DEMISe: Interpretable Deep Extraction and Mutual Information Selection Techniques for IoT Intrusion Detection

Luke R Parker
Defence Equipment and Support
Ministry of Defence
Bristol, UK
luke.parker890@mod.gov.uk

Paul D Yoo*
CSIS, Birkbeck College
University of London
London, UK
paul.d.yoo@ieee.org

Taufiq A Asyhari
School of Computing, Electronics and Mathematics
Coventry University
Coventry, UK
taufiq-a@ieee.org

Lounis Chermak
Centre for Electronic Warfare, Information and Cyber
Cranfield University
Shrivenham, UK
l.chermak@cranfield.ac.uk

Yoonchan Jhi
Security Research Team
Samsung SDS
Seoul, South Korea
yoonchan.jhi@samsung.com

Kamal Taha
ECE Dept
Khalifa University
Abu Dhabi, UAE
kamal.taha@kustar.ac.ae

ABSTRACT

Recent studies have proposed that traditional security technology – involving pattern-matching algorithms that check predefined pattern sets of intrusion signatures – should be replaced with sophisticated adaptive approaches that combine machine learning and behavioural analytics. However, machine learning is performance driven, and the high computational cost is incompatible with the limited computing power, memory capacity and energy resources of portable IoT-enabled devices. The convoluted nature of deep-structured machine learning means that such models also lack transparency and interpretability. The knowledge obtained by interpretable learners is critical in security software design. We therefore propose two novel models featuring a common Deep Extraction and Mutual Information Selection (DEMISe) element which extracts features using a deep-structured stacked autoencoder, prior to feature selection based on the amount of mutual information (MI) shared between each feature and the class label. An entropy-based tree wrapper is used to optimise the feature subsets identified by the DEMISe element, yielding the DEMISe with Tree Evaluation and Regression Detection (DETEReD) model. This affords 'white box' insight, and achieves a time to build of 603 seconds, a 99.07% detection rate, and 98.04% model accuracy. When tested against AWID, the best-referenced intrusion detection dataset, the new models achieved a test error comparable to or better than state-of-the-art machine-learning models, with a lower computational cost and higher levels of transparency and interpretability.

CCS CONCEPTS

• Security and Protection • Applications and Expert Systems • Algorithms

DOI: 10.1145/3339252.3340497

KEYWORDS

Security mobility applications, security of resource constrained devices, IoT, lightweight intrusion detection, feature engineering, mutual information, white-box modelling, deep learning.

1 Introduction

The Internet of Things (IoT) is an expanding network of devices that are predicted to become more mainstream as a result of their proliferation in the healthcare, retail, manufacturing and transportation markets [1,2]. The IoT comprises everyday devices with a degree of networked capability such that they provide an impression of intelligence [2], but they are neither mobile devices nor traditional computers [1]. As the IoT continues to grow, concerns have been raised over the potential misuse of IoT devices to illicitly obtain sensitive personal information (such as location, identity, payment details and health data) or as a means to attack far larger interconnected systems [1,2].

To tackle these security issues, perimeter security comprises preventive measures such as firewall, authentication and access control. Despite the central role of such measures in computer security, recent studies have shown that most computer security incidents are caused by insiders, i.e. people who would not be blocked by these preventative technologies because they often require access with significant privileges to do their daily jobs [3]. Prevention, although necessary, is not a complete security solution. Intrusion detection systems (IDS) not only complement preventive controls as the next line of defence, but also help to defend against the various threats to which networks and hosts are exposed. This is achieved by the network/host-level detection of attacker actions or tools based on misuse or heuristic detection techniques [4].

Most contemporary IDS techniques are not suitable for the IoT because they are not designed to handle the limited computing power, memory capacity and energy resources of portable IoT-enabled devices. Current IDS platforms involve pattern-matching algorithms that check predefined pattern sets consisting of intrusion signatures. More importantly, they only detect known attacks, so a signature must be created for every attack, and novel attacks cannot be detected. Recent studies have proposed that traditional security technology should be replaced with more sophisticated adaptive approaches that combine machine learning and behavioural analytics while minimising the use of computing power and energy.

Machine-learning systems are typically performance driven, and most non-parametric and model-free approaches therefore place high demands on computational resources to find global optima. Designing more accurate machine-learning algorithms to satisfy the market needs is therefore likely to increase the computational costs and energy demands even further.

Deep-learning algorithms are becoming more widespread, as seen in convolutional neural networks (CNNs) that combine feature extraction and classification. Deep-learning methods use feature engineering to achieve a better predictive performance than is possible using a single neural network model. However, these are computationally demanding resources especially when dealing with large-scale or highly-dimensional data. For example, CNNs gradually extract local features from high-resolution feature maps and then combine these features into more abstract feature maps of lower resolution [5]. This is realised by alternating convolution and subsampling layers. The last few layers in the CNN use fully connected multi-layered perceptron-based neural network classifiers to produce the abstracted results. Assuming Q input feature maps and R output feature maps, and a feature map size of $M \times N$, the convolutional kernel size is $K \times L$ and the computation in the convolution layer can be represented in a nested-loop description, as shown below.

```
for (r=0; r<R; r++)             //output feature map
 for (q=0; q<Q; q++)            //input feature map
   for (m=0; m<M; m++)          //row in feature map
     for (n=0; n<N; n++)        //column in feature map
       for (k=0; k<K; k++)      //row in convolution kernel
         for (l=0; l<L; l++)    //column in convolution kernel
           Y[r][m][n] += W[r][q][k][l] * X[q][m+k][n+l];
```

The array X contains the input feature maps, and the array Y contains the output feature maps, which are initialised to zeros. The array W contains the weights in the convolution kernels. The computational workload in the convolution layer only is in the order of $O(R \cdot Q \cdot M \cdot N \cdot K \cdot L)$, whereas the computational workload in the following subsampling layer is in the order of $O(Q \cdot M \cdot N)$. At the output of each layer, an activation function is applied to each vector in the feature maps to mimic neuron activation.

More importantly, the convoluted and complex nature of deep-structured machine learning can make the models difficult to interrogate and interpret, hence generally described as a 'black box'. In intrusion detection algorithm design, interpretability is a critical determinant of the practical utility of the resulting IDS. This not only ensures that the model is aligned with the problem addressed by the intrusion detection algorithm, but also allows the knowledge obtained by the interpretable learners to be used in other security software designs. Model interpretability is of tremendous importance to confirm that the model is performing its task as expected, thus creating trust with end-users.

Nowadays, there is a greater need to develop new lightweight and interpretable machine-learning models to cope with future demands that are in line with similar IoT-related initiatives. Efficient and interpretable modelling is important for cybersecurity, specifically intrusion detection problems that affect many industries. Accordingly, we here propose two machine-learning intrusion detection models for IoT devices which utilise a common Deep Extraction and Mutual Information Selection (DEMISe) element. Within the DEMISe element of the models, additional features are extracted using a two-layer deep-structured stacked autoencoder (SAE), and are then combined with the original features and ranked according to the amount of mutual information (MI) they share with the class label (normal or attack), using a MI theoretic feature selection filter. For resource-constrained IoT devices in which redundant features have already been identified, and MI ranking alone is sufficient to achieve an acceptable IDS performance, we propose the DEMISe model combined with a Radial Basis Function Classifier (DEMISe-RBFC). However, given that rank-based feature selection filters do not utilise machine learning to identify optimal feature subsets [6], a solely rank-based feature selection approach might select a feature subset that still retains some unnecessary dimensionality. Therefore, the proposed DEMISe-RBFC utilises a 'black box' classifier to afford a level of resilience to this additional dimensionality.

Recognising this limitation, we also propose the DEMISe with Tree Evaluation and Regression Detection (DETEReD) model, which utilises a C4.8 tree-based wrapper in a novel manner to identify optimal feature subsets among the top 10 features ranked by the MI theoretic filter. As a result of this subset optimisation, a lower complexity 'white box' logistic regression can be applied in the DETEReD model for instance classification, thus affording a deeper insight into the model's learning process. In terms of this paper's contribution, we will:

- Show MI theoretic feature selection can be effective on its own, but is best employed as a pre-selection filter prior to wrapper-based feature subset selection.
- Build upon current architectural descriptions [7,8,9] to show that a machine-learning IDS has the potential to be lightweight enough to be accommodated within an IoT device[1].

[1]Within this research we define lightweight as IoT devices which utilise some of the smallest microcontrollers available on the market, for example, the D1000 Intel Quark microcontroller with a 32MHz clock speed and 8KB of RAM.

- Build upon earlier work [10] to show the utility of features extracted using a deep SAE, which in conjunction with lightweight MI theoretic feature selection, is able to train a model that outperforms those proposed elsewhere [11,12].

The remainder of this paper is structured as follows. Section 2 introduces the DEMISe-RBFC and DETEReD models and justifies them in more detail, outlining the three key training methods. Section 3 analyses and evaluates the performance of the models, and Section 4 concludes our findings and recommends some areas of further work.

2 Methodology

Based on a selection of the 11 criteria proposed earlier [13], we selected the Aegean Wi-Fi Impersonation Attack Detection (AWID) dataset containing real traces of 802.11 traffic obtained from a dedicated WEP protected Wi-Fi network as the most representative dataset for the IoT environment, and thus the most appropriate dataset for model training and evaluation. We selected the reduced CLS portion of the AWID dataset because it serves as a useful starting point for preliminary research [12], and affords a baseline against which the DEMISe-RBFC and DETEReD models can be compared with the state-of-the-art Deep Feature Extraction and Selection (D-FES) method [10], and other methods using the same reduced CLS portion [11,12]. We pursued the 'divide and conquer' approach for DEMISe-RBFC and DETEReD model development, focusing only on the detection of impersonation attacks. Although flooding and injection attack signatures are also available within the AWID-CLS dataset, impersonation attacks were our focus as Hirte, Honeypot and EvilTwin impersonation attacks have previously been identified as the most severe threats to a network [12] and have been the focus of earlier research [10,11]. As a result, selecting impersonation attacks as the focus of our work allowed us to directly compare the performance of our model to others, a key weakness within the current body of machine-learning-based IoT IDS research. During pre-processing, the values for each instance within the dataset were normalised between zero and one to prevent bias in the machine learner towards features with the largest value ranges [14], and the entire dataset was balanced 50:50 in terms of the normal and impersonation attack instances[2].

Although a balanced dataset is not representative of a real-world network environment, balancing sacrifices some accuracy in terms of detecting the more prevalent class (in this case the normal traffic) in order to achieve greater detection accuracy for the minority class (*i.e.* the attack instances) [14]. Fig. 1 shows the high-level architecture of the DEMISe-RBFC model, and Fig. 2 shows the high-level architecture of the DETEReD model. The key elements of both models are now discussed in further detail.

[2]The balanced training dataset contained 48,522 normal and impersonation attack instances, with the testing dataset containing 20,079 normal and impersonation attack instances.

2.1 Deep Structured Stacked Autoencoder Based Feature Extraction

Although features can be extracted by principal component analysis (PCA), an autoencoder has a significant advantage over PCA in terms of its ability to utilise a non-linear activation (or transfer) function such as the sigmoid function (1) on an input variable x [14,15].

$$f(x) = \frac{1}{1 - e^{-x}} \tag{1}$$

An autoencoder is an unsupervised greedy learning algorithm within artificial neural network (ANN) family, but unlike many other ANNs, an autoencoder has an encoder-decoder architecture that attempts to learn a compressed approximation of the training data, such that it can efficiently predict the correct output for any given input [15].

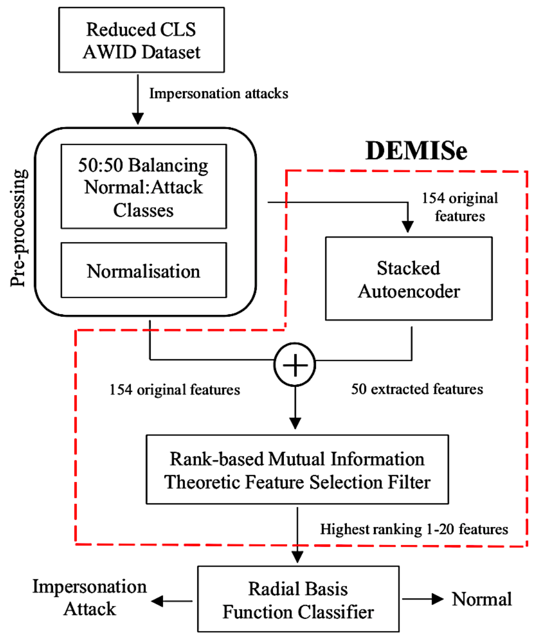

Figure. 1: High-level architecture of the DEMISe-RBFC model showing the core DEMISe element for feature extraction and selection, the Radial Basis Function Classifier, and the pre-processing stages.

Within an ANN such as an autoencoder, a neuron is a computational unit that takes an input $x_1, x_2, \ldots x_n$, and outputs a hypothesis $h_{W,b}(x)$, where the parameters W and b represent the weights and biases within the network that the learning algorithm adjusts to fit the training data [15]. Fig. 3 shows an autoencoder with only 50 neurons within its encoder layer to represent the original input of 154 features. As a result, each neuron within the encoder layer can be considered a newly constructed (or extracted) feature. Having learnt this compressed representation of the data, the decoder layer then attempts to use the learning within the encoder layer to reconstruct an output that resembles the original input as closely as possible, i.e. $h_{W,b}(x) \approx x$.

Mathematically the autoencoder can be expressed in terms of an input vector x being mapped by the encoder to another vector z (2),

$$z = h^{(encoder)}\left(W^{(encoder)}x + b^{(encoder)}\right) \tag{2}$$

prior to the decoder layer attempting to map the vector z back to an estimate of the original input vector $\hat{x}$ (3).

$$\hat{x} = h^{(decoder)}\left(W^{(decoder)}z + b^{(decoder)}\right) \tag{3}$$

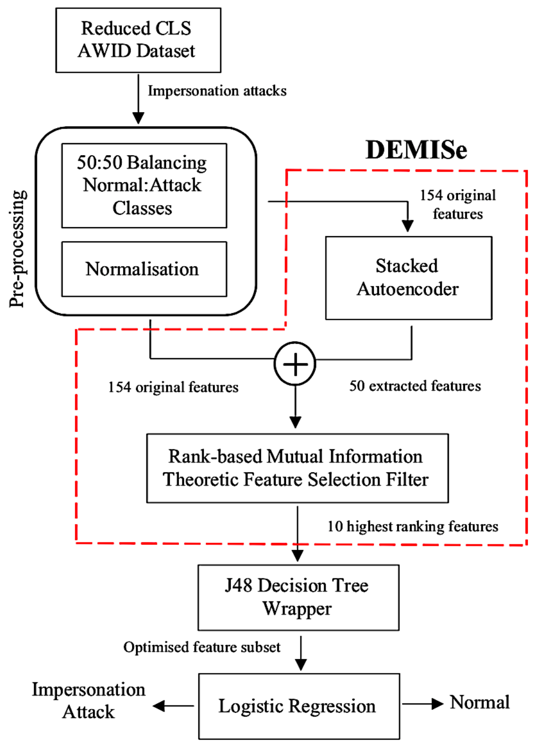

Figure. 2: High-level architecture of the DETEReD model showing the core DEMISe element for feature extraction and selection, the C4.8 tree wrapper, the logistic regression, and the pre-processing stages.

The autoencoder achieves mapping by backpropagation, whereby from the outset of training $h_{W,b}(x)$ is set to $\approx x$, thus forcing the autoencoder to learn a function that satisfies this constraint [15]. However, because autoencoders are greedy algorithms, they have the potential to discard features too early that could still be needed to accurately determine the correct target class for a given instance [10,16]. By learning a compressed representation of the feature space, the autoencoder algorithm assumes that achieving locally optimised weights and biases at each neuron will deliver the best overall performance [17]. To overcome this drawback, we used a stacked autoencoder (SAE) within Matlab for feature extraction. A SAE is a deep-learning implementation of an autoencoder, in which the learning achieved by the encoder layer of the first autoencoder is used to train the second autoencoder and so on. As a result, the SAE architecture allows the number of hidden neurons within the encoder layer to be gradually refined, thus reducing the greediness of the algorithm during its search (Fig. 4).

Furthermore, to construct and extract useful features from the SAE, sparsity must be encouraged within each layer. To achieve this, a cost function is included to penalise any neuron which activates more frequently than the average number of activations across all of the neurons within the network [15]. If m is the number of training instances (in this case 48,522), $a_j^{(encoder)}$ is the activation of the j^{th} neuron within the autoencoder's encoder layer, and the average number of activations of the j^{th} neuron across the training set is $\hat{\rho}_j$, where $\hat{\rho}_j$ is described as shown below (4).

$$\hat{\rho}_j = \frac{1}{m}\sum_{i=1}^{m}\left[a_j^{(encoder)}\left(x^{(i)}\right)\right] \tag{4}$$

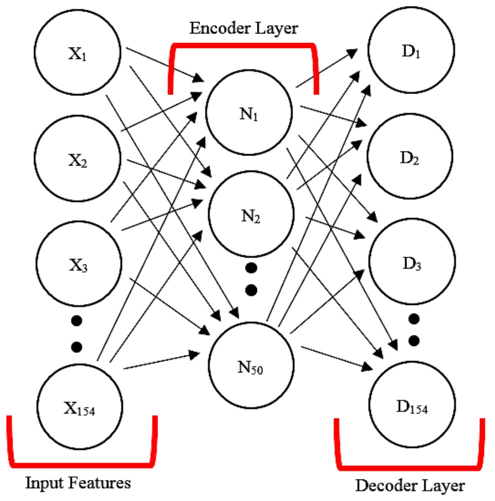

Figure. 3: Autoencoder architecture, where X is an input feature, N is a neuron within the encoder layer, and D is a neuron within the decoder layer.

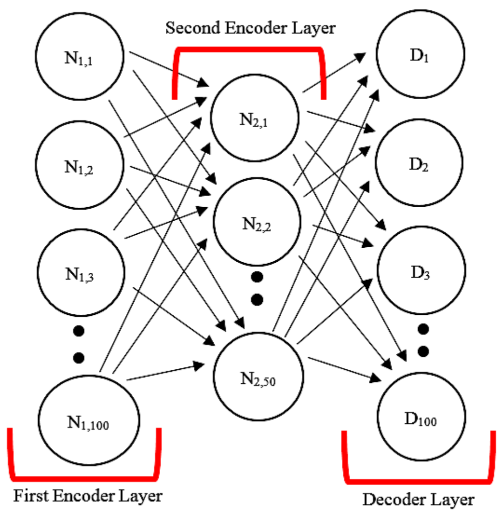

Figure. 4: Stacked autoencoder architecture, where *N_1* is a neuron within the first autoencoder's encoder layer, *N_2* is a neuron within the second autoencoder's encoder layer, and *D* is a neuron within the decoder layer.

As a result, a sparsely trained autoencoder aims to satisfy the constraint that $\widehat{\rho}_j = \rho$, where ρ is known as the sparsity parameter. For our SAE, the commonly used Kullback-Leibler (KL) divergence was used as the sparsity penalty function ($\Omega_{sparsity}$) because it takes the value of zero when $\widehat{\rho}_j = \rho$ as shown below (5).

$$\Omega_{sparsity} = \sum_{i=1}^{Neurons^{(Encoder)}} \rho \log\left(\frac{\rho}{\widehat{\rho}_i}\right) + (1 - \rho) \log\left(\frac{1-\rho}{1-\widehat{\rho}_i}\right) \quad (5)$$

For the DEMISe element of the model, we selected a SAE architecture of 154:100:50, which was previously shown to be effective for the generation of a compressed representation of the same reduced CLS AWID dataset [10]. Therefore, the first encoder layer learns a 100-neuron representation of the original 154 features, from which the second encoder layer learns a 50-neuron representation.

2.2 Mutual Information Theoretic Feature Selection

Within the common DEMISe element of the DEMISe-RBFC and DETEReD models, the 154 original and 50 newly extracted features are ranked and filtered based upon the amount of MI they share with the class label. Although filter methods for feature selection can deliver a sub-optional subset of features for classifier training [18], they present the most lightweight option for feature selection within an already resource-constrained IoT device. The DEMISe-RBFC model therefore employs a 'black box' Radial Basis Classifier to overcome the adverse effect of potentially suboptimal feature subsets being used for model training. To afford greater insight into the learning process, the DETEReD model also uses a C4.8 tree wrapper alongside the MI theoretic filter to optimise the feature subset. Although wrapper methods are more computationally expensive than filter methods because they use a iterative-learning-process to identify optimal feature subsets [18], the computational complexity of the wrapper is reduced because only the 10 features sharing the greatest amount of MI with the class label are considered, rather than all 204 (154 original plus 50 extracted features). Given that the MI theoretic filter will already have identified deterministic features of the class label, the lightweight optimisation afforded by the wrapper enables the use of a lower-complexity 'white box' logistic regression algorithm for instance classification, thereby providing additional valuable insight for the software security expert.

Although embedded methods that undertake feature selection as part of the learning process [6] were also considered for the feature selection element of the model, they are unable to afford the same level of insight in terms of feature importance and are slower and therefore more computationally expensive than filter-based methods [18].

Filter-based methods perform feature selection independently of the machine learner, thus, the feature selection uses heuristics centred on the inherent properties of the data to score and rank features [6]. Although many different measures and techniques are cited in the IDS literature for feature selection filters, we used MI in our model as the measure to score and rank the 204 features because promising results have been achieved by others using this approach [19,20,21]. MI was proposed by Shannon [22] as part of his information theory, which included the concept of entropy: the idea that random signals such as speech have an irreducible complexity, below which no further compression is possible [23]. The entropy *H* of a random variable *x* with a probability mass function *p(x)* can therefore be defined as shown below (6):

$$H(x) = -\sum_{x \in X} p(x) \log_2 p(x) \quad (6)$$

where X is a set of all possible outcomes of x.
The concept of entropy gives rise to: i) conditional entropy

$$H(x|y) = -\sum_{x \in X} \sum_{y \in Y} p(x,y) \log_2[p(x|y)] \quad (7)$$

where the entropy *H* of a random variable *x* is conditional upon the knowledge of another random variable *y*; and ii) mutual information

$$I(x;y) = \sum_{x,y} p(x,y) \log_2 \frac{p(x,y)}{p(x)p(y)} = H(x) - H(x|y), \quad (8)$$

denoting the amount of information gained about *y* as a result of knowing *x* [18,22]. In terms of our MI theoretic feature selection filter, the random variables *x* and *y* can be flexibly used to represent features and class labels. For example, *x* can be used to denote a feature within the dataset, and *y* can be invoked to denote as a class label, *i.e.,* how likely the machine learner is to correctly predict the class label for any given instance as a result of learning a given feature.

Computing the expressions (6)–(8) requires knowledge of probability measures $p(x)$, $p(y)$ and $p(x,y)$. Since these quantities are frequently unknown *a priori* for any given datasets, we invoke a widely-used histogram-based approach [26–28] to estimate the probability distribution. Generally *histogramming* is known to introduce estimation bias due to sensitivity to bin size [23,26–29]. We follow the procedures of discretization via supervised binning [30,31] to minimise this bias and obtain more accurate estimation of MI values.

Rank-based MI theoretic feature selection is inherently based on the assumption that classifier performance is linked to the amount of MI shared between the class label and a feature, *i.e.* the greater the number of high-ranking features selected for classifier training, the better the classifiers perform in terms of correctly identifying the class label for any given instance. However, this approach ignores the potential for more optimal subsets to exist, comprising features that are not sequentially ranked in terms of their MI values. In other words, the DEMISe element of the model assumes that features sequentially ranked 1,2,3,4,5 will train a better performing classifier than (for example) features ranked 1,3,7,32,91. In practice this is likely invalid due to the false assumption that each feature contains different and complementary information about the class label. Nevertheless, for the ultra-lightweight DEMISe-RBFC model, this approach should still identify features that can be used to train an IoT IDS model that achieves a 'good enough' detection performance within highly resource-constrained devices. On the other hand, the DETEReD model addresses the weak assumption made by the DEMISe element's MI theoretic feature selection by including the C4.8 tree wrapper.

A review of the IDS literature indicates that a feature selection filter as a pre-selection stage prior to wrapper subset optimisation is a novel approach to reduce dimensionality within resource-constrained devices. Although various wrapper algorithms including ANNs and support vector machines (SVMs) were considered for the DETEReD model, the C4.8 was selected because of it is less complex and thus has a lower impact on computational resources. Furthermore, the DEMISe element within the DETEReD model will already have identified potentially important features, so the additional complexity associated with SVMs and ANNs is unnecessarily resource-intensive. For DEMISe-RBFC model training, we selected up to 20 of the highest-ranking features in terms of the MI they share with the class label, which is 10% of all features. The state-of-the-art D-FES method which also uses deep feature extraction and selection, achieves strong detection performance with a similar number of selected features [10]. However, for the C4.8 wrapper in the DETEReD model, we only seek to identify optimal subsets among the top 10 features identified by the DEMISe element to minimise the time to build and the resource burden.

2.3 White Box and Black Box Classification

For the linear radial basis function approximation within the DEMISe-RBFC model, the Radial Basis Function Classifier (RBFC) was selected because, unlike other types of ANN, a radial basis function network can learn neuron widths and centres independently of the weights [14], offering advantages in terms of computational complexity (Fig. 5). Furthermore, an RBFC can also express non-linear relationships, which simpler algorithms such as the C4.8 and logistic regression cannot. This is particularly important given the aforementioned potential failure of the solely rank-based MI theoretic feature selection approach (adopted within the DEMISe-RBFC model) to identify optimal feature subsets. As a result, a higher complexity classifier such as the RBFC is required to ensure that model performance is not degraded by the presence of additional noise and outliers within the data. The RBFC achieves lower complexity than SVMs and other ANNs by generalising local representations *i.e.* a radial basis function network represents a 'prototype' of the training data from which the model draws generalised rules to describe the class labels [14].

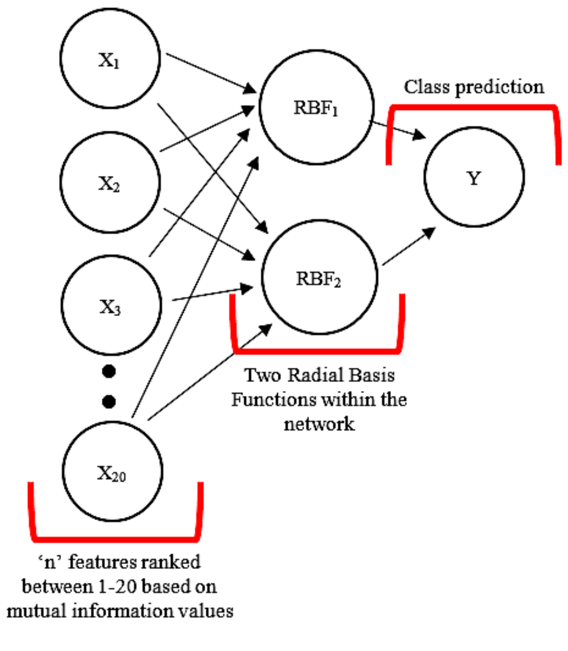
X₁
X₂
X₃
X₂₀
RBF₁
RBF₂
Y
Class prediction
Two Radial Basis Functions within the network
'n' features ranked between 1-20 based on mutual information values

Figure. 5: The Radial Basis Function Classifier architecture, where *X* is a top 20 ranked feature based on the amount of MI it shares with the class label, and *Y* is the class prediction made by the radial basis function network.

For the implementation, the initial centres for the two Gaussian radial basis functions were found using simple *k*-means clustering, whereas the weights were found using an approximation of the logistic function, thereby improving overall performance. For the interpretable regression in the DETEReD model, the implementation of a logistic regression in which ridge estimators improve the parameter estimates and diminish the error made by further predictions [24].

3 Evaluation and Analysis

For all aspects of DEMISe-RBFC and DETEReD model training and testing, we used a workstation running a Unix-based OS with a 3.1 GHz Intel Core i7 CPU and 8 Gb of RAM. In terms of the model evaluation metrics, we elected to use the standard measures of detection accuracy (Acc) (9), detection rate (DR), which is the ratio of the correctly detected attacks to the total number of attacks (10), precision, which measures the number of correctly identified attacks (11), false alarm rate (FAR), which is the ratio of the number of normal instances misclassified as attacks against the total number of normal instances (12), false negative rate (FNR), which represents the number of undetected attacks (13), the harmonic mean between precision and DR (F_1) (14), and the Matthews correlation coefficient (Mcc), which is the coefficient between detected and observed behaviours [10] (15). We also measured the CPU time (second) to build for the model because it indicates the computational burden associated with model training.

$$Acc = \frac{TP + TN}{TP + TN + FP + FN} \tag{9}$$

$$DR(recall) = \frac{TP}{TP + FN} \tag{10}$$

$$Precision = \frac{TP}{TP + FP} \tag{11}$$

$$FAR = \frac{FP}{TN + FP} \tag{12}$$

$$FNR = \frac{FN}{FN + TP} \tag{13}$$

$$F_1 = \frac{2TP}{2TP + FP + FN} \tag{14}$$

$$Mcc = \frac{(TP * TN) - (FP * FN)}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}} \tag{15}$$

Table I: Top 20 features based on mutual information values

Features in order of decreasing MI value	8, 82, 4, 38, 157, 162, 168, 160, 188, 161, 199, 176, 159, 191, 182, 186, 195, 156, 158, 165

3.1 Mutual Information Theoretic Feature Selection

Having extracted the 50 additional features using the deep structured SAE, the continuous variables were discretised using supervised entropy-based discretisation, with the MI between each of the 204 features and the class label calculated as shown above (8). Entropy-based discretisation was applied because it affords a degree of efficiency within the model given that the entropy of each feature is also required to calculate MI. The top 20 features are described in Table I in terms of the MI they share with the class label identified by the MI theoretic feature selection filter within the DEMISe element of both models.

Of the 204 features ranked by the model, only 83 had a MI value > 0, suggesting that 121 features were statistically independent of the class label, and thus irrelevant to the identification of an impersonation attack within the reduced CLS portion of the AWID dataset. This is not an unreasonable discovery, given that not all features within a dataset are important for attack detection [21]. However, this finding suggests that even low-complexity MI theoretic feature selection filters are able to reduce dimensionality within modern complex network datasets such as AWID.

Table II: Optimal number of high-ranking features for different classifiers

Classifier	Optimal Features	Acc (%)	DR (%)	F_1 (%)	Mcc (%)	TTB (s)
RBFC	Top 7	**98.00**	99.04	**97.98**	**96.02**	301.53
LR	Top 3	92.18	98.98	91.59	85.18	294.13
MLP	Top 3	93.16	96.91	92.87	86.60	305.84
SVM	Top 9	90.34	98.89	89.42	81.94	1364.84
C4.8	Top 4	73.99	**100.00**	64.85	56.18	**293.41**
RF	Top 7	53.39	51.76	68.19	18.54	303.40

The time to build (TTB) for all models includes the 293 s required by the deep-structured SAE. Abbreviations not used in main text: logistic regression (LR), multilayer perceptron (MLP), random forest (RF).

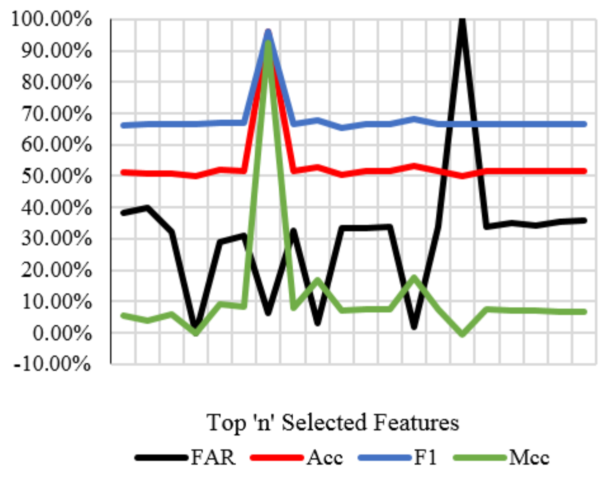

Figure. 6: DEMISe-RBFC model performance when trained with the top n selected features, where 20 ≥ n ≤ 1. The performance of the DEMISe-RBFC model cannot be predicted in advance of training based on the MI shared between each feature and the class label.

3.2 DEMISe-RBFC

For the first round of training and testing, the DEMISe-RBFC model was trained using all of the top 20 features shown in Table I, and then for each subsequent round the number of selected features was reduced by one, with the lowest ranking feature being removed from the training subset each time. We took this approach in order to test the hypothesis that the classifier performance could be predicted using MI ranking alone, whereby the greater the number of high-ranking features used for training, the stronger the model's detection performance. Although our results indicated that training the model's RBFC with the top seven features identified by rank-based MI theoretic feature selection met the objective of achieving a 'good enough' detection performance (Fig. 6), it was not possible to predict this optimal number of high-ranking features in advance.

This conclusion was considered justified because several linear and non-linear classifiers were also tested *in lieu* of the RBFC to ensure the observed behaviour could not be attributed to the RBFC algorithm. As shown in Table II, the top seven features did not prove to be an optimal combination for any of the alternative classifiers. However, MI theoretic feature selection yielded a better-performing model than training using all 204 features and was therefore considered effective (Table III). Furthermore, because the optimal subset comprising the top seven features included the extracted features 157, 162 and 168, the suitability of the deep structured SAE for feature extraction within the DEMISe-RBFC model was also confirmed.

Table III: Mutual information theoretic feature selection versus no feature selection

Features	Acc (%)	DR (%)	F_1 (%)	Mcc (%)	TTB (s)
Top 7	**98.00**	99.04	**97.98**	**96.02**	301.53
All 204	48.50	49.24	65.30	-12.03	356.87

The time to build (TTB) for all models includes the 293 s required by the deep-structured SAE.

3.3 DETEReD

Having identified the top 20 features within the DEMISe element of the model (Table I), the C4.8 wrapper considered only the top 10 for subset optimisation, *i.e.* features 8, 82, 4, 38, 157, 162, 168, 160, 188 and 161. Among these features, the wrapper identified an optimal subset comprising features 4, 8, 82, 160 and 168. Because this subset also included extracted features, the suitability of the deep structured SAE for feature extraction within the DETEReD model was confirmed.

When trained with this five-feature subset, logistic regression achieved the performance shown in Table IV. As above, this performance was compared against other classifiers to validate the effectiveness of the proposed DETEReD model. Accordingly, we found that logistic regression was the best-performing classifier for the DETEReD model.

Table IV: DETEReD model performance with different classifiers

Classifier	Acc (%)	DR (%)	F_1 (%)	Mcc (%)	TTB (s)
LR	**98.04**	**99.07**	**98.01**	**96.09**	603.33
C4.8	51.17	50.59	67.16	10.39	**601.97**
MLP	52.31	51.19	67.61	14.12	635.32
SVM	63.13	58.12	71.82	33.35	1340.84
RBFC	52.61	51.36	67.56	13.48	614.94
RF	52.32	51.19	67.70	15.18	612.07

The CPU time to build (TTB) for all models includes the 293 s required by the deep-structured SAE and the 309 s required by the C4.8 wrapper. Abbreviations not used in main text: logistic regression (LR), multilayer perceptron (MLP), random forest (RF).

3.4 Comparison Against Baselines

Having trained and tested the DEMISe-RBFC and DETEReD models, their suitability was assessed by comparing their performance against other published machine-learning-based IDS solutions. For the purposes of this comparison, we selected the D-FES model because it represents the current state-of-the-art in terms of combining feature extraction and selection techniques within an machine-learning-based IDS [10], as well as two models previously used to detect impersonation attacks within the reduced CLS portion of the AWID dataset [11,12]. The performance of the DEMISe-RBFC and DETEReD models is compared against the D-FES model in Table V, and against the models previously tested against the CLS portion of the AWID in Table VI.

Table V: DETEReD and DEMISe-RBFC vs D-FES models

Classifier	Acc (%)	DR (%)	FAR (%)	F_1 (%)	Mcc (%)	TTB (s)
DEMISe-RBFC	98.00	99.04	3.00	97.98	96.02	301.53
DETEReD	98.04	99.07	2.96	98.01	96.09	603.33
D-FES-CFS	98.80	96.34	0.46	97.37	96.61	1343.00
D-FES-Corr	98.26	95.91	1.04	96.17	95.05	1264.00
D-FES-ANN	99.95	99.88	0.02	99.90	99.87	1444.00
D-FES-SVM	99.97	99.92	0.01	99.94	99.92	12073.00
D-FES-C4.5	99.60	99.55	0.38	99.12	98.86	2595.00

D-FES data from [10]. TTB = CPU time to build.

Given that our aim was to develop a lightweight machine-learning-based IDS solution suitable for the IoT environment, it would be unreasonable to expect the DEMISe-RBFC and DETEReD models to match the performance of the D-FES wrapper methods which consider all 204 features, in contrast to the vastly reduced subset considered by the DETEReD model, and the solely filter-based method used by the DEMISe-RBFC model. Nevertheless, Table V shows that DEMISe-RBFC and DETEReD compared well against the D-FES wrapper methods, and outperform it when using the DR and F_1 classifiers. In terms of the CPU time to build, the DETEReD model was 52.29% quicker than the fastest D-FES-Corr filter-based method, whereas the DEMISe-RBFC model was 76.14% quicker. The lightweight nature of the DEMISe-RBFC and DETEReD models was further demonstrated by comparison to the best-performing D-FES-SVM, with the DETEReD model achieving a 95.01% faster time to build. The DEMISe-RBFC and DETEReD models also outperformed the earlier models tested on the same dataset [11,12] when using the Acc, DR and FAR classifiers, and although these previous studies did not provide data for F_1 and Mcc, the available metrics suggest that the DEMISe-RBFC and DETEReD models would also outperform the earlier models using these classifiers. Accordingly, we suggest that the inclusion of deep feature extraction in the DEMISe-RBFC and DETEReD models provides an advantage in terms of impersonation attack detection within the AWID dataset.

Table VI: DETEReD and DEMISe-RBFC versus Models previously tested against the CLS portion of the AWID dataset [11,12]

Classifier	Acc (%)	DR (%)	FAR (%)	F_1 (%)	Mcc (%)
DEMISe-RBFC	98.00	99.04	3.00	97.98	96.02
DETEReD	**98.04**	**99.07**	2.96	**98.01**	**96.09**
Kolias *et al* [12]	94.91	97.23	74.21	97.37	22.12
Aminanto *et al* [11]	97.60	85.00	**2.36**	NRA	NRA

NRA = No results available.

3.5 Estimating Resource Requirements

Having demonstrated that the DEMISe-RBFC and DETEReD models compete well against the contemporary models in the wider IDS literature, we now consider the resource requirements of these two models in order to determine their suitability for IoT devices.

To estimate the resource requirements of these two models, the 32-bit IEEE 754 binary floating-point standard was selected because the representative D1000, D2000 and C1000 Intel Quark microcontrollers designed for use within IoT devices utilise a 32-bit architecture [25]. We used this binary notation standard to estimate the resource requirements of both models by expressing the coefficients within the DETEReD model's logistic regression and the DEMISe-RBFC radial basis function's unit centres, bias weights, and output weights. Based on these estimates, Table VII shows that despite the longer CPU time to build, the white-box DETEReD model requires an estimated 24 bytes of memory to capture the five weights and one intercept learnt by the logistic regression, whereas the black-box DEMISe-RBFC model requires 84 bytes to capture its more-complex 21 learnt parameters.

Based on the estimated resource requirements in Table VII and the specification for even the smallest D1000 Intel Quark microcontroller, the parameters learnt by the DEMISe-RBFC and DETEReD models could potentially be accommodated within the available non-volatile memory (32 Kb instruction, 4 Kb data) of the microcontroller [25], thus confirming that the DEMISe-RBFC and DETEReD models are suitable for an machine-learning-based IoT IDS.

Table VII: Estimated resource requirements for DETEReD and DEMISe-RBFC

Model	Number of Parameters	Estimated Memory Requirement
DEMISe-RBFC	21 (4 output weights (from 2 layers), 14 unit centres (from 2 layers), 2 bias weights (one for each class) and a scale weight)	84 bytes
DETEReD	5 weights (+1 intercept)	24 bytes

4 Conclusion and Future Work

We built upon earlier architectural descriptions [7,8,9] to propose two lightweight and accurate models for a machine-learning-based IDS with potential suitability for the resource-constrained and dynamic IoT environment. We found that although MI theoretic feature selection is an effective strategy to identify features that are important for the correct classification of any given instance, it is best employed in a novel manner as a pre-selection stage prior to wrapper subset evaluation. The performance of a classifier could not be predicted based on the MI values for each feature alone, so the optimisation of the DEMISe-RBFC model could prove challenging if re-training were required using an unfamiliar dataset, because it may not be valid to assume that the top seven features will always train the best performing model. The DETEReD model overcomes this issue by the novel application of a C4.8 wrapper which only considers the top 10 features identified by the MI theoretic filter, thereby reducing the computational complexity caused by the wrapper needing to consider subsets within all 204 features. The DETEReD model can therefore use a simpler classifier such as logistic regression, and thus delivers white box insight into the model's learning process.

The DEMISe-RBFC and DETEReD models also demonstrate the utility of deep structured feature extraction using a stacked autoencoder, whereby both models use extracted features during learning and outperform earlier models [11,12] by including this deep feature extraction stage.

In terms of the state-of-the-art in the wider machine-learning-based IDS literature, we have shown that our proposed DEMISe-RBFC and DETEReD IoT IDS models also outperform D-FES filter methods [10], achieving a 76.14% and 52.29% faster time to build than even the fastest D-FES-Corr method when using the DR and F_1 classifiers, respectively. In addition to the lower computational cost associated with training the DEMISe-RBFC and DETEReD models, we also showed that the learning achieved by the models could be accommodated within even the most resource-constrained devices, requiring just 84 and 24 bytes of storage, respectively.

Finally, in terms of future work, we intend to investigate the ability of the DEMISe-RBFC and DETEReD models to detect other types of attack that are likely to occur within an IoT network, and demonstrate the lightweight nature of the algorithm by deploying it within a resource constrained device (e.g. Raspberry Pi).

ACKNOWLEDGMENTS

We are grateful to the Laboratory of Information and Communication Systems Security at George Mason University in the U.S. for providing us a copy of AWID dataset as well as their invaluable discussions; and special thanks to Samsung SDS for their constructive criticism and financial support on this work.

REFERENCES

[1] V Adat and BB Gupta (2018) Security in Internet of Things: issues, challenges, taxonomy, and architecture. Telecommunication Systems, 67(3), 423–441.

[2] C Kolias, A Stavrou, J Voas, I Bojanova, and R Kuhn (2016b) Learning Internet-of-Things security "hands-on". IEEE Security & Privacy, 14(1), 37–46.

[3] W Li, W Meng, and HS Horace (2017). Enhancing collaborative intrusion detection networks against insider attacks using supervised intrusion sensitivity-based trust management model. Journal of Network and Computer Applications, 77(1), 135–45.

[4] AL Buczak and E Guven (2016) A survey of data mining and machine learning methods for cyber security intrusion detection. IEEE Communications Surveys & Tutorials, 18(2), 1153–76.

[5] J Cong and B Xiao (2014) Minimizing computation in convolutional neural networks. International conference on artificial neural networks, 15, 281–290.

[6] T Hamed, R Dara and SC Kremer (2018) Network intrusion detection system based on recursive feature addition and bigram technique. Computers and Security, 73. 137–155. 10.1016/j.cose.2017.10.011

[7] E Cho, J Kim and C Hong (2009) Attack model and detection scheme for botnet on 6LoWPAN, 12th Asia-Pacific Network Operations and Management Symposium. Jeju, South Korea, 515–518.

[8] A Gupta, OJ Pandey, M Shukla, A Dadhich, S Mathur and A Ingle (2013) Computational intelligence based intrusion detection systems for wireless communication and pervasive computing networks, 2013 IEEE International Conference on Computational Intelligence and Computing Research. Tamilnadu, India, 26-28 December, 1–7.

[9] NK Thanigaivelan, E Nigussie, RK Kanth, S Virtanen and J Isoaho (2016) Distributed internal anomaly detection system for Internet-of-Things, 2016 13th IEEE Annual Consumer Communications & Networking Conference (CCNC). Las Vegas, USA, 9-12 January, 319–320.

[10] ME Aminanto, R Choi, HC Tanuwidjaja, PD Yoo and K Kim (2018) Deep abstraction and weighted feature selection for Wi-Fi impersonation detection, IEEE Transactions on Information Forensics and Security, 13(3), 621–636.

[11] ME Aminanto and K Kim (2017) Detecting impersonation attack in WiFi networks using deep learning approach, Information Security Applications 17th International Workshop. Jeju Island, South Korea, 25-27 August 2016, 136–147.

[12] C Kolias, G Kambourakis, A Stavrou and S Gritzalis (2016a) Intrusion detection in 802.11 networks : empirical evaluation of threats and a public dataset, IEEE Communication Surveys and Tutorials, 18(1), 184–208.

[13] A Gharib, I Sharafaldin, AH Lashkari and AA Ghorbani (2017) An evaluation framework for intrusion detection dataset, ICISS 2016, 2016 International Conference on Information Science and Security, Jaipur, India, 16-20 December, 1–5.

[14] I Witten, E Frank, M Hall and C Pal, 2017 Data mining: practical machine learning tools and techniques (4th ed). Morgan Kaufmann, Cambridge, USA.

[15] Stanford University (2018) UFLDL tutorial: autoencoders http://ufldl.stanford.edu/tutorial/unsupervised/Autoencoders/

[16] Z Wang (2015) The applications of deep learning on traffic identification, Black Hat, Las Vegas, USA, 5-6 August.

[17] G Kesavaraj and S Sukumaran (2013) A study on classification techniques in data mining. 2013 Fourth International Conference on Computing, Communications and Networking Technologies, Tiruchengode, India, 4-6 July, 1–7.

[18] JR Vergara and PA Estévez (2014) A review of feature selection methods based on mutual information. Neural Computing and Applications, 24(1), 175–186.

[19] OY Al-Jarrah, O.Y., Alhussein, O., Yoo, P.D., Muhaidat, S., Taha, K. and Kim, K. (2016) Data randomization and cluster-based partitioning for botnet intrusion detection'. IEEE Transactions on Cybernetics, 46(8), pp. 1796–1806.

[20] A Shabtai, U Kanonov, Y Elovici, C Glezer and Y Weiss (2012) "Andromaly": a behavioral malware detection framework for android devices'. Journal of Intelligent Information Systems, 38(1), 161–190.

[21] USKPM Thanthrige, J Samarabandu and X Wang (2016) Machine learning techniques for intrusion detection on public dataset. 2016 IEEE Canadian Conference on Electrical and Computer Engineering (CCECE). Vancouver, Canada, 15-18 May, 1–4.

[22] CE Shannon (1948) A mathematical theory of communication. The Bell System Technical Journal, 27, 379-426

[23] TM Cover and JA Thomas (2006) Elements of information theory (2nd ed). John Wiley and Sons: Hoboken, USA.

[24] S Le Cessie and JC Van Houwelingen (1992) Ridge estimators in logistic regression. Applied Statistics, 41(1), 191-201.

[25] Intel Corporation (2018) Intel® QuarkTM microcontrollers. https://www.intel.com/content/www/us/en/embedded/products/quark/overview.html

[26] G Egnal (1999) Image registration using mutual information. Technical Report, University of Pennsylvania.

[27] L Birge and Y Rozenholc (2002) How many bins should be put in a regular histogram. Technical Report, Universite Paris VI, UMR CNRS 7599, Universite du Maine.

[28] PL Davies, U Gather, D Nordman and H Weinert (1997) Constructing a regular histogram - a comparison of methods. Technical Report, Technical University Eindhoven.

[29] PA Legg, PL Rosin, D Marshall and JE Morgan (2013) Improving accuracy and efficiency of mutual information for multi-modal retinal image registration using adaptive probability density estimation. Computerized Medical Imaging and Graphics, 37(7–8), 597-606.

[30] J Dougherty, R Kohavi and M Sahami (1995) Supervised and unsupervised discretization of continuous features. ICML 1995 – 12th International Conference on Machine Learning. Tahoe City, California, USA, 9-12 July 1995. Morgan Kaufmann: San Francisco, CA, 194-202.

[31] I Witten, E Frank, M Hall, and C Pal (2017) Data mining: practical machine learning tools and techniques (4th ed). Morgan Kaufmann: Cambridge, USA.