Two Semi-supervised Approaches to Malware Detection with Neural Networks

Jan Koza[1], Marek Krčál[2], Martin Holeňa[1]

[1] Faculty of Information Technology, Czech Technical University, Thákurova 9, Prague, Czech Republic
[2] Rossum Czech Republic, Dobratická 523, Prague, Czech Republic

Abstract: Semi-supervised learning is characterized by using the additional information from the unlabeled data. In this paper, we compare two semi-supervised algorithms for deep neural networks on a large real-world malware dataset. Specifically, we evaluate the performance of a rather straightforward method called Pseudo-labeling, which uses unlabeled samples, classified with high confidence, as if they were the actual labels. The second approach is based on an idea to increase the consistency of the network's prediction under altered circumstances. We implemented such an algorithm called Π-model, which compares outputs with different data augmentation and different dropout setting. As a baseline, we also provide results of the same deep network, trained in the fully supervised mode using only the labeled data. We analyze the prediction accuracy of the algorithms in relation to the size of the labeled part of the training dataset.

1 Introduction

One of the application domains that pay the most attention to the progress of and new developments in machine learning is malware detection. Vendors of antivirus software cannot keep up with the increasing number of malicious programs and their increasingly sophisticated obfuscation and polymorphism without using more and more advanced machine learning methods, most importantly, methods for anomaly detection, classification and pattern recognition.

The most successful machine learning methods for classification and pattern recognition definitely include artificial neural networks (ANN), especially deep networks. However, they have a high number of degrees of freedom, thus requiring a large amount of labeled training data, whereas most of the data for malware detection is unlabeled because its labeling requires expensive involvement of human experts. One possible way how to tackle the lack of training data is semi-supervised learning. In a narrow sense, this means supervised learning that simultaneously to labels also uses some information from additional unlabeled data, in a broad sense any combination of supervised learning and unlabeled data, e.g., unsupervised learning followed by supervised learning. In the context of malware detection, however, semi-supervised ANN learning is only emerging [20, 21]. The work in progress reported in this paper is a small contribution to it. It restricts attention only to two methods of semi-supervised ANN learning, approaches not relying on neural networks are outside its scope.

The next section briefly reviews using ANN in malware detection and the overlapping area of network intrusion detection. In Section 3, several important methods for semi-supervised ANN learning are recalled, two of which have been implemented for our research. The core Section 4 describes several experiments with a real-world malware dataset, and reports their results.

2 Neural Networks in Malware and Network Intrusion Detection

As malware detection is strongly interconnected with and closely related to network intrusion detection, using ANN will be reviewed here in both areas. Probably the first proposal to use neural networks in them was in 1990 by Lunt [15] and was implemented two years later [4] in a network trained on inputs from audit log files.

The authors of [25] employed user commands as input, but rather than trying to learn benign and malicious command sequences, they were detecting anomalies in frequency histograms of user commands calculated for each user.

The paper by Cannady [3] summarised ANN advantages and disadvantages for misuse detection. As the two main advantages, the flexibility with respect to incomplete, distorted and noisy data, and the generalization ability are viewed, whereas as the main disadvantage, the ANN black-box nature.

In the late 1990s and early 2000s, self-organizing maps were quite popular in this context [2, 5, 24]. In particular, Depren et al. [5] used a hierarchical model where misuse detection based on self-organizing maps (SOMs) was coupled with random forest-based rule system to provide not only high precision, but also some sort of explanation.

Much research has been devoted to comparing different kinds of ANN, or more generally, different classifiers including one or more kinds of ANN, on real-world malware detection or intrusion detection data. Probably the most popular among such data is an extensive intrusion detection dataset that was used at the 1999 KDD Cup [29]. Zhang et al. [33] compared five different kinds of ANN. Mukkamala et al. [19] compared a multilayer perceptron (MLP) with support vector machines.

Among more recent ANN applications to malware and network intrusion detection, [14] should be mentioned for

using synthetically generated attack samples to train an MLP, as well as [30] for a malware detection with recurrent networks. Expectedly, the kinds of ANN applied to these two areas during the last decade are most often deep networks [1, 7, 10]. In [16], deep learning was used together with spectral clustering to improve the detection rate of low frequency network attacks. ability to process raw inputs and learn their own features. Saxe et al. [26] employed a convolutional neural network (CNN) to extract features that were subsequently used as the input for an MLP detecting malicious activities. CNNs seem to be particularly suitable to learn spatial features of network traffic [31, 32]. In [31], a CNN was in addition combined with a long short term memory learning temporal features from multiple network packets.

To our best knowledge, there were so far only two particular ANN applications to malware or network intrusion detection that included semi-supervised learning in the narrow sense. In [20], various settings of semi-supervised ladder networks (see Section 3) were compared on the above mentioned intrusion detection dataset [29]. In [21] (cf. also the thesis [27]), skipgram networks [17] extended with semi-supervised learning based on Pseudo-labels (see Section 3) were used for Android malware detection. Skipgrams are neural networks embedding large sets of structured non-numeric data into low-dimensional vector spaces. Whereas in [17], skipgrams were proposed for the embedding of text (word2vec), the input set in [21] is the set of rooted subgraphs around every node of three dependency graphs representing the API dependencies, permission dependencies, and information source and sink dependencies of the considered Android application. However, skipgrams were not used directly for malware detection in [21], only for representation learning of the structured input, whereas the malware detection itself was performed by a support vector machine. So far, no semi-supervised neural networks have been used directly for malware detection, and also none have been used with unstructured inputs simply listing values of the evaluated features, which are encountered much more frequently than dependency matrices.

3 Semi-supervised Learning of Neural Networks

According to the overview paper [22], the following approaches are most important for semi-supervised learning of neural networks, especially deep networks:

(i) *Pseudo-labels* [13], which are ANN predictions of the correct class for unlabeled data, provided the network has a sufficient confidence in such a prediction. Formally, a prediction serves as a pseudo-label for an unlabeled input x if

$$\arg\max_{c\in C} f_c(x) \geq \vartheta \sum_{c\in C} f_c, \tag{1}$$

where C denotes the set of classes, $f_c(x)$ the activity of the output neuron corresponding to the class $c \in C$ for the input x, and $\vartheta \in (0,1)$ is a given threshold.

(ii) *Increasing the consistency* of predictions for the same input between two instances of a neural network differing through a random perturbation. Such a perturbation is typically introduced through random noise or through dropout. The overall loss function minimized during semi-supervised learning is then the superposition of the loss of supervised learning and a loss reflecting the inconsistency of the considered ANN instances. This approach was first applied in [23] to *ladder networks*, which are basically chained denoising autoencoders. In [12], two similar kinds of neural networks using this approach to semi-supervised ANN learning were proposed that can be viewed as simplifications of ladder networks. The first kind, called Π-*model*, evaluates both randomly differing ANN instances on each minibatch of data. The second kind, called *temporal ensembling*, evaluates only one of them and then uses its predictions in the inconsistency loss. As a compensation, predictions from multiple previous network evaluations are aggregated into an ensemble prediction.

(iii) Due to targets changing only once per epoch, temporal ensembling becomes unwieldy when learning large datasets. To overcome this problem, an approach called *mean teacher* has been proposed in [28]. Instead of aggregating predictions, it aggregates weights, more precisely, averages them.

(iv) In [18], the most sophisticated among the four considered approaches has been proposed, called *virtual adversarial training*, due to using a loss function proposed by Goodfellow et al. to train networks against adversarial inputs [8], and known as *adversarial loss*:

$$L_{\text{adv}}(x,\theta) = D[q(\cdot|x), p(\cdot|x+r_{\text{adv}};\theta)] \tag{2}$$

$$\text{where } r_{\text{adv}} = \arg\max_{\|r\|\leq\varepsilon} D[q(\cdot|x), p(\cdot|x+r;\theta)], \tag{3}$$

In (2)–(3), $q(\cdot|x)$ represents our knowledge of the true conditional distribution of labels given a particular input x, whereas $p(\cdot|x;\theta)$ represents the corresponding distribution implied by the neural network for particular values of their parameters θ, $\varepsilon > 0$ and D is some non-negative function on pairs of probability distribution, such as cross entropy, which was used in [18]. And the term "virtual" refers to the fact that in supervised learning, this loss needs to be minimized on unlabeled inputs instead on adversarial ones

So far, we have managed to implement the first two of those approaches, the second in both variants Π-model and temporal ensembling. Some details of our implementation are given below.

3.1 Our Implementation of ANN Learning

Most parts of the two algorithms we used share the same implementation. Fundamentally, they only differ in the way they compute the unsupervised component of the loss function. Firstly, both methods use the same MLP architecture with ReLU as the activation function in the hidden layers and utilize the same optimizing algorithm Adam [11] with the initial learning rate set to 0.001, $\beta_1 = 0.99$, and $\beta_2 = 0.999$. As was shown above, the optimized loss function is defined as a weighted sum of supervised and unsupervised loss $L = L_S + w(t)L_U$. The weight $w(t)$ depends on the ratio between the number of labeled and all data, and the current epoch. Following a proposal in [22], we ramp up the value of the weight using a Gaussian curve:$w(t) = w_{\max} \frac{|\mathscr{L}|}{|\mathscr{L}|+|\mathscr{U}|} \exp\left(-5(1-t)^2\right)$, where $t = \max(\frac{e}{r_u}, 1)$, e is number of the current epoch,r_u is the length of the rump up period and w_{max} is a parameter specifying the maximum weight. Increasing the weight of the unsupervised loss during the training is necessary as the network needs to learn to classify the supervised data first. Eventually, it can learn to incorporate the unlabeled information as well. Similarly, at the later phase of the training, the learning rate and the β_1 parameter of the Adam optimizer are decreased to improve the exploitation:$lr_e = w_d lr_{e-1}$ and $\beta_1 = 0.4w_d + 0.5$, where $w_d = \exp\left(-12.5t^2\right)$,$t = \max(\frac{e}{r_d}, 1)$ and r_d is the length of the ramp down period. We also included a type of elitism to select the resulting model with the lowest total loss per epoch calculated with the maximal weight for the unsupervised component instead of a weight in the current epoch.

The unsupervised loss in the Pseudo-labeling algorithm is calculated using cross-entropy between network's predictions and pseudo-labels, but only for predictions with confidence above a specified threshold ϑ (cf. (1)). We compute the vector of pseudo-labels y' for every data sample x using the corresponding network output $f(x)$ in the following manner:

$$y'_i = \begin{cases} 1 & \text{if } i = \text{argmax}_{i'} f_{i'}(x) \\ 0 & \text{otherwise} \end{cases} \tag{4}$$

Then the resulting formula based on cross entropy for the unsupervised loss component L_U of a particular data sample x is:

$$L_U(x) = -\sum_{i=1}^{|C|} y'_i \log(y_i), \tag{5}$$

where $|C|$ is the number of classes.

We also implemented two variants of the consistency preserving, self-ensembling algorithms: The Π-model and the temporal ensembling. Both approaches use mean squared error (MSE) to compute unsupervised loss. What is different is the target for which MSE is evaluated. The Π-model compares two predictions of the same state of the network using different inputs and different dropped out neurons. To augment the data for the second prediction, we multiplied the input feature vector with a noise sampled from normal distribution $\mathscr{N}(1, \sigma^2)$. We chose to multiply the data with the noise instead of adding it because it is invariant to the differing variances of the individual features.

The second variant, temporal ensembling, compares the prediction of the network in the current epoch with the predictions obtained in the previous epoch. The dropout and data augmentation can be used as well. So the unsupervised loss L_U for this approach is calculated as follows:

$$L_U(x) = \sum_{i=1}^{|C|} (y_i - \tilde{y}_i)^2, \tag{6}$$

where y is the current output of the network in the training step and $\tilde{y}$ is the output of the network in a different state or for augmented input.

Our open-source implementation is publicly available at https://github.com/c0zzy/semi-supervised-ann.

3.2 Validation using a simple artificial experiment

Firstly, we tried our implementations of two semi-supervised methods mentioned above and a fully supervised baseline on a two-dimensional example. We chose simple generated moon-shaped data, which are often used for testing of classification or clustering algorithms. The data consist of two classes, that are linearly inseparable but do not overlap so that the classification can be performed with no error. The advantage is that we can easily visualize the classification decision border in two dimensions and examine the behavior of the algorithm. For every method in this experiment, we used the same MLP architecture with two hidden layers, the first having 64 neurons and the second 32 neurons.

In Figure 1, we present two different arrangements of labeled and unlabeled data, each solved by the fully supervised learning, Pseudo-labeling, and Π-model. In the first experiment, we tested the ability of the algorithm to learn from a small amount of data, there are two moon-shaped clusters, each having 1000 samples, where only 16 of each are labeled. We let each network to train for 300 epochs. Even though the supervised learning had available samples distributed over the whole cluster, it was not able to learn the correct shape using only 32 samples. The Pseudo-labeling algorithm could not improve the results using the unlabeled data. However, the results of the Π-model are notably better as it managed to capture the moon shape quite well.

In the second experiment, we tried if the algorithms can deal with a drift in the training data. This time we used clusters with 10,000 samples and labeled only 1000 points that lie near the center, for each class. We trained the networks for 100 epochs as having it run longer did not improve the results of either of the methods. The supervised

algorithm could only use the labeled data that are linearly separable. So it learned to classify the labeled data with zero error, and we present it only as a baseline for comparison. Pseudo-labeling again failed to use the information contained in the unlabeled data, and its accuracy was similar to the fully supervised learning. Also in this task, the Π-model was able to use the smoothness of the data and performed the best of three methods. To quantify the results, we summarized the prediction accuracy tested on the whole clusters in Table 1.

Table 1: A summary of test accuracy on the moonshaped data. The table compares Pseudo-labeling, Π-model, and fully supervised learning on a test data covering the whole moon cluster. There are results of two experiments. In the first one, only 16 points out of 1000 were uniformly selected and labeled for both classes. In the second, we labeled 1000 points in the center out of 10,000 samples for both classes.

Method	Test case	
	16 pts uniform	1000 pts in center
Supervised	89.1 %	46.2 %
Pseudo-label	85.4 %	42.9 %
Π-model	95.7 %	76.0 %

Completing these experiments, we observed that the results of the Pseudo-labeling correspond to the idea behind the algorithm. It makes the network's decision more confident as it uses the interim predictions as if they were the true labels. Also, the decision border did not seem to converge to a stable finale state throughout the learning. It kept shifting closer to one or the other class, roughly in the range where the confidence of the supervised learning was low. We managed to get decent results using the Π-model, and it proved to be able to capture the smooth distribution of data. However, the algorithm was susceptible to inappropriate setting of hyperparameters. It often happened that one class became dominant during the training, and the Π-model could not recover from that.

4 Experiments with a Real-World Malware Dataset

4.1 Data

We tested our implementation using a large real-world malware detection dataset containing anonymized data provided by the company Avast. The data concern Windows Portable Executables (PE) files, which were collected during 380 weeks. It consists of 540 real-valued features derived directly from the binary PE files. Unfortunately, the company did not reveal the semantics of the individual features. Each file is labeled with one of the five classes: malware, adware, infected, potentially unwanted program, and clean. There were some features with zero or very low variance in the dataset. Therefore we used principal component analysis (PCA) to reduce the dimensionality of the feature space and speed up the training. First, we min-max normalized the data between 0 and 1, and then we projected them to the subspace spanned by the 128 main components while keeping more than 99 % of the explained variance.

4.2 Experimental Design

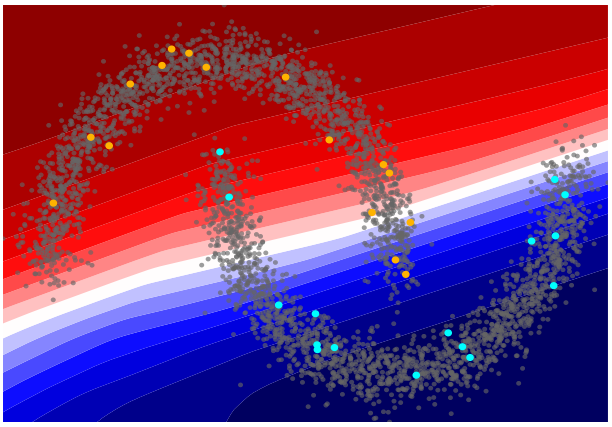
At first, we analyzed the hyperparameters of each algorithm and optimized those that we expected to have the greatest impact on the results during early tests of our implementation. We chose the data from five weeks between 50th and 55th week. We performed stratified random sampling and selected 10,000 training and 5000 testing records. We kept only 5 % of the labeled from the training set, and the rest remained unlabeled. Using this data, we evaluated the classification accuracy for various sets of hyperparameters.

For the Pseudo-labeling algorithm, we optimized the threshold ϑ and the maximal weight w_{max} for the unsupervised loss component. For the consistency preserving algorithms, we optimized the standard deviation σ of the noise used in data augmentation and again the parameter w_{max}. Furthermore, we repeated the search of parameters for all six combinations of variants of the algorithm, which were: Π-model or temporal ensembling and whether to use dropout, augmentation or both. We took the parameters from the following sets:

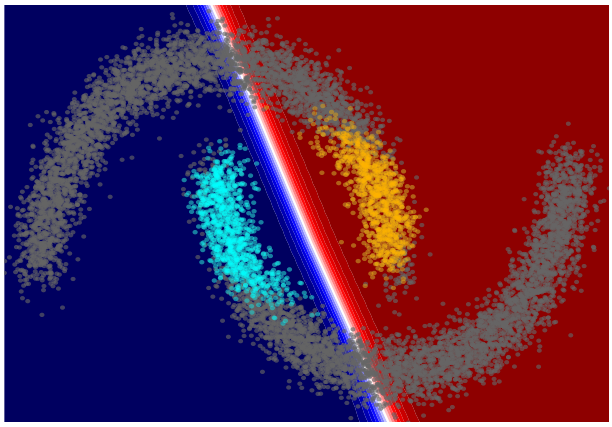
$$w_{max} \in \{0.1, 1, 2, 5, 10, 15, 20, 30, 50\},$$
$$\sigma \in \{0.01, 0.05, 0.1, 0.15, 0.2, 0.3, 0.5\},$$
$$\vartheta \in \{0.5, 0.7, 0.8, 0.9, 0.95, 0.98, 0.99\}.$$

However, because of the high time requirements, we restricted attention among the two similar models proposed in [12] only to the Π-model. For the same reason, we did not perform the full factorial search through all possible combinations. Instead we optimized only one parameter at time, keeping others on default values which were: $w_{max} = 30$, $\sigma = 0.1$ and $\vartheta = 0.9$. Among all these tuned hyperparameters, the most critical from the point of view of the predictive accuracy were the maximal weight, and the standard deviation of the Π-model noise. The rest of the hyperparameters we used as stated in the original papers or we modified them slightly according to our observations because the domain of our dataset is entirely different. The final values of the chosen hyperparameters used in experiments follow in Table 2. For the fully supervised training, we enabled the dropout and the data augmentation in the same manner as with the Π-model. In every experiment, we used the same MLP architecture with five layers and the topology 128-96-64-32-5.

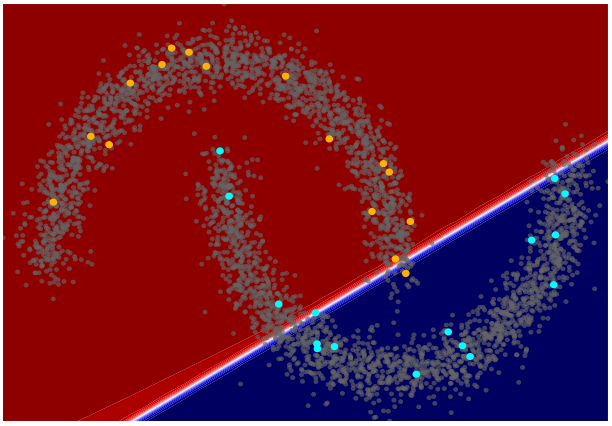
Then we measured the performance of the Pseudo-labeling, Π-model, and the purely supervised baseline



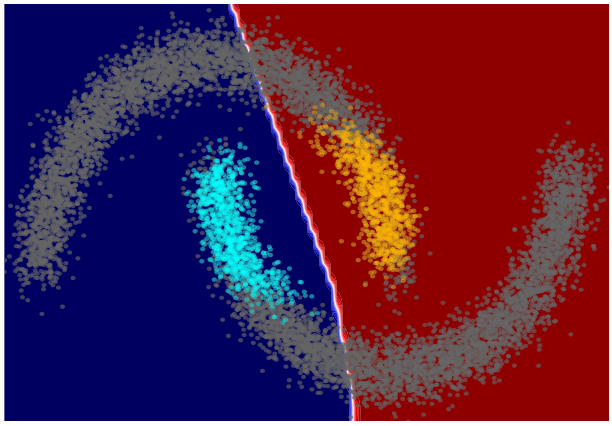
(a) Supervised



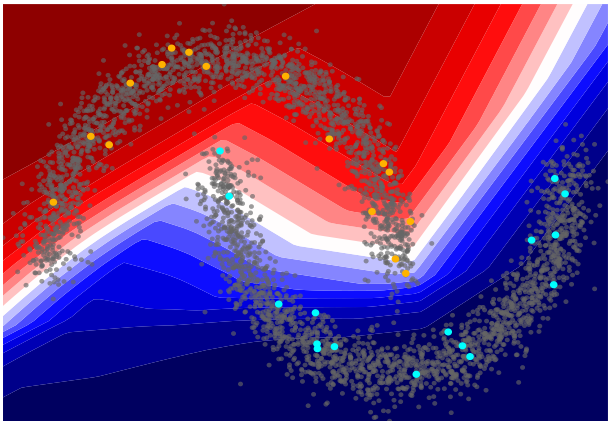
(b) Supervised



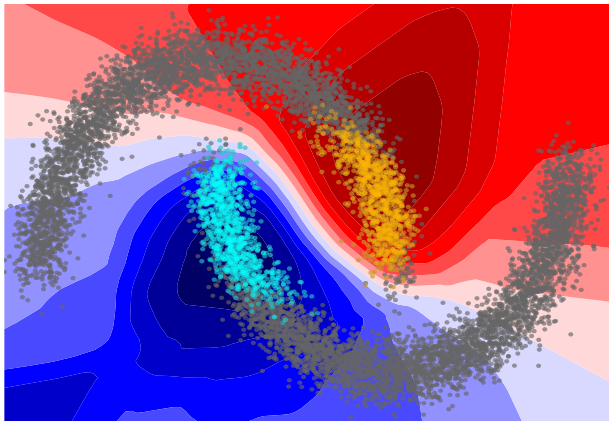
(c) Pseudo-labeling



(d) Pseudo-labeling



(e) Π-model



(f) Π-model

Figure 1: A comparison of the decision border of three algorithms on simple moon-shaped data. The decision border is visualized as a transition from blue to red. The saturation expresses the classification confidence of the network. The labeled data are shown as cyan or orange circles, while unlabeled are drawn in gray. On the left side, we randomly labeled only 16 samples out of 2000 from each class. On the right side, we labeled 1000 samples close to the center out of 5000 from each class.

Table 2: Final setting of model hyperparameters.

Common	
Number of training epochs	100
Training batch size	100
Weight ramp-up period r_u	70
Optimizer ramp-down period r_d	20
Initial learning rate	0.001
Pseudo-Labeling	
Pseudo-labeling threshold ϑ	0.9
Maximal weight w_{max}	10
Consistency preserving	
Consistency preserving variant	Π-model
Use dropout	Yes
Use data augmentation	Yes
Maximal weight w_{max}	20
Standard deviation σ of the noise	0.2

for different proportions of labeled data. We varied the ratio $r = |\mathscr{L}| : (|\mathscr{L}| + |\mathscr{U}|)$ in the set of values $\{0.5\%, 1\%, 2\%, 5\%, 10\%, 25\%, 50\%, 75\%\}$. As the training union of labeled and unlabeled data, we took 10,000 stratified samples from 5 consequent weeks and split them in the considered ratios. Then we trained 20 separate instances of the network and calculated the average accuracy on a stratified test set of size 5000 for them. We repeated this experiment for four arbitrarily chosen distinct groups of weeks: 1-5, 51-55, 101-105, and 151-155. We also evaluated the performance of trained networks on the data from all of the following weeks. This is particularly interesting from the point of view of the considered application domain. Because the structure of malware changes over time, the prediction accuracy of the newer data tends to get worse. That means that if semi-supervised learning could overcome this problem, it could be beneficial. Therefore, we tried to take the data from newer periods than the labeled weeks as the unlabeled training set. So we trained the network using labeled data together with unlabeled data from several weeks later. Unfortunately, we did not manage to outperform the standard fully supervised learning this way using any of the implemented methods, so we refrained from it. We present the results of these experiments in the following section.

4.3 Results and Their Discussion

Using the hyperparameters setting presented in the previous section, we measured the average test accuracy of 20 training runs of our three implementations in relation to the proportion of the labeled data in the training data set. The results can be found in Table 3. We can see that the performance of the fully supervised learning depends on the number of labeled data as it is the only learning source for the network. The results of the semi-supervised algorithms Pseudo-labeling and Π-model are more interesting. Both algorithms bring a slight increase in the accuracy of low ratios of the labels. The most noticeable improvement is when there are only around 1 or 2 % of labels. When the ratio gets above 10 %, the accuracy gain is negligible, and for the higher values, the semi-supervised effect is even negative. Also, it seems that Π-model outperforms Pseudo-labeling, as its accuracy is higher in most of the measurements.

To verify our observations, we tested whether the distributions of predictive accuracy achieved by the three considered methods significantly differ from each other. Those distributions are for the considered ratios of labeled to all data shown in Figure 2, but – due to lack of space – only for the networks trained on data from the first five weeks. Firstly, we applied the Friedman test [6] to reject the hypothesis that all three methods can be considered equal. Then we performed a post hoc pairwise test to find out among which of them there were differences at the 5 % level of family-wise significance with Holm [9] correction. We took the data from all of the following weeks and evaluated the accuracy for all considered ratios of labeled and all data, training for each of them 20 models. A significant difference between the compared methods was found for 80 among the 96 compared pairs corresponding to the 32 combinations of training weeks and ratios. We summarized the results in Table 4, where we compared the average accuracy for the three implemented methods. When we consider only tests with ratio up to 5 %, where the improvement was visible, then the Pseudo-labeling was significantly better than supervised learning in 3 cases and the Π-model in 11 cases. Pseudo-labeling was significantly better than Π-model only in 3 out of 14 significant comparisons.

We also visualized the progress of the classification accuracy over time for networks trained during three arbitrarily chosen sequences of 5 contiguous weeks in Figure 3. To capture the variance of the results, we plotted three quartiles. Because the accuracy oscillated greatly through the individual weeks, we used a moving average with a window size of five weeks to smooth the curves (the accuracy during the first five weeks, for which a window of that size has not been available, is dashed). We can see that both semi-supervised algorithms slightly improved the accuracy of the network on the roughly first 30 weeks. The Pseudo-labeling is around 1 or 2 % better than supervised learning, while Π-model gets another 1 or 2 % above the Pseudo-labeling. However, all three trained networks share the trend of decreasing predictive accuracy during the early weeks when the moving average has been applied, though the number of such weeks is network-specific. After around 40 weeks, the results of all three methods are very similar. As the properties of the data shift over time, the overall results on the data beyond 50 weeks got considerably worse and fluctuated more for all methods.

Table 3: Comparison of the Π-model, Pseudo-labeling and the supervised baseline, for different ratios of labeled and all data. The table depicts the percentage of the average testing accuracy in four different periods. The S columns contain the results of the supervised baseline, the Δ Ps and Δ Π columns show the difference using pseudo-labeling and Π-model, respectively.

Ratio	Weeks: 1–5			Weeks: 51–55			Weeks: 101–105			Weeks: 151–155		
	S	Δ Ps	Δ Π	S	Δ Ps	Δ Π	S	Δ Ps	Δ Π	S	Δ Ps	Δ Π
0.5 %	67.9	+0.4	+3.1	63.9	+2.8	+3.5	56.8	+5.2	+6.8	67.6	+0.3	+1.9
1 %	71.0	+1.7	+4.5	67.1	+5.0	+6.4	61.8	+6.9	+9.0	70.3	+5.7	+6.4
2 %	76.8	+1.3	+2.5	73.9	+3.4	+5.7	69.7	+5.9	+6.2	76.6	+1.9	+2.0
5 %	82.4	-0.1	+1.1	82.2	+0.4	+2.4	77.8	+3.7	+3.3	80.4	+0.2	+0.8
10 %	85.1	+0.0	+1.1	86.1	-0.4	+0.8	83.2	+1.0	+1.1	81.7	+0.3	+0.6
25 %	88.3	-0.4	+0.3	89.2	-0.5	+0.1	87.4	+0.7	+0.3	83.1	+0.3	+0.3
50 %	89.9	-0.4	-0.1	90.6	-0.7	+0.0	89.8	-0.3	-0.2	84.2	-0.1	-0.2
75 %	90.4	-0.1	-0.1	91.2	-0.3	-0.3	90.7	-0.1	-0.4	84.4	+0.3	-0.2
100 %	90.9			91.4			91.3			84.8		

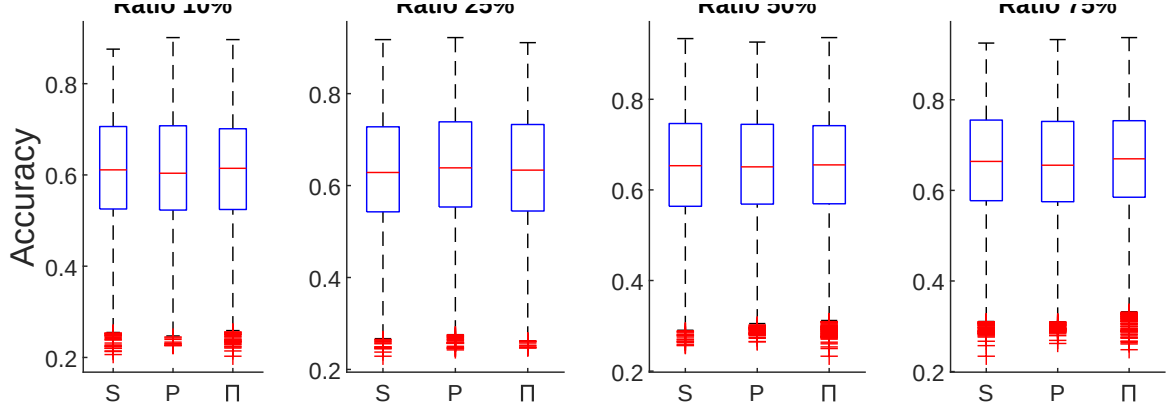


Figure 2: Boxplots summarizing the distributions of predictive accuracy achieved by supervised learning (S), pseudo-labeling (P) and the Π-model (Π) for the considered ratios of labeled to all data and the networks trained on data from the first five weeks

5 Conclusion

In this paper, we presented an application of semi-supervised learning of deep neural networks to malware data. At the beginning, we recalled the current state of detecting malware with artificial neural networks and introduced the principles of neural semi-supervised learning. Then we outlined four semi-supervised approaches to deep learning. We covered two semi-supervised algorithms, Pseudo-labeling and Π-model in more detail and compared them with the fully supervised baseline. We evaluated the classification accuracy on a real-world malware dataset divided to 380 weeks by the time of the first recording of the respective binary file. Despite having been developed for the classification of image data, the results showed that both methods could improve the performance of a neural network on malware data. However, implemented algorithms have the limitation of being beneficial only when the proportion of labeled data is low, ideally around 1 %. We have also found that these semi-supervised methods can increase the accuracy on data newer than the training set, for which drift in structure is likely to occur, but only to a certain extent. Based on our experiments, the slightly more complex algorithm Π-model has got slightly better results than Pseudo-labeling in most cases.

Acknowledgement

The research reported in this paper has been supported by the Czech Science Foundation (GAČR) grant 18-18080S. For the employed data and the work of M. Krčál, his support through Avast fellowship is appreciated. Access to computing and storage facilities owned by parties and projects contributing to the National Grid Infrastructure MetaCentrum provided under the program "Projects of Large Research, Development, and Innovations Infrastructures" (CESNET LM2015042), is greatly appreciated.

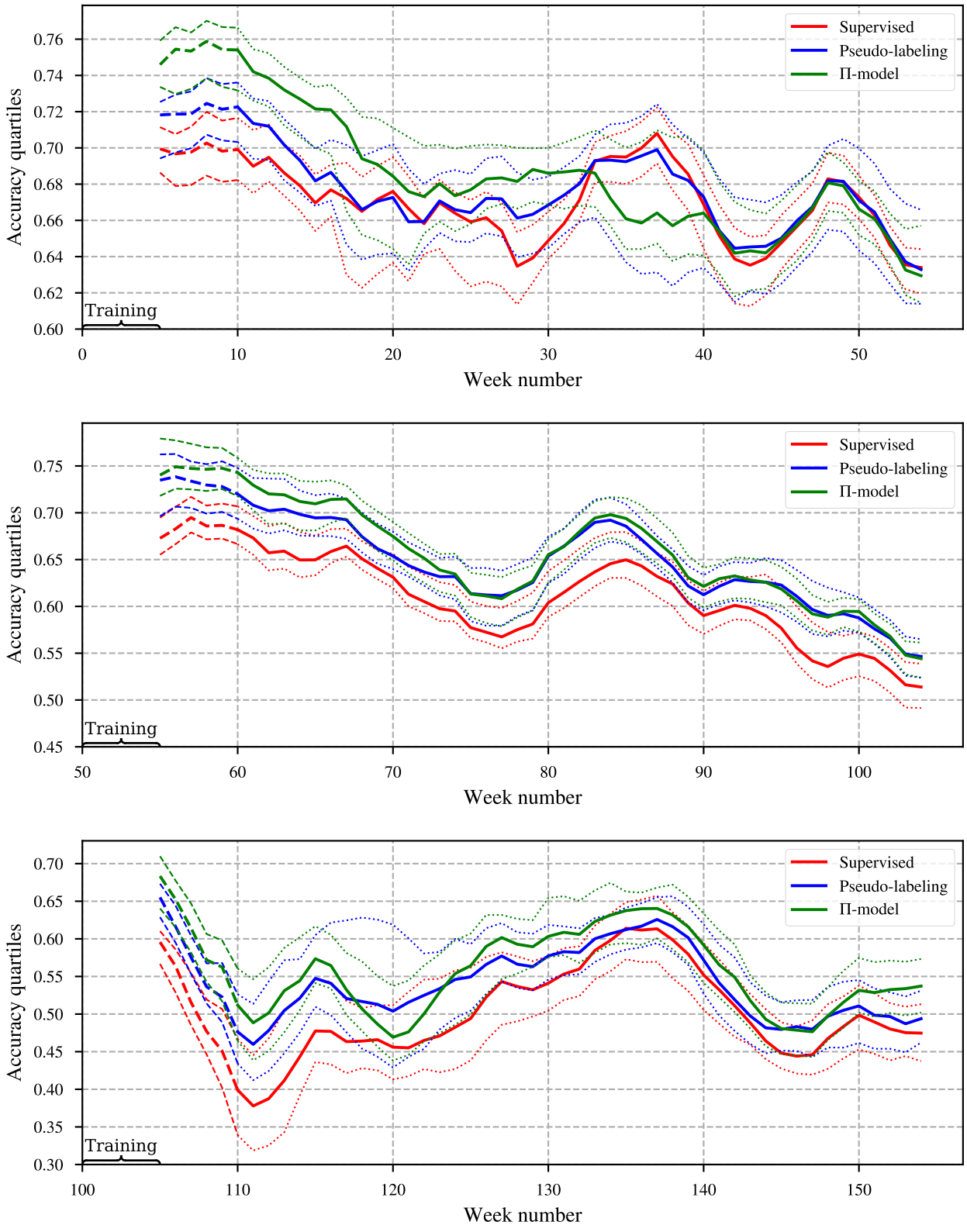


Figure 3: The progression of the classification accuracy on later weeks using Pseudo-labeling, Π-model, and fully supervised learning, trained using set with 1 % of labels. For each plot, there are three quartiles visualized; the median is drawn with a solid line, while the first and the third quartiles are dotted. The curves correspond to the moving average with the window size of five weeks. The first five dashed weeks are means of all previous weeks. The first five weeks at the beginning of each plot were used for the training.

Table 4: Multiple comparisons test of three methods for different ratios of labeled to all data, tested on the data from all of the following weeks till the end. Each cell contains a triplet of symbols representing the results of three post hoc pairwise tests. The order of the comparisons is: supervised to Pseudo-labeling, supervised to Π-model, and Pseudo-labeling to Π-model. The dash means that the difference was not statistically significant and the letters S, P, and Π mark whether supervised, Pseudo-labeling, or Π-model were significantly better than the other compared algorithm.

Ratio	Training weeks			
	1–5	51–55	101–105	151–155
0.5 %	P, –, Π	P, S, Π	P, –, Π	P, –, Π
1 %	P, –, –	S, Π, Π	P, –, Π	P, Π, Π
2 %	S, S, Π	S, S, Π	P, Π, P	S, S, –
5 %	–, S, Π	–, S, P	P, Π, P	P, S, Π
10 %	–, –, Π	P, S, Π	S, S, P	S, S, Π
25 %	P, Π, Π	S, S, Π	S, P, P	S, S, Π
50 %	–, Π, Π	–, S, Π	S, Π, P	S, S, Π
75 %	S, Π, –	S, Π, –	S, –, P	S, S, Π

References

[1] B. Abolhasanzadeh. Nonlinear dimensionality reduction for intrusion detection using auto-encoder bottleneck features. In *IKT: IEEE 7th Conference on Information and Knowledge Technology*, pages 1–5, 2015.

[2] J.M. Bonifacio, A.M. Cansian, A.C.P.L.F. De Carvalho, and E.S. Moreira. Neural networks applied in intrusion detection systems. In *IEEE International Joint Conference on Neural Networks*, pages 205–210, 1998.

[3] J. Cannady. Artificial neural networks for misuse detection. In *National Information Systems Security Conference*, pages 368–381, 1998.

[4] H. Debar, M. Becker, and D. Siboni. A neural network component for an intrusion detection system. In *IEEE Computer Society Symposium on Research in Security and Privacy*, pages 240–250, 1992.

[5] O. Depren, M. Topallar, E. Amarim, and M.K. Ciliz. An intelligent intrusion detection system (IDS) for anomaly and misuse detection in computer networks. *Expert Systems with Applications*, 29:713–722, 2005.

[6] M. Friedman. The use of ranks to avoid the assumption of normality implicit in the analysis of variance. *Journal of the American Statistical Association*, 32(200):675–701, 1937.

[7] N. Gao, L. Gao, Q. Gao, and H. Wang. An intrusion detection model based on deep belief networks. In *IEEE Second International Conference on Advanced Cloud and Big Data*, pages 247–252, 2014.

[8] I.J. Goodfellow, J. Shlens, and C. Szegedy. Explaining and harnessing adversarial examples. In *ICLR*, pages 1–11, 2015.

[9] S. Holm. A simple sequentially rejective multiple test procedure. *Scandinavian Journal of Statistics*, pages 65–70, 1979.

[10] J. Kim, J. Kim, H. Le T. Thu, and H. Kim. Long short term memory recurrent neural network classifier for intrusion detection. In *PlatCon: IEEE International Conference on Platform Technology and Service*, pages 1–5, 2016.

[11] D.P. Kingma and J. Ba. Adam: A method for stochastic optimization. Preprint arXiv:1412.6980, 2014.

[12] S. Laine and T. Aila. Temporal ensembling for semi-supervised learning. In *ICLR*, pages 1–13, 2017.

[13] D.H. Lee. Pseudo-label: The simple and efficient semi-supervised learning method for deep neural networks. In *WREPL: ICML Workshop Challenges in Representation Learning*, pages 1–6, 2013.

[14] O. Linda, T. Vollmer, and M. Manic. Neural network based intrusion detection system for critical infrastructures. In *International Joint Conference on Neural Networks*, pages 1827–1834, 2009.

[15] T.F. Lunt. IDES: An intelligent system for detecting intruders. In *Symposium on Computer Security, Threat and Countermeasures*, pages 30–45, 1990.

[16] T. Ma, F. Wang, J. Cheng, Y. Yu, and X. Chen. A hybrid spectral clustering and deep neural network ensemble algorithm for intrusion detection in sensor networks. *Sensors*, 16:article no. 1701, 2016.

[17] T. Mikolov, I. Sutskever, K. Chen, G. Corrado, and J. Dean. Distributed representations of words and phrases and their compositionality. In *NIPS*, pages 1–9, 2013.

[18] T. Miyato, S.I. Maeda, M. Koyarna, and S. Ishii. Virtual adversarial training: A regularization method for supervised and semi-supervised learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 41:1979–1993, 2019.

[19] S. Mukkamala, G. Janoski, and A. Sung. Intrusion detection using neural networks and support vector machines. In *International Joint Conference on Neural Networks*, pages 1702–1707, 2002.

[20] M. Nadeem, O. Marshall, S. Singh, X. Fang, and X. Yuan. Semi-supervised deep neural network for network intrusion detection. In *Conference on Cybersecurity Education, Research and Practice*, pages 0–11, 2016.

[21] A. Narayanan, C. Soh, L. Chen, Y. Liu, and L. Wang. Apk2vec: Semi-supervised multi-view representation learning for profiling Android applications. In *IEEE International Conference on Data Mining*, pages 357–366, 2018.

[22] A. Oliver, A. Odena, C. Raffel, E.D. Cubuk, and I.J. Goodfellow. Realistic evaluation of deep semi-supervised learning algorithms. In *NIPS*, pages 1–19, 2018.

[23] A. Rasmus, H. Valpola, M. Honkala, M. Berglund, and T. Raiko. Semi-supervised learning with ladder networks. In *NIPS*, pages 1–9, 2015.

[24] B.C. Rhodes, J.A. Mahaffey, and J.D. Cannady. Multiple self-organizing maps for intrusion detection. In *23rd National Information Systems Security Conference*, pages 16–19, 2000.

[25] J. Ryan, M.J. Lin, and R. Miikkulainen. Intrusion detection with neural networks. In *Advances in Neural Information Processing Systems 10*, pages 943–949, 1998.

[26] J. Saxe and K. Berlin. Expose: A character-level convolutional neural network with embeddings for detecting malicious URLs, file paths and registry keys. Arxiv preprint arXiv:1702.08568, 2017.

[27] C.Z.Y. Soh. *Program Analysis and Machine Learning Techniques for Mobile Security*. PhD thesis, Nanyang Technological University, Singapore, 2019.

[28] A. Tarvainen and H. Valpols. Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results. In *NIPS*, pages 1–16, 2017.

[29] M. Tavallaee, E. Bagheri, W. Lu, and A.A. Ghorbani. A detailed analysis of the KDD cup 99 data set. In *IEEE Symposium on Computational Intelligence for Security and Defense Applications*, pages 288–293, 2009.

[30] P. Torres, C. Catania, S. Garcia, and C.G Garino. An analysis of recurrent neural networks for botnet detection behavior. In *ARGENCON: IEEE biennial congress of Argentina*, pages 1–6, 2016.

[31] W. Wang, Y. Sheng, J. Wang, X. Zeng, X. Ye, Y. Huang, and M. Zhu. HAST-IDS: Learning hierarchical spatial-temporal features using deep neural networks to improve intrusion detection. *IEEE Access*, 6:1792–1806, 2017.

[32] W. Wang, M. Zhu, X. Zeng, X. Ye, and Y. Sheng. Malware traffic classification using convolutional neural network for representation learning. In *ICOIN: IEEE International Conference on Information Networking*, pages 712–717, 2017.

[33] Z. Zhang, J. Li, C.N. Manikopoulos, J. Jorgenson, and J. Ucles. HIDE: A hierarchical network intrusion detection system using statistical preprocessing and neural network classification. In *IEEE Workshop on Information Assurance and Security*, pages 85–90, 2001.