On the Evaluation of Restricted Boltzmann Machines for Malware Identification

Kelton Costa, Luis Silva, Guilherme Martins, Gustavo Rosa, Rafael Pires, João Papa

São Paulo State University, Department of Computing, Bauru, SP - Brazil 17033-360. Tel: +55 14 3103-6000
e-mail: {kelton,gustavo.rosa,papa}@fc.unesp.br
luis.silva51@fatec.sp.gov.br, gui.bmartins.unesp@gmail.com, rafapires@gmail.com

Abstract—In the last years, tablets and smartphones have been widely used with the very same purpose as desktop computers: web browsing, social networking, banking and others, just to name a few. However, we are often facing the problem of keeping our information protected and trustworthy. As a result of their popularity and functionality, mobile devices are a growing target for malicious activities. In such context, mobile malwares have gained significant ground since the emergence and growth of smartphones and handheld devices, thus becoming a real threat. The main contribution of this paper is to evaluate Restricted Boltzmann Machines (RBMs) for unsupervised feature learning in the context of malware identification. In order to evaluate the results, we employed two supervised pattern recognition techniques, say that Optimum-Path Forest and Support Vector Machines, as well as a classification approach based on RBMs.

Keywords—Optimum-Path Forest, Restricted Boltzmann Machines, Malware Detection.

1. Introduction

In the past few years, there was a huge increase in the use of tablets and smartphones with the vey same purpose as desktop computers: web browsing, social networking, banking and others [20, 22, 40]. As a result of their popularity and functionality, smartphones are a massive growing target for malicious activities. In this context, mobile malware has gained significant ground since the emergent and growth of smartphones and handheld devices, thus becoming a real threat [18]. As such, many researchers have studied the safety of mobile devices using statistical methods, data mining and intelligent techniques that can somehow cope with the amount of malware attacks on mobile devices [7, 13, 14, 37, 42, 43].

The work by Penning et al. [31] defines concepts in mobile malware threats and attacks, cybercriminal motivations as well as existing prevention methods and their limitations. The same paper further proposes a cloud-based framework for mobile malware detection that requires a collaboration among mobile subscribers, app stores, and IT security professionals. The article describes a step-based framework to download applications, where each downloaded software is analyzed through a malware database scanning (such system is called “Malware Detector”).

Another interesting work was proposed by Guo and Sui [12], which concerns the analysis of the network behaviour in order to foster defense systems for mobile malwares. The system considers the URL features, traffic statistics and files to detect malicious attacks in mobile malwares. The system can track the user identity information, evaluate the mobile malware threat, as well as alert and prevent about mobile malware events according to the security policy. The experiments indicate the real-time execution is very effective and can stop the discharge process of mobile malwares right before the user receives the entire modified file.

Later on, the work described by Arora et al. [2] analyzed network traffic features in order to build a rule-based classifier for the detection of Android malwares. However, such approach is used to those set of malwares that connect to some remote server in the background only, thus generating some network traffic. The experimental results suggest the proposed approach is remarkably accurate, detecting more than 90% of the traffic samples. Kruczkowski and Szynkiewicz [19] employed the well-known Support Vector Machines (SVMs) to detect malwares, and Gavrilut et al. [6] presented a two-step approach for the same purpose, but using SVMs together with Perceptrons. In a recent work, Huda et al. [17] employed SVMs and a Maximum-Relevance-Minimum-Redundancy Filter to detect such threats using features extracted from Application Program Interface (API) calls. Shabtai et al. [33] presented an approach to automatic identify malwares based on network traffic patterns either. Roughly speaking, the idea was to use semi-supervised learning algorithms to recognize anomalous patterns that might correspond to the very same expected behaviour of malwares in the network.

Some years ago, Papa et al. [24, 25] introduced a new pattern recognition technique called Optimum-Path Forest (OPF), which models the problem of pattern classification as a graph partition task, in which each dataset sample is encoded as a graph node and connected to others through an adjacency relation. The main idea is to rule a competition process among some key samples (*prototypes*) that try to conquer the remaining nodes in order to partition the graph into optimum-path trees, each one rooted at one prototype. OPF has obtained promising results in different research areas, being quite similar to Support Vector Machines (SVMs) with respect to recognition rates, but faster for training, since OPF is parameterless and it has a quadratic complexity. The OPF classifier has been used in several applications related to computer security, such as network intrusion detection [4, 32], spam identification [8, 34], and recently malware recognition [5].

Another problem usually faced when working with spam and malware identification concerns the features, which are often based on bag-of-words. Learning dictionaries usually depends on a predefined vocabulary, and its size strongly affects the performance of the learner. In the last years, a considerable attention has been given to the so-called deep learning techniques, which can learn features in a generative manner, i.e. without prior knowledge about the labels. Among the techniques, we can cite Restricted Boltzmann Machines (RBMs), which are essentially stochastic neural networks that aim at recovering the input data based on a hidden layer composed of latent variables. Although a number of RBM-related works can be found in the literature, they mainly focus on image-oriented applications. As a matter of fact, only a few employ RBMs in the context of malicious content identification [9, 35].

In this paper, we propose to use RBMs to learn features from malware-driven data in order to automatic identify such threats in Android-based environments[1], which turns out to be the main contribu-

1. This paper is an extension of the work of Costa et al. [5].

tion of this work. As far as we are concerned, we have observed only two works that aimed at using deep learning techniques for malware detection in mobile data [41, 42]. Both works employed Deep Belief Networks fine-tuned with backpropagation to identify malwares, being the DBN parameters chosen by hand (i.e. empirically). In our work, we propose a single solution based on RBMs and meta-heuristic techniques, being the latter used to fine-tune RBMs, instead of doing by hand.

Roughly speaking, the main contributions of this paper are: (i) to employ RBMs fine-tuned by meta-heuristic techniques in order to automatically identify malware data, and (ii) to foster the related research by evaluating deep learning techniques in computer security. In order to evaluate the proposed approach, we employed OPF, SVM and an RBM-based classifier for comparison purposes. The main reason behind using OPF and SVM concerns the fact such techniques are considered state-of-the-art in supervised learning, thus being widely used by the research community.

The remainder of this paper is organized as follows. Sections 2, and 3 present a theoretical background regarding RBM as a feature learner and as a classifier, respectively. Sections 4 and 5 presents the methodology and experiments, respectively. Finally, Section 6 states conclusions and future works.

2. Restricted Boltzmann Machines

Restricted Boltzmann Machines are energy-based stochastic neural networks composed of two layers of neurons (visible and hidden), in which the learning phase is conducted by means of an unsupervised fashion. The RBM is similar to the classical Boltzmann Machine [1], except that no connections between neurons of the same layer are allowed. Figure 1 depicts the architecture of a Restricted Boltzmann Machine, which comprises a visible layer $\mathbf{v}$ with m units and a hidden layer $\mathbf{h}$ with n units. The real-valued $m \times n$ matrix $\mathbf{W}$ models the weights between visible and hidden neurons, where w_{ij} stands for the weight between the visible unit v_i and the hidden unit h_j.

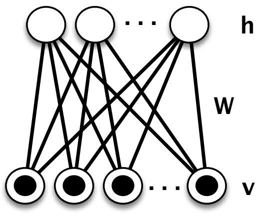

Fig. 1. The RBM architecture.

At first, RBMs were designed using only binary visible and hidden units, the so-called Bernoulli Restricted Boltzmann Machines (BRBMs). Later on, Welling et al. [38] shed light over other types of units that can be used in an RBM, such as Gaussian and binomial units, among others. Since in this paper we are interested in BRBMs, we will introduce their main concepts, which are the basis for other generalizations of RBMs. As a matter of fact, we shall use the term “RBMs” instead of BRBMs for the sake of clarity.

Let us assume $\mathbf{v}$ and $\mathbf{h}$ as the binary visible and hidden units, respectively. In other words, $\mathbf{v} \in \{0,1\}^m$ and $\mathbf{h} \in \{0,1\}^n$. The energy function of a Restricted Boltzmann Machine is given by:

$$E(\mathbf{v},\mathbf{h}) = -\sum_{i=1}^{m} a_i v_i - \sum_{j=1}^{n} b_j h_j - \sum_{i=1}^{m}\sum_{j=1}^{n} v_i h_j w_{ij}, \quad (1)$$

where $\mathbf{a}$ and $\mathbf{b}$ stand for the biases of visible and hidden units, respectively. The probability of a configuration $(\mathbf{v},\mathbf{h})$ is computed as follows:

$$P(\mathbf{v},\mathbf{h}) = \frac{e^{-E(\mathbf{v},\mathbf{h})}}{\sum_{\mathbf{v},\mathbf{h}} e^{-E(\mathbf{v},\mathbf{h})}}, \quad (2)$$

where the denominator of above equation is a normalization factor that stands for all possible configurations involving the visible and hidden units. In short, the RBM learning algorithm aims at estimating **W**, **a** and **b**.

The parameters of an RBM can be optimized by performing stochastic gradient ascent on the log-likelihood of training patterns. Given a training sample (visible unit), its probability is computed over all possible hidden vectors, as follows:

$$P(\mathbf{v}) = \frac{\sum_{\mathbf{h}} e^{-E(\mathbf{v},\mathbf{h})}}{\sum_{\mathbf{v},\mathbf{h}} e^{-E(\mathbf{v},\mathbf{h})}}. \tag{3}$$

In order to update the weights and biases, it is necessary to compute the following derivatives:

$$\frac{\partial \log P(\mathbf{v})}{\partial w_{ij}} = E[h_j v_i]^{data} - E[h_j v_i]^{model}, \tag{4}$$

$$\frac{\partial \log P(\mathbf{v})}{\partial a_i} = v_i - E[v_i]^{model}, \tag{5}$$

$$\frac{\partial \log P(\mathbf{v})}{\partial b_j} = E[h_j]^{data} - E[h_j]^{model}, \tag{6}$$

where $E[\cdot]$ stands for the expectation operation, and $E[\cdot]^{data}$ and $E[\cdot]^{model}$ correspond to the data-driven and the reconstructed-data-driven probabilities, respectively.

In practical terms, we can compute $E[h_j v_i]^{data}$ considering **h** and **v** as follows:

$$E[\mathbf{hv}]^{data} = P(\mathbf{h}|\mathbf{v})\mathbf{v}^T, \tag{7}$$

where $P(\mathbf{h}|\mathbf{v})$ stands for the probability of obtaining **h** given the visible vector (training data) **v**:

$$P(h_j = 1|\mathbf{v}) = \sigma\left(\sum_{i=1}^{m} w_{ij} v_i + b_j\right), \tag{8}$$

where $\sigma(\cdot)$ stands for the logistic sigmoid function[2]. Therefore, it is straightforward to compute $E[\mathbf{hv}]^{data}$: given a training data $\mathbf{x} \in \mathcal{X}$, where $\mathcal{X}$ stands for a training set, we just need to set $\mathbf{v} \leftarrow \mathbf{x}$ and then employ Equation 8 to obtain $P(\mathbf{h}|\mathbf{v})$. Further, we use Equation 7 to finally obtain $E[\mathbf{hv}]^{data}$.

The big question now is how to obtain $E[\mathbf{hv}]^{model}$, which is the model learned by the system[3]. One possible strategy is to perform alternating Gibbs sampling starting at any random state of the visible units until a certain convergence criterion, such as k steps, for instance. The Gibbs sampling consists of updating hidden units using Equation 8 followed by updating the visible units using $P(\mathbf{v}|\mathbf{h})$, given by:

$$P(v_i = 1|\mathbf{h}) = \sigma\left(\sum_{j=1}^{n} w_{ij} h_j + a_i\right), \tag{9}$$

and then updating the hidden units once again using Equation 8. In short, it is possible to obtain an estimative of $E[\mathbf{hv}]^{model}$ by initializing the visible unit with random values and then performing Gibbs sampling, which may be time-consuming. Fortunately, Hinton [15] introduced a faster methodology to compute $E[\mathbf{hv}]^{model}$ based on contrastive divergence. Basically, the idea is to initialize the visible units with a training sample, to compute the states of the hidden units using Equation 8, and then to compute the states of the visible unit (reconstruction step) using Equation 9. Roughly speaking, this is equivalent to perform Gibbs sampling using $k = 1$.

Based on the above assumption, we can now compute $E[\mathbf{hv}]^{model}$ as follows:

$$E[\mathbf{hv}]^{model} = P(\tilde{\mathbf{h}}|\tilde{\mathbf{v}})\tilde{\mathbf{v}}^T. \tag{10}$$

2. The logistic sigmoid function can be computed by the following equation: $\sigma(x) = 1/(1 + \exp(-x))$.

3. We are now writing $E[h_j v_i]^{model}$ in terms of **h** and **v**.

Therefore, the equation below leads to a simple learning rule for updating the weight matrix **W**, as follows:

$$\begin{aligned}\mathbf{W}^{t+1} &= \mathbf{W}^t + \eta(E[\mathbf{hv}]^{data} - E[\mathbf{hv}]^{model}) \\ &= \mathbf{W}^t + \eta(P(\mathbf{h}|\mathbf{v})\mathbf{v}^T - P(\tilde{\mathbf{h}}|\tilde{\mathbf{v}})\tilde{\mathbf{v}}^T), \quad (11)\end{aligned}$$

where $\mathbf{W}^t$ stands for the weight matrix at time step t, and η corresponds to the learning rate. Additionally, we have the following formulae to update the biases of the visible and hidden units:

$$\begin{aligned}\mathbf{a}^{t+1} &= \mathbf{a}^t + \eta(\mathbf{v} - E[\mathbf{v}]^{model}) \\ &= \mathbf{a}^t + \eta(\mathbf{v} - \tilde{\mathbf{v}}), \quad (12)\end{aligned}$$

and

$$\begin{aligned}\mathbf{b}^{t+1} &= \mathbf{b}^t + \eta(E[\mathbf{h}]^{data} - E[\mathbf{h}]^{model}) \\ &= \mathbf{b}^t + \eta(P(\mathbf{h}|\mathbf{v}) - P(\tilde{\mathbf{h}}|\tilde{\mathbf{v}})), \quad (13)\end{aligned}$$

where $\mathbf{a}^t$ and $\mathbf{b}^t$ stand for the visible and hidden units biases at time step t, respectively. In short, Equations 11, 12 and 13 are the vanilla formulation for updating the RBM parameters.

Later on, Hinton [16] introduced a weight decay parameter λ, which penalizes weights with large magnitude[4], as well as a momentum parameter α to control possible oscillations during the learning process. Therefore, we can rewrite Equations 11, 12 and 13 as follows:

$$\mathbf{W}^{t+1} = \mathbf{W}^t + \underbrace{\eta(P(\mathbf{h}|\mathbf{v})\mathbf{v}^T - P(\tilde{\mathbf{h}}|\tilde{\mathbf{v}})\tilde{\mathbf{v}}^T) - \lambda\mathbf{W}^t + \alpha\Delta\mathbf{W}^{t-1}}_{=\Delta\mathbf{W}^t}, \quad (14)$$

$$\mathbf{a}^{t+1} = \mathbf{a}^t + \underbrace{\eta(\mathbf{v} - \tilde{\mathbf{v}}) + \alpha\Delta\mathbf{a}^{t-1}}_{=\Delta\mathbf{a}^t} \quad (15)$$

and

4. The weights may increase during the convergence process.

$$\mathbf{b}^{t+1} = \mathbf{b}^t + \underbrace{\eta(P(\mathbf{h}|\mathbf{v}) - P(\tilde{\mathbf{h}}|\tilde{\mathbf{v}})) + \alpha\Delta\mathbf{b}^{t-1}}_{=\Delta\mathbf{b}^t}. \quad (16)$$

3. Restricted Boltzmann Machines for Classification Purposes

Although RBMs are generative models, there have being some attempts in the literature to make them discriminative models, i.e. to make them capable of classifying patterns and "not only" for feature learning. One interesting model concerns the Discriminative Restricted Boltzmann Machines (DRBMs) [21], which makes use of an additional input layer composed of the labels of a given sample. Such approach models the joint distribution of the inputs and their associated target (label) classes.

In this work, we opted to use a much simpler approach. Actually, if one adds one more input unit with the corresponding label of the sample ('1' or '0' in our case, since we have a binary classification problem, i.e. malware or not malware), we still have a generative RBM, since it "does not know" we are using the label as an input, and the very same formulation presented in Section 2 can be employed. Figure 2 presents the RBM used for classification purposes. The dashed unit in the visible layer stores the label of that input sample.

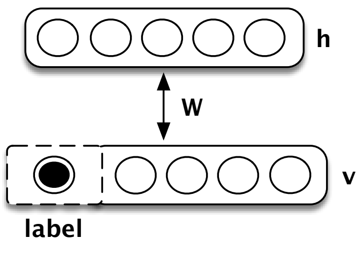

Fig. 2. Restricted Boltzmann Machine used for classification purposes.

Although any other formulation can be used with respect to RBMs or DRBMs, we opted to use the above approach due to its simplicity, as well as we can use the same formulation presented in the previous section. As a matter of fact, the problem of classification now becomes a task of input data reconstruction. After learning the weights over a training set, the unseen samples (test set) can be classified by just using each test data as an input, computing its reconstruction and taking the value of the first visible unit (after reconstruction) as its final (predicted) label.

4. Experimental Evaluation

In this section, we describe the methodology applied to assess the robustness of Restricted Boltzmann Machines to learn features for malware identification.

4.1. RBM Parameter Setting-up

One of the main problems related to RBMs concerns their fine-tuning, since such techniques are very sensitive to their parameter selection. Papa et al. [27, 28, 30] proposed to employ meta-heuristic techniques for such purpose, where the idea is to model the problem of fine-tuning parameters as an optimization task, in which the Minimum Square Error (MSE) of the reconstruction step over the training data is then used as the fitness function.

In this paper, we employed three approaches based on the well-known Harmony Search [10] technique to fine-tune RBMs:

- Naïve Harmony Search (HS): it comprises the original version of the algorithm, that aims at modeling the problem of function minimization based on the way musicians create songs with optimum harmonies. Such technique employs two parameters to address this problem, being the *Harmony Memory Considering Rate* (HMCR) in charge of creating new solutions based on the previous experience of the music player, and the *Pitch Adjusting Rate* (PAR) is responsible for applying some small disturbance in the solution created with HMCR in order to avoid traps from local optima.
- Improved Harmony Search (IHS) [23]: such approach uses dynamic values for both HMCR and PAR variables, which are updated at each iteration with new values that fall within the range [HMCR$_{min}$,HMCR$_{max}$] and [PAR$_{min}$,PAR$_{max}$], respectively. Since PAR is computed using a bandwidth variable ϱ, IHS also requires such variable be bounded within the range $\varrho_{min}, \varrho_{max}$.
- Parameter Setting-Free Harmony Search (PSF-HS) [11]: such approach was proposed in order to avoid the fine-tuning parameter step regarding HMCR, PAR and ϱ variables. Roughly speaking, the idea is to obtain new HMCR and PAR values based on previous computations of such variables whenever a new solution is created using them. However, an initial estimative of such variables is required by the algorithm.

An RBM has four main parameters to be fine-tuned: number of hidden neurons n, learning rate η, momentum α and weight decay λ [27, 28, 30, 36]. In this work, we defined two experiments: EXP1 and EXP2, which aim at evaluating different ranges concerning the number of neurons, which corresponds to the number of features. Both experiments use the very same of ranges for $\alpha \in [0.1, 0.9]$ and $\lambda \in [10^{-5}, 10^{-2}]$, but EXP1 employs $n \in [5, 15]$, and EXP2 uses $n \in [76, 114]$. Therefore, the main idea is to evaluate whether the number of features is really crucial for malware identification or not. Finally, we employed 10 agents over 50 iterations considering all aforementioned techniques. Table 1 presents the parameter configuration for each optimization technique[5]. Also, we employed $T = 100$ as the number of epochs for RBM learning weights procedure. Considering PSF-HS, we start with HMCR=PAR= 0.7 and $\varrho = 0.1$.

TABLE 1
Parameter configuration.

Technique	Parameters
HS	$HMCR = 0.7$, $PAR = 0.7$, $\varrho = 0.1$
IHS	$HMCR = 0.7$, $PAR_{MIN} = 0.1$ $PAR_{MAX} = 1.0$, $\varrho_{MIN} = 0.1$ $\varrho_{MAX} = 0.5$

In order to provide a more robust analysis of the results, we conducted a cross-validation with 10 runnings for all datasets, being 75% of the dataset used for training and the remaining 25% employed for testing purposes.

4.2. Dataset

In this work, we designed the DroidWare[6] dataset. We manually collected 278 benign application samples and 121 malware samples from android OS. Malware samples were obtained and analyzed by VirusTotal[7] web site, which is an on-line scanning tool to detect malwares. Further, the non-malware data were collected from a manifest application by Google Play[8]. The dataset features are composed of 152 permissions ('1' or '0') from the android platform, and are responsible to indicate which resources and device actions can be accessed by the applications. A more detailed explanation about the dataset can be found in its home-page.

5. Notice these values have been empirically set.
6. https://github.com/RECOVI/DroidWare.git
7. http://www.virustotal.com/
8. https://play.google.com/store/

4.3. Classifiers Setting-up

In this work, we compared SVM, OPF and RBM for classification purposes. In regard to SVM implementation, we employed the LibSVM [3] with a radial basis kernel with parameters optimized through a 5-fold cross validation, and with $C \in \{2^{-5}, 2^{-3}, \ldots, 2^{13}, 2^{15}\}$ and $\gamma \in \{2^{-5}, 2^{-3}, \ldots, 2^{13}, 2^{15}\}$. With respect to OPF, we used the LibOPF library [29], and concerning RBM we used LibDEEP[9].

5. Experiments

In this section, we describe the experiments conducted to assess the robustness of the proposed approach. As aforesaid, we first used all 152 features of the DroidWare dataset, hereinafter called ORIGINAL dataset. Then, we conducted experiments in order to learn a more discriminative set of features by means of RBMs optimized with HS (RBM-HS), IHS (RBM-IHS) and PSF-HS (RBM-PSF-HS). Based on ORIGINAL dataset, we created 10 randomly generated training and testing sets, for the further application of the extracted/learned features by RBMs. Later on, such features are used to feed OPF, SVM and the RBM (i.e. the one mentioned in Section 3) classifiers, thus performing a traditional pattern recognition pipeline, i.e. training, testing and accuracy computation.

Table 2 shows the results regarding OPF classifier for the EXP1 and EXP2 experiments, being the accuracy measure the one proposed by Papa et al. [26], which considers unbalanced datasets, as follows:

$$e_{i,1} = \frac{FP_i}{|Z| - |Z_i|} \text{ and } e_{i,2} = \frac{FN_i}{|Z_i|},\ i = 1, \ldots, c, \tag{17}$$

9. https://github.com/jppbsi/LibDEEP

where $FP(i)$ and $FN(i)$ are the false positives and false negatives, respectively. That is, FP_i is the number of samples from other classes that were classified as being from the class i in Z, and FN_i is the number of samples from class i that were incorrectly classified as being from other classes in Z.

The errors $e_{i,1}$ and $e_{i,2}$ are used to define:

$$E_i = e_{i,1} + e_{i,2}, \tag{18}$$

where $E(i)$ is the partial sum error of class i. Finally, the accuracy is written as:

$$Acc = \frac{2c - \sum_{i=1}^{c} E_i}{2c} = 1 - \frac{\sum_{i=1}^{c} E_i}{2c}. \tag{19}$$

The best results according to Wilcoxon signed-rank statistical test [39] are in bold (we used a significance level of 0.05). Therefore, the best recognition accuracies were obtained by ORIGINAL and EXP2 datasets, which is quite interesting, since the average number of features (neurons) used in EXP2 is far less than using all 152 features from ORIGINAL dataset. Tables 3 and 4 present the mean values obtained by each optimization technique considering EXP1 and EXP2 experiments, respectively. If one takes into account the number of features n in EXP1 dataset, we can observe a smaller number of neurons is not interesting, since the accuracy results are quite low. On the other hand, EXP2 shows us there is no need to use all 152 features, since very good recognition rates were obtained with 98 features (RBM-PSF-HS in Table 4).

Technique	n	η	α	λ
RBM-HS	9	0.42	0.84	0.007
RBM-IHS	9	0.64	0.85	0.005
RBM-PSF-HS	9	0.54	0.78	0.005

TABLE 3
Average RBM Parameters (EXP1).

Technique	n	η	α	λ
RBM-HS	103	0.45	0.10	0.004
RBM-IHS	104	0.50	0.12	0.005
RBM-PSF-HS	98	0.58	0.11	0.004

TABLE 4
Average RBM Parameters (EXP2).

In order to provide more insightful results, we employed SVM classifier for classification purposes. Table 5 shows the mean accuracy results concerning SVM-based classification over the test sets. In this case, the ORIGINAL dataset obtained the best results so far, being followed by EXP2 and EXP1. Usually, SVM works better than OPF in higher dimensional feature spaces due to the kernel mapping process, which turns out to benefit SVM when using all 152 features. However, the best RBM-based result was obtained by RBM-IHS, which is around 7.9% less accurate than SVM only.

Table 6 displays the results using RBM as the classifier. In this case, both ORIGINAL and EXP1 datasets obtained similar results considering the Wilcoxon statistical test. Since we are using an additional visible unit for reconstruction purposes, it seems the network adapted with fewer hidden units. Actually, there is no need to use a lot of features when the label information has been encoded together with the input data with respect to the datasets used in work.

6. Conclusions

Network attacks in mobile devices have been of great concern in the last years. Since the number of malwares oriented to mobile devices has increased considerably, a special attention has been devoted to machine learning-based techniques that can handle

		RBM (EXP1)			RBM (EXP2)		
Rounds	Original	HS	IHS	PSF-HS	HS	IHS	PSF-HS
1	72.42	50.00	50.00	50.00	74.61	68.24	70.54
2	61.25	22.00	22.00	44.00	53.49	55.47	57.34
3	73.25	78.00	70.00	68.00	78.72	79.66	78.72
4	80.77	66.15	80.65	50.00	78.35	84.39	85.16
5	61.86	32.00	32.00	63.00	63.41	65.07	66.63
6	72.35	69.00	65.00	57.00	64.56	57.05	62.13
7	68.35	50.00	50.00	50.00	82.74	79.54	82.74
8	77.30	74.36	50.00	70.86	76.37	73.98	66.17
9	72.42	65.00	54.00	61.00	74.72	66.26	76.04
10	73.82	39.08	67.93	51.45	70.19	70.99	70.99
Avg.	**72.42±5.81**	57.50±18.01	52.00±16.84	54.22±8.42	**71.72±8.41**	**70.07±9.10**	**71.65±8.52**

TABLE 2
Mean accuracy considering OPF classifier (EXP1 and EXP2).

		RBM (EXP1)			RBM (EXP2)		
Rounds	Original	HS	IHS	PSF-HS	HS	IHS	PSF-HS
1	79.00	65.00	65.00	65.00	73.00	73.00	70.00
2	78.00	78.00	78.00	78.00	70.00	71.00	70.00
3	94.00	78.00	78.00	78.00	68.00	69.00	69.00
4	92.00	65.00	65.00	65.00	79.00	79.00	78.00
5	77.00	68.00	68.00	68.00	81.00	81.00	79.00
6	77.00	69.00	69.00	69.00	73.00	71.00	73.00
7	89.00	69.00	69.00	69.00	72.00	68.00	72.00
8	90.00	68.00	68.00	68.00	79.00	75.00	79.00
9	79.00	65.00	65.00	65.00	83.00	78.00	78.00
10	83.00	69.00	69.00	69.00	68.00	68.00	73.00
Avg.	**81.00±6.40**	68.50±4.58	68.50±3.58	68.50±3.52	74.60±5.20	73.30±4.49	73.90±3.59

TABLE 5
Mean accuracy considering SVM classifier (EXP1 and EXP2)

such threats in a more effective manner. Such systems are able to learn and detect new attacks, but not always with reasonable efficiency.

In this paper, we addressed the problem of unsupervised feature learning for malware identification by means of Restricted Boltzmann Machines, in which two different experiments were conducted: EXP1 and EXP2. While the former used much less hidden units, the latter one employed around ten times more hidden units (features). In order to access the effectiveness of RBMs, we employed three classifiers at the end of the pipeline: SVMs, OPF and a simple RBM version for classification purposes.

The experiments showed that using more hidden units, the discriminative ability of SVMs and OPF is increased either. On the other hand, RBMs as a classification approach worked better with fewer hidden

		RBM (EXP1)			RBM (EXP2)		
Rounds	Original	HS	IHS	PSF-HS	HS	IHS	PSF-HS
1	63.00	65.00	65.00	65.00	62.00	62.00	62.00
2	73.00	78.00	78.00	78.00	72.00	72.00	70.00
3	81.00	78.00	78.00	78.00	70.00	79.00	71.00
4	74.00	65.00	65.00	65.00	62.00	62.00	61.00
5	74.00	68.00	68.00	68.00	64.00	65.00	65.00
6	73.00	69.00	69.00	69.00	67.00	66.00	65.00
7	78.00	69.00	69.00	69.00	66.00	67.00	67.00
8	79.00	69.00	65.00	68.00	65.00	65.00	66.00
9	66.00	65.00	69.00	65.00	62.00	61.00	64.00
10	71.00	69.00	71.00	69.00	65.00	66.00	66.00
Avg.	**73.20±5.28**	**69.50±4.58**	**69.50±4.56**	**69.40±4.58**	65.50±3.23	66.50±5.12	65.70±2.96

TABLE 6
Mean accuracy considering RBM classifier (EXP1 and EXP2)

units, since they learn how to reconstruct both the input data and its label. In regard to future works, we aim at evaluating Deep Belief Networks for feature learning in the context of malware identification.

Acknowledgments

The authors would like to thank Capes, FAPESP grants #2015/00801-9, #2014/09125-3 and #2014/16250-9, CAPES PROCAD #2966/2014, as well as CNPq grants #470571/2013-6 and #306166/2014-3.

References

[1] D.H. Ackley, G.E. Hinton, and T. J. Sejnowski. A learning algorithm for boltzmann machines. In D. Waltz and J.A. Feldman, editors, *Connectionist Models and Their Implications: Readings from Cognitive Science*, pages 285–307. Ablex Publishing Corp., Norwood, NJ, USA, 1988.

[2] A. Arora, S. Garg, and S.K. Peddoju. Malware detection using network traffic analysis in android based mobile devices. In *International Conference on Next Generation Mobile Apps, Services and Technologies (NGMAST), 2014 Eighth*, pages 66–71, Sept 2014.

[3] C.-C. Chang and C.-J. Lin. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2:27:1–27:27, 2011. Software available at http://www.csie.ntu.edu.tw/~cjlin/libsvm.

[4] K. A. P. Costa, L. A. M. Pereira, R. Y. M. Nakamura, C. R. Pereira, J. P. Papa, and A. X. Falcão. A nature-inspired approach to speed up optimum-path forest clustering and its application to intrusion detection in computer networks. *Information Sciences*, 294(10):95–108, 2015. Innovative Applications of Artificial Neural Networks in Engineering.

[5] K. A. P. Costa, L. A. Silva, G. B. Martins, G. H. Rosa, C. R. Pereira, and J. P. Papa. Malware detection in android-based mobile environments using optimum-path forest. In *2015 IEEE 14th International Conference on Machine Learning and Applications*, ICMLA'15, pages 754–759, 2015.

[6] Gavrilut D., Cimpoesu M., Anton D., and Cior-

tuz L. Malware detection using perceptrons and support vector machines. In *Computation World: Future Computing, Service Computation, Cognitive, Adaptive, Content, Patterns*, pages 283–288, 2009.

[7] A. P. Felt, M. Finifter, E. Chin, S. Hanna, and D. Wagner. A survey of mobile malware in the wild. In *Proceedings of the 1st ACM Workshop on Security and Privacy in Smartphones and Mobile Devices*, SPSM '11, pages 3–14, New York, NY, USA, 2011. ACM.

[8] D. Fernandes, K. A. P. Costa, T. A. Almeida, and J. P. Papa. Sms spam filtering through optimum-path forest-based classifiers. In *14th IEEE International Conference on Machine Learning and Applications*, ICMLA'15, pages 133–137, 2015.

[9] U. Fiore, F. Palmieri, A. Castiglione, and A. Santis. Network anomaly detection with the restricted boltzmann machine. *Neurocomputing*, 122:13–23, 2013. Advances in cognitive and ubiquitous computingSelected papers from the Sixth International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS-2012).

[10] Z. W. Geem. *Music-Inspired Harmony Search Algorithm: Theory and Applications*. Springer Publishing Company, Incorporated, 1st edition, 2009.

[11] Z. W. Geem and K.-B. Sim. Parameter-setting-free harmony search algorithm. *Applied Mathematics and Computation*, 217(8):3881 – 3889, 2010.

[12] D. F. Guo, A. Sui, and T. Guo. A behavior analysis based mobile malware defense system. In *International Conference on Signal Processing and Communication Systems*, pages 1–6, 2012.

[13] J. Haifeng, C. Baojiang, and W. Jianxin. Mining mobile internet packets for malware detection. In *Ninth International Conference on P2P, Parallel, Grid, Cloud and Internet Computing*, pages 481–486, 2014.

[14] J. Hamada. New android threat gives phone a root canal. 2011. Available at http://www.symantec.com/connect/blogs/ new-android-threat-gives-phone-root-canal.

[15] G. E. Hinton. Training products of experts by minimizing contrastive divergence. *Neural Computation*, 14(8):1771–1800, 2002.

[16] G.E. Hinton. A practical guide to training restricted boltzmann machines. In G. Montavon, G.B. Orr, and K.-R. Müller, editors, *Neural Networks: Tricks of the Trade*, volume 7700 of *Lecture Notes in Computer Science*, pages 599–619. Springer Berlin Heidelberg, 2012.

[17] S. Huda, J. Abawajy, M. Alazab, M. Abdollalihian, R. Islam, and J. Yearwood. Hybrids of support vector machine wrapper and filter based framework for malware detection. *Future Generation Computer Systems*, pages –, 2014.

[18] Hyunjae Kang, Jae Wook Jang, Aziz Mohaisen, and Huy Kang Kim. Detecting and Classifying Android Malware using Static Analysis along with Creator Information. *International Journal of Distributed Sensor Networks*, 2015.

[19] M. Kruczkowski and E. N. Szynkiewicz. Support vector machine for malware analysis and classification. In *IEEE/WIC/ACM International Joint Conferences on Web Intelligence and Intelligent Agent Technologies*, volume 2, pages 415–420, 2014.

[20] J. Kwon, J. Jeong, J. Lee, and H. Lee. Droidgraph: discovering android malware by analyzing semantic behavior. In *IEEE Conference on Communications and Network Security, 2014*, pages 498–499, 2014.

[21] H. Larochelle, M. Mandel, R. Pascanu, and

Y. Bengio. Learning algorithms for the classification restricted boltzmann machine. *The Journal of Machine Learning Research*, 13(1):643–669, 2012.

[22] S. Liang and X. Du. Permission-combination-based scheme for android mobile malware detection. In *IEEE International Conference on Communications, 2014*, pages 2301–2306, June 2014.

[23] M. Mahdavi, M. Fesanghary, and E. Damangir. An improved harmony search algorithm for solving optimization problems. *Applied Mathematics and Computation*, 188(2):1567 – 1579, 2007.

[24] J. P. Papa, A. X. Falcão, V. H. C. Albuquerque, and J. M. R. S. Tavares. Efficient supervised optimum-path forest classification for large datasets. *Pattern Recognition*, 45(1):512–520, 2012.

[25] J. P. Papa, A. X. Falcão, and C. T. N. Suzuki. Supervised pattern classification based on optimum-path forest. *International Journal of Imaging Systems and Technology*, 19:120–131, 2009.

[26] J. P. Papa, A. X. Falcão, and C. T. N. Suzuki. Supervised pattern classification based on optimum-path forest. *International Journal of Imaging Systems and Technology*, 19(2):120–131, 2009.

[27] J. P. Papa, G. H. Rosa, K. A. P. Costa, A. N. Marana, W. Scheirer, and D. D. Cox. On the model selection of bernoulli restricted boltzmann machines through harmony search. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 1449–1450, 2015.

[28] J. P. Papa, G. H. Rosa, A. N. Marana, W. Scheirer, and D. D. Cox. Model selection for discriminative restricted boltzmann machines through meta-heuristic techniques. *Journal of Computational Science*, 9:14–18, 2015. Computational Science at the Gates of Nature.

[29] J. P. Papa, C. T. N. S., and A. X. Falcão. *LibOPF: A library for the design of optimum-path forest classifiers*, 2014. Software version 2.1 available at http://www.ic.unicamp.br/~afalcao/LibOPF.

[30] J. P. Papa, W. Scheirer, and D. D. Cox. Fine-tuning deep belief networks using harmony search. *Applied Soft Computing*, 46:875–885, 2015.

[31] N. Penning, M. Hoffman, J. Nikolai, and Yong Wang. Mobile malware security challenges and cloud-based detection. In *International Conference on Collaboration Technologies and Systems*, pages 181–188, 2014.

[32] C. R. Pereira, R. Y. M. Nakamura, K. A. P. Costa, and J. P. Papa. An optimum-path forest framework for intrusion detection in computer networks. *Engineering Applications of Artificial Intelligence*, 25(6):1226–1234, 2012.

[33] A. Shabtai, L. Tenenboim-Chekina, D. Mimran, L. Rokach, B. Shapira, and Y. Elovici. Mobile malware detection through analysis of deviations in application network behavior. *Computers & Security*, 43:1—18, 2014.

[34] L. A. Silva, K. A. P. Costa, P. B. Ribeiro, D. Fernandes, and J P Papa. On the feasibility of optimum-path forest in the context of internet-of-things-based applications. *Recent Patents on Signal Processing*, 5(1):52–60, 2015.

[35] L. A. Silva, K. A. P. da Costa, P. B. Ribeiro, G. H. Rosa, and J. P. Papa. Learning spam features using restricted boltzmann machines. *IADIS International Journal on Computer Science and Information Systems*, 11(1):99–114, 2015.

[36] L. A. Silva, P. B. Ribeiro, G. H. Rosa, K. A. P.

Costa, and J. P. Papa. Parameter setting-free harmony search optimization of restricted boltzmann machines and its applications to spam detection. In *12th International Conference on Applied Computing*, 2015. (accepted for publication).

[37] A. Skovoroda and D. Gamayunov. Review of the mobile malware detection approaches. In *Distributed and Network-Based Processing, 2015 23rd Euromicro International Conference on Parallel*, pages 600–603, 2015.

[38] M. Welling, M. Rosen-zvi, and G.E. Hinton. Exponential family harmoniums with an application to information retrieval. In L.K. Saul, Y. Weiss, and L. Bottou, editors, *Advances in Neural Information Processing Systems 17*, pages 1481–1488. MIT Press, 2005.

[39] F. Wilcoxon. Individual comparisons by ranking methods. *Biometrics Bulletin*, 1(6):80–83, 1945.

[40] S.Y. Yerima, S. Sezer, and G. McWilliams. Analysis of bayesian classification-based approaches for android malware detection. *Information Security, IET*, 8(1):25–36, Jan 2014.

[41] Z. Yuan, Y. Lu, Z. Wang, and Y. Xue. Droid-sec: Deep learning in android malware detection. *ACM SIGCOMM Computer Communication Review*, 44(4):371–372, August 2014.

[42] Z. Yuan, Y. Lu, and Y. Xue. Droiddetector: Android malware characterization and detection using deep learning. *Tsinghua Science and Technology*, 21(1):371–372, 2016.

[43] Z. Yuan, Y. Lu, Y. Xue, and Z. Wang. Droid-sec: deep learning in android malware detection. In *ACM SIGCOMM 2014 Conference, SIGCOMM'14, Chicago, IL, USA, August 17-22, 2014*, pages 371–372, 2014.