all results presented in this paper are an average of 10 different runs. Training accuracy is reported as the outcome of the 10-folding process.

Dataset. In this work, we address the malware detection problem as a binary classification problem (malware vs. goodware). The malware dataset is split according to three different scenarios. The goodware dataset is a generalization of software most users have on their machines. The goodware samples were retrieved from a fresh Windows installation and from the crawling of the most popular applications in Internet software repositories. We ensured all files were labeled as clean by all VirusTotal engines. We used as many goodware samples as needed to provide a 50%-50% balance in the training sets, depending on the availability of malware for each tested scenario. For each tested scenario, we trained the different models with the same incremental set of goodware files, thus ensuring that any observed difference is due to only the different malware files for each scenario.

The malware dataset aims to represent the realistic scenario in which an AV company operates in multiple countries. We considered three datasets of 10K malware samples each, with no overlaps or duplicates (The duplicated rate was 33% for each dataset before we filtered them out). The datasets were collected by an AV company from infected user machines in the United States (US), Brazil (BR), and Japan (JP) during the entire year of 2017 and made exclusively available to us. These datasets were characterized in previous studies [6, 7]. The BR dataset is composed of 3 types of PE files (typical EXEs, DLLs, and CPLs) whereas the US and JP datasets are composed only of EXEs and DLLs, as CPL files were only observed in Brazil. The 3 datasets present more than 100 families, but the BR dataset has a prevalence (53%) of Password Stealers (PSW) and Downloaders whereas US and JP have a prevalence (40%) of Ransomware samples. We consider these datasets a coherent view of the threat landscape since they were collected by the same company, in the same period, from the same type of users, and using the same technique.

Table 1: Dataset Differences. Dynamic analysis events for the US, Brazil, and Japan datasets.

Behavior	US	BR	JP
Hosts file modification	0.04%	1.09%	0.92%
File creation	64%	24%	70%
File deletion	34%	12%	34%
File modification	63%	16%	46%
Browser modification	0%	1.03%	0.59%
Network traffic	53%	96%	52%

Whereas our dataset is of limited size, to the best of our knowledge, it is the most realistic representation of the heterogeneous scenarios AV companies operate to date. With these datasets, we can show the phenomenon and challenges the AV companies are subject to in the actual scenario. To highlight the diversity of these scenarios, we analyzed all samples in a dynamic analysis sandbox. Table 1 shows the prevalence of known malicious behaviors for the samples in each dataset. The most commonly implemented behavior for each scenario is different, as their infection context is different. For instance, the higher frequency of network activity in BR samples is explained by the prevalence of Downloaders in this dataset in contrast to the lower frequency of filesystem activity compared to US and JP, scenarios much more targeted by ransomware samples during the collection snapshot. The difference in the context extends beyond the behaviors and also affects the sample construction, thus challenging static analysis classification, the task investigated in this paper.

4 EXPLORING THE SOLUTION SPACE

4.1 Is it enough to have global models?

Feature Selection for malware classification is a Pareto problem. Models can become larger via two processes: (i) by increasing the feature set; and (ii) by increasing the number of parameters. We here evaluate these two possibilities. In the feature dimension, we trained multiple classifiers with an increasing number of features to evaluate their contribution to the overall accuracy. We ordered the features by their statistical significance, such that our experiments start by adding the most relevant features, aiming to minimize the total number of features required to reach the target accuracy (99%). From the parameter perspective, we trained Random Forest (RF) models with an increasing number of ensemble trees. We varied the number of trees from one (Decision Tree) to 2000. For reference, the default tree number for the `scikit-learn` framework is 100.

We varied the number of features in the models from 2 until the convergence to the 99% accuracy. We ensured that all experiments stopped due to the convergence and not due to the lack of new features, since our original pool of features accounted for a thousand variables. In our tests, Decision Trees (DTs) never reached the 99% accuracy goal. In our graphs, we always present the smallest and the largest ensembles that reached the goal.

We show results for the smallest ensemble to reach the 99% score and the largest ensemble tested. We omit intermediate curves from the plots to increase the graph readability. All curves follow the same characteristics and present values intermediate to the smallest and largest plotted ensembles.

Figure 5 shows the accuracy scores for the experiment with the US dataset. In this scenario, the model was trained and evaluated using the samples collected in the same country. We notice that the accuracy grows following a Pareto law (80/20 law), in which only a few features are required to achieve an already significant accuracy rate, but a much bigger number of features are required to take this accuracy to the next level (the 99% score). This is expected since the first features to be added are the most discriminant ones. As independent discriminant features get scarce, a greater number of features is required to allow discrimination based on their combined information. At this point, the increment caused by every new

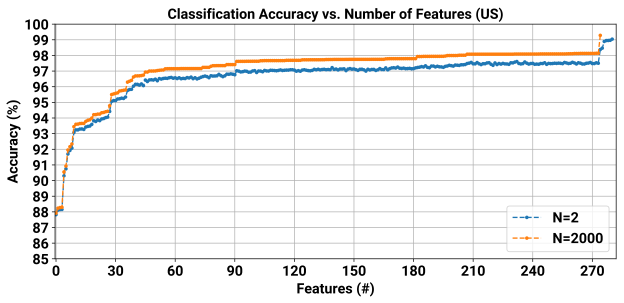

Figure 5: Accuracy rates for the US dataset. Accuracy variation with the increase of the feature set until reaching the 99% value.

feature is marginal. Whereas a few features take the accuracy rate to the 97% level, hundreds of features are required for the model to reach the 99% score.

There is no significant gain in using bigger ensembles in this scenario. Whereas bigger ensembles presented greater accuracy scores for any number of features, this difference has always been smaller than 1%. Models of all ensemble sizes converged to the 99% score. The largest model (N=2000) converged first, with 270 features, whereas the smallest model (N=2) took 290 features to converge.

Model selection from the performance perspective. We previously identified the size of the set of features required to achieve the target 99% accuracy score. Previous work that analyzes the malware classification problem only from the accuracy perspective would tend to say that the larger the feature set, the better. However, we here also propose a performance look at the model size. Thus, in terms of features, there is an accuracy-performance trade-off. The cost-benefit of adding new features is initially high, as every newly added feature contributes little to increasing the model size but a lot to increasing the accuracy score. However, the cost-benefit gets lower over time, since every new feature makes the model bigger but adds only small new discrimination capabilities. Therefore, one should not simply add more features, but add the minimum number of features to make models remain compact and thus more efficient.

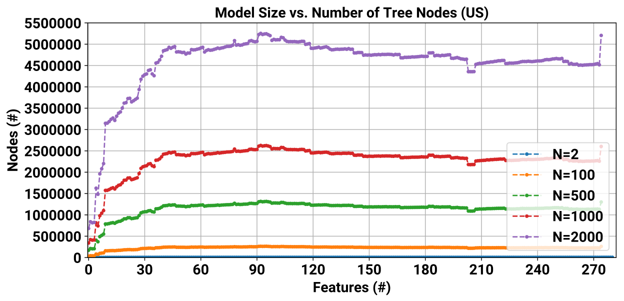

Figure 6: Model size for the US dataset. Number of nodes for an increased number of ensemble trees of increasing feature set sizes.

This same reasoning can be applied to the number of trees in the ensemble. Whereas adding more trees to the ensemble marginally increases the accuracy score, it significantly increases the size of the model. Figure 6 shows the total number of nodes for the models for the US dataset. Initially, the models significantly grow when more features are added, as more information must be processed. After a break-even point, even though more features are added, the models do not significantly grow, since most features present redundant information that can be merged with other conditions. The most important observation, however, is that adding trees linearly increases the size of the models, i.e., doubling the number of trees doubles the model size. It happens because the decision nodes are replicated over multiple trees. We observe in the figure that whereas the number of nodes for the simplest ensemble (N=2) is almost negligible at this graph scale, the number of nodes for the largest ensemble (N=2000) is millions. There is a significant disparity in the number of nodes that can be used to achieve the same 99% accuracy, such that performance and storage requirements must be considered in the deployment decision.

The different needs for the Brazilian scenario. Previous works analyzing feature selection considered only a global model for all datasets. This is unrealistic for an AV company, since distinct scenarios tend to present different characteristics, and thus requirements. To show these differences in practice, we repeated the previously presented experiment for the BR samples.

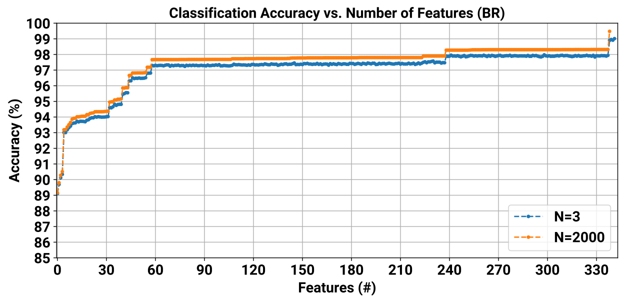

Figure 7: Accuracy rates for the BR dataset. Accuracy variation with the increase of the feature set until reaching the 99% value.

Figure 7 shows the accuracy rate growth with the number of features until the convergence for the BR dataset. Overall, the curve behavior is similar to the one for the US dataset, with the accuracy score initially growing significantly and followed by a period of marginal increase, until convergence. As a difference for the US scenario, the number of features required for this scenario to converge is greater than for the US one. Whereas the previous scenario required less than 300 features, the BR scenario required 340 features to converge. The application of the previous, smaller feature selector to this scenario would lead to sub-optimal accuracy results.

The characteristics of the models are also similar in terms of the ensemble size. Figure 8 shows the number of nodes in the ensembles of different numbers of trees. Once again, the models grow linearly, doubling the size when the number of trees doubles. However, the models for the BR dataset are significantly smaller than for the US scenario. The largest BR ensemble is ≈ 50% the size of the largest US ensemble. It shows that the features in the BR dataset are much

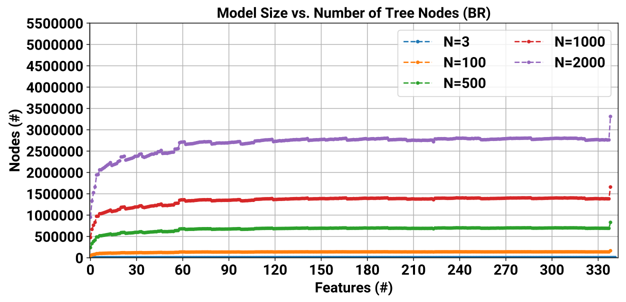

Figure 8: Model size for the BR dataset. Number of nodes for an increased number of ensemble trees of increasing feature set sizes.

more related than in the US scenario, such that can be mixed in the decision nodes, thus reducing the total model size.

The different needs for the Japanese scenario. We complement our analysis by repeating the previous experiments for the JP dataset.

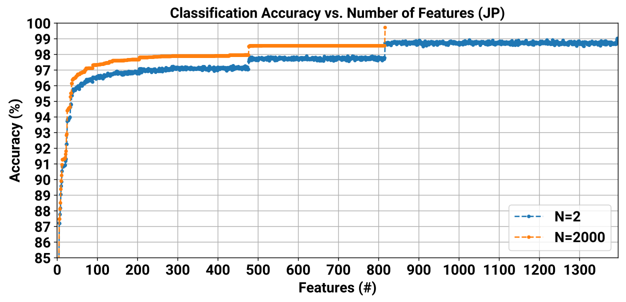

Figure 9: Accuracy rates for the JP dataset. Accuracy variation with the increase of the feature set until reaching the 99% value.

Figure 9 shows the accuracy rate scores for an increased number of features in the JP dataset. Overall, this curve also presents the same characteristics as the ones for the US and BR datasets: the accuracy initially grows significantly with the first features and then the growth is marginally increasing. However, unlike the previous scenarios, the JP dataset presents convergence challenges. In multiple runs, starting from different seeds, the accuracy score was close to 99% but without crossing the bar, thus requiring more and more features until converging. Thus, in multiple cases, such as the worst case plotted in the figure, the number of required features to converge (1400) is almost double that of the average case (800).

In this scenario, having more trees in the ensemble helped the models to deterministically converge with a smaller number of features than the worst-case for a smaller ensemble. Even in the case of converging early, the JP dataset requires significantly more features to converge (800) than the US and BR ones (≈ 300). Applying a feature extractor tuned for these scenarios would lead to an accuracy score of 3% to 4% lower than the target.

Figure 10 shows the model size increase with the addition of features until the average number of features required for the convergence. We limited the plot to this range to highlight the curve

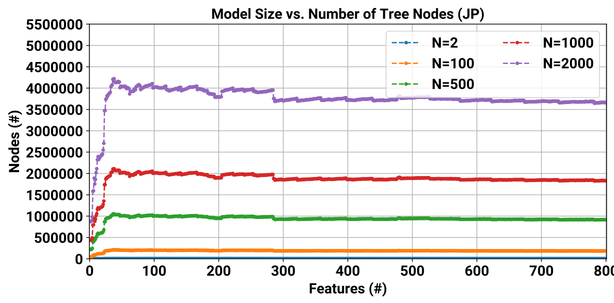

Figure 10: Model size for the JP dataset. Number of nodes for an increased number of ensemble trees of increasing feature set sizes.

format as adding more features does not increase the total number of nodes. With the addition of features, the incoming features are each time more related, such that they only cause nodes to reorganize and fine-tune their decision parameters, but not to split, thus not increasing the total model size. Therefore, the model size is dominated once again by the number of trees rather than by the number of features, such that the trade-off between early convergence due to multiple trees and the addition of more features is better solved from the storage perspective by adding more features.

The different needs for a global model. We showed how results differ from dataset to dataset. We claim it is key to look at the individual scenarios in addition to the global one, as done by most of the previous works. To highlight the differences between local and global models, we created a global model mixing the 3 datasets.

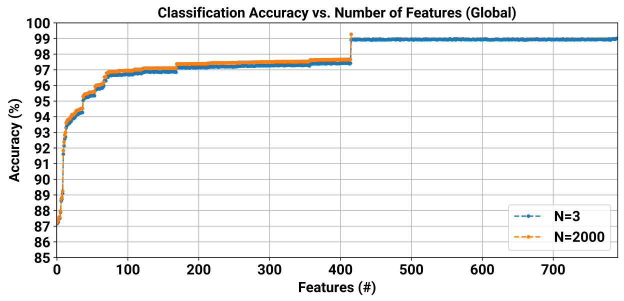

Figure 11: Accuracy rates for the combined dataset. Accuracy variation with the increase of the feature set until reaching the 99% value.

Figure 11 shows how the accuracy grows with feature addition. Overall, the global curve presents the same behavior as the local ones: an initial period of significant growth followed by a period of marginal increments. The global model converges to the 99% accuracy with 400 features if multiple trees are used, and with 800 features if the minimum number of trees (N=3, in this case) is used. It is important to observe that the global model is not a sum of the previous models, but a mix of them. In comparison to the BR and US scenarios, the global model requires more features ($\approx$ 300 to 400) and more trees (2 to 3) to converge, thus showing that creating local models for these scenarios is more performance-efficient. The model requires the same number of trees for the JP scenario, but it requires fewer features to converge, both when considering a large (400 vs 800) or a reduced (800 vs 1300) number of trees, which shows that the data distribution from the other scenarios helped to make the model converge faster. These facts show that if we only look at the global model, we would not have a good understanding of each one of the particular scenarios.

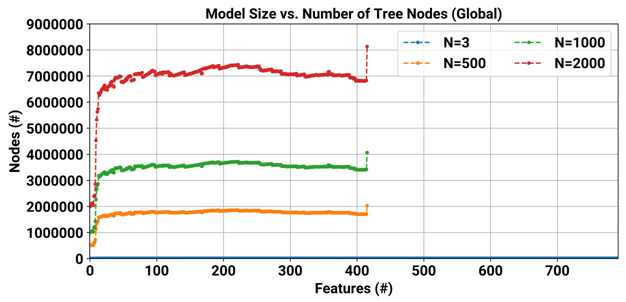

Figure 12: Model size for the combined dataset. Number of nodes for an increased number of ensemble trees of increasing feature set sizes.

A similar mix of scenarios can be observed in the analysis of the model size. Figure 12 shows the total number of decision nodes for the multiple ensemble sizes. We notice that the biggest ensemble in this model has more nodes (7M) than any of the previous scenarios (5M for the US), thus showing that adding more data increased the information diversity, which resulted in more node splits. However, the total ensemble size is still smaller than the sum of the size of the three individual models, thus showing redundancy in the decision data. The ensemble size still grows linearly with the number of trees on it, thus the number of nodes for the models with multiple trees is significantly larger than for the smallest model, even though they use fewer features (the smallest model uses the double). The trade-off between features and trees is again better handled storage-wise with more features.

4.2 Does a global model help?

Why do we need a large, global model? We previously demonstrated that global models do not reflect the reality of specific countries and that large models are not necessarily better than smaller models. So, *is there a reason to have a large, global model?* The reason for that is that in practice AV companies do not have a complete view of each local scenario. In turn, they have to cross-relate data from multiple scenarios to identify trends and attackers' moves. Whereas local, small models are good at detecting the particularities of each scenario, global models are supposed to have better generalization abilities. To demonstrate that in practice, we took the classifiers trained for each local scenario and applied them against the samples from the other scenarios to simulate the case in which an AV company is observing a wave of new, unknown samples coming from a different place.

Figure 13 shows the accuracy rate variation with feature set size and the number of ensemble trees when the trained US model is used to predict the samples from the BR and JP datasets. Notice that in this experiment we significantly expanded the number of

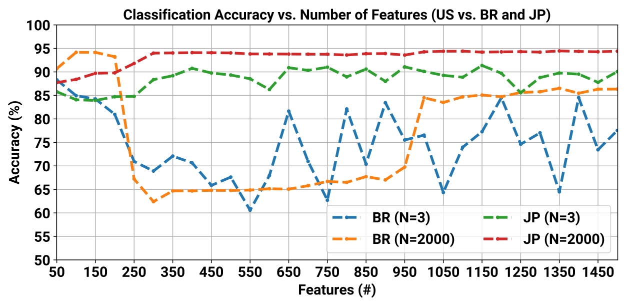

Figure 13: Cross-dataset accuracy rate. Trained US model classifying the samples from the other datasets.

considered features beyond the minimum required to achieve the 99% score in the original dataset. Although the target accuracy goal was achieved for the original samples, it was never achieved for the other datasets. The maximum accuracy value reached was 95% for the JP dataset, thus showing that the characteristics between the scenarios are really different. The detection of BR malware samples is significantly lower (60% in the worst case), which shows that models do not necessarily generalize well over multiple datasets. We notice that in both scenarios it is advantageous to use a larger number of trees. Both for JP and for BR, the large-tree models presented higher accuracy rates. In the BR case, the larger tree presented significantly fewer variations, as clearly seen in the interval with more than 1000 features. Therefore, whereas few-tree models are better at detecting known threats (AV end-points), large-tree models generalize more, thus they are worth using in less constrained devices (AV backends).

Models for different regions really detect different patterns. The generalization ability a model will have is strongly tied to the origin dataset it was trained on. We highlight that aspect by repeating the previous experiments taking the BR and JP datasets as a basis.

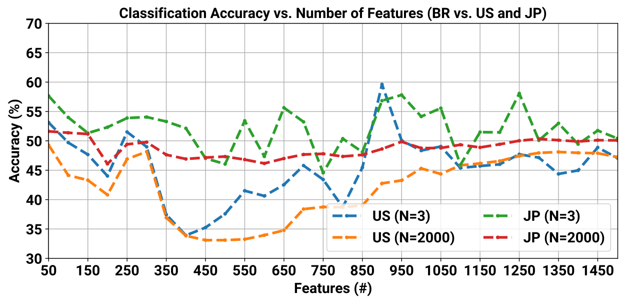

Figure 14: Cross-dataset accuracy rate. Trained BR model classifying the samples from the other datasets.

Figure 14 shows the accuracy score when the BR model is used to predict the US and JP samples. Once again, whereas the original model is able to achieve a 99% accuracy in the original dataset, the results with the datasets from different regions are far from the target. In the BR case, the original model presents less than 50% accuracy when tested with different datasets, indicating that the criteria that are used in the BR dataset to separate goodware from malware are totally distinct from the ones used in the other scenarios, leading to label flipping.

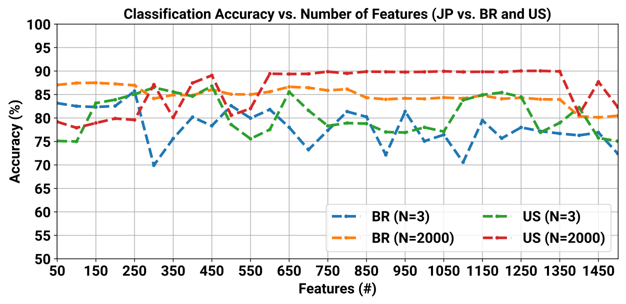

Figure 15: Cross-dataset accuracy rate. Trained JP model classifying the samples from the other datasets.

Figure 15 shows the accuracy score when the JP model is used to predict the US and BR samples. The JP scenario presents the smallest amplitude variation among the datasets. However, the maximum accuracy reached was 90%, far from the 99% target. In this scenario, once again having bigger ensembles helps in sustaining the generalization, as clearly observed between 600 and 1300.

4.3 How to best build local-to-global models?

Federated Learning (FL) helps build global models. We previously showed that global models are good for generalizing and discovering new threats and that local models are required to better understand the local scenarios. Therefore, it is key to have a mechanism to bridge the gap between them. A solution for that is to build a FL mechanism inside the AV company in which the local models send data to the AV server to build a global model. A challenge is that AV companies will not have an entire dataset at some point, but the dataset is incrementally built as more samples are collected over time. We evaluate this scenario via an experiment in which we send the same proportion of the different datasets to a global server to simulate the effect of the collection over time. The sent samples are randomly chosen. The trained global model is used to predict the remaining samples in the three datasets.

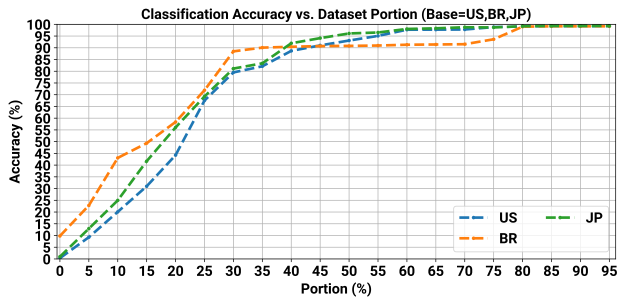

Figure 16: Building a global model. Accuracy rate for building a global model from different portions of the source datasets.

Figure 16 shows the accuracy rate achieved in detecting the samples of the three datasets when different portions of these datasets are used to build a global model. In this experiment, the feature set size is the one that allows the global model to achieve the 99% accuracy rate. The accuracy rate starts from zero, when no data is available, and grows as more data is provided to the model. The global model achieved the 99% accuracy rate for the three entire datasets when 80% of all datasets were considered. It is interesting to notice that the global model does not need an entire view of the local datasets to scale well. With 30% of all the source datasets, the total detection rate is already over 80%. With 50% of the datasets, the accuracy rate is already over 90%. This shows that more important than the amount of data, sharing data is the key step for increased detection capabilities.

Enriching an existing scenario is more efficient than training from scratch. We have previously shown that sending data from the local model to the global model helps create a model that generalizes more and thus discovers more new samples. We are not the first to observe this possibility, but previous works present an assumption that does not fit reality. They assume that the models will be trained from scratch, as we previously presented. In reality, AV companies will have a baseline model that they already operate, therefore, the correct way to evaluate that is by enriching the existing model with data from the other scenarios. The problem with assuming a model created from scratch is that it will report that the portion of the dataset required to train a model is different it actually is in practice. To show this difference, we repeated the experiment by refining the existing models. In all cases, a random selection of samples is sent to the main model.

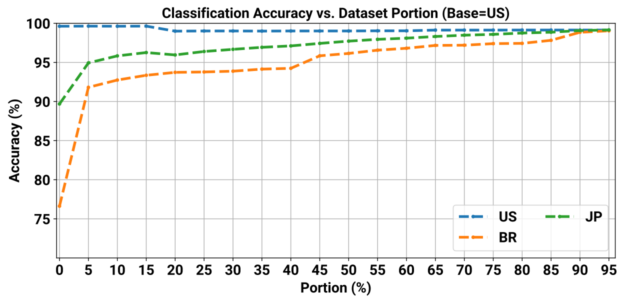

Figure 17: Extending the existing US model. Accuracy rates on the different datasets for different portions of the source datasets using random sample selection.

Figure 17 shows the accuracy rate when the US model is enriched with different portions of the BR and JP datasets and it is used to predict malware samples in these two datasets. We notice the model already starts classifying the samples from the US dataset at the 99% accuracy rate it was trained on. The increase of data from other scenarios marginally diminished the accuracy to a bit over 98%. In turn, adding data from other scenarios significantly increased the accuracy rate for them. Adding 5% of data from other scenarios significantly increased the detection rate. For the BR samples, for instance, it was enough to grow the accuracy rate from 75% to over 95%. After this point, the accuracy marginally grows with more data. In the end, the US model becomes able to detect BR and JP samples with 99% accuracy. The advantage in this scenario is the ability to quickly start detecting more samples as the first chunks of data are added to the classifier.

Figure 18 shows the accuracy rate when the BR model is enriched with different portions of the US and JP datasets and it is used to

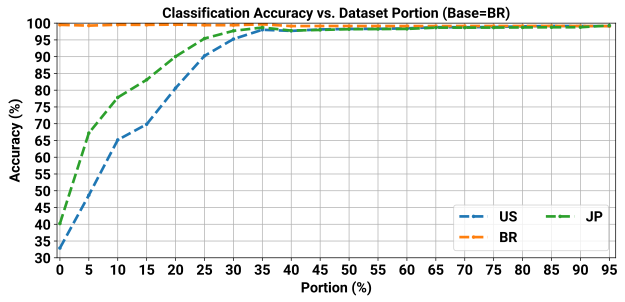

Figure 18: Extending the existing BR model. Accuracy rates on the different datasets for different portions of the source datasets using random sample selection.

predict malware samples in these two datasets. Once again, the detection in the BR dataset is not affected by the data addition. In turn, adding data from the other datasets significantly increased the detection capabilities for these other datasets. In comparison to the US scenario, the BR scenario is more challenging as a base scenario, thus it requires more data. It requires adding 35% of the other datasets to achieve the 95% detection rate. In the end, all scenarios achieved the 99% accuracy.

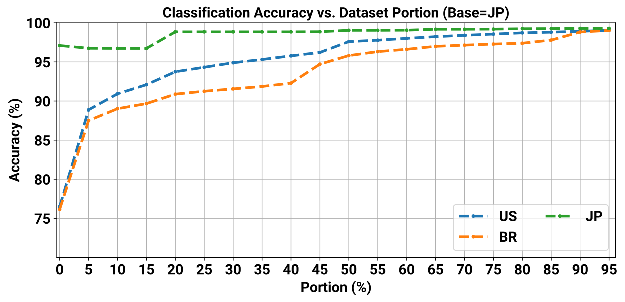

Figure 19: Extending the existing JP model. Accuracy rates on the different datasets for different portions of the source datasets using random sample selection.

Figure 19 shows the accuracy rate when the JP model is enriched with different portions of the US and BR datasets and it is used to predict malware samples in these two datasets. This scenario exhibits mixed characteristics between the BR and US scenarios. Whereas it presents most of its growth with only 5% of the other datasets, it also took a significant time to converge (50% of the dataset to achieve 95%), like the BR scenario.

Global models are more efficiently built with confidence-based sample selection than random sample selection. Whereas the previously-presented strategy is already more efficient than training from scratch, it is still not ideal, because it considers all samples as contributing equally to the global model. In reality, some samples contribute more than others and these should be prioritized for fast learning. A prioritization strategy is to check which samples are classified with less confidence by the main classifier and send them to the global classifier. We evaluated this strategy by repeating the previous experiment under this new setting.

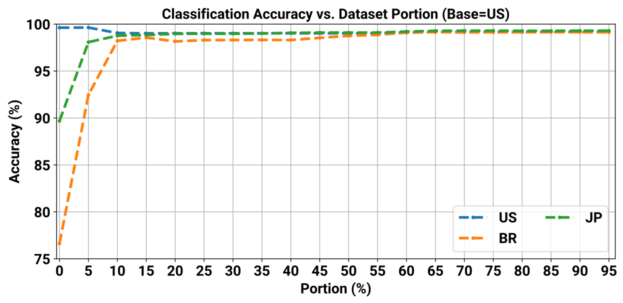

Figure 20: Extending the existing US model. Accuracy rates on the different datasets for different portions of the source datasets using confidence-based sample selection.

Figure 20 shows the accuracy rate increase when the US model is enriched with samples from the BR and JP datasets selected by the low confidence of the US classifier on detecting them. Like in the previous scenario, most of the detection increase happens in the first 5%, taking detection rates over to 90%. In this case, however, adding another 5% (up to 10%) is enough to take the accuracy rate closer to the 99% target. The accuracy rate is sustained for additional portions of the dataset.

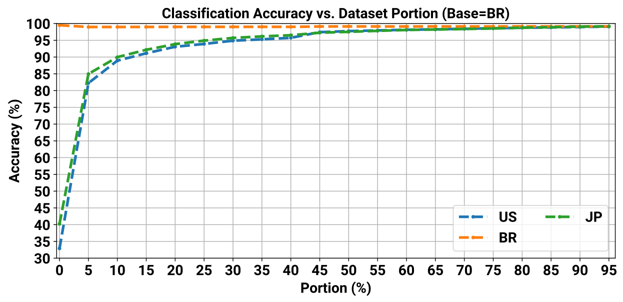

Figure 21: Extending the existing BR model. Accuracy rates on the different datasets for different portions of the source datasets using confidence-based sample selection.

Figure 21 shows the accuracy rate increase when the BR model is enriched with samples from the US and JP datasets selected by the low confidence of the BR classifier on detecting them. The BR model also benefits from a confidence-based sample selection strategy. Whereas in the random selection setting it takes 25% of the other datasets to achieve the 90% accuracy rate, now this same rate is achieved with only 10% of the datasets. The BR dataset has been revealed challenging since it required a significant portion of the datasets to be added to reach the 99% accuracy level. Despite that, the overall result is considered positive because detecting most samples earlier is positive by reducing the attack opportunity window [8].

Figure 22 shows the accuracy rate increase when the JP model is enriched with samples from the BR and US datasets selected by the low confidence of the JP classifier on detecting them. The selection strategy is revealed to be efficient as the JP model enriched with only 10% of the samples from the other datasets is able to achieve the

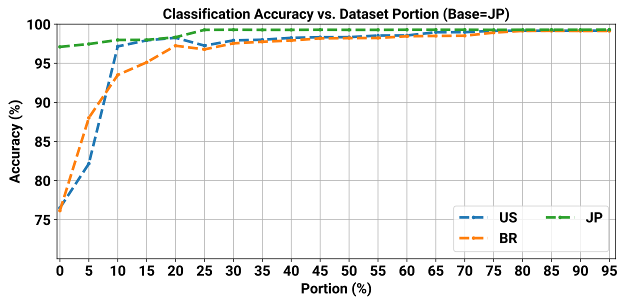

Figure 22: Extending the existing JP model. Accuracy rates on the different datasets for different portions of the source datasets using confidence-based sample selection.

99% accuracy rate for the US scenario. The BR was once again more challenging, requiring $\approx$ 15% of the dataset to pass the 90% level. The JP scenario however once again presented convergence problems, with the accuracy rates varying significantly until converging to the target level.

4.4 Are real models trained from scratch?

In practice, models are not retrained, but distilled. We so far investigated the problem of retraining models in different conditions. However, assuming that model retraining is always feasible is also a common pitfall for realistic ML deployments. In reality, AV companies might not be able to retrain their models from scratch due to multiple reasons. For instance, they might not have the original samples used to train the samples, but only the model parameters. Therefore, in practice, multiple models are only derived from previous ones (i.e., distilled). To better understand the implications of a model distillation process, we developed a series of experiments employing the Teacher-Student (TS) distillation technique [20, 23], where the labels of the original model were used to train the distilled model. We started distilling the local models.

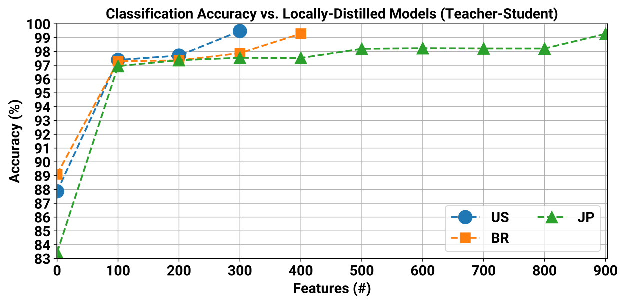

Figure 23: Self-Model Distilling. Number of features required to achieve the maximum accuracy rate for the different datasets.

Figure 23 shows the accuracy score variation according to the number of features considered in the model for the case in which we distill the original local models for each one of the three datasets into new versions of them. We notice that effective model distillation is possible, i.e., the distilled model can also achieve the targeted 99%

accuracy score. However, it comes at the cost of some extra features. It required 10 (290 to 300), 60 (340 to 400), and 100 (800 to 900) additional features to make the US, BR, and JP datasets converge to the target accuracy. This increase is explained by the loss of accuracy caused by considering the labels provided by the trained model and not by the ground truth ones. This loss of accuracy must be compensated with additional data. Therefore, there is also a trade-off between the model accuracy and the model size in the distillation procedure.

Distilled models are bigger than retrained models. Considering that models can be built via distillation is important to report accurate results about models' sizes. As the challenges to distilling a model might be different than to building from scratch, the number of features required by the models might be different. To evaluate that, we extended our evaluation from distilling local models to local models to distill the global model from the previous experiments into multiple local models.

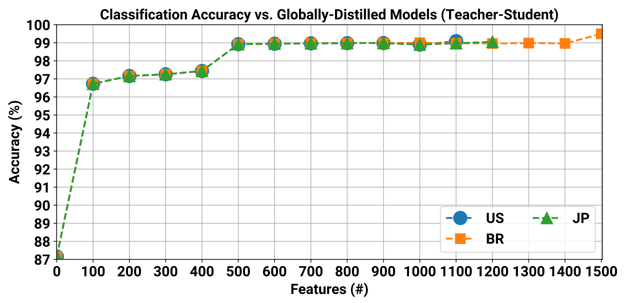

Figure 24: Global to Local Model Distilling. Number of features required to achieve the maximum accuracy rate for the different datasets.

Figure 24 shows the accuracy rate for different feature sets for the three datasets. As the curves overlap, we added markers to them to highlight the differences. We notice that distilling a global model to a local model requires much more features than self-model distillation. This happens because the errors in the labels of all samples are accumulated, which requires more data to be fixed. In the long term, all local models converged to the target 99% accuracy score when applied to their local scenarios, but they took a different number of features for that. The global US model converged with 1000 features, when it stopped being displayed in the graph. The JP dataset converges with 1200 features, when it stops being displayed in the graph. The BR dataset was once again the most challenging one, requiring 1500 features until converging.

The feature set size for the distilled models varies with the dataset proportion. The previously shown effects of model rebuilding via distilling are not the only ones affecting model construction in reality. The effects of partial global model construction via FL also play a key role. Therefore, it is key to evaluate these two effects in conjunction to not report inaccurate feature set sizes. The addition of different dataset portions makes the feature selection problem dynamic, such that we should understand how the feature requirement progresses. We here investigate how ideal feature size varies by repeating previous experiments with different proportions of the source datasets.

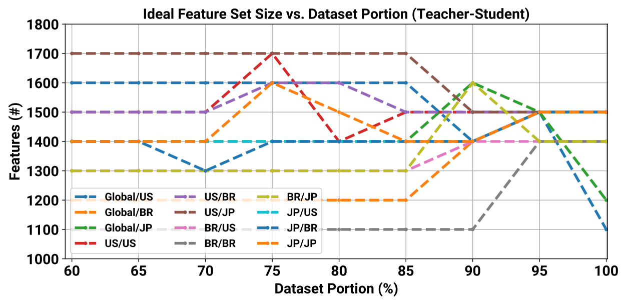

Figure 25: Global to Local Model Distilling. Variation on the number of features required to achieve the maximum accuracy rate for different portions of the source datasets.

Figure 25 shows how the number of features required to reach the target accuracy varies with the portion of the datasets that are used to train multiple classifiers via distillation. We report the results both for the distilling of the local and global models and their application to multiple local scenarios. Whereas there is some initial stability, we notice that the maximum number of features for each scenario varies significantly, with a tendency to decrease when the full dataset is used, since more data is available. This variation suggests that the adoption of a dynamic distilling strategy is desirable. Although it would be possible to identify a larger number of features that would cover all scenarios (1700) and keep the feature set size constant, it would not be ideal performance-wise, since it requires traversing much more nodes than needed in some scenarios (e.g., 1100), making the prediction process slower.

4.5 What is the real impact of ML on AVs?

The performance impact of different feature set sizes is bigger in actual rules than in the ML models. The impact of selecting a different number of features is highlighted when we consider the detection task in endpoint machines. A common assumption of most papers in the literature is that the local model will be directly used in the end user machine (i.e., the AV), but this is not the common case in practice. Most AVs deploy rules as their matching mechanisms [10]. The use of model-derived rules rather than the actual models can be motivated by multiple reasons: (i) do not reveal the entire model; (ii) make human interpretation easier; (iii) mitigate FPs by removing individual rules; and (iv) making updates faster by deploying only new rules rather than an entire model. Thus, we should evaluate the impact of different feature set sizes on the performance by inspecting the rules.

The rules can be directly derived from the local model by traversing the decision tree. We used this strategy to derive YARA [38] rules, the industry standard for pattern matching, from the model. Whereas the difference in the matching time of a rule on a single file is small, the accumulated time for scanning a large set of files might be significant. Thus, we simulated this scenario in our experiment. We distributed all malicious files that the model was expected to detect (the same used in the ML experiment) over the entire filesystem of a fresh Windows 10 installation. In total, we asked YARA to scan 127GB of data in this system, simulating a full system scan by an AV. All YARA rules were precompiled to avoid the compilation overhead and were run in the same Intel I7, 20 core, 8GB system. The number and complexity of the matching rules varied with the number of features, as detailed in Appendix A. On average, 900 rules of 25 levels of depth were considered.

Table 2 shows the total matching time for the set of rules derived from the models using different feature set sizes. Our goal is to motivate the use of smaller models for the scenarios that allow it rather than standard large models for all scenarios. The results show that the overhead caused by using larger and increasingly complex rules is significant. Using a standard feature set of 1700 features is 40% slower than using a minimal set of 1100 features (for the scenario where it is possible). This 40% overhead represents an additional 5m36s in the matching for the simulated scenario. This result highlights the benefits of a feature-aware distill schema.

5 CASE STUDY: TEMPORAL EVALUATION

Once we explored the impact of different model settings on the multiple datasets, we now evaluate how the proposed new setting would help increase malware detection in a more realistic scenario. To do that, we evaluate malware detection as a time series, as new samples appear over time, and not in a single batch. Thus, we ordered the samples in the three datasets by the *"first seen"* date in Virustotal [39], as done by related work [12]. The datasets we had access to were imbalanced in time, thus we were not able to provide a comparison of the whole year. However, the datasets are reasonably balanced in the second semester of the year. Thus, we split the dataset in two, training with the first half of the year (≈50% of samples) for each dataset, and predicting the next six months of the year (normalized to 980 samples per month per dataset).

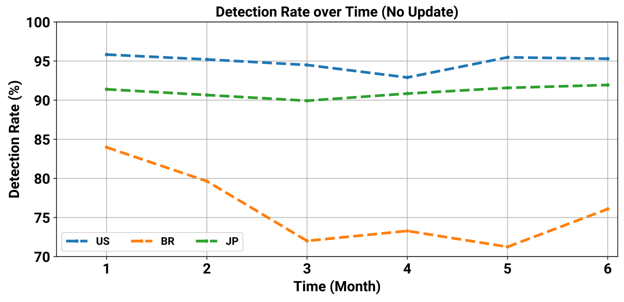

Figure 26: Detection rate as a time-series for the individual static models. Previously trained classifiers attempt to detect new threats. Performance degradation due to concept drift is observed.

In the first experiment, each classifier was trained with the best set of features for the training step (500 for US, 300 for BR, and 800 for JP). The same set of features was kept during the whole year. Figure 26 shows the detection rate for the three datasets in the six months of the second semester of the collection year. We notice that each dataset presents a distinct detection rate, according to their samples' characteristics. In this experiment, the BR samples were the hardest to classify, which is in line with the reports of previous works on the particularities of BR malware [7]. It is possible to notice a clear degradation in the detection performance of the BR classifier over time, indicating the occurrence of concept drift. While our experiment is of reduced scale, it is in line with literature reports about the frequent occurrence of concept drift in BR malware [15].

The typical strategy to handle concept drift occurrences is to retrain the classifier when drift occurs. We simulated this scenario by detecting drift via the Early Drift Detection Methods (EDDM) [3]. We allowed the feature extractor to be retrained as well since previous research demonstrated it leads to better results [12]. It implies that the number of features used by the classifiers varied over time (but never got larger than 800), being optimal at every retraining.

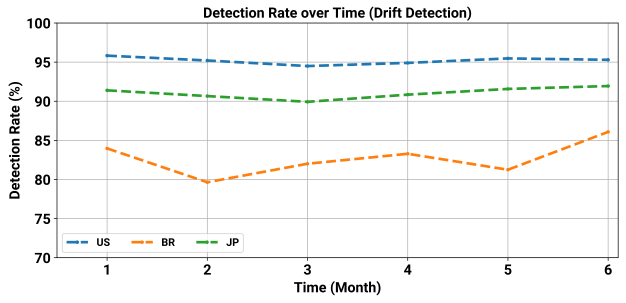

Figure 27: Detection rate as a time-series for the individual, drift-aware models. The retraining of models when concept drift is detected takes the detection rate back to its original level.

Figure 27 shows the detection rates when retraining on drift detection is in place. We notice that the BR detector does not degrade its performance anymore. Also, we notice that a drift point was identified for the US curve in the fourth month. No concept drift occurrence was detected for the JP scenario. As a drawback of this strategy, the detection rates did not increase from their original levels even with retraining.

A hypothesis for that is that whereas retraining on drift detection allows reorganizing the feature importance distribution, it does not add new information, which can only be added externally to the local models. To evaluate that, we considered our proposition of training a global model and distilling local versions. The global model was initially trained with all samples of the 3 countries for the first semester. Later, different models were distilled every month, considering the ideal feature set size for each scenario (ranging from 1100 to 1500), as previously discussed. The global model is also updated every month with the samples from the previous months.

Figure 28 shows the detection results for the scenario with global model distillation to local models. We notice that, like in the previous scenario, drift events are mitigated by the periodic distill, which is equivalent to model retraining. However, unlike the previous scenario, the detection rates are at a higher level due to the sharing of data between the datasets. Whereas the BR malware samples remain harder to detect due to their unique characteristics, their difference from the other scenarios is much smaller in this case.

The main takeaway for this experiment is that whereas retraining on drift detection mitigates the performance degradation, only federated learning actually increases detection rates. A derived takeaway is thatthe increased detection rate of distilled global models comes at the cost of an additional number of features.

Table 2: Matching performance. Wall time (s) for matching Yara rules derived from ML models of different feature sets sizes against a real, infected filesystem.

Features	1100	1200	1300	1400	1500	1600	1700
Time	13m57s	14m00s (+0.3%)	14m05s (+1%)	14m50s (+6%)	15m57s (+14%)	17m58 (+29%)	19m33s (+40%)

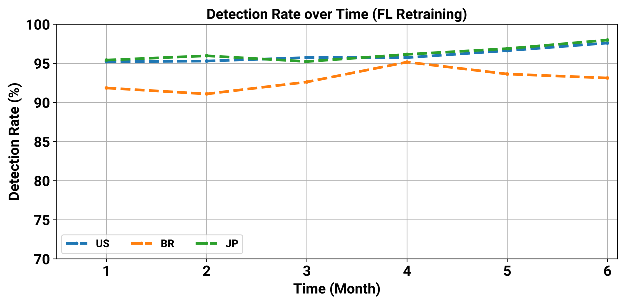

Figure 28: Detection rate as a time-series for the globally-distilled models. The use of data from a global model not only mitigated the drift effects but also increased the detection rate for all datasets.

6 DISCUSSION

Impact on Threat Models. Our work tackles the problem of malware detection in heterogeneous scenarios. We are not the first to claim that different scenarios require different solutions. In this sense, our contribution is to explicit what are **the types** of differences observed in analyzing heterogeneous datasets. We use the identified requirement for a different number of features per dataset to motivate the adoption of an ensemble of heterogeneous trees.

Impact on Federated Learning. When we demonstrate **(i)** the need for specifically considering the heterogeneity of geographically distributed malware and **(ii)** the need for integrating individual knowledge for increased detection rates, we are also **(i)** pinpointing the trade-off between local and global instances, and **(ii)** the need for designing architectures that balance these factors. Our key contribution is to notice that we do not need to design a completely new solution for that, but we can achieve this goal by taking a different look at federated learning, by considering as clients of the global models the local AV subsidiaries and not the endpoints.

FL without poisoning risks. Our proposal for the adoption of FL comes along with the idea of distributing the AV operation. Our proposal does not cause a paradigm shift in the way AV companies operate but implies in the replication of the AV operation model into multiple subsidiaries. When we move the FL client from the endpoint device to the local AV subsidiary model, we increase the trust level the global AV model has in the data it receives in comparison to directly receiving data from an endpoint device. In our threat model, although the malware files are sourced from the endpoint devices (as in all AVs [10]), they are not directly incorporated into the local or global models, but curated by the local AV subsidiary via their analysts (as in typical AV operations [29]). Therefore, a collusion of endpoints at a single subsidiary would not be able to poison the model. Poisoning the global models would only be possible via collusion of subsidiary models, which is out of the assumed threat model of an AV company trusting its subsidiaries.

Why are malware samples regionally different? To lead to an effective infection, malware campaigns must be meaningful to the victim's context (e.g., use known languages, known topics, known companies, and so on). The types of assets affected by malware samples also only make sense to an attacker if the attacker have access to them (e.g., same payment methods, same banks, same currency, and so on). Therefore, malware samples are naturally shaped by the population they target. Previous work has demonstrated, for instance, how the ecnomic development of Brazil has led to particular types of malware [7], such as a proliferation of custom banking samples due to the unique payments methods developed in this country [11]. This phenomenon is reflected in this research, whose BR dataset has filetypes (e.g., CPL) unique to it.

How are the features affected by the regional differences? The pointed contextual differences shape the technical decisions made by the malware creators and thus affect the way malicious behaviors are implemented. This is reflected in the detection of samples. Previous research has shown that due to the local differences BR samples are less detected by AVs than global samples [8] and that BR malware also causes more classifiers drifts [15]. The contextual difference in our study is that while the BR scenario was targeted by data exfiltration (e.g., banking) malware, the US and JP datasets were mroe targeted by ransomware. If we look to the most discriminative feature for each scenario, we notice that the top-10 most discriminative feature for the BR dataset are network functions and for the US and JP datasets are filesystem functions, exactly what one would hypothesize for scenarios dominated by data exfiltrators and ransomware samples.

Will malware remain regionalized in the future? It is key to highlight that although our datasets are representative of the BR, US, and JP scenarios by that time, these datasets are not representative of the current threat scenario, which is very dynamic. However, we understand that our work is permanentely relevant as contextual differences between the countries will also be permanent. Since the technical design decisions are coupled to the contextual differences, any operation in heterogeneous scenarios will present significant differences in the feature requirements, as here exemplified. However, instead of the difference between data exfiltrators and ransomware, different malware families are expected according to the scenario's developments.

Generalization Limitations. We selected for our experiments a coherent set of malware examples: they are collected by the same company, during the same period, from the same type of users (infected machines). However, we acknowledge that these datasets are only a small fraction of all threats that exist out there, although this is the best representation presented to the moment in comparison to the literature works. Thus, we do not claim that the statistics obtained from the experiments with these samples will generalize for all scenarios or over time. Instead, we limit our claims to demonstrate that it is possible to find in the reality scenarios in which the problem addressed in this research is of actual impact. For the future, it is key to conduct a large-scale study to answer the questions derived from our observations and their generalization for multiple datasets over time. At the present, we evaluate the generalization of our claim that heterogeneous datasets inherently cause classification impact via the experiments presented in Appendix B.

7 RELATED WORK

Realistic Malware Detection Pipelines. Recent works point out that ML-based detectors need to be more practical [14] and present more realistic assumptions in their evaluations [1]. Previous work addressed issues such as improper dataset imbalances [31]. We complement them by considering performance requirements.
Concept Drift issues have been addressed in multiple works [4, 12, 22, 32]. However, none of them considered the effect of simultaneous heterogeneous datasets, as in this work.
Model Distillation. Whereas the distill of Random Forest (RF) has been proposed in the past in the literature [23], our contribution here is to build the ensemble tree in a way that favors direct distillation. Previous distillation works focused on creating models that are stronger against adversaries [21], but not necessarily suitable for easy distillation. Also, whereas previous works observed that distilled models might have a mix of characteristics [33], they did not explore it to handle the differences in regional datasets.
Federated Learning. Whereas previous works suggested that AV companies could benefit from FL [18], this idea was never completely developed. Also, their focus is often on keeping users' data external to some company environment, whereas our scenario is internal to the AV company. A key difference from previous works is that whereas most literature works propose models to be run in the edge devices [25, 28], we take the performance constraints into consideration and propose the FL to be run inside the AV company infrastructure. The agents of the FL process are the AV company subsidiaries, not the users. This solves this risk of poisoning associated with FL [17, 37]. Whereas in client-side FL the dataset can be poisoned by a set of malicious actors, in our scenario the clients are trusted because they are internal to the AV company.
Combining FL and Distilling. Whereas combining the two techniques has been proposed for other domains [24], this is the first use for malware detection. Whereas previous works distilled an ensemble into another [26], we propose building a heterogeneous ensemble to facilitate single-tree distillation.
Adaptive Classifiers. Random Forest modifications have been proposed in the past [19], but only a few works modified RF for easing model distilling. The closest proposal to it was the creation of heterogenous RF with different features per tree [2]. We here sorrt the RF trees by an incremental number of features (with overlap with the previous ones) which allows easy model distills steps.
Classification Performance Trade-Offs. This work sheds light on ML performance. Although large models tend to present greater detection rates, they also tend to be too big to run in edge devices [40]. Thus, finding a good trade-off between model size and accuracy is key. A typical strategy for that is to partition the models into cloud and edge versions [36], a strategy also leveraged in this work. However, whereas the typical trade-off is in the number of trees [16], we here approached it also in the number of features.

8 CONCLUSION

We investigated how malware detection pipelines proposed in the literature often do not consider the challenges involved in the actual operation in heterogeneous scenarios, such as that: (i) the ML model that runs in a client machine is different from the model that runs in an AV company backend; and (ii) datasets of malware samples collected in different regions of the world present different detection requirements in terms of model complexity. We evaluate the impact of overlooking these aspects by modeling a ML-based malware detection pipeline to be applied to 3 datasets of 10K malware samples each collected in the same period of time in three different countries: USA, Brazil, and Japan. We show that (i) the ideal model for each scenario requires a different number of features; (ii) increasing the model size does not lead to significant detection gains; and (iii) integrating data from all the scenarios in a global model indeed raises the detection rates for all datasets. We proposed the use of FL in combination with model distilling to build the global dataset. Unlike previous proposals, our FL approach is run inside the AV company, and not on the endpoints. Also, we modified the RF algorithm to use a heterogeneous number of features in its ensemble, thus favoring the distillation of different models. We expect this work to foster further research on cross-regional malware.
Reproducibility. All developed codes for this research is available at: https://github.com/marcusbotacin/Malware.Federated.Distill

ACKNOWLEDGMENTS

We thank the anonymous reviewers and shepherd for all the helpful insights. Marcus Botacin thanks NSF for the support via the CNS 2327427 grant. Heitor Gomes thanks the Marsden Fund for the award number VUW2213.

REFERENCES

[1] Daniel Arp, Erwin Quiring, Feargus Pendlebury, Alexander Warnecke, Fabio Pierazzi, Christian Wressnegger, Lorenzo Cavallaro, and Konrad Rieck. 2022. Dos and Don'ts of Machine Learning in Computer Security. In *31st USENIX Security Symposium (USENIX Security 22)*. USENIX Association, Boston, MA, 3971–3988. https://www.usenix.org/conference/usenixsecurity22/presentation/arp
[2] Mohamed Bader-El-Den. 2014. Self-adaptive heterogeneous random forest. In *2014 IEEE/ACS 11th International Conference on Computer Systems and Applications (AICCSA)*, Vol. 1. IEEE, Qatar, 640–646. https://doi.org/10.1109/AICCSA.2014.7073259
[3] Manuel Baena-García, José Campo-Ávila, Raúl Fidalgo-Merino, Albert Bifet, Ricard Gavald, and Rafael Morales-Bueno. 2006. Early Drift Detection Method. *Fourth International Workshop on Knowledge Discovery from Data Streams* 1 (01 2006), 1.
[4] Federico Barbero, Feargus Pendlebury, Fabio Pierazzi, and Lorenzo Cavallaro. 2022. Transcending TRANSCEND: Revisiting Malware Classification in the Presence of Concept Drift. In *2022 IEEE Symposium on Security and Privacy (SP)*. IEEE, US, 805–823. https://doi.org/10.1109/SP46214.2022.9833659
[5] Albert Bifet, Geoff Holmes, Richard Kirkby, and Bernhard Pfahringer. 2010. MOA: Massive Online Analysis. *Journal of Machine Learning Research* 11, 52 (2010), 1601–1604. http://jmlr.org/papers/v11/bifet10a.html
[6] Marcus Botacin. 2021. Does Your Threat Model Consider Country and Culture? A Case Study of Brazilian Internet Banking Security to Show That It Should!. In *USENIX Enigma*. USENIX Association, USA, 1.
[7] Marcus Botacin, Hojjat Aghakhani, Stefano Ortolani, Christopher Kruegel, Giovanni Vigna, Daniela Oliveira, Paulo Lício De Geus, and André Grégio. 2021. One Size Does Not Fit All: A Longitudinal Analysis of Brazilian Financial Malware. *ACM Trans. Priv. Secur.* 24, 2, Article 11 (jan 2021), 31 pages. https://doi.org/10.1145/3429741
[8] Marcus Botacin, Fabricio Ceschin, Paulo de Geus, and André Grégio. 2020. We need to talk about antiviruses: challenges & pitfalls of AV evaluations. *Computers & Security* 95 (2020), 101859. https://doi.org/10.1016/j.cose.2020.101859
[9] Marcus Botacin, Fabricio Ceschin, Ruimin Sun, Daniela Oliveira, and André Grégio. 2021. Challenges and pitfalls in malware research. *Computers & Security*

106 (2021), 102287. https://doi.org/10.1016/j.cose.2021.102287
[10] Marcus Botacin, Felipe Duarte Domingues, Fabrício Ceschin, Raphael Machnicki, Marco Antonio Zanata Alves, Paulo Lício de Geus, and André Grégio. 2022. AntiViruses under the microscope: A hands-on perspective. *Computers & Security* 112 (2022), 102500. https://doi.org/10.1016/j.cose.2021.102500
[11] Marcus Botacin, Anatoli Kalysch, and André Grégio. 2019. The Internet Banking [in]Security Spiral: Past, Present, and Future of Online Banking Protection Mechanisms based on a Brazilian case study. In *Proceedings of the 14th International Conference on Availability, Reliability and Security* (Canterbury, CA, United Kingdom) *(ARES '19)*. Association for Computing Machinery, New York, NY, USA, Article 49, 10 pages. https://doi.org/10.1145/3339252.3340103
[12] Fabrício Ceschin, Marcus Botacin, Heitor Murilo Gomes, Felipe Pinagé, Luiz S. Oliveira, and André Grégio. 2023. Fast & Furious: On the modelling of malware detection as an evolving data stream. *Expert Systems with Applications* 212 (2023), 118590. https://doi.org/10.1016/j.eswa.2022.118590
[13] Fabricio Ceschin, Marcus Botacin, Gabriel Lüders, Heitor Murilo Gomes, Luiz Oliveira, and Andre Gregio. 2021. No Need to Teach New Tricks to Old Malware: Winning an Evasion Challenge with XOR-Based Adversarial Samples. In *Reversing and Offensive-Oriented Trends Symposium* (Vienna, Austria) *(ROOTS'20)*. Association for Computing Machinery, New York, NY, USA, 13–22. https://doi.org/10.1145/3433667.3433669
[14] Fabrício Ceschin, Heitor Murilo Gomes, Marcus Botacin, Albert Bifet, Bernhard Pfahringer, Luiz S. Oliveira, and André Grégio. 2020. Machine Learning (In) Security: A Stream of Problems. https://doi.org/10.48550/ARXIV.2010.16045
[15] Fabricio Ceschin, Felipe Pinage, Marcos Castilho, David Menotti, Luiz S. Oliveira, and Andre Gregio. 2018. The Need for Speed: An Analysis of Brazilian Malware Classifiers. *IEEE Security & Privacy* 16, 6 (2018), 31–41. https://doi.org/10.1109/MSEC.2018.2875369
[16] Lingling Fan, Minhui Xue, Sen Chen, Lihua Xu, and Haojin Zhu. 2016. POSTER: Accuracy vs. Time Cost: Detecting Android Malware through Pareto Ensemble Pruning. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security* (Vienna, Austria) *(CCS '16)*. Association for Computing Machinery, New York, NY, USA, 1748–1750. https://doi.org/10.1145/2976749.2989055
[17] Minghong Fang, Xiaoyu Cao, Jinyuan Jia, and Neil Gong. 2020. Local model poisoning attacks to {Byzantine-Robust} federated learning. In *29th USENIX security symposium (USENIX Security 20)*. USENIX, US, 1605–1622.
[18] Hossein Fereidooni, Samuel Marchal, Markus Miettinen, Azalia Mirhoseini, Helen Möllering, Thien Duc Nguyen, Phillip Rieger, Ahmad-Reza Sadeghi, Thomas Schneider, Hossein Yalame, et al. 2021. SAFELearn: Secure aggregation for private federated learning. In *2021 IEEE Security and Privacy Workshops (SPW)*. IEEE, US, 56–62.
[19] Heitor M Gomes, Albert Bifet, Jesse Read, Jean Paul Barddal, Fabrício Enembreck, Bernhard Pfharinger, Geoff Holmes, and Talel Abdessalem. 2017. Adaptive random forests for evolving data stream classification. *Machine Learning* 106 (2017), 1469–1495.
[20] Jianping Gou, Baosheng Yu, Stephen J. Maybank, and Dacheng Tao. 2021. Knowledge Distillation: A Survey. *Int. J. Comput. Vision* 129, 6 (jun 2021), 1789–1819. https://doi.org/10.1007/s11263-021-01453-z
[21] Kathrin Grosse, Nicolas Papernot, Praveen Manoharan, Michael Backes, and Patrick McDaniel. 2017. Adversarial examples for malware detection. In *Computer Security–ESORICS 2017: 22nd European Symposium on Research in Computer Security, Oslo, Norway, September 11-15, 2017, Proceedings, Part II 22*. Springer, Norway, 62–79.
[22] Roberto Jordaney, Kumar Sharad, Santanu K. Dash, Zhi Wang, Davide Papini, Ilia Nouretdinov, and Lorenzo Cavallaro. 2017. Transcend: Detecting Concept Drift in Malware Classification Models. In *26th USENIX Security Symposium (USENIX Security 17)*. USENIX Association, Vancouver, BC, 625–642. https://www.usenix.org/conference/usenixsecurity17/technical-sessions/presentation/jordaney
[23] Sangwon Kim, Mira Jeong, and Byoung Chul Ko. 2022. Lightweight surrogate random forest support for model simplification and feature relevance. *Applied Intelligence* 52, 1 (2022), 471–481.
[24] Daliang Li and Junpu Wang. 2019. Fedmd: Heterogenous federated learning via model distillation. *arXiv preprint arXiv:1910.03581* 1 (2019), 1.
[25] Kuang-Yao Lin and Wei-Ren Huang. 2020. Using Federated Learning on Malware Classification. In *2020 22nd International Conference on Advanced Communication Technology (ICACT)*. IEEE, South Korea, 585–589. https://doi.org/10.23919/ICACT48636.2020.9061261
[26] Tao Lin, Lingjing Kong, Sebastian U Stich, and Martin Jaggi. 2020. Ensemble distillation for robust model fusion in federated learning. *Advances in Neural Information Processing Systems* 33 (2020), 2351–2363.
[27] Tao Lin, Lingjing Kong, Sebastian U. Stich, and Martin Jaggi. 2020. Ensemble distillation for robust model fusion in federated learning. In *Proceedings of the 34th International Conference on Neural Information Processing Systems* (Vancouver, BC, Canada) *(NIPS'20)*. Curran Associates Inc., Red Hook, NY, USA, Article 198, 13 pages.
[28] Arvind Mahindru and Himani Arora. 2022. Dnndroid: Android malware detection framework based on federated learning and edge computing. In *International Conference on Advancements in Smart Computing and Information Security*. Springer, India, 96–107.
[29] Brad Miller, Alex Kantchelian, Michael Carl Tschantz, Sadia Afroz, Rekha Bachwani, Riyaz Faizullabhoy, Ling Huang, Vaishaal Shankar, Tony Wu, George Yiu, Anthony D. Joseph, and J. D. Tygar. 2016. Reviewer Integration and Performance Measurement for Malware Detection. In *Proceedings of the 13th International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment - Volume 9721* (San Sebastián, Spain) *(DIMVA 2016)*. Springer-Verlag, Berlin, Heidelberg, 122–141. https://doi.org/10.1007/978-3-319-40667-1_7
[30] MOA. 2017. AdaptiveRandomForest. https://moa.cms.waikato.ac.nz/adaptive-random-forest/.
[31] Borja Molina-Coronado, Usue Mori, Alexander Mendiburu, and Jose Miguel-Alonso. 2023. Towards a fair comparison and realistic evaluation framework of android malware detectors based on static analysis and machine learning. *Computers & Security* 124 (2023), 102996. https://doi.org/10.1016/j.cose.2022.102996
[32] Feargus Pendlebury, Fabio Pierazzi, Roberto Jordaney, Johannes Kinder, and Lorenzo Cavallaro. 2019. TESSERACT: Eliminating Experimental Bias in Malware Classification across Space and Time. In *28th USENIX Security Symposium (USENIX Security 19)*. USENIX Association, Santa Clara, CA, 729–746. https://www.usenix.org/conference/usenixsecurity19/presentation/pendlebury
[33] Hemant Rathore, Adithya Samavedhi, Sanjay K Sahay, and Mohit Sewak. 2021. Robust malware detection models: learning from adversarial attacks and defenses. *Forensic Science International: Digital Investigation* 37 (2021), 301183.
[34] scikit learn. 2020. Machine Learning in Python. https://scikit-learn.org/stable/.
[35] scikit learn. 2020. SelectKBest. https://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.SelectKBest.html.
[36] Tianyi Shen, Cyan Subhra Mishra, Jack Sampson, Mahmut Taylan Kandemir, and Vijaykrishnan Narayanan. 2022. An Efficient Edge-Cloud Partitioning of Random Forests for Distributed Sensor Networks. *IEEE Embedded Systems Letters* 1, 1 (2022), 1–1. https://doi.org/10.1109/LES.2022.3207968
[37] Vale Tolpegin, Stacey Truex, Mehmet Emre Gursoy, and Ling Liu. 2020. Data poisoning attacks against federated learning systems. In *Computer Security–ESORICS 2020: 25th European Symposium on Research in Computer Security, ESORICS 2020, Guildford, UK, September 14–18, 2020, Proceedings, Part I 25*. Springer, UK, 480–501.
[38] VirusTotal. 2016. The pattern matching Swiss knife. https://github.com/VirusTotal/yara.
[39] Virustotal. 2023. Virustotal. https://www.virustotal.com/gui/home/upload.
[40] Carole-Jean Wu, David Brooks, Kevin Chen, Douglas Chen, Sy Choudhury, Marat Dukhan, Kim Hazelwood, Eldad Isaac, Yangqing Jia, Bill Jia, Tommer Leyvand, Hao Lu, Yang Lu, Lin Qiao, Brandon Reagen, Joe Spisak, Fei Sun, Andrew Tulloch, Peter Vajda, Xiaodong Wang, Yanghan Wang, Bram Wasti, Yiming Wu, Ran Xian, Sungjoo Yoo, and Peizhao Zhang. 2019. Machine Learning at Facebook: Understanding Inference at the Edge. In *2019 IEEE International Symposium on High Performance Computer Architecture (HPCA)*. IEEE, USA, 331–344. https://doi.org/10.1109/HPCA.2019.00048

A DERIVING YARA RULES FROM ML MODELS

Most works in the literature tend to complete their evaluations at the ML model level. However, in practice, AV companies do not use the models directly in the end users' machines, but to derive rules that will be deployed in the endpoints. There are important differences between ML models and the rules and there are phenomenons that can only be observed at the rule level. Therefore, we took a step further and generated the rules from the model for evaluation purposes. We here characterize the derived rules and point out important facts about their nature to bridge the gap in the literature about the application of rules derived from ML models.

YARA rules. Rules can be derived from a tree-based model (e.g., RF or DT) by traversing all paths of the tree and aggregating the node conditions. If all conditions are satisfied (logic AND), the rule matches. The paths can be represented via multiple frameworks. In this work, we chose YARA, for two reasons. First, because it is the *de-facto* standard, used by many security products, which gives us realistic results. Second, it natively supports the PE format. Since our features are PE entities, they can be directly mapped to YARA rules as the paths are traversed. When using it, each traversed path results in a new rule.

```
1  import "pe"
2
3  rule rule_from_ml_0 {
4   condition:
5    pe.imports(/(.).dll/i, /closehandle/i)
6    and
7    pe.characteristics & pe.EXECUTABLE_IMAGE
8    and
9    pe.exports(/dllunregisterserver/i)
10 }
```

Code 1: Yara rule generated from the ML model.

Code 1 exemplifies a YARA rule generated from traversing one ML model path. The rule relies on the support for PE files (line 1) to match all conditions (lines 4 to 6) via logic ANDs. The rule matches different types of features. It checks (1) if the scanned file imports a given API function (line 4); (2) if the binary image has specific characteristics (line 5), and (iii) if the binary exports a given symbol. All checks are case-insensitive (/i),

The number of rules. Since each traversed path results in a new rule, it is plausible to hypothesize that when using more features in the model, more rules are generated, since more features are available to be matched. To test this hypothesis, we generated rules from the models using the diverse number of features identified in the previous experiments.

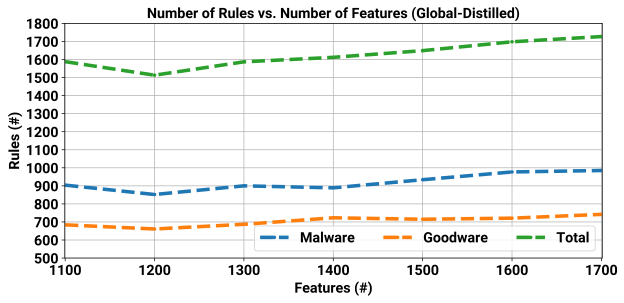

Figure 29: Number of rules vs. feature size. The number of generated rules moderately increases with the number of features.

Figure 29 shows the number of totals, malware, and goodware rules generated from the models of different feature set sizes. Our first observation is that the number of features in the model indeed has an impact on the number of generated rules (100 new rules were generated). However, this impact is moderate (100 new rules represent 6% of all rules).

Our second observation is that the total number of rules is not an appropriate proxy for the model complexity because, in the case of a binary problem, the model generates tree branches, thus rules, covering both classes. In practice, however, the AV does not verify the benign samples but only checks for the malicious ones, such that rules should be generated only for the malware class.

Our final observation is that there is not an equal number of malware and goodware rules, regardless of the feature set sizes. The model generates more paths for the malware ones, which indicates that these are more complex to classify.

The complexity of the rules. Another plausible hypothesis is that the number of features in a model affects the complexity of the model, i.e., there are more features in each path, which requires more checks to be performed, thus increasing the matching time. We repeated the experiments to evaluate how many comparisons there are in the paths of the rules generated from each model.

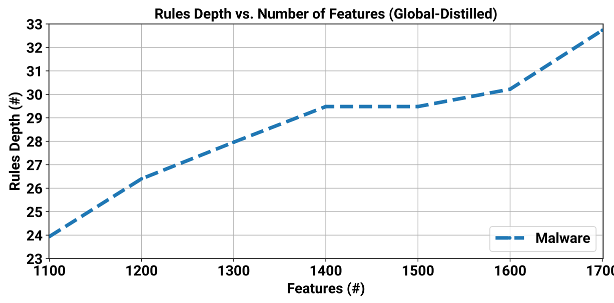

Figure 30: Rules depth vs. feature size. The average depth of the rules increases with the number of features.

Figure 30 shows the average number of comparisons in the set of malware rules derived from each model. It is noticeable that increasing the number of features in the model significantly increases the complexity of the rules (from 24 to 33), which on average makes the matching slower. Therefore, the performance overhead observed in the conducted experiments is explained in a minor part by the addition of more rules and to a larger extent by the rules becoming more complex with an increased number of features.

The average coverage of the rules. A desirable characteristic of ML models in comparison to byte-based signatures is generalization. The same model can detect multiple samples, reducing the rule storage requirements, and the number of rules to be matched, thus increasing the performance of the security application. Ideally, this same property should be achieved for the rules derived from the model. Although the YARA framework can be used for byte-based pattern matching, it can be used also to match broader rules that generalize more. We repeated the experiments to evaluate how many samples are covered (i.e., detected) by each derived rule.

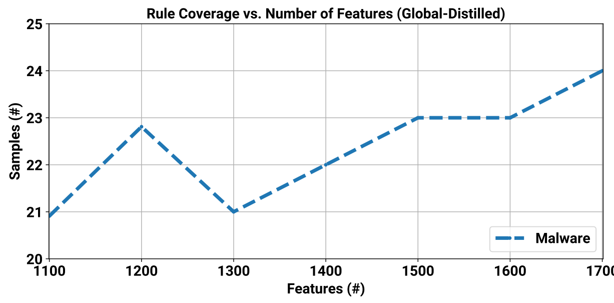

Figure 31: Rules coverage vs. feature size. The average number of samples covered by each rule moderately increases with the number of features.

Figure 31 shows the average number of samples identified via each rule. The rules indeed generalize to multiple samples. Identifying more than 20 samples with a single rule significantly contributes to reducing the storage requirement by this same magnitude. The number of samples covered increased by a little with an increase in the number of features (from 21 to 24), which is not enough to cause a significant impact on thousands of samples. This finding reinforces the previous point about the performance being mainly impacted by the rules' complexity rather than number.

The maximum coverage of the rules. In addition to the average case previously discussed, the matching rules also present interesting corner cases. One of them is when the same rule matches a significant number of files. This is desired by the AVs as a way to detect malware variants. We repeated the experiments to identify if the number of features affect the maximum coverage.

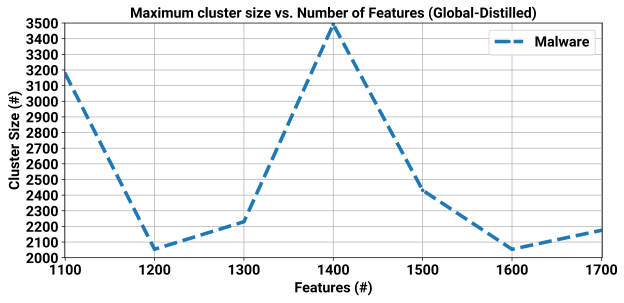

Figure 32: Rules maximum coverage vs. feature size. The maximum coverage significantly varies over time, with no direct relation to the feature set size.

Figure 32 presents how many samples are covered by the rule of maximum coverage for the different feature set sizes. We notice that the rules of maximum coverage are very efficient, allowing the match of thousands of samples with the same rule, thus leading to significant storage and performance gains. We notice also a significant variation with feature size increase (from 2000 to 3400 samples). However, the variation is not coherent, i.e., it does not sustain over time. It happens because the new features do not necessarily make rules to generalize more, but they cause the trees to split, such that in the cases in which the maximum coverage decreased, it happened because the maximal rule was split into two similar ones with the difference of a few features.

The minimum coverage of the rules. Another relevant corner case is when the rules fail to generalize and are able to detect a single sample (singletons). This happens, for instance, when a sample is hard to classify and thus it requires additional features that are not required by any other one to be matched. In this case, although the rule was produced by a ML model, the scenario is somehow similar to the individual signatures previously generated by the AVs.

Figure 33 shows the number of singleton rules for the different feature set sizes. Once again, there is no coherent trend, with a significant variation. The number of singletons is significantly smaller than the number of the samples covered by the maximal rule (300 vs. 3000). This highlights the fact that the majority of the samples are covered by the average rules.

B GENERALIZATION EVALUATION

This work aims to shed light on the impact of heterogeneous datasets on malware detection. Our key claim is that datasets from

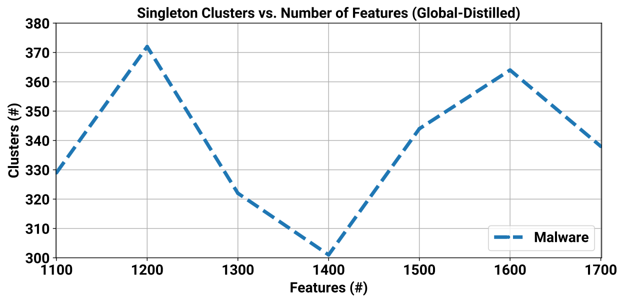

Figure 33: Number of Singletons vs. feature size. The number of singleton clusters significantly varies over time, with no direct relation to the feature set size.

different regions have inherently different patterns requiring specialized handling. To demonstrate that, it is key to show that the effect happens independently of the type of detector. We here complement experiments presented in the main text with experiments showing the impact on different detector settings.

B.1 Varying Feature Selectors

The number of features required to detect 99% of the samples in each dataset is the most prominent example of a different requirement for each dataset. Whereas the main experiments have demonstrated the results for the feature selection process using the ANOVA's `F-Score` method, the most powerful one, we here complement the results to demonstrate that other feature selection metrics lead to the same result. To that, we repeated previous experiments with the `SelectKBest` [35] method from `scikit-learn`, but now using different feature selection methods. In addition to the `F-Score`, we considered the `Mutual Information` and `Chi2` methods, as they are the most popular feature selectors for classification tasks.

Table 3: Feature Selection Method. Ideal feature set size for the multiple regional malware datasets.

	US	BR	JP
F-Score	290	340	800
Chi2	292	342	803
Mutual Info	294	345	812

Table 3 shows the number of features required for the detection of each dataset to converge to 99% when using different feature selectors. Whereas the `F-Score` metric, whose results were presented in the main text, is slightly superior to the other metrics, in the sense of requiring fewer features, the performance of all feature selectors (i.e., number of features) is similar. Regardless of the considered feature selector, the number of required features by each dataset is different, which shows that this is due to inherent dataset characteristics rather than due to the feature selection method.

B.2 Varying Classifiers

As for the feature selectors, we also investigated the impact of different classifiers. Whereas we only considered in the main text RF results, given its prevalence and frequently superior performance according to previous works, we here extend the experiments to other popular classifiers implemented in the `scikit-learn` framework (RF, SGD, AdaBoost, and SVM). Once again, we hypothesize that there is an inherent phenomenon originating from dataset characteristics that is independent of classifier architecture.

Table 4: Classifier Influence on the detection of different regional malware datasets. Feature set sizes.

	95%			99%		
	US	**BR**	**JP**	**US**	**BR**	**JP**
RF	35	40	45	290	340	800
SGD	35	40	45	292	342	805
AdaBoost	35	40	45	292	342	805
SVM	36	41	46	295	345	813

Table 4 shows the number of features required for each dataset detection converge to 95% and 99% respectively when using different classifiers. Whereas the RF classifier's performance is slightly superior to its counterparts, as hypothesized, the results for all classifiers are overall similar, thus reinforcing the hypothesis that the datasets have unique characteristics that should be handled. The Pareto characteristic of the classification problem is present in all classifiers, with much fewer features being required to achieve 95% than the 99% detection rate.

B.3 Varying Distillation Techniques

As for the feature selection and classifiers, we also evaluated the impact of different distillation strategies on the datasets. We complemented the experiments with the Teacher-Student (TS) strategy presented in the main text with experiments leveraging the closest distillation proposal to ours: Federated Model Fusion (FMF) [27]. This approach proposes combining the outputs of multiple models in a global knowledge database, as we propose to combine region-specific models into a global one. Whereas originally evaluated with images, we here extend it to work with malware. To that, we adapted the proposed algorithm to work in the same conditions as ours. For instance, instead of operating with N clients, we limited it to operating with a single one, as each regional model is stored only in one node (the regional server), not on multiple endpoint nodes. Similarly, we limited the number of rounds to one, as for experimental purposes, we can add all the data at once, without the need of querying individual nodes. With that, the proposed algorithm is reduced to a version of the TS strategy.

Table 5: Distillation Technique Influence on the detection of different regional malware datasets. Feature set sizes.

	US	**BR**	**JP**
TS [20, 23]	300 (+3%)	400 (+17%)	900 (+12.5%)
FMF [27]	299 (+3%)	402 (+18%)	902 (+12.5%)

Table 5 shows the number of features required for each dataset detection converge to the 99% detection rate when using different distillation techniques and its increase in comparison to the base models. As hypothesized, the performance of FMF is very close to the TS, as their operation became very similar under these conditions. Therefore, we reinforce the claim that the differences observed in the required number of features are due to the dataset's inherent characteristics and that we need to add this work; 's presented flexibility to handle it.