
# GreenForensics: Deep Hybrid Edge-Cloud Detection and Forensics System for Battery-Performance-Balance Conscious Devices

By:
Mohit Sewak, Sanjay K. Sahay and Hemant Rathore

From the proceedings of
The Digital Forensic Research Conference
**DFRWS APAC 2022**
Sept 28-30, 2022

Forensic Science International: Digital Investigation 43 (2022) 301445

Contents lists available at ScienceDirect

Forensic Science International: Digital Investigation

journal homepage: www.elsevier.com/locate/fsidi

DFRWS 2022 APAC - Proceedings of the Second Annual DFRWS APAC

# GreenForensics: Deep hybrid edge-cloud detection and forensics system for battery-performance-balance conscious devices

Mohit Sewak [a, *], Sanjay K. Sahay [b], Hemant Rathore [b]

[a] Security & Compliance Research, Microsoft R&D, India Pvt. Ltd., India
[b] Dept. of CS & IS, Goa Campus, BITS Pilani, India

ARTICLE INFO

*Article history:*

*Keywords:*
Efficient forensics
Battery-performance balance aware deep learning
Deep clustering
Hybrid edge

ABSTRACT

Motivated by the advancements made by the recently proposed DRo algorithm to uplift the performance of data scarce Deep Learning malware detector for edge, we propose an adaptive and efficient system for hybrid edge-cloud detection and forensics, named GreenForensics. The proposed adaptive enhancement, makes the system more suitable for devices with custom battery-performance optimization mandates like tablets and laptops. Further, the enhancements offer various discrete and continuous controls for influencing the detection coverage and model footprints in real time. To further enhance the detection efficiency and making the detection resilient to adversarial-attacks, the proposed system can work with adversarial-DL immune algorithms. In the experiments conducted, GreenForensics was able to significantly outperform even the best baseline deep architectures and improved the detection and forensics robustness by up to **100%** and performance by up to **40%**. This gains further significance as the incumbent baseline DL architecture had up to **6700%** higher neural inference complexity, and had its performance and robustness benchmarks had remained unchallenged for a long time.


## 1. Introduction

As per the *Work Trend Index* report published by Microsoft (Wiseman, 2021), more than 73% of employees would like to continue to work in a hybrid manner post pandemic, and 66% of business leader are considering redesigning their work culture to enable this. This sudden and extreme shift to hybrid work cultures has led to the adoption of more personal and mobile devices like Windows laptops and tablets and raising the concerns on enabling similarly high level of security mechanism on these battery-performance conscious machines. Since such battery powered client devices can ill-afford the current advanced and computationally involved detection mechanisms as were designed for their more powerful and AC connected workstations and server counterparts, this adoption trend is also seen as an opportunity by many ransomware operators and malware developers to specifically target such devices. Therefore, the recent issue of Microsoft's *Digital Defence Report* (Hogan-Burney, 2021) observes that human-operated-ransomware (HumOR) operators are increasingly attacking devices belonging to this segment. In 2021, the individual consumers were the highest engagement segment for the (13%) Ransomware as a Service (RaaS) operators, followed by financial and manufacturing sectors (at 12% each). Therefore, innovation for performant and efficient mechanisms to protect such battery-conscious Windows clients from ransomware and other advanced malware has increasingly become critical.

Recently Sewak et al. presented a novel and innovative algorithm named DRo (Sewak et al., 2021a) (for **D**eep **Ro**uter), and proposed many end-to-end (e2e) Deep Learning (DL) architectures for implementing DRo under multiple unique scenarios; including some for malware detection (followed by forensics via vault) at the edge device itself. The promising experimental results indicated that DRo could revolutionize the performance of DL based security systems even under extremely data-scarce scenarios. The DRo algorithm was benchmarked using an Android based on-edge architecture (DRoID), and it was demonstrated that DRo could help design a performant and efficient malware detector even with low-information features extracted from sparsely labeled data. Further, it was demonstrated that the model could be tuned for varying degrees of performance (accuracy/recall) and false-positive-rates (FPR) balance mandates for the downstream DL model.

Motivated by the performance of DRo and some of its architectures that intrinsically support a vault based-mechanism for

* Corresponding author.
*E-mail address:* mohit.sewak@microsoft.com (M. Sewak).

https://doi.org/10.1016/j.fsidi.2022.301445

subsequent forensics, we propose new and unique architectures and application areas for DRo. In particular, we propose multiple adaptive enhancements to the architecture of DRo and apply them to the Windows endpoint security domain. We further strengthen the vault mechanism to have adaptive coverage to support efficient forensics. We call the resulting system as GreenForensics. We take one of DRo's (hierarchical) architecture that has been proposed for low-powered edge devices, and adapt it to suit multitude of hybrid edge-cloud inference requirements. The adaptive enhancement proposed by us, makes DRo more suitable for use with custom battery-performance-balance mandate devices like Windows clients, tablets, and laptops; and simultaneously also provides opportunities for influencing performance and coverage balance in real time. This is achieved by offering various discrete and continuous controls options with varying complexity and coverage outcomes that takes the *Power Mode* mandate and battery drain status as inputs. Moreover, to further enhance the detection and forensics efficiency and to make the system resilient to adversarial-DL attacks (Rathore et al., 2021; Sewak et al., 2021b), we adapted DRo to also work with adversarial-attack immune classical Machine Learning (ML) algorithms like the Random Forest (RF) (Sewak et al., 2018). In line with the results claimed in the DRo's paper (of improving the robustness and accuracy of downstream DL detection by up to ≈ 67% and 11% respectively (Sewak et al., 2021a)), GreenForensics offers similar advantages and further improves upon these benchmarks and has more controlled coverage of malware samples for subsequent forensics. On a popular and standardized malware and ransomware dataset (Nappa et al., 2015), GreenForensics was able to improve the detection robustness by up to **100%** (0% FPR across multiple experiments) and performance/area-under-curve statistics (AUC) by up to ≈ **40%** as compared to even the best baseline e2e DL configurations. This is significant, as the comparison was made with some of best baseline e2e DL architectures, which had up to ≈ **6700%** higher neural inference complexity as compared to GreenForensics, and their established benchmarks had remained unchallenged by any other DL architecture, technique, or model since a long time now.

Following are the key research contributions of this paper:

1. We propose a new architecture for DRo, named GreenForensics, which could balance between the edge detection:forensics coverage ratio and detection complexity (without compromising with the detection performance or robustness); and hence is extremely useful for hybrid edge-cloud detection.
2. The original DRo introduced hyperparameters to influence a desirable trade-off between accuracy and FPR; we propose further enhancements to extend such flexibility to balance between on-demand coverage and on-demand complexity. Moreover, we provide two unique types of controls, one discrete control and another continuous control for this purpose.
3. Where the primary aim of the original DRo proposal was to enhance the performance of a downstream DL classifier under scarce label scenario (as an alternative to Transfer Learning, but under more extreme conditions); we further demonstrate that DRo, despite being a highly-efficient and low-complexity algorithm, could (mostly) by itself surpass the benchmarks of most DL architectures, even e2e DL pipelines comprising of multiple DL models, all of which being many times larger and more complex than DRo; and does not need an assistance of existing e2e DL pipeline to achieve this objective. To this end, we combine DRo with exceedingly simple, efficient and adversarial-DL immune classical ML algorithms and obtain several benchmarks to support this claim.
4. Finally, we further established the universality and wide applicability of the DRo mechanism by implementing it on datasets beyond Android malware, and beyond simplistic/non-intuitive features. We apply DRo on informative features, on which exceedingly high benchmarks already exists. Our experiments demonstrate the DRo continue to break all benchmarks established by any existing DL models as used in security, including e2e DL pipelines and several layered deep malware detection systems.

The remaining the paper is organized as follows. In section 2 we cover the related work with several aspect of DRo and the problem we solve with GreenForensics. Next, we cover the mathematics of (original) DRo and our proposed architecture for GreenForensics in section 3. We cover details of an illustrative implementation of GreenForensics for a Windows based power-performance optimized device (WinDRo) in section 3.4. Then we provide the details on the dataset and baseline experiments in section 4, and next the benchmark experiments and their results in section 4.1. Finally, we provide conclusion in section 5.

## 2. Related work

In this section, we first describe some related work on the mathematical basis of DRo, which is built over recent advances in the field of Deep Clustering. Next, since GreenForensics at a high level does data sampling for training and inferencing at the edge vs. on the cloud, we produce some art related to these aspects.

### 2.1. Deep clustering

End-to-End Deep Learning for discrete representation learning (clustering and hashing) has is becoming popular in research fraternity and different algorithms have been proposed that have been proposed to this end. Such discrete representation could be for clustering or hashing. When Deep Learning is used for generating discrete representations like hashing or clustering, we in general refer to it as Deep Clustering (DC). Many of the conventional and popular *representation* learning algorithms like Gaussian Mixture Modeling (GMM) and K-Means are incapable of modelling non-linear boundary separation between groups/clusters. Others like the *kernel* (Xu et al., 2005; Kulis and Darrell, 2009) and *spectral* (Weiss et al., 2009) clustering-based techniques though can model arbitrary cluster boundaries, but can not efficiently scale to large datasets. Here is where DL, especially deep clustering can create a large impact, as it is capable of modeling and learning non-linear and complex separation boundaries and is extremely scalable and flexible. Many of the recent algorithms in this field are also capable of producing data embeddings along with the cluster representations.

Besides learning discrete data representation, the reason why deep clustering became popular as a semi-supervised pre-processing approach for DL algorithms is because it could generate embeddings that could be used to make e2e DL architectures. In this regard, the first DC algorithm that could simultaneously also generate embedding and clustering representations was Deep Embedding for Clustering (DEC) (Xie et al., 2016). Since this is an evolving field of research different ideas have gained popularity on how to make discrete representation learning more robust. In this regard, some approaches for generating adversarial robustness to different types of DL models exist (Jiang et al., 2017). In these approaches, the distribution of data is modeled by a *generative* algorithm, and then GMM models are used to represent the prior distributions for these models. This approach is atypical to data generation using Variational Auto Encoders (VAE) (Sewak et al., 2020). Leen (1995) proved that applying data augmentation to a DL classifier gives a similar effect as applying regularization to the

original cost function of a conventional machine learning (ML) model. Miyato et al. (2015), and Sajjadi et al. (2016) successfully adapted this approach to semi-supervised DL algorithms and demonstrated successful results. Self Augmentation Training (SAT) is inspired by this approach of regularization. Dosovitskiy et al. (2014) proposed to use data augmentation to model the invariance of learned representations for unsupervised algorithms like clustering. IMSAT takes a similar approach but applies invariance directly to the learned representation instead of applying it to the surrogate classes, and directly learns discrete representations (clusters), instead of learning continuous representations that are later converted to discrete class representations or clusters predictions. Further, it combines Information Maximization (IM) along with SAT to produce more robust cluster formations; a special coefficient $\lambda$ is used to balance between entropy loss and augmentation loss for learning cluster representations.

### 2.2. Sampling for forensics and hybrid edge-cloud detection

Where the role of a malware classifier is to detect a malware sample, the role of a forensic sampler is to analyze which sample could or could not be detected well by a malware detector and hence sample them either to an automated detector or to a dedicated forensics team or a vault. Where there exists good research on designing effective and efficient malware classifiers, there is a dearth of research on automated forensic/detection sampler systems. Whatever work exists, use complex Deep Reinforcement Learning (DRL) algorithms and offer black-box solutions to one of the variables in the integrated problem scenario of training data sampling, associated inference data routing, hybrid mobile-edge detection, and model complexity-based routing. Some selected works in each of these problem areas are given as follows. Recently some suggestions have come from the research in Network Intrusion Detection System (NIDS), where Lopez et al. (Lopez-Martin et al., 2020) used DRL to train agents to learn to sample anomaly logs for training an anomaly detection system. Similarly, DRL solutions have been proposed by Xiaoyue et al. (Wan et al., 2017) for selective offloading of mobile-based inference to the cloud. Similarly, for the problem of overall detection complexity management, a DRL based solution to route appropriate inference candidates to one of the many available models of different complexity is proposed by Yoni et al. (Birman et al., 2020).

## 3. Background and the proposed architecture

In this section we provide a brief on the DRo algorithm (in section 3.1) cover the mathematical basis of (original) DRo (in section 3.2). Next, we cover an illustrative architecture for implementing a GreenForensics based system for hybrid edge-cloud inference (in section 3.3) and compare it with that of one of the architectures (for edge devices) of DRo. Finally, we cover the formulation of the WinDRo solution that applies the GreenForensics architecture to a battery-performance optimized connected Windows client (tablet/laptop) for hybrid edge-cloud detection and routing (in section 3.4).

### 3.1. About DRo

Sewak et al. (2021a) used the ideas of Deep Clustering to solve a novel problem faced by most DL application domains, i.e. learning robust detection under scarce label data scenarios. In this regard, they further modified the deep clustering algorithms to form DRo (**D**eep **Ro**uter) such that the cluster assignment directly ends in providing routing suggestions for data samples based on the cluster-separation-margin which is indirectly an indication of the associated noise level in the sample. The DRo mechanism proves useful for effective, noise-free training of a (label data affine) DL classifier. Further, the same routing suggestions were used for selectively routing inference candidates to a classifier trained by DRo sampled data. Since the training and inferencing are both controlled by the same model, DRo enabled downstream DL discriminators to effectively learn meaningful discriminative features with little representative data. In doing so, the downstream DL classifiers consistently outperformed even more sophisticated models trained on the same input data. With an end-objectively to influence the recall and robustness of the downstream discriminator, Sewak et al. introduced another hyperparameter, $\mu$, that strike a balance between the absolute and conditionally entropy losses to alter the routing suggestions. We cover the related mathematical details on DRo in section 3.2.

### 3.2. Mathematical basis of DRo

Representation Learning and unsupervised learning like clustering are difficult tasks as the data associations are not defined. But acquiring labeled data is expensive, and sometimes even infeasible. Also, for many tasks, the labels are not determined apriori. Hence, discrete representation learning, like hashing and clustering, from unlabeled is a critical task. Classical Machine Learning (ML) offers many popular clustering algorithms like the k-means (Hartigan and Wong, 1979), hierarchical clustering (Johnson, 1967) etc. For this purpose; but these algorithms do not scale to large datasets and complex non-linear patterns. These are the areas where Deep Learning (DL) has invariably replaced many classical ML solutions. DL offers complex algorithms to model sophisticated non-linear patterns in the data. This also makes DL over-fit to the training data easily, and in the process lose the ability to identify meaningful domain representations from noise. In unsupervised learning, since the target is not specified, the problem is unconstrained, which further exacerbates this problem. Therefore, such an algorithm needs optimal regularization to perform under different scenarios. In DL invariance of different types can be introduced to make it immune to slight alterations in patterns (Leen, 1995) and subsequently regularize the learning. One type of invariance is local invariance, which uses Self Augmented Training (SAT). Such augmented learning in the simplest form could be generated by random perturbations or strategic adversarial perturbations. Perturbations-based SAT generate local perturbations from original data samples and in the loss function tries to minimize the loss between surrogate classes and their predicted representation to make the model invariant to such local perturbations.

IMSAT (Hu et al., 2017) (refer section 2.1 for a discussion on other deep clustering algorithms), uses perturbation based SAT. Besides using the SAT penalties to provide regularization, IMSAT also use the (Regularized) Information Maximization (RIM) criteria (Krause et al., 2010), (Bridle et al., 1992) for identifying cluster boundaries. These two criteria are balanced in the total loss function of clustering by a regularization coefficient $\lambda$. This can be represented as in equation (1), where $R_{SAT}$ represents the regularization loss for SAT and $R_{IM}$ the loss due to cluster representation learning (negative information gain).

$$Loss_{total} = R_{SAT} + \lambda R_{IM} \tag{1}$$

#### 3.2.1. $R_{SAT}$ loss

Assume that $T$: $\mathbf{X} \rightarrow \mathbf{X}$ is the transformation function that generates the local perturbations (data augmentation) to ensure invariance. If $p$ is a small perturbation that does not alter the

discrete representation of the data sample in the given context, then in the simplest form, the transformation function, $T$, function could be expressed as $T(x) = x + p$. SAT against such small, local perturbations, $p$, generates local-invariance in the learned representations, and hence pushes that the cluster-separation boundaries to the low sample density regions. Such cluster-separation boundaries are desired as these abide by the low-density-separation principle. For a sample with feature vector $x$, with surrogate label $y$, the probability $p_{\widehat{\theta}}(y_m|x)$ of the learned representation $y_m$ in an $M$ class discrete representation learning problem ($m \in [1, \ldots, M]$), using a function parameterized over hyper-parameter vector $\widehat{\theta}$, the SAT regularization penalties, $\mathcal{R}_{SAT}(\theta; x, T(x))$, is given in equation (2).

$$\mathcal{R}_{SAT}(\theta; x, T(x)) = -\sum_{m=1}^{M} \sum_{y_m=0}^{V_m-1} p_{\widehat{\theta}}(y_m|x) \log p_\theta(y_m|T(x)) \tag{2}$$

The SAT regularization loss for the entire batch $\mathbf{X}$ of data (where $x \in \mathbf{X}$) is the average of the individual sample level SAT loss $\mathcal{R}_{SAT}(\theta; x, T(x))$. This is given as in equation (3).

$$\mathcal{R}_{SAT}(\theta; T) = \frac{1}{N} \sum_{n=1}^{N} \mathcal{R}_{SAT}(\theta; x_n, T(x_n)) \tag{3}$$

#### 3.2.2. $R_{IM}$ loss

The Regularized Information Maximization algorithm minimizes the objective as in equation (4). In this, $x \in \mathbf{X}$ are the data samples, and $Y \in \mathbf{Y} \equiv \{1, \ldots, M\}$ are the cluster predictions (for $M$ class clusters assignment problem). $\mathcal{R}_{IM}$ is the RIM regularization penalty, and $\mathbf{I}(\mathbf{X}; \mathbf{Y})$ is Mutual Information (MI) between surrogate classes of the data samples and the cluster assignment representations.

If $\mathbf{I}(\mathbf{X}; \mathbf{Y})$ represents the Mutual Information (MI) between surrogate classes ($y_m$) of the data sample $x (x \in \mathbf{X})$ and the cluster assignment representations, $y_k([0, \ldots, k] \in K)$, the $R_{IM}$ loss could be given as in equation eq: rim-objective.

$$\mathcal{R}_{IM} = -\mathbf{I}(\mathbf{X}; \mathbf{Y}) \tag{4}$$

The RIM's cluster probability prediction function could be expressed as $p_\theta(y_1, \ldots, y_m|x)$ by the associated DNN architecture. Assuming that the data dimensions are conditionally independent, the joint probability could be expressed as a product of individual conditional probabilities as in equation (5).

$$p_\theta(y_1, \ldots, y_m|x) = \prod_{m=1}^{M} p_\theta(y_m|x) \tag{5}$$

If the marginal-entropy (ME) $H(Y)$ of a classes in a data is expressed as in the equation (6), and the conditional-entropy (CE) (equation (7)) of the cluster-assignments of the data conditioned on the input features is $H(Y|X)$, then the Mutual Information Gain ($-R_{IM}$) due to *clustering* is as a difference between ME and CE ($-(H(Y) - H(Y|X))$) (Cover and Thomas, 2006); where the entropy function is given in equation (8).

$$H(Y) \equiv h(p_\theta(y)) = h\left(\sum_{i=1}^{N} p_\theta(y|x)\right) \tag{6}$$

$$H(Y|X) \equiv \frac{1}{N} \sum_{i=1}^{N} h(p_\theta(y|x)) \tag{7}$$

$$h(p(y)) \equiv -\sum_{y'} p(y') \log p(y') \tag{8}$$

#### 3.2.3. $L_{DRo}$

An enhanced the Marginal Entropy $H(Y)$ could enforce the cluster sizes to be uniform. Whereas a diminished Conditional Entropy $H(Y|X)$ could enforce unambiguous and low-noise cluster assignments (Bridle et al., 1992). To influence this balance desirably, an additional hyperparameter $\mu \in \mathbb{Z}$, was added to further tune the Conditional Entropy in DRo. This mechanism offered added advantage of controlling the coverage across different routes of DRo, thus enabling different modes like the *Vault Mode* and *Hierarchical Mode* settings. The modified loss function of the DRo algorithm could be given as in equation (9).

$$Loss_{DRo} = R_{SAT} - \lambda((H(Y) - \mu H(Y|X)) \tag{9}$$

Experimentally the coefficient $\mu \in$ (Sewak et al., 2021a, 2021b) provides a decent range for default settings, and $\mu \in$ (Wiseman, 2021; Weiss et al., 2009) provides a decent range to influence recall and FPR across a wide range of applications.

### 3.3. Proposed architecture for GreenForensics

The original DRo (Sewak et al., 2021a) offers multiple architectures. The prominent ones of those being a single-layer architecture, mostly suitable for an unconnected edge device with on-device vault provision, and a second one for slightly more powerful edge devices, that can use multiple layers of detection (e.g., as in Advanced Threat Protection (ATP)) which might be sparingly connected to online services for dynamic analysis, feedback, and model updates. Both architectures are remarkably useful for mobile devices that are mostly configured of extreme power efficiency and in-frequent network usage. The hierarchical (multi-routing) DRo architecture however could be configured for a variety of usage. Therefore, we take this second architecture as a base, and modify it to propose an architecture for more powerful clients (e.g., tablets, and laptops), which though value battery efficiency, but largely desires an optimal battery-performance balance (as is available in Windows 11), and are more often connected. The architecture for the proposed GreenForensics system is given in Fig. 2 and for comparison, the architecture of Hierarchical multi-routing DRo is given in Fig. 1. The following are the key difference between these architectures:

- As compared to the hierarchical DRo, GreenForensics transfers the subsequent detection routing beyond the first layer *selectively* to the cloud infrastructure.
- Both the architectures use multiple DRo models, but where the hierarchical DRo uses these in a tree based hierarchical section, and all the models could have similar model complexity and footprint, GreenForensics selective choose one of the multiple models for edge-detection routing, and each model has a varying level of neural complexity and footprint.
- Instead of assisting a DL classifier and forming an e2e DL architecture with shared features, GreenForensics use a classical ML algorithm (e.g., Random Forest) for 2 main reasons; first to make the systems more robust to adversarial-DL attacks, and second to further lower the model neural complexity and footprints, and make it ideal for devices which do not have a powerful/dedicated GPU.

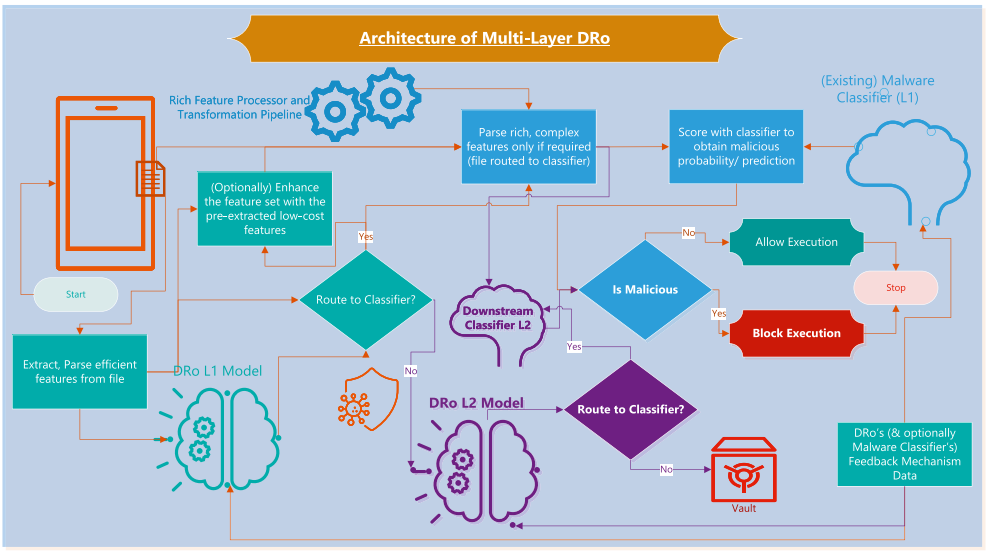


**Fig. 1.** Architecture of hierarchical (multi-routing) DRo.

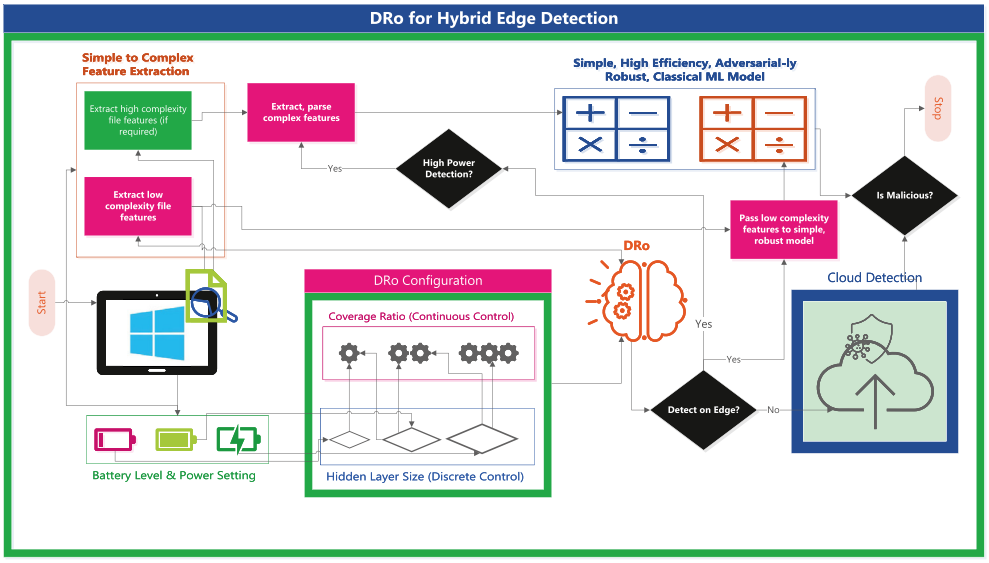


**Fig. 2.** Architecture of GreenForensics

### 3.4. Adaptive formulations for WinDRo

We use the GreenForensics architecture in the Windows context, to design a hybrid edge-cloud detection system, named WinDRo. The WinDRo system is based on GreenForensics architecture (section 3), is trained on Windows PE dataset (section 4), and the different DRo configurations with varying discrete and continuous battery-performance optimization control dimensions (section 4.1). This solution is illustrated in Fig. 3. For this, we take 2 inputs, one is the *Performance Mode* set by the user for their device (e.g., Windows Laptop), second is the battery level of the device. Where the first adaptive control input is purely discreet, the second option is largely based on continuous inputs but may use discrete configurations. Based on the *Performance Mode*, a suitable (pre-compiled) architecture of the DRo model is loaded into the model. All of these architecture are single layer ones to offer high efficiency and low model footprint (a function of trainable parameters and computations), and differs only in the size of the single hidden layer. Next, based on the battery level, an equally performant configuration, but with varying coverage ratio is activated. Based on the configuration parameters ($\lambda$, $\mu$), the neuron connection weights may also change, and hence these are also modified without re-compiling the model or changing its architecture. For lower battery levels, a configuration with lower edge-detection coverage/ forensics is loaded and vice-versa.

## 4. Dataset and baseline models

We use the malware from the overly popular Malicia dataset (Nappa et al., 2015). One reason for choosing this highly popular dataset is that many benchmarks exist on this dataset, and these are so high that it is considered futile to break. The most recent and the highest benchmark claimed on this dataset using e2e DL architectures is by Sewak et al. (2018). They obtained an accuracy of 99.21% at an FPR of 0.2% while using a 3-layer Auto-Encoder, coupled with a 4-layer MLP-DNN. The features used were the frequency vector of

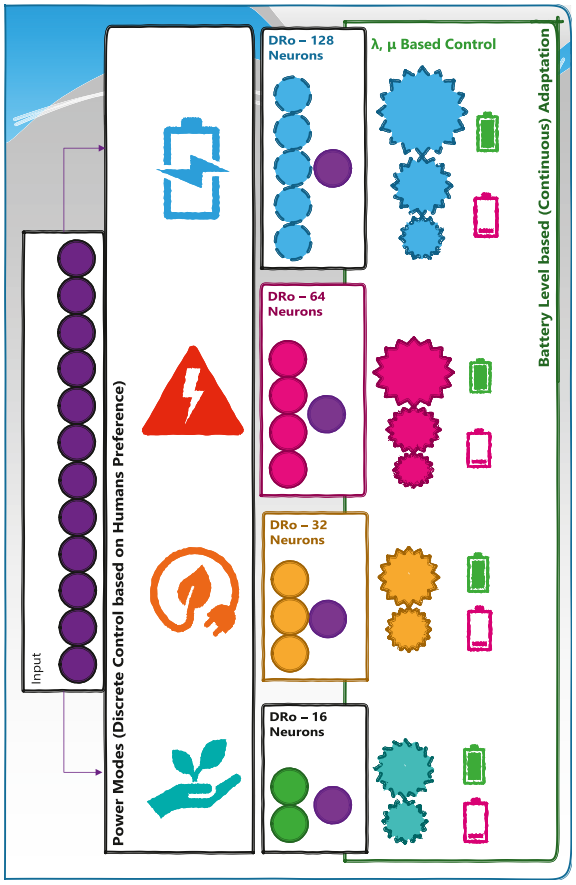


**Fig. 3.** Adaptive Configurations for DRo with adaptive Detection:Forensics coverage ratio.

the different opcodes present in the assembly of the respective files. Sewak et al. however used a sub-sample of Malicia for creating a balanced dataset. But in practical scenarios security datasets are not balanced, therefore we compute the baseline with an unbalanced Malicia dataset using a similar e2e Stacked Auto Encoder (SAE) and Multi-Layer Perceptron based Deep Neural Network (MLP-DNN) combinations (Sewak et al., 2020). This process is shown in Fig. 4 and the baseline results are presented in Table 1. Additionally, Table 2 indicates the neural complexity of these models, i.e. the neurons that need to be activated during the training and inference-ing/scoring for each input (across each pass). These should not be confused with the trainable parameters of a model, as those are a function of the weight-vectors which are in turn computed as a product of neurons across each subsequent layer, which for a multi-layer network is astronomical. Since the focus here is to arrive at the computation load during inference (and not necessarily model footprint during training), the number of activation that needs to be invoked during inference is a better indicator for this purpose in endpoint protection scenarios on hybrid edge devices.

The training neurons ($\mathcal{N}_{Tain}$) and inference ($\mathcal{N}-Inference$) neurons for the SAE and MLP architectures are computed as in equations (10)–(12) respectively, where all symbols have their standard meaning as per the SAE-MLP architecture in Fig. 5.

$$\mathcal{N}_{Train}^{SAE} = \mathcal{N}^{Encoder} + \mathcal{N}^{Embedding} + \mathcal{N}^{Decoder} = 2 \times \mathcal{N}^{Encoder} + \mathcal{N}^{Embedding} \quad (10)$$

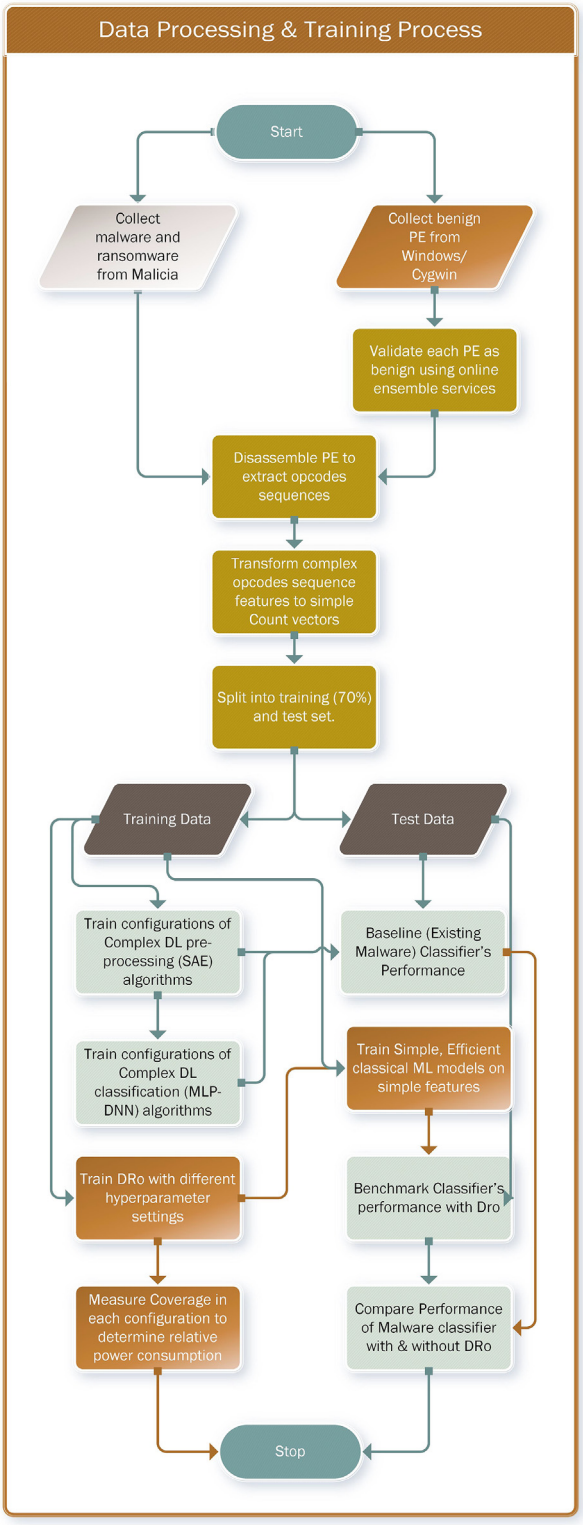


**Fig. 4.** Data processing and training process.

$$\mathcal{N}_{Inference}^{SAE} = \mathcal{N}^{Encoder} + \mathcal{N}^{Embedding} \quad (11)$$

**Table 1**
Baseline Configurations and performance.

| C. | $Layers_{Encoder}$ | $D_{Enc}$ | $D^{MLP}_{Hidden}$ | AUC | Acc. | FPR |
|---|---|---|---|---|---|---|
| **1** | [256] | 64 | [64] | 0.5 | **0.881** | 1 |
| 2 | [256] | 128 | [128] | 0.476 | 0.777 | 0.919 |
| 3 | [256, 128] | 64 | [64, 64] | 0.476 | 0.834 | 0.994 |
| 4 | [512, 256] | 128 | [128, 64] | 0.562 | 0.804 | 0.756 |
| **5** | [512, 256, 128] | 64 | [64, 64, 64] | **0.715** | 0.783 | **0.375** |
| 6 | [1024, 512, 256] | 128 | [128, 64, 64] | 0.509 | 0.874 | 0.969 |
| 7 | [1024, 512, 256, 128] | 64 | [64, 64, 64, 64] | 0.561 | 0.855 | 0.825 |
| 8 | [1024, 512, 256, 128] | 64 | [64, 64, 32, 32] | 0.538 | 0.872 | 0.9 |

**Table 2**
Neural (activation) complexity of the baseline Configurations.

| C. | $N_{Encoder}$ | $N_{MLP}$ | $N_{Training}$ | $N_{Inferencing}$ |
|---|---|---|---|---|
| 1 | 256 | 64 | 1840 | 385 |
| 2 | 256 | 128 | 1968 | 513 |
| 3 | 384 | 128 | 2160 | 577 |
| 4 | 768 | 192 | 3056 | 1089 |
| **5** | 896 | 192 | **3248** | **1153** |
| 6 | 1792 | 256 | 5168 | 2177 |
| 7 | 1920 | 256 | 5360 | 2241 |
| 8 | 1920 | 192 | 5296 | 2177 |

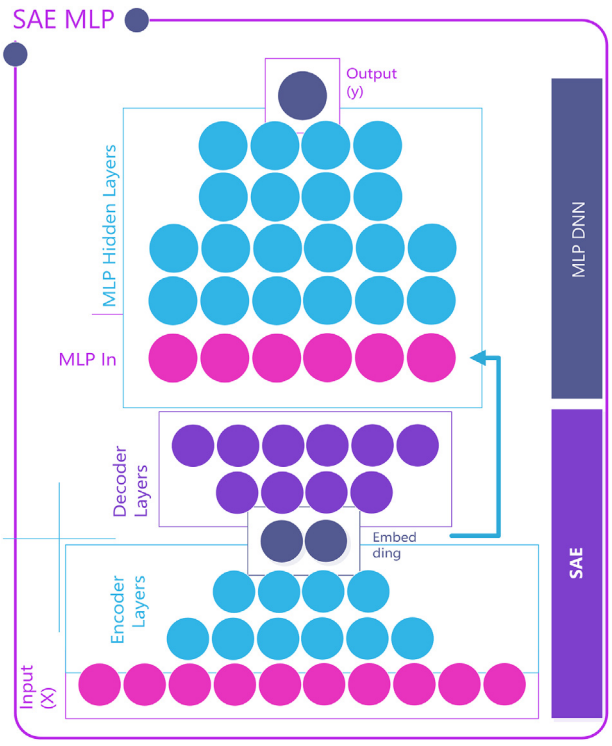


**Fig. 5.** Existing (baseline) DL architecture.

$$\mathcal{N}^{MLP}_{Train.Inference} = \sum_{HiddenLayer=1}^{L} \mathcal{N}^{HiddenLayer} + \mathcal{N}^{Output} = \sum_{HiddenLayer=1}^{L} \mathcal{N}^{HiddenLayer} + 1 \quad (12)$$

All configurations were trained for 1000 epochs and with a batch size of 2048 to eliminate the training pass bias across all baselines and benchmarks. The baseline configuration 5 generated the best results of 78.3% accuracy with 0.375 FPR, and had an $N^{SAE}_{Inference}$ load of 1153.

### 4.1. GreenForensics benchmark results

Next, we experiment with more than 100 configurations of DRo, but in 4 different groups. The first 3 group represents one of the 'Power Mode' settings for the Windows table and battery, and the 4th group represents the state of being active charged (by AC adapter) for the device. As shown in the architecture for GreenForensics in Fig. 2, instead of deploying the complete hierarchical DRo model on the edge, we deploy only the 'Definitive' route of the DRo on the device and the remaining hierarchy could be deployed on the cloud or replaced by more complex dynamic analysis. Therefore, the benchmarks here refers to the performance of the part architecture deployed on the edge device.

All configurations here comprise of a single layer MLP, and same $\mathcal{N}^{HiddenLayer}$ in a group. This ensure that the neural complexity is homogeneous for all configurations in a group and is given by equation (12). The neural complexity of the configuration represents the discrete control as described in section 3.3, whereas the coverage (records routed to be processed on edge) provides the aspect of continuous control as per the architecture. From each group we select some configurations with similar performance, but with varying proportion of edge-overage. The continuous coverage is varied as per the battery level of the device.

Table 3 shows the benchmark configurations for the 'Best Power Efficiency' Power Mode settings. This has the least (16) neurons in a single hidden layer. From these we select the Conf. 10, and 15 which have perfect performance (100% accuracy and 0.00 FPR), and use the varying coverage across these to alter the proportion of samples for edge-based detection or subsequent forensics in real time bases on battery status. As compared to the best baseline configuration, Conf. 5, both of these configurations demonstrate an improvement in robustness by up to **100%** (no FPRs) and performance (AUC) by up to ≈**40**% (100 × ((1.0/0.715) − 1)%). This gains further significance as the baseline DL architecture in comparison had up to ≈**6700**% higher neural inference complexity (100 × ((1153/17) − 1)%). All things remaining same, the uplift is attributable to the sampling done by the DRo algorithm for both the training and the inferencing (test/validation).

Table 4 shows the benchmark configurations for the 'Balanced' Power Mode settings. This has the next higher, 32 neurons in a single hidden layer. From these we select the Conf. 34, and 45 which again have perfect performance, and similarly use the varying coverage across these to alter the proportion of samples for edge-based detection or subsequent forensics in real time bases on battery status.

Table 5 shows the benchmark configurations for the 'Best Performance' Power Mode settings. This has the next higher, 64

**Table 3**
Edge 'best power efficiency' power mode Configurations.

| C. | $D^{MLP}_{Hidden}$ | $\lambda$ | $\mu$ | Acc. | FPR | AUC | Coverage | $N_{Total}$ |
|---|---|---|---|---|---|---|---|---|
| 1 | 16 | 0.01 | 3 | 0.99 | 0.021 | 0.997 | 0.663 | 17 |
| **2** | 16 | 0.01 | 4 | **0.996** | **0.032** | **1.0** | **0.335** | 17 |
| 3 | 16 | 0.01 | 5 | 0.988 | 0.031 | 0.998 | 0.539 | 17 |
| 4 | 16 | 0.01 | 6 | 0.997 | 0.034 | 0.999 | 0.546 | 17 |
| 5 | 16 | 0.01 | 7 | 0.984 | 0.041 | 0.997 | 0.525 | 17 |
| 6 | 16 | 0.05 | 3 | 0.992 | 0.05 | 0.996 | 0.531 | 17 |
| 7 | 16 | 0.05 | 4 | 0.983 | 0.031 | 0.995 | 0.446 | 17 |
| 8 | 16 | 0.05 | 5 | 0.986 | 0.019 | 0.996 | 0.462 | 17 |
| 9 | 16 | 0.05 | 6 | 0.984 | 0.099 | 0.998 | 0.478 | 17 |
| **10** | 16 | 0.05 | 7 | **0.998** | 0 | **1.0** | **0.496** | 17 |
| 11 | 16 | 0.1 | 3 | 1 | 0 | 0 | 0.527 | 17 |
| 12 | 16 | 0.1 | 4 | 0.99 | 0.096 | 0.987 | 0.503 | 17 |
| 13 | 16 | 0.1 | 5 | 0.984 | 0.031 | 0.996 | 0.472 | 17 |
| 14 | 16 | 0.1 | 6 | 0.99 | 0.079 | 0.989 | 0.507 | 17 |
| **15** | 16 | 0.1 | 7 | 0.999 | 0 | 1.0 | **0.525** | 17 |

**Table 4**
Edge 'balanced' power mode Configurations.

| C. | $D_{Hidden}^{MLP}$ | $\lambda$ | $\mu$ | Acc. | FPR | Coverage | $N_{Total}$ |
|---|---|---|---|---|---|---|---|
| 16 | 32 | 0.01 | 3 | 0.991 | 0.047 | 0.489 | 33 |
| 17 | 32 | 0.01 | 4 | 0.982 | 0.019 | 0.413 | 33 |
| 18 | 32 | 0.01 | 5 | 0.983 | 0.02 | 0.403 | 33 |
| 19 | 32 | 0.01 | 6 | 0.994 | 0.071 | 0.362 | 33 |
| 20 | 32 | 0.01 | 7 | 0.984 | 0.019 | 0.42 | 33 |
| 21 | 32 | 0.05 | 3 | 0.99 | 0.033 | 0.528 | 33 |
| 22 | 32 | 0.05 | 4 | 0.994 | 0.01 | 0.49 | 33 |
| 23 | 32 | 0.05 | 5 | 0.993 | 0.022 | 0.53 | 33 |
| 24 | 32 | 0.05 | 6 | 0.979 | 0.038 | 0.467 | 33 |
| 25 | 32 | 0.05 | 7 | 0.989 | 0.102 | 0.49 | 33 |
| 26 | 32 | 0.1 | 2 | 0.993 | 0 | 0.454 | 33 |
| 27 | 32 | 0.1 | 3 | 0.999 | 0 | 0.557 | 33 |
| 28 | 32 | 0.1 | 4 | 0.997 | 0.036 | 0.482 | 33 |
| 29 | 32 | 0.1 | 5 | 0.982 | 0.038 | 0.455 | 33 |
| 30 | 32 | 0.1 | 6 | 0.991 | 0.075 | 0.503 | 33 |
| 31 | 32 | 0.1 | 7 | 0.991 | 0.031 | 0.502 | 33 |
| 32 | 32 | 0.2 | 3 | 0.982 | 0.031 | 0.452 | 33 |
| 33 | 32 | 0.2 | 4 | 0.984 | 0.031 | 0.466 | 33 |
| **34** | 32 | 0.2 | 5 | **1** | **0** | **0.534** | 33 |
| 35 | 32 | 0.2 | 6 | 0.986 | 0.025 | 0.466 | 33 |
| 36 | 32 | 0.3 | 3 | 0.99 | 0.162 | 0.536 | 33 |
| 37 | 32 | 0.3 | 4 | 0.985 | 0.019 | 0.45 | 33 |
| 38 | 32 | 0.3 | 5 | 0.993 | 0.059 | 0.507 | 33 |
| 39 | 32 | 0.3 | 6 | 1 | 0 | 0.539 | 33 |
| 40 | 32 | 0.4 | 3 | 0.983 | 0.025 | 0.447 | 33 |
| 41 | 32 | 0.4 | 4 | 1 | 0 | 0.55 | 33 |
| 42 | 32 | 0.4 | 5 | 0.982 | 0.038 | 0.458 | 33 |
| 43 | 32 | 0.4 | 6 | 0.98 | 0.038 | 0.478 | 33 |
| 44 | 32 | 0.5 | 3 | 0.982 | 0.025 | 0.448 | 33 |
| **45** | 32 | 0.5 | 4 | **1** | **0** | 0.551 | 33 |
| 46 | 32 | 0.5 | 5 | 0.999 | 1 | 0.543 | 33 |

**Table 5**
Edge 'best performance' power mode Configurations.

| C. | $D_{Hidden}^{MLP}$ | $\lambda$ | $\mu$ | Acc. | FPR | Coverage | $N_{Total}$ |
|---|---|---|---|---|---|---|---|
| 48 | 64 | 0.01 | 3 | 0.979 | 0.031 | 0.42 | 65 |
| 49 | 64 | 0.01 | 4 | 0.995 | 0 | 0.545 | 65 |
| 50 | 64 | 0.01 | 5 | 0.987 | 0.019 | 0.446 | 65 |
| 51 | 64 | 0.01 | 6 | 0.992 | 0.024 | 0.478 | 65 |
| 52 | 64 | 0.01 | 7 | 0.984 | 0.019 | 0.411 | 65 |
| 53 | 64 | 0.05 | 3 | 1 | 0 | 0.551 | 65 |
| 54 | 64 | 0.05 | 4 | 0.992 | 0.032 | 0.479 | 65 |
| 55 | 64 | 0.05 | 5 | 0.987 | 0.041 | 0.518 | 65 |
| 56 | 64 | 0.05 | 6 | 0.991 | 0.046 | 0.486 | 65 |
| 57 | 64 | 0.05 | 7 | 0.981 | 0.031 | 0.463 | 65 |
| 58 | 64 | 0.1 | 2 | 0.987 | 0.07 | 0.469 | 65 |
| 59 | 64 | 0.1 | 3 | 0.988 | 0.07 | 0.482 | 65 |
| 60 | 64 | 0.1 | 4 | 0.99 | 0.039 | 0.539 | 65 |
| 61 | 64 | 0.1 | 5 | 0.989 | 0.07 | 0.525 | 65 |
| 62 | 64 | 0.1 | 6 | 0.985 | 0.025 | 0.46 | 65 |
| **63** | 64 | 0.1 | 7 | **1** | **0** | **0.539** | 65 |
| 64 | 64 | 0.2 | 3 | 1 | 0 | 0.549 | 65 |
| 65 | 64 | 0.2 | 4 | 0.991 | 0.024 | 0.516 | 65 |
| 66 | 64 | 0.2 | 5 | 0.997 | 1 | 0.522 | 65 |
| 67 | 64 | 0.2 | 6 | 0.983 | 0.031 | 0.496 | 65 |
| 68 | 64 | 0.3 | 3 | 0.983 | 0.025 | 0.444 | 65 |
| 69 | 64 | 0.3 | 4 | 0.984 | 0.119 | 0.513 | 65 |
| 70 | 64 | 0.3 | 5 | 0.991 | 0.07 | 0.495 | 65 |
| 71 | 64 | 0.3 | 6 | 0.986 | 0.059 | 0.492 | 65 |
| 72 | 64 | 0.4 | 3 | 0.999 | 1 | 0.562 | 65 |
| 73 | 64 | 0.4 | 4 | 0.98 | 0.038 | 0.45 | 65 |
| 74 | 64 | 0.4 | 5 | 0.988 | 0.071 | 0.505 | 65 |
| 75 | 64 | 0.4 | 6 | 1 | 0 | 0.538 | 65 |
| 76 | 64 | 0.5 | 3 | 0.985 | 0.025 | 0.447 | 65 |
| 77 | 64 | 0.5 | 4 | 0.999 | 1 | 0.564 | 65 |
| **78** | 64 | 0.5 | 5 | **1** | **0** | **0.55** | 65 |
| 79 | 64 | 0.5 | 6 | 1 | 0 | 0.541 | 65 |

neurons in a single hidden layer. From these we select the Conf. 63, and 78 for similar reasons and purpose. Here, there are multiple configurations with similarly higher coverage (at perfect performance), so we chose the one which has the highest regularization ($\lambda$).

Finally, Table 6 shows the benchmark configurations for the 'Charging' State settings. This has the next highest, 128 neurons in a single hidden layer. From these we select the Conf. 88, and 111 for similar reasons and purpose.

## 5. Conclusion

Deep Learning, though is enormously powerful, but to learn non-linear representations from noisy data, these algorithms invariably become overly complex and require expensive training and scoring. Such architectures are not suitable for edge and hybrid devices and network payloads. DRo, a novel representation learning algorithm, based on deep clustering solves many of these problems. We further improved DRo, to design GreenForensics, a mechanism that is more suitable for battery-performance optimizations aware devices and is also immune to adversarial-DL attacks. The GreenForensics mechanism not only improves upon the detection performance (by up to 40%) and robustness (by up to 10%) of even the best of DL architectures, but is several times (67X) more efficient to infer on edge devices and network payloads. Despite its simplistic model design, GreenForensics surpasses the performance of all existing Machine and Deep Learning algorithms and architecture, on an extremely popular dataset, on which for several years it was assumed that peak performance has been reached, and no improvement in benchmarks could be claimed. With GreenForensics we were able to design a powerful, efficient, and robust hybrid edge-cloud suitable malware detection system for devices

**Table 6**
Edge 'connected/charging' state Configurations.

| C. | $D_{Hidden}^{MLP}$ | $\lambda$ | $\mu$ | Acc. | FPR | Coverage | $N_{Total}$ |
|---|---|---|---|---|---|---|---|
| 80 | 128 | 0.01 | 3 | 0.997 | 0 | 0.564 | 129 |
| 81 | 128 | 0.01 | 4 | 0.981 | 0.025 | 0.433 | 129 |
| 82 | 128 | 0.01 | 5 | 0.999 | 0 | 0.518 | 129 |
| 83 | 128 | 0.01 | 6 | 0.995 | 0.03 | 0.478 | 129 |
| 84 | 128 | 0.01 | 7 | 0.982 | 0.031 | 0.448 | 129 |
| 85 | 128 | 0.05 | 3 | 0.978 | 0.031 | 0.436 | 129 |
| 86 | 128 | 0.05 | 4 | 0.984 | 0.067 | 0.511 | 129 |
| 87 | 128 | 0.05 | 5 | 0.993 | 0.061 | 0.455 | 129 |
| **88** | 128 | 0.05 | 6 | **1** | **0** | **0.503** | 129 |
| 89 | 128 | 0.05 | 7 | 0.997 | 0 | 0.507 | 129 |
| 90 | 128 | 0.1 | 2 | 0.992 | 0.04 | 0.47 | 129 |
| 91 | 128 | 0.1 | 3 | 0.999 | 0 | 0.563 | 129 |
| 92 | 128 | 0.1 | 4 | 0.992 | 0.025 | 0.531 | 129 |
| 93 | 128 | 0.1 | 5 | 0.98 | 0.031 | 0.452 | 129 |
| 94 | 128 | 0.1 | 6 | 0.988 | 0.051 | 0.509 | 129 |
| 95 | 128 | 0.1 | 7 | 0.982 | 0.038 | 0.459 | 129 |
| 96 | 128 | 0.2 | 3 | 0.978 | 0.044 | 0.439 | 129 |
| 97 | 128 | 0.2 | 4 | 0.995 | 0.025 | 0.481 | 129 |
| 98 | 128 | 0.2 | 5 | 0.99 | 0.093 | 0.508 | 129 |
| 99 | 128 | 0.2 | 6 | 0.992 | 0.019 | 0.485 | 129 |
| 100 | 128 | 0.3 | 3 | 0.987 | 0.04 | 0.519 | 129 |
| 101 | 128 | 0.3 | 4 | 0.978 | 0.044 | 0.44 | 129 |
| 102 | 128 | 0.3 | 5 | 0.99 | 0.079 | 0.522 | 129 |
| 103 | 128 | 0.3 | 6 | 0.979 | 0.044 | 0.451 | 129 |
| 104 | 128 | 0.4 | 3 | 0.981 | 0.031 | 0.439 | 129 |
| 105 | 128 | 0.4 | 4 | 0.999 | 0 | 0.562 | 129 |
| 106 | 128 | 0.4 | 5 | 0.991 | 0.031 | 0.487 | 129 |
| 107 | 128 | 0.4 | 6 | 0.98 | 0.038 | 0.451 | 129 |
| 108 | 128 | 0.5 | 3 | 0.979 | 0.025 | 0.429 | 129 |
| 109 | 128 | 0.5 | 4 | 0.991 | 0.054 | 0.504 | 129 |
| 110 | 128 | 0.5 | 5 | 0.983 | 0.025 | 0.448 | 129 |
| **111** | 128 | 0.5 | 6 | **1** | **0** | **0.548** | 129 |

like Windows tablets and laptops. The GreenForensics mechanism based WinDRo architecture could dynamically alter the coverage of the detection at edge based upon device's power settings mandate and also being cognizant of the real-time battery status of the device. Therefore, GreenForensics, provides the best of all worlds for a hybrid edge-cloud detection for battery-power balance optimized devices.

## References


Birman, Y., Hindi, S., Katz, G., Shabtai, A., 2020. Cost-effective malware detection as a service over serverless cloud using deep reinforcement learning. In: 2020 20th IEEE/ACM International Symposium on Cluster, Cloud and Internet Computing. CCGRID), pp. 420–429.

Bridle, J.S., Heading, A.J.R., MacKay, D.J.C., 1992. Unsupervised classifiers, mutual information and 'phantom targets. In: Advances in Neural Information Processing Systems, vol. 4. Morgan-Kaufmann, pp. 1096–1101.

Cover, T.M., Thomas, J.A., 2006. Elements of Information Theory (Wiley Series in Telecommunications and Signal Processing). Wiley-Interscience, USA.

Dosovitskiy, A., Springenberg, J.T., Riedmiller, M., Brox, T., 2014. Discriminative unsupervised feature learning with convolutional neural networks. In: Advances in Neural Information Processing Systems, vol. 27. Curran Associates, Inc., pp. 766–774

Hartigan, J.A., Wong, M.A., 1979. Algorithm as 136: a k-means clustering algorithm. Journal of the royal statistical society. series c (applied statistics) 28 (1), 100–108. https://doi.org/10.2307/2346830.

Hogan-Burney, A., 2021. How cyberattacks are changing according to new microsoft digital defense report. www.microsoft.com/security/blog/2021/10/11/how-cyberattacks-are-changing-according-to-new-microsoft-digital-defense-report/. (Accessed 22 October 2021).

Hu, W., Miyato, T., Tokui, S., Matsumoto, E., Sugiyama, M., 2017. Learning discrete representations via information maximizing self-augmented training. In: 34th International Conference on Machine Learning, vol. 70, pp. 1558–1567. ICML'17.

Jiang, Z., Zheng, Y., Tan, H., Tang, B., Zhou, H., 2017. Variational deep embedding: an unsupervised and generative approach to clustering. In: 26th International Joint Conference on Artificial Intelligence, IJCAI'17. AAAI Press, pp. 1965–1972.

Johnson, S.C., 1967. Hierarchical clustering schemes. Psychometrika 32 (3), 241–254. https://doi.org/10.1007/BF02289588.

Krause, A., Perona, P., Gomes, R.G., 2010. Discriminative clustering by regularized information maximization. In: Advances in Neural Information Processing Systems, vol. 23. Curran Associates, Inc., pp. 775–783

Kulis, B., Darrell, T., 2009. Learning to hash with binary reconstructive embeddings. In: Advances in Neural Information Processing Systems, vol. 22. Curran Associates, Inc., pp. 1042–1050

Leen, T.K., 1995. From data distributions to regularization in invariant learning. Neural Comput. 7 (5), 974–981.

Lopez-Martin, M., Carro, B., Sanchez-Esguevillas, A., 2020. Application of deep reinforcement learning to intrusion detection for supervised problems. Expert Syst. Appl. 141, 112963.

Miyato, T., ichi Maeda, S., Koyama, M., Nakae, K., Ishii, S., 2015. Distributional Smoothing with Virtual Adversarial Training arXiv:1507.00677.

Nappa, A., Rafique, M.Z., Caballero, J., 2015. The malicia dataset: identification and analysis of drive-by download operations. Int. J. Inf. Secur. 14 (1), 15–33.

Rathore, H., Samavedhi, A., Sahay, S.K., Sewak, M., 2021. Robust malware detection models: learning from adversarial attacks and defenses. Forensic Sci. Int.: Digit. Invest. 37, 301183.

Sajjadi, M., Javanmardi, M., Tasdizen, T., 2016. Regularization with stochastic transformations and perturbations for deep semi-supervised learning. In: Advances in Neural Information Processing Systems, vol. 29. Curran Associates, Inc., pp. 1163–1171

Sewak, M., Sahay, S.K., Rathore, H., 2018. An investigation of a deep learning based malware detection system. In: Proceedings of the 13th International Conference on Availability, Reliability and Security. ARES 2018, Association for Computing Machinery. https://doi.org/10.1145/3230833.3230835, 26:1–26:5.

Sewak, M., Sahay, S.K., Rathore, H., 2020. An overview of deep learning architecture of deep neural networks and autoencoders. J. Comput. Theor. Nanosci. 17 (1), 182–188. https://doi.org/10.1166/jctn.2020.8648.

Sewak, M., Sahay, S.K., Rathore, H., Dro, 2021. A data-scarce mechanism to revolutionize the performance of dl-based security systems. In: 2021 IEEE 46th Conference on Local Computer Networks (LCN). IEEE, pp. 581–588. https://doi.org/10.1109/LCN52139.2021.9524929.

Sewak, M., Sahay, S.K., Rathore, H., Drldo, 2021. A novel drl based de-obfuscation system for defence against metamorphic malware. Defence Sci. J. 71 (1), 55–65.

Wan, X., Sheng, G., Li, Y., Xiao, L., Du, X., 2017. Reinforcement learning based mobile offloading for cloud-based malware detection. In: GLOBECOM 2017 - 2017 IEEE Global Communications Conference, pp. 1–6.

Weiss, Y., Torralba, A., Fergus, R., 2009. Spectral hashing. In: Advances in Neural Information Processing Systems, vol. 21. Curran Associates, Inc., pp. 1753–1760

Wiseman, B., 2021. The next great disruption is hybrid work - are we ready? www.microsoft.com/en-us/worklab/work-trend-index/hybrid-work. (Accessed 22 October 2021).

Xie, J., Girshick, R., Farhadi, A., 2016. Unsupervised deep embedding for clustering analysis. In: 33rd International Conference on Machine Learning, vol. 48. of Proceedings of Machine Learning Research, pp. 478–487.

Xu, L., Neufeld, J., Larson, B., Schuurmans, D., 2005. Maximum margin clustering. In: Advances in Neural Information Processing Systems, vol. 17. MIT Press, pp. 1537–1544.