Swarm Intelligence Approaches for Parameter Setting of Deep Learning Neural Network: Case Study on Phishing Websites Classification

Grega Vrbančič
University of Maribor, Faculty of Electrical Engineering and Computer Science
SI-2000 Maribor, Slovenia
grega.vrbancic@um.si

Iztok Fister Jr.
University of Maribor, Faculty of Electrical Engineering and Computer Science
SI-2000 Maribor, Slovenia
iztok.fister1@um.si

Vili Podgorelec
University of Maribor, Faculty of Electrical Engineering and Computer Science
SI-2000 Maribor, Slovenia
vili.podgorelec@um.si

ABSTRACT

In last decades, the web and online services have revolutionized the modern world. However, by increasing our dependence on online services, as a result, online security threats are also increasing rapidly. One of the most common online security threats is a so-called Phishing attack, the purpose of which is to mimic a legitimate website such as online banking, e-commerce or social networking website in order to obtain sensitive data such as usernames, passwords, financial and health-related information from potential victims. The problem of detecting phishing websites has been addressed many times using various methodologies from conventional classifiers to more complex hybrid methods. Recent advancements in deep learning approaches suggested that the classification of phishing websites using deep learning neural networks should outperform the traditional machine learning algorithms. However, the results of utilizing deep neural networks heavily depend on the setting of different learning parameters. In this paper, we propose a swarm intelligence based approach to parameter setting of deep learning neural network. By applying the proposed approach to the classification of phishing websites, we were able to improve their detection when compared to existing algorithms.

CCS CONCEPTS

• **Computing methodologies → Search methodologies**; **Neural networks**; • **Security and privacy** → *Human and societal aspects of security and privacy*;

KEYWORDS

Machine learning, neural networks, optimization, phishing, website classification

ACM Reference Format:
Grega Vrbančič, Iztok Fister Jr., and Vili Podgorelec. 2018. Swarm Intelligence Approaches for Parameter Setting of Deep Learning Neural Network: Case Study on Phishing Websites Classification. In *WIMS '18: 8th International Conference on Web Intelligence, Mining and Semantics, June 25–27, 2018, Novi Sad, Serbia.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3227609.3227655

1 INTRODUCTION

In recent years, we are witnessing an enormous increase of online services such as online banking, entertainment, shopping, education and social networking. Numerous people use such internet amenities because of the ease of their use and because it saves them time and money.

On the other side, with such rapid growth of online services, we are also witnessing a rapid increase of online security threats such as Pharming, Spoofing and Phishing. The latter is according to the Internet Crime Report from 2016 [8], one of the most popular online crime types which affected around 20 thousand people just last year, resulting in around 31 million dollars of financial loss. Statistics from the Anti-Phishing Working group published in June 2017 [2] are showing a 10% increase of unique phishing attacks worldwide from attacks identified in 2015. Most of the attacks are targeted at primary industry sectors, with financial institutions, e-commerce, social networking and money transfer companies being targeted in over three-quarters of all phishing attacks.

Phishing attack is a type of extensive fraud that happens when a malicious website acts, looks and feels almost identical to the legitimate one, keeping in mind that the end goal is to obtain victim's sensitive data. Phishing attacks impact many actors, from individuals to the corporate and government agencies whose brands are deceptively used [15]. The victims of phishing attacks often find their personal or financial information, such as their credit card numbers, health or insurance information, email, addresses, login credentials and other sensitive data, stolen. Once that kind of information is stolen, it can be used to create fake accounts in the victim's name which can have a severe impact on their credit ratings or prevent the victim from accessing their own accounts, leading to a lack of financial credibility [13].

The aforementioned consequences of phishing attacks clearly demonstrate that there is a need for a robust anti-phishing solution for this continuously evolving internet threat. The literature suggests several conventional techniques for detecting phishing websites. However, the decision regarding phishing websites using those techniques was predicted imprecisely [3, 23], which led to the classification of legitimate websites as phishing. In general, two most popular approaches are used to detect the phishing websites:

blacklist and whitelist based approaches and intelligent heuristic-based approaches [4, 23]. The downside of the first approach is that the blacklist usually cannot cover all phishing websites since a newly created phishing website takes a considerable amount of time before it is added to the list. In contrast to blacklist based approach, the use of intelligent heuristic-based approaches can recognize freshly created phishing websites in real time [19].

In last few years, with recent advancements in the field of machine learning, some intelligent phishing detection methods have been suggested in order to effectively predict phishing websites. In [22] authors are tackling the problem with the self-structuring neural network, in [29] a complex hybrid model was proposed, combined from various different classification methods such as Naive Bayes, Bayes Net, Random Forest, Decision Tree, Support Vector Machines (SVM), etc. In addition, different kinds of neural networks (NNs) have been employed in the detection of phishing websites. For example, in [27] authors have used a convolutional neural network (CNN) as a part of their approach to automating feature extraction and in [30, 31] the authors also presented the different approaches for tackling the problem of detecting phishing websites with the use of NNs.

While comparing the results of different classification methods used, we can see that in most cases NNs fall behind the conventional classification methods such as Random Forest, Random Tree, J48, etc. which is not surprising at all. The biggest drawback in the usage of NNs is definitely finding the right learning parameters and topology configuration, to get the best results out of them. The biggest challenge in finding the right parameters settings for the NNs is that there is no general rule or recipe to follow, which would guarantee you a good outcome. It more or less depends on our previous experience and trying out different configurations.

In our work we propose a novel method, titled as TDLBA or TDLHBA (Tuning Deep Learning using Bat/Hybrid Bat Algorithm) which combines swarm intelligence approaches for parameter settings of deep learning neural network. The main goal of our research is to study whether a NN with parameters set by utilizing our proposed method will give us better classification accuracy than NN with recommended, sane settings. The main advantage of the proposed method is the very straightforward usage with various feed-forward NN topologies and different datasets, without need to manually search for right learning parameters.

The remaining of the paper is organized as follows. Section 2 briefly describes methods we used. In Section 3 we present the proposed TDLBA/TDLBHA method, whose results on predicting phishing websites are presented in Section 4. Our conclusions and final thoughts are gathered in Section 5.

2 METHODS

2.1 Swarm Intelligence

In order to tackle the optimization problems that were arising in almost every domain of human endeavor, scientists were looking back to the nature to get inspiration for the design of complex algorithms. Such algorithms mostly mimic the biological features of some fascinating animal species, as for example ants, bees, bats or fireflies. Roughly speaking, these species are capable of decentralized decision making, coordinated movement as well as collective behavior. Algorithms that are based on these features belong to the family of swarm intelligence algorithms. In the past years, many surveys have shown how promising these algorithms are for solving problems in various domains. Due to the popularity of this research area, many swarm intelligence algorithms were proposed. However, our study is based on bat algorithm (BA) [34] and hybrid bat algorithm (HBA) [10]. Both algorithms are characterized in the next subsection.

2.1.1 Bat algorithm. Bat algorithm is the member of swarm intelligence family that is inspired by the behavior of micro-bats. BA was developed by Xin-She Yang in 2010 [34]. Interestingly, some origins of BA can also be found in particle swarm optimization algorithm and simulated annealing heuristics. Pseudocode of basic BA is presented in Alg. 1, where parameter D denotes dimension of the problem, Np is population size, MAX_FE is number of function evaluations, A_i is loudness and r_i is pulse rate.

Algorithm 1 Original Bat algorithm

```
Input: Bat population $\mathbf{x_i} = (x_{i1}, \dots, x_{iD})^T$ for $i = 1 \dots Np$, $MAX\_FE$.
Output: The best solution $\mathbf{x}_{best}$ and its corresponding value $f_{min} = \min(f(\mathbf{x}))$.
1:  init_bat();
2:  $eval$ = evaluate_the_new_population();
3:  $f_{min}$ = find_the_best_solution($\mathbf{x}_{best}$);
4:  while termination_condition_not_meet do
5:      for $i = 1$ to $Np$ do
6:          y = generate_new_solution($\mathbf{x}_i$);
7:          if rand(0, 1) > $r_i$ then
8:              y = improve_the_best_solution($\mathbf{x}_{best}$)
9:          end if
10:         $f_{new}$ = evaluate_the_new_solution(y);
11:         $eval = eval + 1$;
12:         if $f_{new} \le f_i$ and N(0, 1) < $A_i$ then
13:             $\mathbf{x}_i = \mathbf{y}$; $f_i = f_{new}$;
14:         end if
15:         $f_{min}$=find_the_best_solution($\mathbf{x}_{best}$);
16:     end for
17: end while
```

Main components of BA are the following:

- initialization: the initial population is being generated as well as evaluated,
- generation of the new solution: virtual bats are moved within the search space according to the physical rules of echolocation,
- local search step: the best solution is improved using random walk direct exploitation heuristic,
- evaluation of new solution: evaluating the new solution,
- conditional saving of best solution: the new best solution is saved under some probability A_i,
- finding the best solution: finding the current best solution.

Interested readers are invited to check a more detailed description of BA in paper [34].

2.1.2 Hybrid bat algorithm. Hybrid bat algorithm or simply HBA [10] is one of the first hybrid variants of BA. HBA is hybridized with the mutation strategies of differential evolution. In other words, random walk step from the original bat algorithm is eliminated and changed by the mutation strategy. A short pseudocode is presented in Algorithm 2. HBA has achieved very promising performance when solving small-scale global optimization problems. For that reason, HBA is also evaluated on the problem of tuning neural network parameters that belong to small-scale optimization problems.

Algorithm 2 HBA

```
1: if $rand(0,1) > r_i$ then
2:     $r_{i=1...3} = \lfloor \text{rand}(0,1) * Np + 1 \rfloor$;
3:     $n = \text{rand}(1, D)$;
4:     for $i = 1$ to $D$ do
5:         if $((\text{rand}(0,1) < CR) || (n == D))$ then
6:             $\mathbf{y}_n = \mathbf{x}_{r1,n} + F * (\mathbf{x}_{r2,n} - \mathbf{x}_{r3,n})$;
7:             $n = (n+1)\%(D+1)$;
8:         end if
9:     end for
10: end if
```

2.2 Deep learning

A standard NN as we know for decades, consists of many simple connected processors known as neurons, each producing a sequence of real-valued activations. In the most widely used type of NN, neurons are stacked together in form of layers. The first layer, known as an input layer, consists of neurons which get activated through sensors perceiving the environment. The outputs of the previous layer become the weighted input to the next layer, with no interconnections of neurons within each layer. Learning such NN is about finding the right weights that make the NN exhibit desired behaviour [11, 28].

Depending on problems, the process of training a NN may take long causal chains of computational stages, where each stage transforms (mostly in a non-linear way) the aggregate activation of NN. Deep learning is about accurately assigning credit across many such stages. Since 1980 back-propagation played an important role as an efficient gradient descent algorithm. It trains the NNs with a teacher-based supervised learning approach [18]. Many of deep learning applications use feed-forward NN topologies (Fig. 1), which learn fixed-size input to a fixed size output (e.g. probability for each of several categories). Going from one layer to the next, a set of weighted sum is computed from the inputs of the previous layer and passed through a non-linear function. Currently most popular non linear function is the rectified linear unit (ReLU), which is the half-wave rectifier $f(z) = max(z, 0)$. Opposed to other previously most commonly used smoother non-linearities, such as $tanh(z)$ or $1/(1 + exp(-z))$, ReLU typically learns much faster, especially in networks with many hidden layers, allowing training of deep supervised network without unsupervised pre-training [12, 16].

Beside feed-forward NN the most widely used type of NN topology is a recurrent neural network (RNN), which contains feedback connections from neurons in the subsequent layers to neutrons in the preceding layers. This implied that the output of such NN not only depends on the external inputs but also on the state of the network in the previous training iteration [22]. RNNs are very powerful dynamic systems, mostly used for tasks which involve sequential inputs such as speech and language, but training them has proved to be problematic [16].

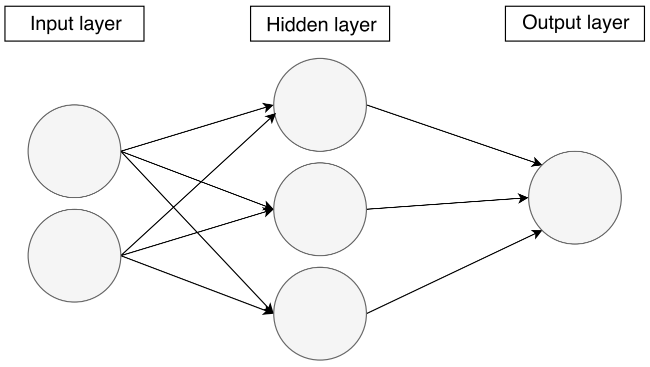

Figure 1: Simple feed-forward NN topology with one hidden layer

3 PROPOSED METHOD

Our proposed method for parameter setting of deep learning NN is presented in Alg. 3. The method is based on the modified bat and hybrid bat algorithm variants named TLDBA and TDLHBA. The task of optimizers TLDBA/TLDHBA is to find optimal parameter settings for the NN. In our case, we took the following parameters in account: the number of epochs, batch size, learning rate and the number of neurons in the first hidden layer (the number of neurons in the second hidden layer is fixed).

As we can see from Alg. 3, most of the steps are actually similar to steps presented in Alg. 1. However, the most important components that differ from Alg. 1 are the following:

- representation of individuals,
- design of fitness function, and
- design of the NN.

All three components are described in detail in the next subsections.

Algorithm 3 Proposed method

```
Output: The best model with parameters set based on best solution
1: BA.init();
2: while termination_condition_not_meet do
3:     solution = BA.get_best_solution();
4:     epoch = map_epoch(solution[0]);
5:     batch = map_batch(solution[1]);
6:     learning_rate = map_learning_rate(solution[2]);
7:     num_neurons = map_num_neurons(solution[3]);
8:     fitness = train_and_eval(epoch, batch,
       learning_rate, num_neurons);
9:     BA.generate_new_solution(fitness);
10: end while
11: best = create_model(BA.get_best_solution());
```

3.1 Representation of individuals

Individuals in TDLBA and TDLHBA are presented as real-valued vectors:

$$\mathbf{x}_i^{(t)} = (x_{i,0}^{(t)}, \ldots, x_{i,n}^{(t)}), \quad \text{for } i = 0, \ldots, Np-1, \tag{1}$$

where each element of the solution is in the interval $x_{i,1}^{(t)} \in [0,1]$.

Real values are then mapped according to equations 2, 3 and 4 where y_1 stands for number of epochs, y_2 for batch size and y_4 for learning rate used to train NN. The value y_3 is representing the number of neurons in first hidden layer of our NN topology.

$$y_1 = \lfloor x[i] * 100 + 100 \rfloor; y_1 \in [100, 200] \tag{2}$$

$$y_2, y_3 = \left\lfloor \frac{x[i] * 100}{2} + 10 \right\rfloor; y_2, y_3 \in [10, 60] \tag{3}$$

$$y_4 = \frac{x[i]}{10} \in [0, 0.1] \tag{4}$$

3.2 Fitness function

We defined fitness function using the accuracy of classification calculated on the validation set. Formally we can express fitness function as presented in equation (5), where *acc* stands for previously mentioned accuracy. Because BA and HBA are basically designed to search for the global minimum, we are converting the problem of searching maximal accuracy to the problem of searching for the minimum with the subtraction of the accuracy from value 1.

$$f(acc) = 1 - acc \tag{5}$$

3.3 Neural network

For topology of NN, we propose a feed-forward NN with at least one fully-connected hidden layer. The width of the latter would be a y_3 value from equation (3) in the above subsection. For binary classification problems, we propose using non-linear activation function ReLU on hidden layers and sigmoid function on output layer. Instead of conventional stochastic gradient descent optimizer we propose ADAM [14] optimizer, which is designed to combine the advantages of two recently popular methods: AdaGrad [7] and RMSProp [32].

4 EXPERIMENTS AND RESULTS

4.1 The Phishing Websites dataset

To conduct the experiments of predicting phishing websites with the proposed approach, the Phishing Websites Data Set [20] from the UCI Machine Learning repository [17] was used, prepared by Mohammad et al. [21]. The basic information of this dataset is presented in Table 1.

There are different approaches to tackle the problem of identification of phishing websites reported in literature [3]. Typically, the two most technical methods in fighting phishing attacks are the blacklist and the heuristic-based [1, 9]. In the blacklist method, the requested URL is compared with a predefined phishing URLs. The downside of this method is that it typically doesn't deal with all phishing websites since a newly launched fake website takes a substantial amount of time before being added to the list. In contrast to the blacklist approach, the heuristic-based approach can recognize newly created fake websites in real-time [19].

Parameter	Value
Number of features	31
Number of instances	11,055
Number of classes	2 classes
Distribution of classes	3,793 phishing websites
	7,262 legitimate websites

Table 1: The basic information about the Phishing Websites dataset

The heuristic-based approach has been also applied in the case of preparing the Phishing Websites dataset [21]. The authors defined an approach for extracting features from the web page itself rather than user experience. First, the authors examined whether a page contains any text fields since a phishing web page requires users to input credentials through those fields [5]. If a page has at least one text input then they proceeded to extract the other features. After extracting the features, the authors collected a number of URLs from the PhishTank data archives [25], which are free community sites for sharing phishing data, using their tool. Based on the analysis of each extracted feature's frequency within the collected addresses, the importance of a feature was reflected, and finally, the heuristic rules for determining whether a specific website is a phishing one or not regarding a specific feature were devised.

There are many features that can possibly distinguish phishing websites from other (legitimate) types of websites in the research literature of phishing. In this manner, the constructed features belong to one of the following groups: Address Bar based features, Abnormal based features, HTML and JavaScript-based features, and Domain-based features. Altogether 30 features were constructed. Some of the most important features are presented in Table 2.

The constructed features, shown in Table 2 can take either a binary or ternary values, where binary features hold either "phishy" or "legitimate" status because the existence or lack of the feature within the website determines the value assigned to it. For ternary value features, which can also hold the value "suspicious", the existence of the feature in a specific ratio determines the value assigned for that feature.

All the features' values have been determined in accordance with a set of heuristic rules, devised to best reflect the presence of potentially phishy websites [3, 21]. For example, let us examine the feature "Sub-domain and multi sub-domain". A technique used by phishers to scam users is by adding a sub-domain to the URL so users may believe they are dealing with an authentic website. In this manner, the following heuristic rule can be devised:

$$IF \begin{cases} dots\ in\ domain\ part < 3 \Longrightarrow legitimate \\ dots\ in\ domain\ part = 3 \Longrightarrow suspicious \\ else \Longrightarrow phishy \end{cases}$$

Group	Features
Address bar based features	Using the IP address Long URL to hide the suspicious part URL having @ symbol Adding prefix or suffix to domain Sub-domain and multi sub-domain Fake HTTPS and SSL ...
Abnormal based features	Request URL URL of anchor Server form handler Abnormal URL ...
HTML and JavaScript based features	Redirect page Hide the link using `onMouseOver` Disabling right click Using pop-up window ...
Domain based features	Age of domain DNS record Web traffic ...

Table 2: Some of the most important features, used to detect the phishing websites

Similar heuristic rules have been determined for all other features as well. For more information please see [3, 21].

4.2 Experimental settings

4.2.1 BA and HBA parameters. For initialization parameters of BA and HBA algorithms, we used values presented in Table 3.

Initialized with given parameters, BA and HBA are used for searching for optimal values for the number of epochs, batch size, learning rate and the number of neurons in the first hidden layer of NN. In our experiment, we used a 20% validation split for the purpose of calculating accuracy needed for evaluation of fitness function presented in Section 3.3.

4.2.2 Neural network. For the purpose of the experiment, we defined feed-forward NN with two fully connected hidden layers and a fully-connected output layer with a single neuron. The width (number of neurons) of the first hidden layer is set dynamically, based on the parameters passed in the initialization phase. The number of neurons in the second hidden layer is set to 30, the same as in the input layer which is matching the number of features in the dataset.

Both hidden layers are using a non-linear activation function ReLU, while the output layer is using sigmoid. As proposed in Section 3.3, we used ADAM as the optimizer function.

Parameter	BA	HBA
Dimension of the problem	4	4
Population size	40	40
Number of generations	100	100
Loudness	0.5	0.5
Pulse rate	0.5	0.5
Min. frequency	0.0	0.0
Max. frequency	2.0	2.0
Lower bound	0.0	0.0
Upper bound	1.0	1.0
F (Scaling factor)		0.5
CR (Crossover probability)		0.5

Table 3: Used parameter values for BA and HBA algorithms

The number of neurons in the first hidden layer of NN for the experiment for handpicked NN was set to 30.

4.2.3 Learning parameters. Based on [14] and our previous experience in machine learning, we handpicked as optimal as possible learning parameters. For batch size we choose 32, for learning rate 10^{-3} and for the number of epochs 150.

Performance of NN with these parameters should give us a good starting point in comparison of performance between conventional classification methods and our proposed method.

4.3 Results

For the proposed deep learning approach the experiments were implemented with Python programming language using the following libraries: Keras [6], NumPy [24], and scikit-learn [26]. For all of the existing classification algorithms, the experiments were performed using the standard Weka library [33]. All classification models were used with their default settings.

To objectively evaluate the performance of the proposed classification approach, and to compare it with the existing classification algorithms, we followed an established methodology. The dataset was divided into train and test sets in a ratio of 90:10 using tenfold cross-validation procedure. In this manner, 9 out of 10 folds were used for training and the last fold for testing purposes. In the case of swarm intelligence approaches for parameter settings of deep learning neural network, which are iterative in nature, the 80% of the train set was used in the iterative process of optimizing parameters, while the remaining 20% of the train set was used as a validation set for calculating the fitness and thus directing the evolution. Folds were made by Weka with its stratification method, which splits set in such way that the distribution of classes remains the same across all of the folds.

The presented results are minimum, maximum and average as well as median accuracies of the ten folds achieved by a specific method on a test set. The standard deviations for each specific classification algorithm over the ten folds are also reported.

4.4 Neural Network performance

In this experiment, we assessed the performance of our NN with handpicked training parameters described in sub-sections 4.2.2. and 4.2.3 in comparison to Logistic regression (LR), Naive Bayes (NB), J48, SimpleCart and Random Tree (RT) classifiers. The experiment results are presented in a form of box plots (as known as box and whisker diagram) in Fig. 2. The statistics are showing that our handpicked NN performs slightly worse, with a minimum accuracy of 94.4%, maximum accuracy of 96.9% and median of 95.7% than the best performing classifier RT with a minimum accuracy of 95.5%, maximum accuracy of 97.1% and the median accuracy of 96.1%. Also, in comparison with SimpleCart and Random Tree classifier, our handpicked NN has a higher standard deviation from mean accuracy value. As we can see, the LR and NB classifiers fall behind the others noticeably in every performance aspect.

We expect that the mentioned drawbacks in the performance of NN can be addressed with the usage of our proposed TDLBA/TDLHBA parameter setting method.

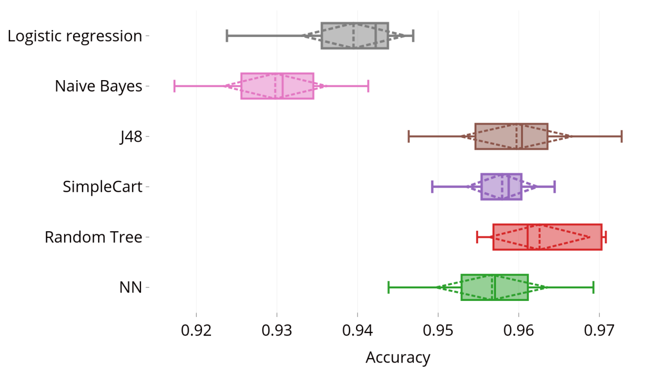

Figure 2: Comparison of accuracy between our handpicked NN and other conventional classifiers using 10-fold cross validation

4.5 Improved Neural Network performance

Utilizing our proposed method we conducted another experiment in which we used the obtained optimal parameter solutions presented in Table 4 to train our NN.

Parameter	NN	TDLBA	TDLHBA
Number of epoch	150	130	100
Batch size	32	52	10
Learning rate	10^{-3}	0.0043526	0.0017470
Number of neurons	30	35	40

Table 4: Optimal parameter solutions using TDLBA and TDLHBA

In Fig. 3, the performance results of all classification algorithms are presented, with exception of NB and LR, which performed significantly worse than all others. Observing mean accuracy as well as median accuracy presented in Fig. 3, both TDLBA and TDLHBA did outperform not just our handpicked NN, but also other tested classifiers. With the average accuracy of 96.5%, TDLBA gave us slight performance increase in comparison with previously best-performing classifier RT. Also, looking at the distribution of each fold's accuracy for TDLBA/TDLHBA, we can see a noticeable improvement in the much smaller range between the minimum and maximum accuracy than any other tested classifier.

Taking a closer look at the comparison of results for handpicked NN and TDLBA/TDLHBA in Fig. 4, we can see a major improvement in all performance accuracy aspects in favor of our two proposed optimization methods. With the utilization of the proposed methods, we did not only address all of the performance drawbacks mentioned in the previous section but also improved the general accuracy of the model.

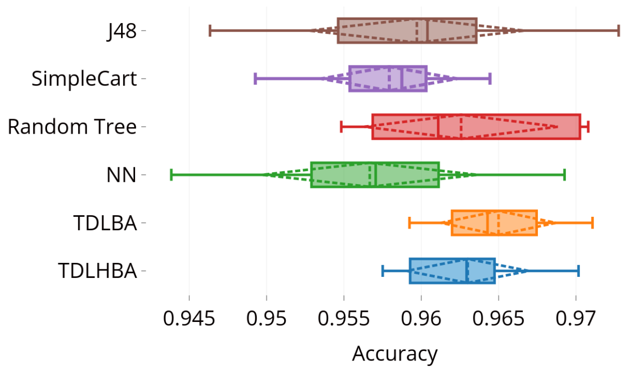

Figure 3: Comparison of accuracy between all classifiers using 10-fold cross validation

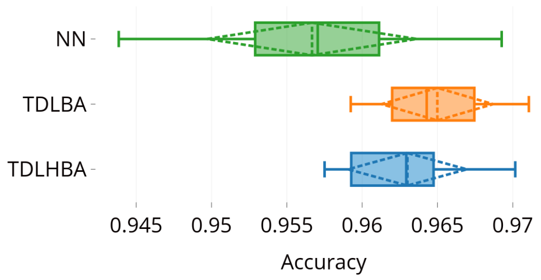

Figure 4: Comparison of accuracy between all NNs using 10-fold cross validation

4.6 Statistical comparison of methods

To evaluate the statistical significance of the obtained results, we first applied the Friedman test by calculating the Friedman asymptotic significance for various classification algorithms. The test confirmed (p<0.001) that there are statistically significant differences between all used classifiers, as was expected already from observing Fig. 3.

The post-hoc Wilcoxon signed ranks test with Holm-Bonferroni correction revealed that both Naive Bayes (p=0.005) and Logistic Regression (p=0.005) classifiers were significantly outperformed by all other algorithms. On the other hand, NN (with handpicked parameters) was not significantly different from J48 (p=0.285), SimpleCart (p=0.575) and RT (p=0.169), but performed slightly worse than them, being better in 4 and worse in 6 out of 10 folds.

When considering the two proposed optimization methods, they both significantly outperformed the NN with hand-picked parameters: TDLBA (p=0.008) and TDLHBA (p= 0.012), being better in 9 out of 10 folds. Their mutual comparison revealed that the difference is not statistically significant; however, the TDLBA performed better than TDLHBA in 7 out of 10 folds.

The comparison of TDLHBA with J48 (p=0.333), SimpleCart (p=0.074) and RT (p=0.799) showed no significant difference between them. On the other hand, the TDLBA turned out to significantly outperform SimpleCart (p=0.013), being better in 8 out of 10 folds. Finally, the difference between TDLBA and J48 (p=0.093) and RT (p=0.333) turned out not to be statistically significant. However, TDLBA outperformed J48 in 6 out of 10 folds, and RT in 7 out of 10 folds.

5 CONCLUSIONS

In this paper, we presented a novel method utilizing a swarm intelligence based approach to set parameters of deep learning neural network. The results, obtained from the conducted experiments, have proven to be very promising, giving us noticeable accuracy performance improvements in the classification of phishing websites in comparison with conventional classification methods, as well as with the manually tuned NN. In general, the proposed TDLBA method showed the best results in classifying phishing websites, which was also statistically confirmed.

In the future, based on these encouraging results, we would like to expand our work with the use of different swarm intelligence algorithms (e.g. Firefly algorithm, Cuckoo search algorithm), to search for optimal solution of larger number of training parameters and to self-construct different NN topologies with various depth, width and types of layers. We would also like to do a more elaborated performance comparison of our proposed method, using various different datasets in comparison with other state of the art approaches.

ACKNOWLEDGMENTS

The authors acknowledge the financial support from the Slovenian Research Agency (Research Core Funding No. P2-0057).

REFERENCES

[1] Greg Aaron and Ronnie Manning. [n. d.]. APWG Phishing Reports. APWG. 2014.
[2] Greg Aaron and Rod Rasmussen. 2017. Global Phishing Survey 2015-2016 Global Phishing Survey: Trends and Domain Name Use in 2016 An APWG Industry Advisory Global Phishing Survey 2016: Trends and Domain Name Use. *Global Phishing Survey* (2017). https://docs.apwg.org/reports/APWG{_}Global{_}Phishing{_}Report{_}2015-2016.pdf
[3] Neda Abdelhamid, Aladdin Ayesh, and Fadi Thabtah. 2014. Phishing detection based Associative Classification data mining. , 5948–5959 pages. https://doi.org/10.1016/j.eswa.2014.03.019
[4] Waleed Ali. 2017. Phishing Website Detection based on Supervised Machine Learning with Wrapper Features Selection. *IJACSA) International Journal of Advanced Computer Science and Applications* 8, 9 (2017), 72–78. www.ijacsa.thesai.org
[5] Ram B Basnet, Andrew H Sung, and Quingzhong Liu. 2011. Rule-based phishing attack detection. In *International Conference on Security and Management (SAM 2011), Las Vegas, NV.*
[6] François Chollet et al. 2015. Keras. https://keras.io.
[7] John Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive Subgradient Methods for Online Learning and Stochastic Optimization. *Journal of Machine Learning Research* 12 (2011), 2121–2159. https://doi.org/10.1109/CDC.2012.6426698 arXiv:arXiv:1103.4296v1
[8] Federal Bureau of Investigation of USA - Internet Crime Complaint Center. 2016. *Internet Crime Report.* Technical Report. Federal Bureau of Investigation of USA. http://www.geoinform.ru/?an=gng2009-all-en
[9] Ian Fette, Norman Sadeh, and Anthony Tomasic. 2007. Learning to detect phishing emails. In *Proceedings of the 16th international conference on World Wide Web.* ACM, 649–656.
[10] Iztok Fister, Dusan Fister, and Xin She Yang. 2013. A hybrid bat algorithm. *Elektrotehniški Vestnik/Electrotechnical Review* 80, 1-2 (2013), 1–7. arXiv:1303.6310

[11] Iztok Fister, Ponnuthurai Nagaratnam Suganthan, Iztok Fister, Salahuddin M Kamal, Fahad M Al-Marzouki, Matjaž Perc, and Damjan Strnad. 2015. Artificial neural network regression as a local search heuristic for ensemble strategies in differential evolution. *Nonlinear Dynamics* 84, 2 (2015), 895–914. https://doi.org/10.1007/s11071-015-2537-8

[12] Xavier Glorot, Antoine Bordes, and Yoshua Bengio. 2011. Deep sparse rectifier neural networks. *AISTATS '11: Proceedings of the 14th International Conference on Artificial Intelligence and Statistics* 15 (2011), 315–323. https://doi.org/10.1.1.208.6449 arXiv:1502.03167

[13] R. Gowtham and Ilango Krishnamurthi. 2014. A comprehensive and efficacious architecture for detecting phishing webpages. *Computers and Security* 40 (feb 2014), 23–37. https://doi.org/10.1016/j.cose.2013.10.004

[14] Diederik P Kingma and Jimmy Lei Ba. 2015. Adam: A Method for Stochastic Optimization. *ICLR* (2015). https://doi.org/10.1145/1830483.1830503 arXiv:arXiv:1412.6980v9

[15] David Lacey, Paul Salmon, and Patrick Glancy. 2015. Taking the Bait: A Systems Analysis of Phishing Attacks. *Procedia Manufacturing* 3 (2015), 1109–1116. https://doi.org/10.1016/j.promfg.2015.07.185

[16] Yann Lecun, Yoshua Bengio, and Geoffrey Hinton. 2015. Deep learning. , 436–444 pages. https://doi.org/10.1038/nature14539 arXiv:arXiv:1312.6184v5

[17] M. Lichman. 2013. UCI Machine Learning Repository. Available at http://archive.ics.uci.edu/ml, Accessed: 2018-01-15.

[18] Weibo Liu, Zidong Wang, Xiaohui Liu, Nianyin Zeng, Yurong Liu, and Fuad E Alsaadi. 2017. A survey of deep neural network architectures and their applications. *Neurocomputing* 234 (2017), 11–26. https://doi.org/10.1016/j.neucom.2016.12.038

[19] Daisuke Miyamoto, Hiroaki Hazeyama, and Youki Kadobayashi. 2008. An evaluation of machine learning-based methods for detection of phishing sites. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, Vol. 5506 LNCS. Springer, Berlin, Heidelberg, 539–546. https://doi.org/10.1007/978-3-642-02490-0_66

[20] Rami M Mohammad, Fadi Thabtah, and Lee McCluskey. [n. d.]. Phishing Websites at UCI Machine Learning Repository. Available at http://archive.ics.uci.edu/ml/datasets/Phishing+Websites, Accessed: 2018-01-17.

[21] Rami M Mohammad, Fadi Thabtah, and Lee McCluskey. 2012. An assessment of features related to phishing websites using an automated technique. In *Internet Technology And Secured Transactions, 2012 International Conference for.* IEEE, 492–497.

[22] Rami M Mohammad, Fadi Thabtah, and Lee McCluskey. 2014. Predicting phishing websites based on self-structuring neural network. *Neural Computing and Applications* 25, 2 (2014), 443–458. https://doi.org/10.1007/s00521-013-1490-z

[23] Rami M. Mohammad, Fadi Thabtah, and Lee McCluskey. 2015. Tutorial and critical analysis of phishing websites methods. , 24 pages. https://doi.org/10.1016/j.cosrev.2015.04.001

[24] Travis E Oliphant. 2006. *A guide to NumPy*. Vol. 1. Trelgol Publishing USA.

[25] OpenDNS. [n. d.]. PhishTank data archives. Available at https://www.phishtank.com/, Accessed: 2018-01-17.

[26] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research* 12 (2011), 2825–2830.

[27] Joshua Saxe and Konstantin Berlin. 2017. eXpose: A Character-Level Convolutional Neural Network with Embeddings For Detecting Malicious URLs, File Paths and Registry Keys. *CoRR* abs/1702.08568 (2017). arXiv:1702.08568 http://arxiv.org/abs/1702.08568

[28] Jürgen Schmidhuber. 2015. Deep Learning in neural networks: An overview. *Neural Networks* 61 (2015), 85–117. https://doi.org/10.1016/j.neunet.2014.09.003 arXiv:1404.7828

[29] M. Amaad Ul Haq Tahir, Sohail Asghar, Ayesha Zafar, and Saira Gillani. 2016. A Hybrid Model to Detect Phishing-Sites Using Supervised Learning Algorithms. In *2016 International Conference on Computational Science and Computational Intelligence (CSCI)*. IEEE, 1126–1133. https://doi.org/10.1109/CSCI.2016.0214

[30] Abdeljaber Fadi Thabtah, T.L. McCluskey, and Rami M Mohammad. 2013. Predicting Phishing Websites using Neural Network trained with Back-Propagation. *Proceedings of the 2013 World Congress in Computer Science, Computer Engineering, and Applied Computing. WORLDCOMP 2013* . January (2013), 682–686.

[31] F. Thabtah, R. M. Mohammad, and L. McCluskey. 2016. A dynamic self-structuring neural network model to combat phishing. In *2016 International Joint Conference on Neural Networks (IJCNN)*. 4221–4226. https://doi.org/10.1109/IJCNN.2016.7727750

[32] Tijmen Tieleman and Geoffrey Hinton. 2012. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural networks for machine learning* 4, 2 (2012), 26–31.

[33] Ian H Witten, Eibe Frank, Mark A Hall, and Christopher J Pal. 2016. *Data Mining: Practical machine learning tools and techniques*. Morgan Kaufmann.

[34] Xin She Yang. 2010. A new metaheuristic Bat-inspired Algorithm. *Studies in Computational Intelligence* 284 (2010), 65–74. https://doi.org/10.1007/978-3-642-12538-6_6 arXiv:1004.4170