A Comparison of Machine and Deep Learning Models for Detection and Classification of Android Malware Traffic

Giampaolo Bovenzi, Francesco Cerasuolo, Antonio Montieri,
Alfredo Nascita, Valerio Persico, Antonio Pescapé
DIETI, University of Napoli "Federico II", Napoli, Italy
giampaolo.bovenzi@unina.it, fran.cerasuolo@studenti.unina.it, antonio.montieri@unina.it,
alfredo.nascita@unina.it, valerio.persico@unina.it, pescape@unina.it

***Abstract*—With the increasing popularity of mobile-app services, malicious software is increasing as well. Accordingly, the interest of the scientific community in Machine and Deep Learning solutions for detecting and classifying malware traffic is growing. In this work, we provide a fair assessment of the performance of a number of data-driven strategies to detect and classify Android malware traffic. Three models are taken into account (Decision Tree, Random Forest, and 1-D Convolutional Neural Network) considering both flat (i.e. non-hierarchical) and hierarchical approaches. The experimental analysis performed using a state-of-art dataset (CIC-AAGM2017) reports that Random Forest exhibits the best performance in a flat setup, while moving to a hierarchical approach could cause significant variation in precision and recall. Such results push for further investigating advanced hierarchical setups and learning schemes.**

***Index Terms*—Android Malware, Malware Traffic Detection, Adware Traffic Detection, Traffic Classification, Machine Learning, Deep Learning, Hierarchical Classification.**

I. Introduction

According to the latest Ericsson mobility report [1] at the end of 2021, total mobile traffic was 65 EB per month (one order of magnitude more than 5 years before) fueled by the rising number of smartphone subscriptions and the increasing data-volume per subscription. More than 50% of mobile traffic is generated by Android devices, the most popular mobile operating system in the world, which dominates the smartphone market scene together with Apple devices [2]. Together with this trend, an increase of malicious software potentially compromising users' privacy and security (e.g., tracing of user activities or theft of private information) is observed: in 2021 more than 80% of attacks was carried out using mobile malware with adware accounting for the largest share (42.42%) of all detected threats [3]. Therefore, the *detection* and *prevention* of malware and adware of various types has become a challenge of primary importance for the entire Internet community. Android malware detection techniques can be primarily divided into *static techniques* and *dynamic techniques* [4]: the former aim at detecting malicious or incorrect code without running the mobile app but processing the app package; the latter monitor the traffic generated attempting to identify notable patterns highlighting unwanted and unexpected behaviors.

In this work, we focus on *detecting and classifying malware traffic*. In this regards, data-driven techniques have recently attracted the attention of the networking research community, resulting in a number of solutions involving *Machine Learning (ML)* and *Deep Learning (DL)* approaches [2, 5–10]. Indeed, these latter have proved to be a viable solution to attain high performance in the dynamic and challenging mobile context [11]. Specifically, we investigate the performance of various strategies to identify and classify Android malware traffic. More specifically, we evaluate the benefits of *hierarchical* approaches against non-hierarchical (i.e. *flat*) ones, with the aim of reaping the benefits of model parallelism and problem splitting in sub-problems (viz. *divide-et-impera*) provided by the former. Concerning the specific algorithms whose performance are investigated, we consider two classic ML tree-based models, namely single (*Decision Tree*) and ensemble (*Random Forest*), and a DL network (*1-Dimensional Convolutional Neural Network*). For the latter, we have also evaluated the benefit of considering *embedding* layers as a way to augment information carried by input data to the classifier. For the experimental validation, we analyze a public dataset containing Android malware traffic—namely Android Adware and General Malware Dataset CIC-AAGM2017 [2]—as a relevant case study to *fairly compare* the performance of the proposed detection and classification methods. Results witness the superiority of Random Forest and show that hierarchical approaches (besides intrinsic benefits [12, 13]) do not achieve significantly better performance in the considered evaluation scenario (in spite of a higher precision) but provide room for improvements through the design of more advanced setups.

The rest of the paper is organized as follows: Section II positions our contribution in the context of previous work; Section III describes the experimentation, introducing the data-driven solutions, the dataset, and the performance metrics; Section IV discusses the results of the experiments; Section V concludes the paper.

II. Related Work

This section provides a brief overview of works performing malware classification mainly by means of ML/DL ap-

TABLE I: Details on the 1D-CNN architecture: layers and hyperparameters. Two configurations with and without the embedding layer have been evaluated.

Layer	Hyperparameters
Input	Input Shape: (N_b, 1)
Embedding	Input Dim: 256, Output Dim: 10, Input Length: 512
Convolutional1D	Filters: 16, Activation: ReLU
MaxPooling1D	Pool Size: 3
Convolutional1D	Filters: 32, Activation: ReLU
MaxPooling1D	Pool Size: 3
Convolutional1D	Filters: 64, Activation: ReLU
MaxPooling1D	Pool Size: 3
Flatten	-
Dropout	Rate: 0.25
Dense	Nodes: 256, Activation: ReLU
Dropout	Rate: 0.25
Dense	Nodes: 3, Activation: Softmax

proaches. Previous works concerning classification of Android malware exploit string matching on HTTP headers and IP/DNS blacklisting [14], behavioral analysis [15], or ML-based *flat classifiers* [2, 6–10] possibly considering the CIC-AAGM2017 dataset [2, 9, 10].

Despite ML-based hierarchical learning [16–18] and DL [19–21] for malware classification purposes are broadly explored in the networking literature, these studies are usually conducted leveraging outdated datasets (such as KDD-Cup-99 and NSL-KDD). Given their collection periods, such datasets hardly exhibit a current real-world network traffic profile, and the counter-productively use of handcrafted statistics they provide, does not permit automatic extraction of knowledge from raw traffic data, nullifying a key advantage of DL. The sole exceptions are represented by more recent works [22–25] which extract different engineered input data from raw traffic traces but mainly considering IoT or non-mobile traffic.

For what concerns considered DL algorithms most works apply AutoEncoders (AE) [22], Deep Neural Networks (DNNs) [20, 23], Convolutional Neural Networks (CNNs) [21, 24, 25], and variants of Recurrent Neural Networks (RNNs) [19, 25]. Regarding ML ones, ensembles commonly outperform single classifiers, with hierarchical approaches providing further benefits by design [12, 13].

In the present work, we evaluate on a state-of-the-art Android malware dataset [2], both ML and DL classification models, with the former models being evaluated in a flat and a hierarchical setup. ML and DL models are fed with properly designed statistical features and raw-packet payload data, thus supporting an *offline* and an *early-detection* [26] application scenario, respectively. Finally, the benefits carried by the adoption of embedding layers are also experimentally measured.

III. Experimental Setup

A. Malware Classification via ML and DL

Herein, we conduct traffic classification of Android malwares: given a *traffic object* [26] (i.e. an aggregation of traffic packets sharing common properties), the considered classification task aims to assign a label among L classes (each corresponding to benign traffic or to a malware family) within the set $\{1, \cdots, L\}$.[1] Specifically, we segment traffic into *bidirectional flows* (viz. *biflows*), defined as a stream of packets sharing the same 5-tuple (i.e. transport-level protocol, source and destination IP addresses and ports) regardless of the direction of communication.[2] In the following, we describe ML and DL techniques exploited for Android malware classification in either *flat* or *hierarchical* fashion.

Machine Learning Approaches: Firstly, we employ two ML-based models commonly used as state-of-the-art traffic classifiers, fed with 34 statistical features extracted from the sequence of packet lengths and inter-arrival times of each biflow and selected based on previous state-of-the-art results [9, 12, 25, 27], namely: min, max, mean, standard deviation, variance, mean absolute deviation, skewness, kurtosis, and 10th to 90th percentiles.[3] In detail, we evaluate a *Decision Tree (DT)*—a model that makes predictions by learning simple decision rules from the data features–and a *Random Forest (RF)* classifier—a meta-estimator operating as an ensemble of multiple DTs trained on various sub-samples of the dataset and taking decision on testing samples based on the majority voting or soft combination of the outcomes of DTs.

Deep Learning Approaches: Additionally, with the aim of performing an "early" classification of Android malware traffic, we leverage a *1-Dimensional Convolutional Neural Network (1D-CNN)* having a base architecture (i.e. layers and related hyperparameters) inspired by that proposed in [29] being an enhancement of the architecture successfully used for malware classification in [30].[4]

Table I reports the details of the 1D-CNN considered in this work. More specifically, the architecture is composed of three 1D convolutional layers (with 16, 32, and 64 filters, respectively), each followed by a 1D max-pooling, one dense layer (with 256 nodes) and is terminated with one softmax layer. It is worth noting the presence of 2 dropout layers to avoid the overfitting phenomenon. For training this network, we employ the Adam optimizer with its default parameters and the early stopping technique measured on the training accuracy (as an additional overfitting countermeasure)[5].

As a further refinement to provide a better representation of the input, an *embedding layer* has been added to the 1D-CNN architecture described above. The embedding layer is defined as the first hidden layer of a network and is described

[1] In the case of benign vs. malware detection, $L = 2$ and the classification task is binary.

[2] To distinguish biflows occurring multiple times in the dataset, we construct an unique 6-tuple exploiting the timestamp of the first packet of each biflow.

[3] We adopt the established procedure described in [28] for feature extraction.

[4] Starting with state-of-the-art proposals [29, 30], these choices have been driven by both our past experience [11] and further preliminary analyses (not shown for brevity).

[5] We highlight that the most common approach requires a validation set to monitor early-stopping. However, due to the class-imbalance of the CIC-AAGM2017 dataset (see Sec. III-B), some classes in the training set have a limited number of samples. Hence, using part of the (whole) training set for validation could impair performance associated with such minority classes. For this reason, we use early-stopping on training data by monitoring the "knee" of the training accuracy, and exiting when this condition is satisfied. Similar approaches were also recently proposed in [31].

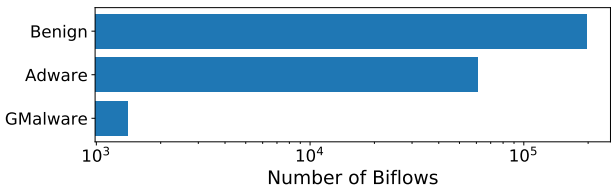


Fig. 1: Amount of biflows for each traffic category (`Benign`, `Adware`, and `GMalware`) in the CIC-AAGM2017 dataset.

by 3 parameters: input dimension, output dimension, and input length. The input dimension represents the size of the vocabulary in the input data; the output dimension is the size of the vector space in which words will be embedded; the input length is the length of input sequences. Since our inputs are integers constrained within the range $[0, 255]$, we set the input dimension to 256, the output dimension to 10, and the input length to 512, the latter being the number of bytes N_b we consider in our experimentation. Indeed, embeddings project the (unnormalized) inputs into a fixed-dimensional space, where each byte is represented by a dense vector which represents the projection of the byte into a continuous vector space. The position of a byte within the vector space is learned from input data and is based on other bytes surrounding it.

For each biflow, we feed the 1D-CNN with the first N_b bytes of transport-layer payload (from up to the first 32 packets) arranged in a byte-wise way and normalized within $[0, 1]$. Samples longer than N_b bytes are truncated to the designed length, whereas shorter instances are padded with zeros.

Flat vs. Hierarchical Approaches: In our experimentation, we have also considered the benefits deriving from the adoption of a *hierarchical approach* as opposed to a flat one. We aim to attain gains—in terms of malware classification performance or increased scalability—by splitting the classification task in sub-problems and employing a set of local classifiers each tackled to a specific sub-problem (i.e. a top-down approach). Although hierarchical approaches may result in slight increased training complexity, we can leverage model parallelism, due to their scalability and modularity. We resort to this approach as it is particularly suitable for the considered task, being the information associated with the latter naturally structured in a hierarchy of levels (i.e. benign vs. malware and specific multi-class malware classification). Specifically, we adopt the widely used *Local Classifier per Parent Node (LCPN)* hierarchical structure [12, 13, 32] which requires a multi-class classifier for each parent node in the class hierarchy, trained to distinguish among its children nodes.

B. Dataset Description

For the experimental evaluation, we used the public CIC-AAGM2017 dataset [33], generated and labeled by the *Canadian Institute for Cybersecurity* in 2017. Concerning the generation of benign traffic, the first 1500 apps of the Google Play Store ranking were installed and launched in rounds of 20 at a time. As for malicious traffic, 400 malware apps were chosen from adware and general malware, including popular families of malware, such as Airpsuh, Dowgin, Kemoge, Mobidash, and Shuanet (for adware) and AVpass, FakeAV, FakeFlash, GGtracker, and Panetho (for general malware). Malware traffic was captured with the same procedure used for benign traffic, but configuring the smartphone at each installation.

Overall, the dataset has a dimension of 9.5 GB and is composed of raw traffic traces (available in PCAP format) labeled as follows (see Fig. 1): *(i)* `Benign`: 173 traces, 1500 apps, 197670 biflows; *(ii) General Malware* (`GMalware`): 7 traces, 150 apps, 1411 biflows; *(iii)* `Adware`: 62 traces, 250 apps, 60997 biflows. The evident unbalance between benign and malicious traffic (and the higher number of adware samples) reflects the desire to represent a scenario as realistic as possible. Indeed, according to the recent reports [3, 34], the normal distribution of benign and malware apps in the *real world* is 80% to 20%, which is the *proportion kept in the dataset*, with a predominance of adware threats.

C. Performance Metrics

The experimental evaluation is based on a *stratified hold-out technique*: we split the dataset into training (80%) and test (20%) sets *by keeping the realistic proportion of samples* as they appear in the overall dataset. The following performance metrics are then employed: *precision* (the per-class ratio of decisions being correct), *recall* (the class-conditional accuracy), and *F-measure* (the harmonic mean of precision and recall).[6] Specifically, we consider their *macro* (i.e. arithmetically-averaged over classes) version for multi-class tasks. Formal definitions of binary and multi-class considered metrics are given in the following [11]. Eqs. 1–3 report the formula of binary precision (prec), recall (rec), and F-measure (F-meas) for a binary classification task (e.g., benign vs. malware), while Eqs. 4–6 generalize the formulas to a multi-class classification task (macro averaged over the L classes):

$$prec = \frac{TP}{TP + FP} \tag{1}$$

$$rec = \frac{TP}{TP + FN} \tag{2}$$

$$F\text{-}meas = \frac{2 \cdot TP}{2 \cdot TP + FP + FN} \tag{3}$$

$$prec = \frac{1}{L}\sum_{l=1}^{L} prec_l = \frac{1}{L}\sum_{l=1}^{L} \frac{\sum_{i \in C_l} TP_i}{\sum_{i \in C_l} TP_i + FP_i} \tag{4}$$

$$rec = \frac{1}{L}\sum_{l=1}^{L} rec_l = \frac{1}{L}\sum_{l=1}^{L} \frac{\sum_{i \in C_l} TP_i}{\sum_{i \in C_l} TP_i + FN_i} \tag{5}$$

$$F\text{-}meas = \frac{1}{L}\sum_{l=1}^{L} \frac{2 \cdot prec_l \cdot rec_l}{prec_l + rec_l} \tag{6}$$

[6]Given the imbalance between the classes encompassing CIC-AAGM2017, we do not use the accuracy which would provide a performance figure biased toward majority classes.

TABLE II: Performance measures [%] of ML and DL classifiers predicting finer-granularity classes. F stands for *flat*, H for *hierarchical*.

Traffic Classifier	*Family*	*Approach*	*Precision*	*Recall*	*F-measure**
RF	ML	F	80	**97**	**86**
1D-CNN (Emb)	DL	F	**97**	77	83
1D-CNN	DL	F	**97**	75	82
DT	ML	F	80	79	80
RF	ML	H	**97**	80	85

*F-measure is considered for ranking purposes.

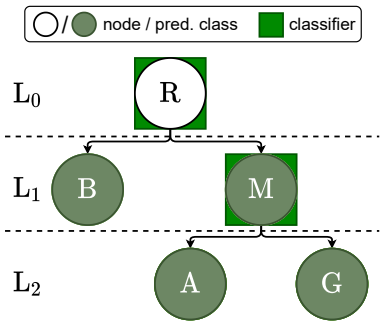


Fig. 2: Hierarchical classifier structure. R stands for *root*, B for *benign*, M for *malware*, A for *adware*, and G for *general malware*.

where TP are the True Positives, FP the False Positive, FN the False Negatives, and TN the True Negatives; C_l is the lth class with $l \in \{1, \cdots, L\}$.

In addition, we exploit *confusion matrices* to provide a graphical representation that showcases (mis)classification patterns at a finer grain. Clearly, a higher concentration toward the diagonal (where predicted app equals the actual one) implies better performance of the classifier.

IV. Experimental Results

In this section, we analyze the performance of the ML/DL malware classification solutions under investigation. Table II briefly reports their performance figures in terms of precision, recall, and F-measure.

Focusing on ML models evaluated in a flat setup, we have performed a preliminary sensitivity analysis to evaluate the performance of RF with respect to the number of estimators—reported in Appendix A for the sake of completeness—attaining the best trade-off with 100 trees. Considering this configuration, from Tab. II it is evident that DT and RF practically result in the same precision (80%). However, the ensemble method (i.e. RF) provides better performance overall, with a significant increase in terms of recall (+18%) and F-measure (+6%).

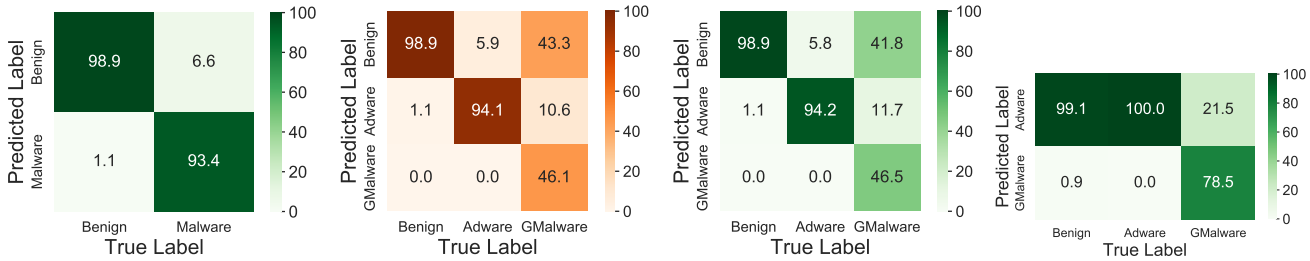
Concerning DL models, we have evaluated the performance of 1D-CNN with different values for the number of bytes N_b used as input. The assessed dimensions are 512, 784, and 1024 (according to the most common choices in recent literature [30, 35]). Since results highlighted no appreciable differences when varying N_b, in the following we refer to the performance obtained with $N_b = 512$ bytes. Indeed, this configuration has the benefit of resulting in the lowest number of trainable parameters, therefore reporting the shortest training (and inference) time. In terms of classification effectiveness, 1D-CNN achieves a high precision (97%). However, it exhibits a recall lower than DT. Overall, a loss of −4% in terms of F-measure with respect to RF is observed. As a further improvement, for the 1D-CNN network, we evaluated the performance obtained with the introduction of an embedding layer. With this refinement, it is possible to achieve a gain of +2% and +1% for recall and F-measure, respectively.

As the RF resulted the most promising model for malware classification in our experimentation, we have also tested the beneficial impact of implementing it in a hierarchical setup. Notably, in the preliminary evaluation provided in this work, we only consider a naïve implementation of the LCPN hierarchical classifier where all the nodes implement the same classification algorithm (RF) and are fed with the same feature set (see Sec. III-A). We leave further refinements in the design as a future work.

In this case, the classification architecture results in two classifiers operating in cascade (see Fig. 2): *(i)* the upstream classifier is in charge of telling `Benign` from `Malware` traffic and *(ii)* the downstream classifier distinguishes between `Adware` and `GMalware`. In detail, the classification tree starts from a binary classifier (R) that discriminates samples between `Benign` (B) and `Malware` (M) classes [12]. We recall that at this level (L_0) the instances whose labels are either `Adware` or `GMalware` are grouped into the `Malware` class. If an instance is predicted to be `Malware`, then we move to the next classification level (L_1), where another binary classifier (M) discriminates between `Adware` (A) and `GMalware` (G) being the leaves of the tree (L_2).

Deepening into obtained results, in Fig. 3 confusion matrices are shown for both hierarchical and flat RF, in order to better analyze the per-class errors at different levels of the aforementioned hierarchy of labels. A first outcome is shown in Fig. 3a, with the root (binary) classifier (R in Fig. 2) obtaining 98.9% TNR (True Negative Rate, viz. the fraction of correctly classified `Benign` samples) and scoring 6.6% FNR (False Negative Rate, viz. the fraction of `Malware` samples wrongly labeled as `Benign`).

In addition, Figs. 3b and 3c show the overall performances obtained by the flat and the hierarchical approach, respectively. Despite results seem to be quite similar, some differences can be spotted looking at the `GMalware` family, with the hierarchical approach better confining wrongly classified `GMalware` samples within the malware classes. In particular, `GMalware` traffic is less confused with `Benign`, passing from a misclassification ratio of 43.3% (Fig. 3b) to 41.8% (Fig. 3c). It is worth noting that improvements regard the minority class, showing some mitigation of the class-imbalance issue. Finally, Fig. 3d shows the fine-grained behavior of the malware classifier (M in Fig. 2), with the `Benign` instances wrongly classified by the R node (i.e. errors propagated downward the hierarchy to L_1) which are mainly confused with the `Adware` class (99.1%), an effect probably due to the intra-malware class-imbalance (between `Adware` and `GMalware`) that is worth to be further investigated (and possibly mitigated) in future works.



(a) Flat/Hierarchical L_1. (b) Flat L_2. (c) Hierarchical L_2. (d) Hierarchical L_2 (M classifier).

Fig. 3: Confusion matrices of Random Forest classifier performing flat classification (a, b) and hierarchical classification (a, c, d). It is worth noting that Hierarchical L_1 (a) is also Flat L_1.

Summarizing, although overall performance figures reach values around 86% F-measure, applying hierarchical approaches in the naïve setup provides results that are not entirely satisfactory. Compared against the flat approach (first and last row in Tab. II), the hierarchical approach achieves almost the same results in terms of F-measure. However, looking at precision and recall, we have a completely opposite situation. In detail, the hierarchical approach shows a lower recall but a higher precision (+17%), thus obtaining *more robust* results, i.e. less malicious samples are classified as `Benign`. In conclusion, applying hierarchical solutions has a better potential, since it leaves room for possible optimizations, such as optimizing (i) the way of using the local information at each node or (ii) the hierarchical architecture itself by selecting different/optimal algorithms and number/type of features at each node of the hierarchy [12]. Such improvements would provide a higher recall without sacrificing the precision (i.e. an overall better F-measure), thus mitigating the possibility of treating a benign sample as malicious and consequently impacting the quality of experience of mobile users. We leave this aspect for future works.

V. Conclusion and Future Work

Leveraging a state-of-the-art Android malware dataset, in this paper we have evaluated the effectiveness of ML and DL models to identify and classify Android malware with the aim of detecting possible vulnerabilities that could affect such complex *distributed systems*. Different models have been considered: DT, RF, and 1D-CNN, with the implementation of the latter also refined with the adoption of an embedding layer. Experimental results show that among these solutions, RF provides the best performance when considering a flat (i.e. non-hierarchical) setup, with 97% recall and 86% F-measure. Hence, RF is also evaluated in a hierarchical setup, implementing a cascade of two binary classifiers, distinguishing between *(i)* Benign and Malware traffic and *(ii)* Adware and General Malware, respectively. Results witness that the considered implementation of the hierarchical approach—besides its intrinsic benefits by design—does not provide improved performance figures overall, even if a significant improvement in terms of precision is observed (+17%) leaving also room for enhancements.

The outcomes of the present work are particularly relevant for the distributed systems community. Indeed, they constitute a starting point toward the adoption of federated solutions for classification of malware traffic that would exploit model parallelism (intrinsic to a hierarchical framework) and data parallelism (e.g., by integrating DL architectures with big data platforms) [13]. Both are appealing future perspectives. As a future work, we also aim to further investigate the benefits of hierarchical approaches, relaxing the constraints of having the same model and features for each node of the hierarchy. Moreover, we plan to exploit more complex DL architectures but with a lightweight design originated from pruning, quantization, and distillation techniques. Finally, we aim at capitalizing the heterogeneity of traffic via multimodal architectures, as well as jointly tackling benign traffic classification and malware classification in a multitask fashion.

Acknowledgment

The authors would like to thank Ms. Anna Lamboglia, Mr. Daniel Parisi, and Mr. Francesco Ottata for their collaboration to the preliminary experiments to this study.

References

[1] F. Jejdling *et al.*, "Ericsson Mobility Report." *Ericsson AB, Business Area Networks, Stockholm, Sweden, Tech. Rep. EAB-21*, vol. 10887, November 2021.

[2] A. Habibi Lashkari, A. F. Abdul kadir, H. Gonzalez, K. Mbah, and A. Ghorbani, "Towards a Network-Based Framework for Android Malware Detection and Characterization," in *2017 15th Annual conference on privacy, security and trust (PST)*. IEEE, 2017, pp. 233–23 309.

[3] T. Shishkova and A. Kivva, "Kaspersky Lab - Mobile malware evolution 2021," 2021, https://securelist.com/mobile-malware-evolution-2021/105876/.

[4] T. Sharma and D. Rattan, "Malicious application detection in android—a systematic literature review," *Computer Science Review*, vol. 40, p. 100373, 2021.

[5] S. Shamshirband, M. Fathi, A. T. Chronopoulos, A. Montieri, F. Palumbo, and A. Pescapè, "Computational intelligence intrusion detection techniques in mobile cloud computing environments: Review, taxonomy, and open research issues," *Journal of Information Security and Applications*, vol. 55, p. 102582, 2020.

[6] A. H. Lashkari, A. F. A.Kadir, L. Taheri, and A. A.Ghorbani, "Toward Developing a Systematic Approach to Generate Benchmark Android Malware Datasets and Classification," *IEEE Network*, pp. 1–7, 2018.

[7] A. Arora and S. K. Peddoju, "Minimizing Network Traffic Features for Android Mobile Malware Detection," in *Proceedings of the 18th International Conference on Distributed Computing and Networking*, ser. ICDCN '17, 2017.

[8] M. Abuthawabeh and K. Mahmoud, "Enhanced Android Malware Detection and Family Classification, using Conversation-level Network Traffic Features," *The International Arab Journal of Information Technology*, vol. 17, no. 4A, pp. 607–614, 2020.

[9] K. Lee and H. Park, "Malicious adware detection on android platform using dynamic random forest," in *International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing*. Springer, 2019, pp. 609–617.

[10] A. Karami, "An anomaly-based intrusion detection system in presence of benign outliers with visualization capabilities," *Expert Systems with Applications*, vol. 108, pp. 36–60, 2018.

[11] G. Aceto, D. Ciuonzo, A. Montieri, and A. Pescapé, "Mobile encrypted traffic classification using deep learning: Experimental evaluation, lessons learned, and challenges," *IEEE Transactions on Network and Service Management*, vol. 16, no. 2, pp. 445–458, 2019.

[12] A. Montieri, D. Ciuonzo, G. Bovenzi, V. Persico, and A. Pescapé, "A Dive into the Dark Web: Hierarchical Traffic Classification of Anonymity Tools," *IEEE Transactions on Network Science and Engineering*, vol. 7, no. 3, pp. 1043–1054, 2020.

[13] G. Bovenzi, G. Aceto, D. Ciuonzo, V. Persico, and A. Pescapé, "A big data-enabled hierarchical framework for traffic classification," *IEEE Transactions on Network Science and Engineering*, vol. 7, no. 4, pp. 2608–2619, 2020.

[14] D. Iland, A. Pucher, and T. Schauble, "Detecting Android Malware on Network Level," *University of California, Santa Barbara*, vol. 12, no. 2011, 2011.

[15] L. Tenenboim, O. Barad, A. Shabtai, D. Mimran, L. Rokach, B. Shapira, and Y. Elovici, "Detecting Application Update Attack on Mobile Devices Through Network Features," in *2013 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*. IEEE, 2013, pp. 91–92.

[16] C. Guo, Y. Ping, N. Liu, and S.-S. Luo, "A two-level hybrid approach for intrusion detection," *Neurocomputing*, vol. 214, pp. 391–400, 2016.

[17] S.-Y. Ji, B.-K. Jeong, S. Choi, and D. H. Jeong, "A multi-level intrusion detection method for abnormal network behaviors," *Elsevier JNCA*, vol. 62, pp. 9–17, 2016.

[18] F. A. Khan, A. Gumaei, A. Derhab, and A. Hussain, "TSDL: A Two-stage Deep Learning Model for Efficient Network Intrusion Detection," *IEEE Access*, 2019.

[19] C. Yin, Y. Zhu, J. Fei, and X. He, "A deep learning approach for intrusion detection using recurrent neural networks," *IEEE Access*, vol. 5, pp. 21 954–21 961, 2017.

[20] A. A. Diro and N. Chilamkurti, "Distributed attack detection scheme using deep learning approach for Internet of Things," *Future Generation Computer Systems*, vol. 82, pp. 761–768, 2018.

[21] K. Wu, Z. Chen, and W. Li, "A novel intrusion detection model for a massive network using convolutional neural networks," *IEEE Access*, vol. 6, pp. 50 850–50 859, 2018.

[22] G. Bovenzi, G. Aceto, D. Ciuonzo, V. Persico, and A. Pescapé, "A hierarchical hybrid intrusion detection approach in IoT scenarios," in *GLOBECOM 2020-2020 IEEE Global Communications Conference*. IEEE, 2020, pp. 1–7.

[23] T. M. Booij, I. Chiscop, E. Meeuwissen, N. Moustafa, and F. T. den Hartog, "ToN_IoT: The Role of Heterogeneity and the Need for Standardization of Features and Attack Types in IoT Network Intrusion Datasets," *IEEE IoT Journal*, 2021.

[24] O. Barut, Y. Luo, T. Zhang, W. Li, and P. Li, "Multi-task hierarchical learning based network traffic analytics," in *ICC 2021-IEEE International Conference on Communications*. IEEE, 2021, pp. 1–6.

[25] A. Nascita, F. Cerasuolo, D. Di Monda, J. T. A. Garcia, A. Montieri, and A. Pescapé, "Machine and Deep Learning Approaches for IoT Attack Classification," in *2022 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*. IEEE, 2022, pp. 1–6.

[26] G. Aceto, D. Ciuonzo, A. Montieri, and A. Pescapé, "Toward effective mobile encrypted traffic classification through deep learning," *Neurocomputing*, vol. 409, pp. 306–315, 2020.

[27] V. F. Taylor, R. Spolaor, M. Conti, and I. Martinovic, "Robust smartphone app identification via encrypted network traffic analysis," *IEEE Transactions on Information Forensics and Security*, vol. 13, no. 1, pp. 63–78, 2017.

[28] G. Aceto, D. Ciuonzo, A. Montieri, V. Persico, and A. Pescapé, "MIRAGE: Mobile-app Traffic Capture and Ground-truth Creation," in *2019 4th International Conference on Computing, Communications and Security (ICCCS)*. IEEE, 2019, pp. 1–8.

[29] W. Wang, M. Zhu, J. Wang, X. Zeng, and Z. Yang, "End-to-end encrypted traffic classification with one-dimensional convolution neural networks," in *2017 IEEE international conference on intelligence and security informatics (ISI)*. IEEE, 2017, pp. 43–48.

[30] W. Wang, M. Zhu, X. Zeng, X. Ye, and Y. Sheng, "Malware Traffic Classification using convolutional neural network for representation learning," in *IEEE International Conference on Information Networking*, 2017, pp. 712–717.

[31] D. Duvenaud, D. Maclaurin, and R. Adams, "Early stopping as nonparametric variational inference," in *Artificial Intelligence and Statistics*. PMLR, 2016, pp. 1070–1077.

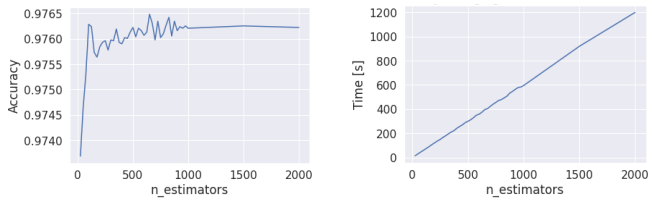
[32] L. Grimaudo, M. Mellia, and E. Baralis, "Hierarchical learning for fine grained internet traffic classification," in *8th International Wireless Communications and Mobile Computing Conference*, 2012, pp. 463–468.

[33] C. I. for Cybersecurity, "Android Adware and General Malware Dataset (CIC-AAGM2017)," 2017, data retrieved for the first time in 03/31/2021 at https://www.unb.ca/cic/datasets/android-adware.html.

[34] S. S. Center, "Internet Security Threat Report," 2016, https://www.symantec.com/security-center/threat-report.

[35] G. Aceto, D. Ciuonzo, A. Montieri, and A. Pescapé, "DISTILLER: Encrypted traffic classification via multimodal multitask deep learning," *Journal of Network and Computer Applications*, vol. 183, p. 102985, 2021.

APPENDIX A
SENSITIVITY ANALYSIS FOR RANDOM FOREST



(a) Accuracy. (b) Training Time.

Fig. 4: Sensitivity Analysis for the Random Forest classifier with respect to the number of estimators.

In this supplementary analysis, we have evaluated how the performance of the Random Forest classifier (in terms of accuracy, for the sake of simplicity) changes as the number of estimators (i.e., the trees considered as base classifiers) increases. We also consider the training time to account for the complexity of the obtained classifier. Looking at the results in Fig. 4, we can notice that using 100 estimators is a good trade-off between accuracy and training time. Indeed, increasing the number of trees increases the accuracy only marginally ($< 1\%$ improvement) at the cost of much higher training times (one order of magnitude).