

This work was written as part of one of the author's official duties as an Employee of the United States Government and is therefore a work of the United States Government. In accordance with 17 U.S.C. 105, no copyright protection is available for such works under U.S. Law.

Public Domain Mark 1.0

<https://creativecommons.org/publicdomain/mark/1.0/>

Access to this work was provided by the University of Maryland, Baltimore County (UMBC) ScholarWorks@UMBC digital repository on the Maryland Shared Open Access (MD-SOAR) platform.

Please provide feedback

Please support the ScholarWorks@UMBC repository by emailing scholarworks-group@umbc.edu and telling us what having access to this work means to you and why it's important to you. Thank you.

Catch'em all: Classification of Rare, Prominent, and Novel Malware Families

Maksim E. Eren*, Ryan Barron†, Manish Bhattarai†, Selma Wanna*,
Nicholas Solovyev†, Kim Rasmussen†, Boian S. Alexandrov†, and Charles Nicholas‡

*Advanced Research in Cyber Systems, Los Alamos National Laboratory. Los Alamos, USA.

†Theoretical Division, Los Alamos National Laboratory. Los Alamos, USA.

‡CSEE, University of Maryland, Baltimore County. Maryland, USA.

Abstract—National security is threatened by malware, which remains one of the most dangerous and costly cyber threats. As of last year, researchers reported 1.3 billion known malware specimens, motivating the use of data-driven machine learning (ML) methods for analysis. However, shortcomings in existing ML approaches hinder their mass adoption. These challenges include detection of novel malware and the ability to perform malware classification in the face of class imbalance: a situation where malware families are not equally represented in the data. Our work addresses these shortcomings with **MalwareDNA**: an advanced dimensionality reduction and feature extraction framework. We demonstrate stable task performance under class imbalance for the following tasks: malware family classification and novel malware detection with a trade-off in increased abstention or reject-option rate.

Index Terms—non-negative matrix factorization, novel malware, semi-supervised learning, reject-option, class-imbalance

I. INTRODUCTION

Approximately half a million new malware specimens are reported every day, totaling 1.3 billion specimens as of 2024 [1]. This immense quantity of malware requires the utilization of Machine Learning (ML) based automated security systems for detection and family classification. The goal of malware family classification is to assign malware family labels to known-malware examples to better understand specimen behavior [2]. Malware authors actively generate new specimens to evade detection and to introduce novel threats, resulting in variations within existing malware families or evolution of new/novel families. Our work focuses on classifying existing and novel malware families, an important task for risk analysis to assess threat severity and develop mitigation strategies for emerging threats. While ML-based solutions may reduce time and costs for malware detection and recovery, the adoption of such strategies has been slow. We attribute this to real-world complexities pertaining to malware analysis [2], [3], and seek to address these shortcomings in this work.

Recent works often overlook relevant evaluation criteria for real-world applications of malware family classification. These core criteria include assessing the model's ability to identify new or novel malware and to classify malware families in the face of class imbalance [2], [4]–[6]. Determining that a given specimen is not a member of a known malware family with

certainty is an important malware analysis task. At the same time, popular supervised models trained on known malware families may fail to generalize to new data, resulting in false negatives on novel specimens which may lead to security incidences or missed threats [2], [6]. Similarly, the ML-based models should be able to operate under conditions of class imbalance. In malware analysis, class imbalance refers to a large disparity in data class counts—instances for specific malware families significantly outnumber (prominent malware) instances of other low-count classes (rare malware) in the dataset. The models trained with prominent malware families may fail to generalize and detect rare specimens. However, it is still important to detect the rare specimens as they can also cause security breaches. Finally, while semi-supervised learning can help address these shortcomings, they have not been widely studied for Windows malware field [2]. With the growing quantities of malware in the wild, there is an urgent need to develop methods that address these shortcomings, and motivate increased adaption of ML solutions.

In this paper, we showcase our semi-supervised method: *MalwareDNA* [5], for classification of both rare and prominent malware families (class imbalance) as well as identification of novel malware families. Our method uses hierarchical non-negative matrix factorization (NMF) with automatic model determination [7]. This automatic estimation of the number of latent (hidden) signatures helps avoid under/over-fitting, which enables data modeling with high specificity and accuracy, which in turn lets us build an archive of latent signatures (identifiers) of malware families. These signatures are then used for precise real-time downstream classification of malware families. Our method also includes a fast optimization method to perform abstaining classification (*reject-option*) using distinct confidence metrics [8]. This reject-option lets us see the confidence of the model, and gives the model the ability to say "I do not know", and gives the user the ability to select between performance and coverage (non-abstaining classification) [9]. The reject-option capability enables *MalwareDNA* to identify novel malware families, and maintain its performance under class imbalance by reducing its coverage rate. Our contributions include:

- Comparing and contrasting our method's malware family

classification capability with different inference confidence metrics including Projection Similarity, Ensemble Voting, and Data Augmentation.

- Demonstrating our method’s capability to classify both rare and prominent malware families, and identify novel malware families all at the same time, using the Windows Portable Executable (PE) format malware specimens from EMBER-2018 dataset [10], outperforming our supervised and semi-supervised baseline models.

II. PRIOR PUBLICATION NOTICE

We have previously introduced the MalwareDNA method in [5], where we showcased the novel malware detection capabilities of the method. In this paper, we provide an in-depth analysis of the methodology and results, extend our experiments to a larger dataset, addresses class imbalance, test our approach against this issue, and introduce new techniques for confidence measurement in inference. Specifically, under a new set of experiments, this paper showcases the method’s capability for handling the class-imbalance problem where we classify both prominent and rare malware families. While the original paper used the *Projection Similarity* as a confidence metric, we introduce two new confidence metrics to perform classification on malware. Finally, while the first paper looked at 1,000 malware specimens, our experiments in this paper are scaled up to 10,000 specimens.

III. RELEVANT WORK

Several prior works used tree- and deep learning-based ML methods for malware classification. Raff et al. introduced a deep learning architecture named MalConv, aiming to classify malware directly based on the entire raw byte-sequences of the binary [11]. Kumar et al. demonstrated that XGBoost is an effective model for classifying Windows PE malware using the EMBER-2018 dataset [10], achieved through low-resource feature selection [12]. Pham et al. illustrated that statistical summaries of the original PE features can enhance detection results. They employed LightGBM, which surpassed the previously introduced deep learning solution MalConv while requiring fewer resources [13]. In our experiments, we also test our model against EMBER-2018, benchmarking it against XGBoost and LightGBM, which are considered state-of-the-art baseline models on the EMBER-2018 dataset. While these methods are supervised solutions, we explore semi-supervised learning for its superior generalization capability.

As part of the semi-supervised scheme, our method leverages clustering and similarity scores for the categorization of novel samples. A number of previous works have also used clustering approaches, where the ensemble of clustering algorithms with distinct characteristics has been shown to yield accurate results for malware classification [14], [15]. Likewise, similarity metrics to extract embeddings (distance-based feature vectors) have also proven successful [16]. In addition, a similarity-based approach was employed by Raff et al., where they introduced the Burrows Wheeler Markov Distance (BWMD), an efficient similarity metric. This metric

is based on embedding data into a fixed-size vector space, demonstrating its effectiveness in clustering malware [17]. Finally, the malware similarity for clustering IoT malware in an unsupervised manner was presented in [18]. However, these methods exclusively concentrate on malware/benignware detection or malware family classification and do not possess the ability to identify novel families.

Another area that has garnered increasing interest is the application of ensemble learning to augment the predictive capabilities of malware classifiers. Atluri et al. demonstrated that various tree-based ensemble models, such as Random Forest, Bagging Decision Tree Classifier, and Gradient Boosting Classifier, among others, can be used together in a single framework, named Voting Ensemble Classifier (VEC), to achieve enhanced detection of Windows PE malware [19]. Ramadhan et al. explored a comparable method by creating a voting-based ensemble model employing LightGBM, XGBoost, and Logistic Regression [20]. Their study showed that an ensemble of classifiers, each with its distinct inductive biases, could result in increased accuracy compared to any individual model alone, as each member of the ensemble complements the weaknesses of the others. In addition, the framework of ensemble learning has been applied in the realm of deep learning for malware detection by Dahl et al. [21]. The authors demonstrated that an ensemble of neural networks employing voting, alongside a novel feature selection method based on dimensionality reduction and random projections, significantly improves malware identification. Inspired by the success of ensemble learning, we incorporate an ensemble based inference confidence metric to our model.

It has already been shown that the class-imbalance problem degrades the performance of popular methods developed for large-scale malware analysis [22]. Rajvardhan et al. used BERT to classify imbalanced malware data with high accuracy [23]. While this work only focused on malware/benignware classification, our aim is to classify malware families. A handful of prior works discuss the class-imbalance problem in large-scale malware analysis where they attempted to detect rare specimens by grouping multiple rare families into a single “others” class [24]–[26]. The most realistic malware family classification work was done by Huang et al. which targeted 100 classes where one of the classes included “others” [24]. While this approach introduced an ability to detect rare specimens by the “others” class, it yields poor generalization to new or never before seen specimens as was also pointed out by Loi et al [25]. They report that their false positives are heavily represented by the families collected within the “others” class due to the supervised method’s inability to learn the patterns of these families from a small number of specimens. Conversely, our method does not require training with rare specimens, since it possesses the abstaining prediction ability i.e. the *reject-option*. This allows our method to combine the abilities of malware family classification under class imbalance and novel malware family identification where we make an increased number of abstaining predictions (lower coverage-rate) to maintain the performance or accurate decisions.

IV. METHOD

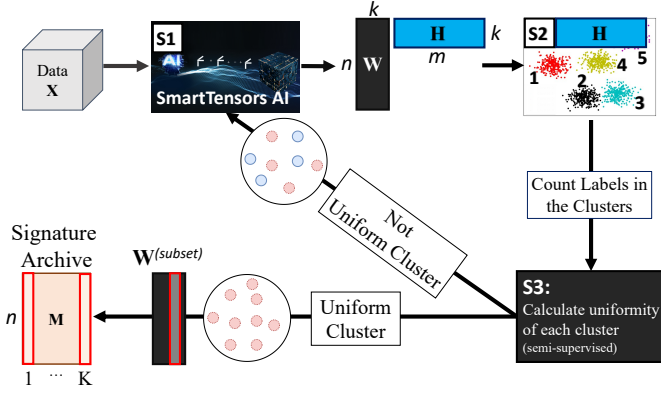


Fig. 1. Overview of how the archive of latent signatures is built from multi-dimensional data in a hierarchical manner. The patterns from the data are first extracted (S1). These patterns have the corresponding clusters among the samples (e.g. malware specimens, S2). If we identify a cluster where each sample belongs to the same class (uniform), we place the patterns (or latent signatures) corresponding to this cluster into the archive (S3). Otherwise, we separate the mixed signatures of samples belonging to a non-uniform cluster by successive factorization (going back to S1).

In this section, we first introduce the latent signature archive's construction through MalwareDNA, then introduce how the signature archive is used for classification. We further introduce different confidence metrics used for the classification and definition of the reject-option.

A. Building Signature Archive

An overview of how the signature archive is built is shown in Figure 1. MalwareDNA first applies NMF to the observational data \mathbf{X} (Figure 1, S1). NMF is an unsupervised learning method based on a low-rank matrix decomposition [27]. NMF approximately represents an observed non-negative matrix, $\mathbf{X} \in \mathbb{R}_+^{n \times m}$, as a product of two (unknown) non-negative matrices, $\mathbf{W} \in \mathbb{R}_+^{n \times k}$ whose k columns are the latent signatures each with n features, and $\mathbf{H} \in \mathbb{R}_+^{k \times m}$ whose rows are the activities of each one of the k signatures (latent features) in each m samples, where usually $k \ll m, n$. This approximation is performed via non-convex minimization with a given distance, $\|\dots\|_{dist}$, constrained by the non-negativity of \mathbf{W} and \mathbf{H} : $\min \|\mathbf{X}_{ij} - \sum_{s=1}^k \mathbf{W}_{is} \mathbf{H}_{sj}\|_{dist}$.

The NMF minimization requires prior knowledge of the latent dimensionality k for accurate data modeling, which is usually unavailable [28]. Excessively small values of k lead to poor approximation of observables in \mathbf{X} (*under-fitting*), while excessively large values of k fit the data's noise (*over-fitting*). In this work, we use *NMFk* that incorporates automatic model selection for estimating k [7], [29]. *NMFk* integrates NMF-minimization with custom clustering and Silhouette statistics, and combines the accuracy of the minimization and robustness/stability of the NMF solutions. A bootstrap procedure (i.e., generation of a random ensemble of perturbed matrices) is applied to estimate the number of latent features k . Recently, *NMFk* was applied to a large number of

big synthetic datasets with a predetermined number of latent features, and it has demonstrated its superior performance of correctly determining k in comparison to other heuristics clustering techniques [30], [31]. MalwareDNA uses a publicly available implementation of *NMFk* [32]¹.

After extracting the accurate factors \mathbf{W} and \mathbf{H} , we apply a custom H -clustering method, the *Argmax* operator, to assign each sample (represented by the columns of \mathbf{X}) to one of the k signature clusters (S2) i.e. the label assignment for $X_{:,j}$ is given as $y_j = \text{argmax}(\mathbf{H}_{:,j})$ if $\max(\mathbf{H}_{:,j}) > \tau$ for some confidence probability/threshold τ . In each of these clusters, some of the samples may have different labels (non-uniformity) based on this confidence probability if $\max(\mathbf{H}_{:,j}) \leq \tau$. We evaluate the uniformity of the samples in each cluster, determining whether all labels are the same (S3). When a uniform cluster is identified, we separate the samples of this cluster from the data, \mathbf{X} , and add the annotated (by the labels) cluster centroid, corresponding column of \mathbf{W} , to our archive of signatures $\mathbf{M} \in \mathbb{R}_+^{n \times K}$, where n is the number of features and K is the number of unique latent signatures. Otherwise, we continue with successive factorization in a hierarchical manner to separate the mixed latent signatures as shown in Figure 1.

B. Inference Using the Signature Archive

We use the latent signature archive \mathbf{M} , after it is built, for inference - or classification - tasks. During testing for real-time inference, we project each new sample \mathbf{x} onto the signature archive using Non-negative Least Squares Solver (NNLS) [33] where the optimization problem is given as $\arg \min_{\mathbf{h} \geq 0} \|\mathbf{x} - \mathbf{M}\mathbf{h}\|_2^2$ to extract coefficient vector $\tilde{\mathbf{h}}$. This allows us to perform real-time identification by representing each new sample as a combination of signatures recorded in the archive $\mathbf{x} \approx \sum_{i=1}^K \tilde{\mathbf{h}}_i * \mathbf{M}_i \Rightarrow \tilde{\mathbf{x}} = \mathbf{M}\tilde{\mathbf{h}}$ and estimating the accuracy, or similarity score, of this representation. We utilize the cosine similarity score of the NNLS projection of the new sample to the signatures $\mathbf{m} \in \mathbf{M}$ given as $S(\mathbf{m}, \tilde{\mathbf{x}}) = \frac{\mathbf{m} \cdot \tilde{\mathbf{x}}}{\|\mathbf{m}\|_2 \|\tilde{\mathbf{x}}\|_2}$. Techapanurak et al. observed that cosine similarity is effective in identifying out-of-distribution samples [34], and Zhang et al. demonstrates cosine similarity as an effective metric to define the confidence of methods with reject-option capability [9]. We further define three different confidence metrics – Projection Similarity, Ensemble Voting, and Data Augmentation – using the cosine similarity scores from the NNLS projections.

1) *Projection Similarity*: We utilize the similarity scores, together with a threshold, τ , to define the malware family and novel malware family classification. Once we extract $\tilde{\mathbf{h}}$ based on NNLS approach discussed above, the prediction is then defined, using cosine similarity score S where $y_j = \arg \max_{0 \leq j \leq K} S(\mathbf{M}_{:,j}, \tilde{\mathbf{h}})$. where the given prediction j is labeled $\mathbf{y}_j \in \{1, 2, \dots, C\}$ for C classes. i.e. the most similar signature is selected based on distance measurement. When a signature possesses a similarity score above τ , the labels of

¹NMFk is available in <https://github.com/lanl/T-ELF>.

the signature will be determined as the classification result. Likewise, when the similarity score is below τ , the reject-option or abstaining classification will be selected.

2) *Ensemble Voting*: Ensemble learning can further enhance the accuracy of our confidence calculation [9]. If we define a second threshold $\tilde{\tau}$ against the cosine similarity between $\tilde{\mathbf{h}}$ and each K signatures in \mathbf{M} , we can obtain votes for each class $\mathbf{y}_i^M \in \{1, 2, \dots, C\}$. Given a sample \mathbf{x} , for each class C we can obtain V_C number of votes if the cosine similarity score between $\tilde{\mathbf{h}}$ and columns of \mathbf{M} belonging to class C are above the given threshold $\tilde{\tau}$ ($\tilde{\tau} = 0.5$ in our experiments). We normalized the votes based on the number of signatures present for a given class in \mathbf{y}^M , such that $\hat{V}_C = V_C/|\mathbf{I}^C|$, where $|\mathbf{I}^C|$ is essentially is the number of latent signatures belonging to class C .

3) *Data Augmentation*: We also test our method with data augmentation to define the confidence, where the idea is confidence stability under perturbation for post-processing during testing time [9], [35], which is done with instance-level perturbations (*test-time data augmentation*) [36]. Here we add some error ϵ to \mathbf{x} to generate p different perturbations ($\tilde{\mathbf{x}}_1, \tilde{\mathbf{x}}_2, \dots, \tilde{\mathbf{x}}_p$) where $\tilde{\mathbf{x}}_i = \mathbf{x} + \epsilon|_{i=1}^p$, that is centered around \mathbf{x} with distance $\|\epsilon\| = 0.015$ in our experiments), and average the corresponding cosine similarities of the predictions $S_{i=1}^p$ to define the confidence. The idea is that a truly confident outcome should remain stable under noise/perturbations, and that instance-level perturbations may yield more robust confidence measurements. We apply this bootstrap approach 50 times in our experiments.

V. EXPERIMENTS

In this section we first introduce the dataset used in our experiments, and the experimental setup. Then, we showcase the capabilities of our method with different confidence metrics, and then compare to our baselines.

A. Dataset and Experiment Setup

TABLE I
DISTRIBUTION OF MALWARE FAMILIES IN TRAINING AND TESTING SETS REPORTED WITH MEAN NUMBER OF INSTANCES AND THE CONFIDENCE INTERVAL OVER 10 SAMPLE TRIALS.

Malware Family	Training Set	Testing Set
xtrat	4853.9 (+ 12.6)	543.1 (+ 12.2)
installmonster	3750.3 (+ 10.2)	416.7 (+ 11.5)
adposhel	3216.4 (+ 6.6)	361.6 (+ 5.6)
zusy (rare family)	638.0 (+ 7.0)	67.0 (+ 6.9)
emoted (rare family)	232.2 (+ 3.8)	25.8 (+ 3.8)
fareit (rare family)	97.2 (+ 1.9)	11.8 (+ 1.4)
ramnit (novel family)	0.0	1029.0 (+ 2.4)

We ten times randomly sample 10k malware specimens from seven top populous families (ramnit, adposhel, emotet, fareit, installmonster, xtrat, and zusy) using a popular benchmark dataset, EMBER-2018 [10]. We select ramnit to represent a novel/unseen malware family. Further, we select zusy, emotet, and fareit to represent the rare malware families

and randomly under-sample these classes ten times with an increasing under-sampling rate (i.e. fareit is the rarest class). We summarize the distribution of the malware families in the training and testing sets in Table I. We use the static analysis features byte histogram and entropy, print table distribution, strings entropy, number of strings/exports/imports/sections, file size, and code size. In our normalization, Z-scores are used to remap the outliers that are more than or less than 3 standard deviations away from the mean to the point that is exactly 3 standard deviations away from the mean. We report our results with a 95% confidence interval (CI) for the ten runs.

We baseline our method against the popular supervised malware classifiers XGBoost [37] and LightGBM [38]. We further extend these baselines with the SelfTrain [39] algorithm to create semi-supervised models. We note that the previous work has used these models to report benchmarking against this dataset [10], [40]; however, we expose these models to a more challenging task of classifying malware families under extreme class imbalance and detecting novel malware families all at the same time. Our baselines are tuned using Optuna [41] over 200 trials with 5-fold stratified shuffle cross-validation.

B. Compare Confidence Measurement Techniques

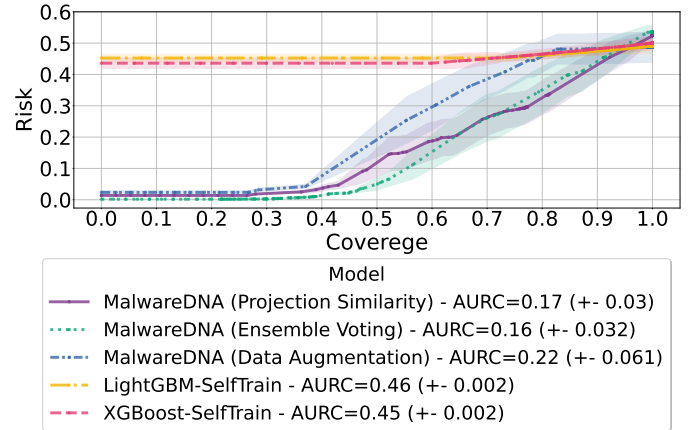


Fig. 2. Risk-Coverage (RC) curve when classifying malware families and novel malware together with the area under the RC (AUC) for different MalwareDNA confidence metrics and our semi-supervised baselines.

The performance of our method is first reported with the Area Under the Curve of Risk-Coverage (AUC) [8] in Figure 2 where we compare the different confidence metrics. AUC models the trade-off between the coverage (the number of samples for which the non-rejecting predictions were made) and the risk which is measured as $1 - F1$ score. AUC score is reported between 0 and 1, and lower AUC is preferred over higher AUC. MalwareDNA with Ensemble Voting achieves the best AUC score of 0.16 when classifying the malware families. It can also be seen that Ensemble Voting maintains lower-risk (higher F1 score) with the increasing coverage rate (reducing fraction of abstaining predictions) until around the 0.65 coverage rate. In addition, MalwareDNA for each confidence metrics yields lower risk score for the majority of the coverage rate than our semi-supervised baselines.

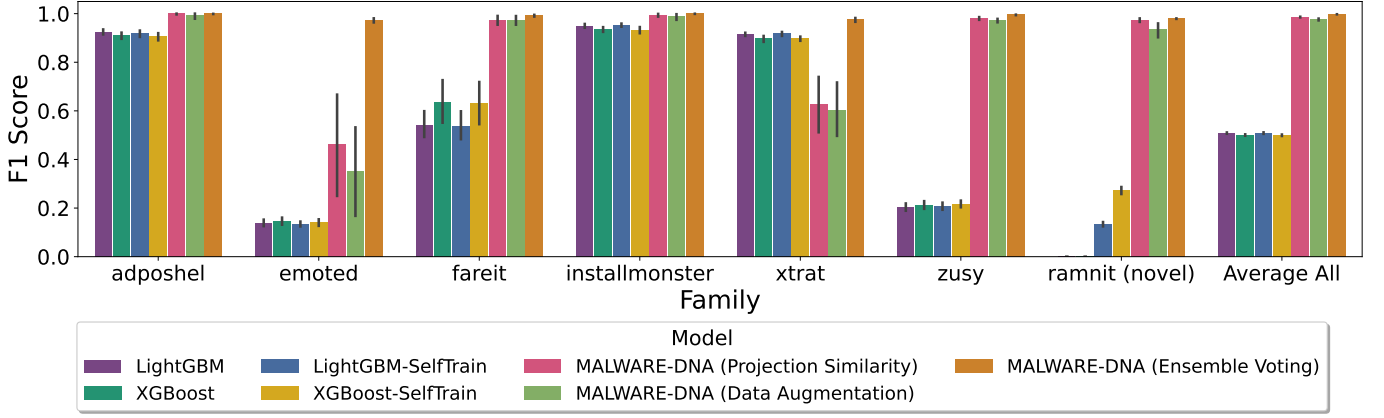


Fig. 3. Mean F1 scores with CI is reported for each malware family when comparing MalwareDNA with different confidence metrics to both our supervised and semi-supervised baselines. It can be seen that, while our baseline’s performance degrade for the rare malware families (emoted, fareit, and zusy), our method maintains its performance.

C. Classification of Malware, Rare and Prominent Malware Families, and Novel Malware

TABLE II
PERFORMANCE OF MALWAREDNA WHEN CLASSIFYING MALWARE FAMILIES COMPARED TO BASELINES.

Model	F1	Precision	Recall
MalwareDNA (Projection Similarity)	.966 (+-.008)	.973 (+-.007)	.960 (+-.005)
MalwareDNA (Ensemble Voting)	.995 (+-.002)	.993 (+-.002)	.996 (+-.002)
MalwareDNA (Data Augmentation)	.966 (+-.012)	.971 (+-.009)	.967 (+-.008)
XGBoost	.500 (+-.005)	.468 (+-.039)	.823 (+-.013)
LightGBM	.509 (+-.006)	.460 (+-.031)	.825 (+-.019)
XGBoost-SelfTrain	.499 (+-.007)	.466 (+-.034)	.819 (+-.015)
LightGBM-SelfTrain	.510 (+-.006)	.460 (+-.031)	.824 (+-.014)

TABLE III
NOVEL MALWARE DETECTION OF MALWAREDNA COMPARED TO BASELINES. REJECTION SEEN PROVIDES THE FALSE REJECTION PREDICTIONS FOR THE SAMPLES THAT BELONGS TO KNOWN CLASSES. REJECTION NOVEL IS THE TRUE REJECTION PREDICTIONS FOR THE SAMPLES THAT BELONG TO A NOVEL MALWARE FAMILY.

Model	Rejection Seen	Rejection Novel
MalwareDNA (Projection Similarity)	67.16% (+- 3.38)	85.84% (+- 0.76)
MalwareDNA (Ensemble Voting)	70.11% (+- 0.40)	95.34% (+- 0.09)
MalwareDNA (Data Augmentation)	69.23% (+- 3.32)	84.67% (+- 2.76)
XGBoost	NA	NA
LightGBM	NA	NA
XGBoost-SelfTrain	11.80% (+- 1.48)	27.50% (+- 3.44)
LightGBM-SelfTrain	5.75% (+- 0.88)	13.50% (+- 2.07)

At around the 30% coverage rate, MalwareDNA with Ensemble Voting achieves an F1 score of 0.995 when classifying the malware families (Table II), and 95.34% true-rejection predictions for the chosen unseen family ramnit, which illustrates our method’s ability to identify novel malware families (Table III). Note that the relatively high rejection-seen percentage for our method, for example 70.11% for Ensemble Voting in Table III, is the result of the trade-off for giving up coverage, and in return obtaining lower risk (i.e. higher performance or F1 score) and higher rate of detecting novel malware families. A user may choose the trade-off between the coverage and risk, see for example Figure 2 for determining the model performance for different coverage rates. In Table II it can also be seen that our baselines, including the semi-supervised ones, obtain much lower scores. The lower performance of

our baselines are mainly caused by the miss-classified rare-classes as well as the novel families. For example, as seen in Table III, our semi-supervised baselines with SelfTrain, while obtaining low rejection-seen percentage, they are also not able to reject much of the novel malware families and miss-classify them (XGBoost+SelfTrain only rejects 27.50% of the novel specimens). We further notice the performance degradation of our baselines for the class-imbalance problem in Figure 3.

Figure 3 shows that our baseline models yield lower performance on each of the rare malware families (emoted, fareit, and zusy), while our method maintains a higher F1 score. Note that MalwareDNA with Data Augmentation and Projection Similarity yields lower scores for the rare family emoted, while Ensemble Voting still manages to maintain its performance. Therefore, we believe that Ensemble Voting is an ideal confidence metric for handling the class imbalance problem. Our benchmarking against the baseline models and the poor performance of these models, points out both the difficulty of the task, and MalwareDNA’s unique ability to both accurately classify families under class imbalance, while simultaneously detecting novel malware families.

VI. CONCLUSION

In this paper, we showcased MalwareDNA’s capability to classify malware families under class imbalance, and detect novel malware families. Our preliminary results showcased the novel malware detection capability of our system, while also outperforming state-of-the-art methods in a more difficult problem of solving inference tasks under class imbalance and with the presence of novel malware.

ACKNOWLEDGMENT

This manuscript has been assigned LA-UR-24-21917. This research was funded by the LANL LDRD grant 20230067SR and the LANL Institutional Computing Program, supported by the U.S. Department of Energy National Nuclear Security Administration under Contract No. 89233218CNA000001.

REFERENCES

- [1] The Independent IT Security Institute, “Malware statistics & trends report: Av-test,” February 2024.
- [2] E. Raff and C. Nicholas, “A survey of machine learning methods and challenges for windows malware classification,” *ArXiv*, vol. abs/2006.09271, 2020.
- [3] IBM, “Cost of a data breach report,” IBM, Technical Report, 2021. [Online]. Available: <https://www.ibm.com/security/data-breach>
- [4] A. T. Nguyen, E. Raff, C. Nicholas, and J. Holt, “Leveraging uncertainty for improved static malware detection under extreme false positive constraints,” *arXiv preprint arXiv:2108.04081*, 2021.
- [5] M. E. Eren, M. Bhattarai, K. Rasmussen, B. S. Alexandrov, and C. Nicholas, “Malwaredna: Simultaneous classification of malware, malware families, and novel malware,” in *2023 IEEE International Conference on Intelligence and Security Informatics (ISI)*, 2023.
- [6] M. E. Eren, M. Bhattarai, R. J. Joyce, E. Raff, C. Nicholas, and B. S. Alexandrov, “Semi-supervised classification of malware families under extreme class imbalance via hierarchical non-negative matrix factorization with automatic model selection,” *ACM Trans. Priv. Secur.*, vol. 26, no. 4, nov 2023. [Online]. Available: <https://doi.org/10.1145/3624567>
- [7] B. Alexandrov, V. Vesselinov, and K. O. Rasmussen, “Smartensors unsupervised ai platform for big-data analytics,” Los Alamos National Lab.(LANL), Los Alamos, NM (United States), Tech. Rep., 2021, LA-UR-21-25064.
- [8] Y. Ding, J. Liu, J. Xiong, and Y. Shi, “Revisiting the evaluation of uncertainty estimation and its application to explore model complexity-uncertainty trade-off,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, 2020, pp. 4–5.
- [9] X.-Y. Zhang, G.-S. Xie, X. Li, T. Mei, and C.-L. Liu, “A survey on learning to reject,” *Proceedings of the IEEE*, vol. 111, no. 2, pp. 185–215, 2023.
- [10] H. Anderson and P. Roth, “Ember: An open dataset for training static pe malware machine learning models,” *ArXiv*, vol. abs/1804.04637, 2018.
- [11] E. Raff, J. Barker, J. Sylvester, R. Brandon, B. Catanzaro, and C. K. Nicholas, “Malware detection by eating a whole exe,” in *AAAI Workshops*, 2018.
- [12] R. Kumar and S. Geetha, “Malware classification using xgboost-gradient boosted decision tree,” *Adv. Sci. Technol. Eng. Syst.*, vol. 5, pp. 536–549, 2020.
- [13] H.-D. Pham, T. D. Le, and T. N. Vu, “Static pe malware detection using gradient boosting decision trees algorithm,” in *Future Data and Security Engineering*, T. K. Dang, J. Küng, R. Wagner, N. Thoi, and M. Takizawa, Eds. Cham: Springer International Publishing, 2018, pp. 228–236.
- [14] Y. Ye, T. Li, Y. Chen, and Q. Jiang, “Automatic malware categorization using cluster ensemble,” in *Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD ’10. New York, NY, USA: Association for Computing Machinery, 2010, pp. 95–104.
- [15] Y. Zhang, C. Rong, Q. Huang, Y. Wu, Z. Yang, and J. Jiang, “Based on multi-features and clustering ensemble method for automatic malware categorization,” in *2017 IEEE Trustcom/BigDataSE/ICSS*, 2017, pp. 73–82.
- [16] D. Kong and G. Yan, “Discriminant malware distance learning on structural information for automated malware classification,” in *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD ’13. New York, NY, USA: Association for Computing Machinery, 2013, pp. 1357–1365.
- [17] E. Raff, C. K. Nicholas, and M. McLean, “A new burrows wheeler transform markov distance,” in *AAAI*, 2020.
- [18] M. Bak, D. Papp, C. Tamás, and L. Buttyán, “Clustering iot malware based on binary similarity,” in *NOMS 2020-2020 IEEE/IFIP Network Operations and Management Symposium*. IEEE, 2020, pp. 1–6.
- [19] V. Atluri, “Malware classification of portable executables using tree-based ensemble machine learning,” in *2019 SoutheastCon*. IEEE, 2019, pp. 1–6.
- [20] F. H. Ramadhan, V. Suryani, and S. Mandala, “Analysis study of malware classification portable executable using hybrid machine learning,” in *2021 International Conference on Intelligent Cybernetics Technology Applications (ICICyTA)*, 2021, pp. 86–91.
- [21] G. E. Dahl, J. W. Stokes, L. Deng, and D. Yu, “Large-scale malware classification using random projections and neural networks,” in *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, 2013, pp. 3422–3426.
- [22] Z. Sawadogo, G. Mendy, J. M. Dembele, and S. Ouya, “Android malware detection: Investigating the impact of imbalanced data-sets on the performance of machine learning models,” in *2022 24th International Conference on Advanced Communication Technology (ICACT)*. IEEE, 2022, pp. 435–441.
- [23] R. Oak, M. Du, D. Yan, H. Takawale, and I. Amit, “Malware detection on highly imbalanced data through sequence modeling,” in *Proceedings of the 12th ACM Workshop on Artificial Intelligence and Security*, ser. AISec’19. New York, NY, USA: Association for Computing Machinery, 2019, p. 37–48.
- [24] W. Huang and J. Stokes, “Mtnet: A multi-task neural network for dynamic malware classification,” in *Proceedings of 13th International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment (DIMVA 2016)*. Springer, July 2016, pp. 399–418.
- [25] N. Loi, C. Borile, and D. Ucci, “Towards an automated pipeline for detecting and classifying malware through machine learning,” *arXiv preprint arXiv:2106.05625*, 2021.
- [26] A. Mohaisen, O. Alrawi, and M. Mohaisen, “Amal: High-fidelity, behavior-based automated malware analysis and classification,” *Computers & Security*, vol. 52, pp. 251–266, 2015.
- [27] J.-P. Brunet, P. Tamayo, T. R. Golub, and J. P. Mesirov, “Metagenes and molecular pattern discovery using matrix factorization,” *Proceedings of the national academy of sciences*, vol. 101, no. 12, pp. 4164–4169, 2004.
- [28] V. Y. Tan and C. Févotte, “Automatic relevance determination in nonnegative matrix factorization with the/spl beta/-divergence,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 7, pp. 1592–1605, 2012.
- [29] R. Vangara, M. Bhattarai, E. Skau, G. Chennupati, H. Djidjev, T. Tierney, J. P. Smith, V. G. Stanev, and B. S. Alexandrov, “Finding the number of latent topics with semantic non-negative matrix factorization,” *IEEE Access*, 2021.
- [30] I. Boureima, M. Bhattarai, M. Eren, E. Skau, P. Romero, S. Eidenbenz, and B. Alexandrov, “Distributed out-of-memory nmf on cpu/gpu architectures,” *The Journal of Supercomputing*, pp. 3970–3999, 2024.
- [31] B. T. Nebgen, R. Vangara, M. A. Hombrados-Herrera, S. Kuksova, and B. S. Alexandrov, “A neural network for determination of latent dimensionality in non-negative matrix factorization,” *Machine Learning: Science and Technology*, vol. 2, no. 2, p. 025012, 2021.
- [32] M. Eren, N. Solovyev, R. Barron, M. Bhattarai, D. Truong, I. Boureima, E. Skau, K. Rasmussen, and B. Alexandrov, “Tensor Extraction of Latent Features (T-ELF),” Oct. 2023. [Online]. Available: <https://github.com/lanl/T-ELF>
- [33] R. Bro and S. De Jong, “A fast non-negativity-constrained least squares algorithm,” *Journal of Chemometrics: A Journal of the Chemometrics Society*, vol. 11, no. 5, pp. 393–401, 1997.
- [34] E. Techapanurak, M. Suganuma, and T. Okatani, “Hyperparameter-free out-of-distribution detection using cosine similarity,” in *Proceedings of the Asian conference on computer vision*, 2020.
- [35] Y. Bahat and G. Shakhnarovich, “Confidence from invariance to image transformations,” *arXiv preprint arXiv:1804.00657*, 2018.
- [36] —, “Classification confidence estimation with test-time data-augmentation,” *arXiv preprint arXiv:2006.16705*, 2020.
- [37] T. Chen, T. He, M. Benesty, V. Khotilovich, Y. Tang, H. Cho, K. Chen, R. Mitchell, I. Cano, T. Zhou *et al.*, “Xgboost: extreme gradient boosting,” *R package version 0.4-2*, vol. 1, no. 4, pp. 1–4, 2015.
- [38] G. Ke, Q. Meng, T. Finley, T. Wang, W. Chen, W. Ma, Q. Ye, and T.-Y. Liu, “Lightgbm: A highly efficient gradient boosting decision tree,” *Advances in neural information processing systems*, vol. 30, 2017.
- [39] D. Yarowsky, “Unsupervised word sense disambiguation rivaling supervised methods,” in *Proceedings of the 33rd Annual Meeting on Association for Computational Linguistics*, ser. ACL ’95. USA: Association for Computational Linguistics, 1995, p. 189–196.
- [40] B. Marais, T. Quertier, and C. Chesneau, “Malware analysis with artificial intelligence and a particular attention on results interpretability,” in *Distributed Computing and Artificial Intelligence, Volume 1: 18th International Conference 18*. Springer, 2022, pp. 43–55.
- [41] T. Akiba, S. Sano, T. Yanase, T. Ohta, and M. Koyama, “Optuna: A next-generation hyperparameter optimization framework,” in *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*, 2019, pp. 2623–2631.