Neural AutoForensics: Comparing Neural Sample Search and Neural Architecture Search for Malware Detection and Forensics

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) 301444

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

Neural AutoForensics: Comparing Neural Sample Search and Neural Architecture Search for malware detection and forensics

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:
Malware detection and forensics
AutoDL
Neural architecture search
Neural sample search
Edge ML

ABSTRACT

Neural Architecture Search (NAS) desired to bring Machine Learning to the common masses. But ironically, because of its high-resources requirements, it remained exclusive to the elite. After several efficiency enhancements, its most efficient version (ENAS), found a place across some commonly used Deep Learning libraries, but it still could not gain mass popularity. Especially in the field of malware forensics, there exists no popular implementation of NAS. AutoML, as it stands today, comprises NAS and hyperparameter tuning as sub-domains. But both from effort and impact perspectives, the data dimension has 80% weight in an ML problem, but still, the data dimension of ML is currently missing from AutoML. In forensics, optimal sample discovery may have more impact than an optimal model discovery. Therefore, in this paper, we propose Neural Sample Search (NSS) using DRo, to comprise the data discovery dimension in AutoML. Further, we prove that, for malware forensics, NSS outperforms all expert-curated and NAS-suggested models by an exceptionally large margin. This gains further significance, as the baseline expert model had over *6700%* higher neural inference complexity than the NSS model, and was curated with efforts of several forensic experts across several years to reach that performance level; and the Efficient-NAS model had (ironically) over *100,000%* higher neural inference complexity than the proposed NSS mechanism. With such high performance at such minimal model footprint and complexity that NSS brings, we can claim that by including NSS, AutoML can *truly* be ready for mass adoption in the field of malware forensics.

1. Introduction

Recently a novel algorithm named DRo (Sewak et al., 2021) (**D**eep **Ro**uter) was proposed for the purpose of efficient malware detection and forensics on edge devices and it offered many end-to-end (e2e) Deep Learning (DL) architectures for improving the performance of a Machine Learning (ML) malware detection and forensics system under different extreme data and resource-scarce scenarios. Neural Sample Search (NSS) refers to the science of using neural architectures to sample the dataset to improve the performance of an existing model. Therefore, in simple terms, DRo performs the function of NSS and finds the optimal samples to train and infer from a breach detection/forensics system. The promising experimental results showed that DRo-NSS could significantly uplift the performance of a DL based malware forensics system even under extreme resource and data-scarce scenarios. The DRo-NSS algorithm uses recent advancements in Deep Clustering (DC) and proposes further algorithmic and architectural enhancements to design a system that could selectively sample data for training (any) DL/ML malware detection system. It was demonstrated that using the data samples suggested by DRo, any DL based malware detection, and analysis system could outperform all other DL/ML based systems that are trained on the entire corpus of the breach/malware data. DRo could similarly also sample inference data to provide superior forensic candidate identification accuracy at a reduced false-positive rate (FPR).

In this paper, we compare DRo based NSS (DRo-NSS) with existing state-of-the-art (SOTA) in AutoML/AutoDL literature. For this purpose, we benchmark the DRo-NSS based breach detection system against a similar custom-developed system that uses Auto-PyTorch (Tabular) (Zimmer et al., 2021) based detection models. Auto-PyTorch is one of the most popular AutoDL libraries and uses customization of the Efficient-NAS (ENAS) (Pham et al., 2018) mechanism for tabular data in the Neural Architecture Search (NAS) space, along with enhancements for auto hyperparameter tuning (Mendoza et al., 2016) (multi-fidelity system) for structured data. Although an approach that uses a SOTA NAS algorithm for

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

https://doi.org/10.1016/j.fsidi.2022.301444
2666-2817

determining an optimal architecture, followed by using another SOTA approach for hyperparameter-tuning algorithm could have also been proposed to benchmark the SOTA in AutoML, such an approach would have been oblivious to the interaction effect between the architecture and the hyperparameters and hence are known to cause instability (Zela et al., 2020). Therefore, we chose *Auto-PyTorch Tabular* for this purpose. This is because other similar AutoML mechanisms either do not offer multi-fidelity, or do not have proven benchmarks on tabular/structured data, or are not compatible with DL (AutoDL) architectures.

We compare both Multi-Fidelity-(E)NAS (combination of NAS and associated hyperparameter selection for AutoML) and NSS (using DRo-NSS), against the baseline results from (human) expert-curated results on a standardized multi-breach malware dataset (Malicia Malware Dataset (Nappa et al., 2015)). The reason for choosing cybersecurity forensics as the test domain is that:

1. First, DRo was originally proposed for malware detection systems.
2. Second, security forensics offers challenging and high-impact scenarios both in terms of performance and efficiency mandates.
3. Third, the data available in forensics is usually challenging, and relatively less explicable using human cognizance, and hence DL based feature extraction systems and DL based candidate forensic-sample selection system offers a high-impact application area.
4. Fourth, forensics is inherently a (labeled) data-scarce system, and hence training a DL based system (which is preferable owing to its auto feature extraction prowess) is not always possible (models may not converge), or may not be effective (offer sub-optimal performance) (Sewak et al., 2018a).

Experimental results show that on the standardized multi-breach malware and ransomware dataset, DRo-NSS could improve the detection robustness over the expert-baseline DL model by up to **100%** (0% FPR across multiple experiments) and detection performance by up to $\approx$ **40%**. This is significant, as:

1. The expert baseline was established using some of the best e2e DL architectures, which had up to $\approx$ **6700%** higher neural inference complexity as compared to DRo-NSS, and their established benchmarks had remained unchallenged by any other DL architecture, technique, or model since a long time. On the other hand, Multi-Fidelity-NAS proposed model was significantly more complex than all expert models and performed the worst.
2. Another advantage offered by DRo-NSS is that it could be implemented with both DL and ML models.
3. Third, and maybe the most important advantage of DRo-NSS, is that its internal model size could be varied to suit multiple deployments (and training) constraints.

This opens many new avenues for NSS currently untouched even by the most efficient implementations of NAS. This aspect is relevant not just for forensics-ML, but for the larger ML research, especially edge-ML and embedded-ML as well. To demonstrate this aspect of DRo-NSS, we experiment with multiple (adaptive) hybrid edge deployment scenarios for DRo-NSS with varying battery and complexity (model-footprint) constraints. Specifically, the adaptive inference scenarios covered are enumerated next.

- An extremely low performance, battery-operated endpoint; for example (budget) mobile endpoints, or IoT sensors.
- A low-performance endpoint (like a battery-powered endpoint on low battery); for example Windows Laptop with *Efficiency Mode* power setting or premium mobiles
- A low to medium performance, battery-powered endpoint; for example Windows Laptop with *Balanced Mode* power setting
- A medium performance, and battery-powered endpoint; for example Windows Laptop with *Best Performance Mode* power setting
- A medium performance, and connected endpoint; for example connected Windows (Consumer) Desktop with limited resources

We use (relatively) larger footprint models for each of the above scenarios for DRo-NSS, where even the largest DRo model was 67 $\times$ smaller than the expert model, which in turn is 47 $\times$ smaller than the Multi-Fidelity-NAS suggested model. Despite its extremely small footprints, each of the DRo-NSS models could outperform the expert model, which in turn had superior results than even the several 10−100 $\times$ times larger Multi-Fidelity-NAS suggested models. The performance benchmarks indicate that the DRo-NSS, despite having a 10000 $\times$ lower footprint than the AutoML model, can outperform not just multiple expert models, curated over several years, by multiple experts, but also even the best Multi-Fidelity-NAS model.

The remaining paper is organized as follows. In section 2, we cover the related work on deep clustering and sample search. Next, we cover the mathematics of DRo-NSS in section 3. Then we provide the details on the dataset, baseline and benchmark performance in section 4, and conclusion in section 5.

2. Related work

In this section, we first provide some prior arts which have (implicitly) attempted to do sample-search for different applications, especially in malware detection and forensics for edge detection. Next, we provide related work on the mathematical aspects of DRo, which is built on recent advances in the field of Deep Clustering.

2.1. Sample search and hybrid edge-cloud detection

The performance of any classifier is largely dependent on 2 factors, i.e. the learning algorithm, and the data that is used to learn. While there is an abundance of research in the Deep Learning domain on the different types of approximator architecture, the research on algorithms to sample effective training to train these models is unfortunately abysmal. Whatever work exists, use complex Deep Reinforcement Learning (DRL) algorithms and offer partial and black-box solutions to solve only the 'sampling for training' problems, 'sampling for inferencing' problem, or the 'complexity based routing' problem. 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).

2.2. Deep clustering

End-to-End Deep Learning for discrete representation learning is becoming popular in the 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, we 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 modeling 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, cannot efficiently scale to large datasets. Here is where DL, especially deep clustering, creates a large impact. Besides learning discrete data representation, the reason why DC became popular as a semi-supervised pre-processing approach for DL algorithms is that 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). Next, suggestions were made to make the 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 (Miyato et al., 2015), and Sajjadi (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 (Dosovitskiy et al., 2014) proposed to use data augmentation to model the invariance of learned representations for unsupervised algorithms like clustering. IMSAT (Hu et al., 2017) takes a similar approach but applies invariance directly to the learned representation instead of applying it on the surrogate classes, and directly learns discrete representations (clusters), instead of learning continuous representations that are later converted to discrete class representations or cluster predictions. Further, it combines Information Maximization (IM) along with SAT to produce more robust cluster formations; a special coefficient λ is used to balance between entropy loss and augmentation loss for learning cluster representations.

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).

3.1. About DRo

Sewak et al. (2021) used the ideas of Deep Clustering to solve a novel problem faced by most DL application domains, i.e. learning robust classification 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 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 inference-ing are both controlled by the same model, DRo enabled downstream DL discriminators to learn meaningful discriminative features effectively with little representative data. By 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, μ, that strikes 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 is a 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 a priori. 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 generates 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 2.2 for a discussion on other deep clustering algorithms), uses perturbation based SAT. Besides using the SAT penalties to provide regularization, IMSAT also uses 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 λ. 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, 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 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 hyperparameter 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 class in a data is expressed as in 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 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 $\mu \in$ (Pham et al., 2018; Zela et al., 2020) provides a decent range for default settings, and $\mu \in$ (Sewak et al., 2021; Birman et al., 2020) provides a decent range to influence recall and FPR across a wide range of applications.

3.3. Model and deployment architectures

The initial work on DRo was done on Android based system and corresponding datasets, and there was no comparison with other expert-curated or NAS based systems. We compare DRo-NSS with both expert-curated architectures, which we refer to as baseline models here, and the AutoML suggested models. The expert-curated baseline models used in this work (Sewak et al., 2018b) claim the highest performance to date on a very popular Windows malware dataset (Nappa et al., 2015). We use these models and the resulting performance as the baseline. The baseline model architecture is as shown in Fig. 2. Each of the models is a combination of a (stacked) auto encoder (AE) with 1, or 3 encoders, followed by an embedding and a similar number of decoder layers. The AE layers are followed by Multi-Layer Perceptron (MLP) based Deep Neural Network (DNN) layers. Each baseline model had 1, 4, or 7 layers. The model footprint in terms of the number of neurons for all models is provided in section 4.

The Multi-Fidelity-AutoML determined model is as shown in Fig. 3. The most optimal model for the data consists of 11 blocks (S1−S11). Each block consists of a *residual* sub-block and, optionally, a dense sub-block. In residual sub-block where the inputs are aggregated, in dense blocks, these are concatenated. Each sub-block could consist of one or more layers of varying size. This model consists of a total of **18,624** neurons, of which 12,602 were distributed across the residual layers and 6022 across dense layers. As compared to the baseline models, this model is 855% more complex than the largest model and 4837% more complex than the smallest model, and **1615%** more complex than the best performing baseline model.

Since DRo-NSS is adaptable, we could have different model architectures to suit the computational and efficiency constraints across different scenarios. As shown in Fig. 1, for each of the scenarios as listed in section 1, we use a similar model architecture, with a single dense (fully-connected) layer, but vary the number of neurons in that single hidden layer. Based upon the power-performance balance mandate of the system an appropriate architecture of DRo is loaded for the inference. Correspondingly, for each DRo configuration, we alter the hyperparameters (μ, λ) of DRo-

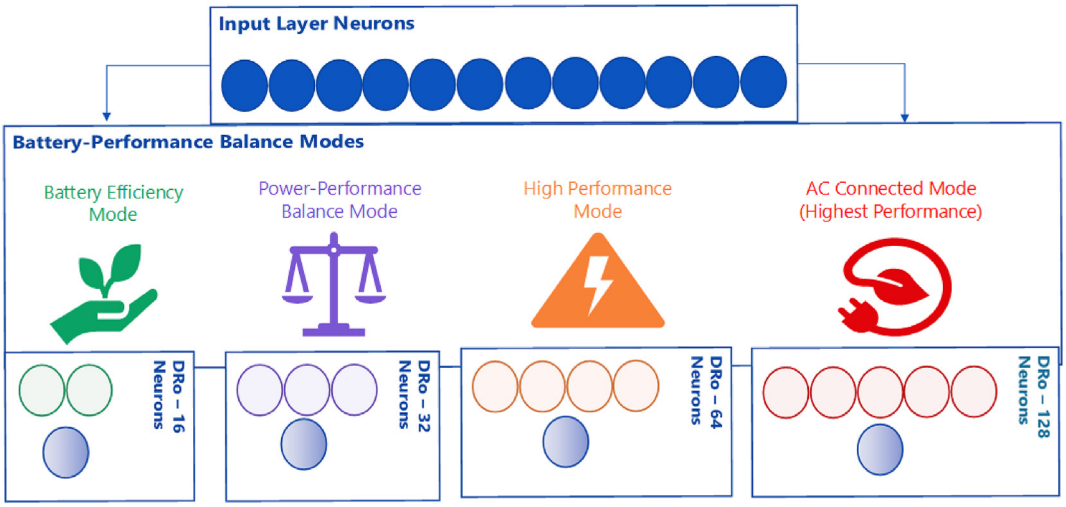

Fig. 1. Multiple efficiency adaptive DRo Configurations.

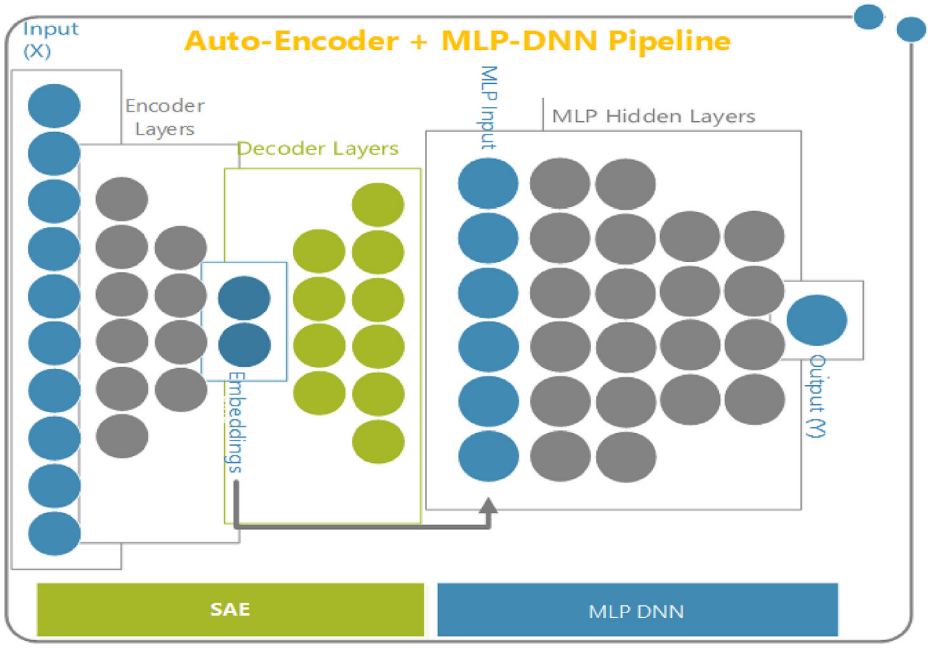

Fig. 2. E2E DL (Pipeline) Architecture of the baseline expert-curated models.

NSS architecture to search for the best samples to train that configuration (training sample search) optimally and also to vary the coverage for on-edge detection (inference sample search).

4. Baseline models and benchmarks

We use the malware from the overly popular Malicia dataset (Nappa et al., 2015). This is an old dataset, and multiple expert baselines exist on this dataset. The baselines model that claimed the highest performance (Sewak et al., 2018b) on this dataset using e2e DL was reevaluated under the standardized test conditions for our setup. The standardized configuration used a balanced dataset, where the malware (positive) samples came from the Malicia dataset and negative samples were extracted from standard Cygwin and Windows system libraries. We also use the same dataset in this work for a valid comparison. The reconstituted baselines are provided in Table 1. The best baseline model architecture 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*. 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).

The performance of the AutoML suggested architecture across multiple budgets is provided in Table 3. The model delivered very low accuracy of 24.8% at an unacceptable FPR of 0.994. Such a model could not be considered worthy of production in security forensics (or most other domains).

Table 4 shows the benchmark configurations across different required scenarios (Sc.). The model for Sc.1 has the least (16) neurons in a single hidden layer and has perfect performance ($\approx$100% accuracy and 0.00 FPR), and uses the varying coverage across these to alter the proportion of samples for edge-based detection in real-