International Journal on Artificial Intelligence Tools

Parameter Setting for Deep Neural Networks using Swarm Intelligence on Phishing Websites Classification

Grega Vrbančič, Iztok Fister Jr. and Vili Podgorelec
Faculty of Electrical Engineering and Computer Science, University of Maribor
Koroška cesta 46, Maribor, SI-2000, Slovenia
grega.vrbancic@um.si
iztok.fister1@um.si
vili.podgorelec@um.si

Received 23 November 2018
Revised 3 May 2019
Accepted 24 July 2019

Over the past years, the application of deep neural networks in a wide range of areas is noticeably increasing. While many state-of-the-art deep neural networks are providing the performance comparable or in some cases even superior to humans, major challenges such as parameter settings for learning deep neural networks and construction of deep learning architectures still exist. The implications of those challenges have a significant impact on how a deep neural network is going to perform on a specific task. With the proposed method, presented in this paper, we are addressing the problem of parameter setting for a deep neural network utilizing swarm intelligence algorithms. In our experiments, we applied the proposed method variants to the classification task for distinguishing between phishing and legitimate websites. The performance of the proposed method is evaluated and compared against four different phishing datasets, two of which we prepared on our own. The results, obtained from the conducted empirical experiments, have proven the proposed approach to be very promising. By utilizing the proposed swarm intelligence based methods, we were able to statistically significantly improve the predictive performance when compared to the manually tuned deep neural network. In general, the improvement of classification accuracy ranges from 2.5% to 3.8%, while the improvement of F1-score reached even 24% on one of the datasets.

Keywords: Machine Learning, Neural Networks, Optimization, Swarm Intelligence, Phishing, Website Classification

Citation detail: G. Vrbančič, I. Jr. Fister, & V. Podgorelec, Parameter Setting for Deep Neural Networks Using Swarm Intelligence on Phishing Websites Classification, International Journal on Artificial Intelligence Tools, Vol. 28 No. 6, pp. 1 - 28, 2019. http://dx.doi.org/10.1142/S021821301960008X

Preprint: Preprint of an article published in International Journal on Artificial Intelligence Tools, Vol. 28, No. 06, pp. 1-28, http://dx.doi.org/10.1142/S021821301960008X © World Scientific Publishing Company https://www.worldscientific.com/worldscinet/ijait

1. Introduction

With the noticeable advancements in the field of machine learning, we can observe the horrendous increase in utilization of different types of deep neural network (DNN) against various kinds of tasks. While state-of-the-art convolutional neural networks (CNN) and recurrent neural networks (RNN) can deliver the performance comparable or in some cases even superior to humans, especially in the fields such as image recognition [1,2,3], speech recognition [4] and natural language processing [5], on the other side some kinds of DNNs can fall behind the conventional classification methods, mostly against the tasks involving structured data. Regardless of the strengths and weaknesses of different kinds of DNNs, one biggest weakness is common to them all – finding the right learning parameters and network topology to achieve the best performance for a given task. The biggest challenge in finding the right parameters settings for the DNNs is that there is no general rule or recipe to follow, which would guarantee a good outcome. It more or less depends on our previous experience and trying out different parameter settings.

Not many studies were conducted addressing the problem of parameters settings of DNN. One such early attempt is presented by authors [6], where the tuning of the structure and parameters of a neural network using an improved genetic algorithm is presented. The use of a genetic algorithm is not surprising, as the problem of parameter setting can be regarded and represented as an optimization problem, and evolutionary algorithms have proven to be successful in solving such problems. In this manner, the use of evolutionary approaches were common to few studies that followed [7,8], until in the recent year, based on the published studies in this field, the trend of optimizing learning parameters for a NN seems to pick up. There are several studies [9,10,11], where the authors are trying to optimize the architecture and training parameters of feed-forward DNNs, CNNs and RNNs, again with the use of evolutionary approaches.

Swarm intelligence algorithms, like evolutionary algorithms, are fast and efficient algorithms for solving discrete as well as continuous optimization problems as authors presented in [12,13]. These families of algorithms consist of a population of individuals that undergo variation operators governed by some principles inspired by natural and biological systems [14]. For instance, bat algorithm [15] that mimic the echolocation of micro-bats, undergo the variation operator guided by this physical phenomenon, while individuals in firefly algorithm [16] undergo variation operator that is based on bioluminescence phenomenon of fireflies.

It has been shown recently, that swarm intelligence algorithms have some advantages over evolutionary algorithms. They generally require less computational resources [17] and perform especially well when dealing with a smaller number of function evaluations [18], while genetic algorithms and differential evolution perform relatively better when the computational budget is large [18]. As the learning process of DNNs is already computationally very demanding, the introduction of a population-based iterative optimization method over it requires a lot of computa-

tional resources. For this purpose, in our recent work, we adopted the swarm intelligence approach to attempt to optimize the training parameters of DNNs [19] as well as the architecture of a feed-forward DNNs [20]. By setting the learning parameters in this way, the prediction performance of the used DNNs improved.

Encouraged by these promising results, in this paper we present a new, expanded method, titled as PSDNN.BA, PSDNN.HBA, or PSDNN.FA (Parameter Setting for Deep Neural Network, using Bat Algorithm / Hybrid Bat Algorithm / Firefly Algorithm), which utilizes swarm intelligence approaches for parameter settings of DNNs. The conceptual architecture of our expanded method is presented in a Fig. 1.

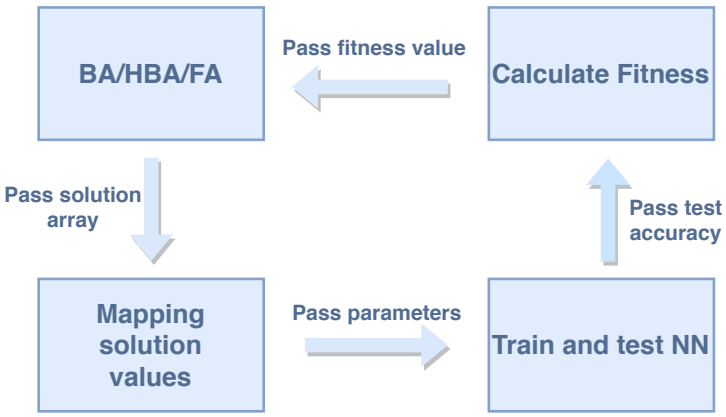

Fig. 1: The conceptual diagram of proposed PSDNN.BA/PSDNN.HBA/PSDNN.FA method.

The main goal of our research is to study whether a NN with parameters set by utilizing our proposed method will provide better classification performance than NN with recommended, sanely handpicked settings. The main advantage of the proposed method is the very straightforward usage with various feed-forward NN topologies and different datasets, without the need to manually search for right learning parameters.

To validate the proposed method, we applied its three variants to the problem of distinguishing between phishing and legitimate websites. A 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 [21]. 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 them from accessing their accounts, leading to a lack of financial credibility [22].

For the purpose of in-depth performance evaluation of our proposed method

variants, we conducted experiments against four different kinds of phishing websites datasets:

- The Phishing Websites dataset by Mohhamad with 30 features and 11,055 instances,
- The Phishing Websites dataset by Abdelhamid with 9 features and 1,353 instances,
- The Phishing Websites dataset by Vrbančič - small with 111 features and 58,645 instances and
- The Phishing Websites dataset by Vrbančič with 111 features and 88,657 instances.

As can be observed from the list, the datasets with various number of instances, features and decision classes were used. The last two of the listed datasets we prepared on our own.

The main contributions of this paper can be summarized as follows:

- a new, expanded method for parameter setting of deep neural networks utilizing nature-inspired algorithms is presented,
- the presented method variants are evaluated on a four different Phishing datasets, two of which we collected on our own and are presented for the first time and
- the in-depth performance comparison study is conducted against the conventional classifier logistic regression and baseline neural network.

The remaining of the paper is organized as follows. Section 2 briefly describes the methods we used. In Section 3 we present the proposed PSDNN.BA/PSDNN.HBA/PSDNN.FA method, followed by the description of the experimental setup in Section 4. Results of our conducted experiments are presented in Section 5 and finally, our conclusions and final thoughts are gathered in Section 6.

2. Methods

In this section the utilized methods and algorithms of our proposed method are presented more in-depth. First, we describe the principles of swarm intelligence and present the algorithms which were used and secondly we present the principles and motivation behind the deep learning. Our proposed method based on presented methods and algorithms is in-depth described in the next section.

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 nature to get inspiration for the design of complex algorithms. Such algorithms mostly mimic the biological

features of some fascinating animal species, like 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 the bat algorithm (BA) [15], hybrid bat algorithm (HBA) [23] and firefly algorithm (FA) [16,24]. All mentioned algorithms are characterized in the next subsection.

2.1.1. *Bat algorithm*

Bat algorithm is a member of the swarm intelligence family that is inspired by the behavior of micro-bats. BA was developed by Xin-She Yang in 2010 [15]. 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 Algorithm 1, where parameter D denotes dimension of the problem, Np is population size, MAX_FES is number of function evaluations, A_i is loudness and r_i is pulse rate.

Algorithm 1 Canonical bat algorithm

Input: Bat population $\mathbf{x_i} = (x_{i1}, \ldots, x_{iD})^T$ for $i = 1 \ldots Np$, MAX_FES.
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(x_best);
4: while termination_condition_not_meet do
5:   for i = 1 to Np do
6:     y = generate_new_solution(x_i);
7:     if rand(0,1) > r_i then
8:       y = improve_the_best_solution(x_best)
9:     end if
10:    f_new = evaluate_the_new_solution(y);
11:    eval = eval + 1;
12:    if f_new ≤ f_i and N(0,1) < A_i then
13:      x_i = y; f_i = f_new;
14:    end if
15:    f_min=find_the_best_solution(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 newly generated solution,
- conditional saving of best solution: the new best solution is saved under some probability that is denoted by parameter 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 [15].

2.1.2. *Hybrid bat algorithm*

Hybrid bat algorithm [23] 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 replaced by the mutation strategy. Hybridization 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 finding parameter settings for deep neural network training that belong to small-scale optimization problems.

Algorithm 2 Hybridization step in bat algorithm

if $rand(0,1) > r_i$ **then**
  $r_{i=1\ldots3} = \lfloor \text{rand}(0,1) * Np + 1 \rfloor$;
  $n = \text{rand}(1, D)$;
  for $i = 1$ **to** D **do**
    if $((\text{rand}(0,1) < CR) || (n == D))$ **then**
      $\mathbf{y}_n = \mathbf{x}_{r1,n} + F * (\mathbf{x}_{r2,n} - \mathbf{x}_{r3,n})$;
      $n = (n+1)\%(D+1)$;
    end if
  end for
end if

2.1.3. *Firefly algorithm*

Firefly algorithm is another swarm intelligence algorithm that was developed by Xin-She Yang in 2008 [16,24]. The phenomenon of fireflies lies in the flashlights. Flashlights attract mating partners as well as serve as a mechanism of protection. Their light intensity I decreases when the distance r from the light source increases according to term $I \propto r^2$.

Yang proposed three idealized rules that govern FA:

- all fireflies are unisex,
- their attractiveness is proportional to their brightness, and
- the brightness of a firefly is affected or determined by the landscape of the objective function.

According to the Algorithm 3, main components of FA algorithms are the following:

- InitializeFA(): the initial population is being generated,
- EvaluateFA(): evaluating the new solution,
- OrderFA(): forming an intermediate population by ordering the solutions into the original population according to the ascending values of the objective function,
- FindTheBestFA(): determining the best solution in the current population. The best solution found so far is preserved.
- MoveFA(): moving the fireflies towards the search space according to the attractiveness of their neighbor's solutions.

Algorithm 3 Canonical Firefly algorithm

1: $t = 0; s^* = \emptyset; \gamma = 1.0;$
2: $P^{(0)}$ = initialize_FA();
3: **while** ($t <$ MAX_FES) **do**
4: $\alpha^{(t)}$= AlphaNew();
5: evaluate_FA($P^{(t)}, f(s)$);
6: order_FA($P^{(t)}, f(s)$);
7: s^* = find_the_best_FA($P^{(t)}, f(s)$);
8: $P^{(t+1)}$ = move_FA($P^{(t)}$);
9: $t = t + 1;$
10: **end while**

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 [25,26].

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 [27]. Many deep learning applications use feed-forward NN topologies, 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) [28], 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 [29,30].

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 [31]. 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 [30].

3. Proposed method

The proposed method is based on applying a selected swarm intelligence algorithm (we used BA, HBA and FA) to the problem of parameter setting for a DNN (PS-DNN), giving us three variants named PSDNN.BA, PSDNN.HBA, and PSDNN.FA (Fig. 1). The task of optimizers BA/HBA/FA is to find an optimal solution – the parameter values to train a given DNN. The solution consists of one-dimensional array with four items, each representing one of the parameters we are trying to optimize: 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 and it matches the number of neurons of the input layer). The task of finding the optimal solution is in an iterative manner. Each solution produced by the utilized optimizer is evaluated through the process of training and testing DNN. Based on the achieved performance, fitness value for a used solution (parameter settings) is calculated. The fitness value represents the quality of found solution (the predictive performance of a neural network, trained using the given set of parameters, in our case). In the final step, the fitness value is reported back to the optimizer algorithm, which constructs a new solution. In this manner, the solution with the best fitness value in the last iteration cycle is taken as the final solution, and always represents the best solution found during the optimization.

The Algorithm 4 is presenting our proposed method more in depth. As we can see

most of the steps are actually similar to steps presented in Algorithm 1 [a]. However, the most important components that differ from Algorithm 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 4 Proposed method (PSDNN.BA)

Input: Control parameter settings, Dataset

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 PSDNN.BA, PSDNN.HBA, and PSDNN.FA 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, 4 and 5 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 the first hidden layer of our NN topology and the n is representing the numbers of neurons in the first

[a]Pseudo-code presents the proposed method variation utilizing BA optimizer. The utilization of HBA and FA optimizers is done in the same manner.

hidden layer of NN while the m is denoting the number of features based on given dataset.

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

$$y_2 = \left\lfloor \frac{x[i] * 100}{2} + 1 \right\rfloor; y_2 \in [1, 100] \tag{3}$$

$$y_3 = \lfloor x[i] * (m * 2 - 1) + 1 \rfloor; y_3 \in [1, m * 2], m \in \mathbb{N} \tag{4}$$

$$y_4 = x[i] * (0.1 - 10^{-6}) + 10^{-6}; y_4 \in [10^{-6}, 0.1] \tag{5}$$

3.2. *Fitness function*

We defined fitness function using the accuracy of classification calculated on a test part of our search subset, which represents the 20% of the initially given dataset. Formally, we can express fitness function as presented in equation (6), where $search\_test\_acc$ stands for previously mentioned classification accuracy. Because BA, HBA and FA 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 a value 1.

$$f(search\_test\_acc) = 1 - search\_test\_acc \tag{6}$$

3.3. *Neural network*

For the topology of a NN, we propose a feed-forward NN with two fully-connected hidden layers presented in Fig. 2. The arhitecture of the NN is constructed based on out previous experience with feed-forward NNs [19,20] and practical recommendations presented in [32]. The number of neurons on the input layer is equal to the number of features of the given dataset, while the number of neurons in the output layer is matched with the number of unique target classes of the dataset. The width of the first hidden layer is defined with a y_3 value from equation (4) in the previous subsection, while the number of neurons is the same as the number of features. For the classification problems, we propose using a non-linear activation function ReLU on hidden layers and a softmax function on the output layer. Instead of conventional stochastic gradient descent (SGD) [33] optimizer we propose ADAM [34] optimizer, which is designed to combine the advantages of two recently popular methods: AdaGrad [35] and RMSProp [36].

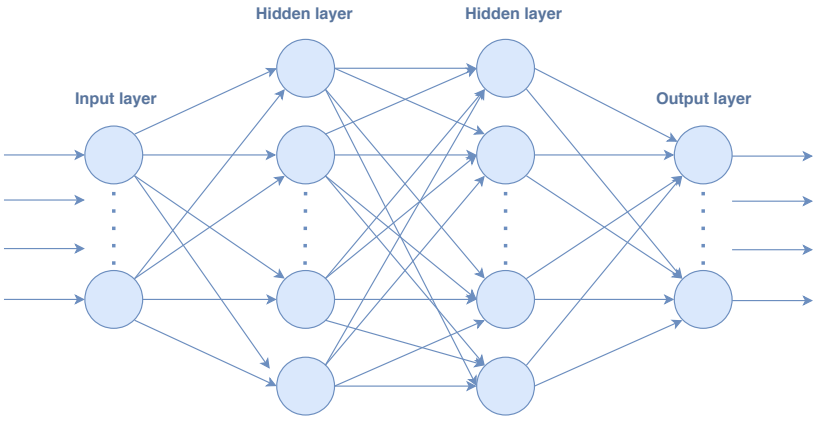

Fig. 2: The topology of used feed-forward neural network.

4. Experiment setup

In this section, we describe the experimental setup that we followed to evaluate the performance of the proposed approach. First, we describe the data collection and composition of the four phishing websites datasets, which were used for performing the empirical experiments. Then we list the classification algorithms, which were used to identify the phishing websites in four different datasets. Finally, we explain the evaluation method and the classification performance metrics used in the comparison. The obtained experimental results are presented and discussed in the next section.

4.1. *The Phishing Websites datasets*

There are different approaches to tackle the problem of identification of phishing websites reported in literature [37]. Typically, the two most technical methods in fighting phishing attacks are the blacklist and the heuristic-based [38,39]. In the blacklist method, the requested URL is compared with a predefined phishing URLs. The downside of this method is that it typically does not 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 [40].

In general, the majority of publicly available phishing datasets are prepared using the heuristic-based approach. The most common representatives of such datasets, used in research studies are The Phishing Websites dataset by Mohammad [41] and The Phishing Websites dataset by Abdelhamid [37]. Regardless of the manner of technical methods used in the later preparation process of the dataset, there is a generally used recipe on how to obtain the data and which or what kind of features to extract from the data. The mentioned recipe roughly consists of the following steps:

- manually obtain the phishing websites URLs,
- manually obtain the legitimate websites URLs,
- automatically extract and/or calculate features from URLs and/or websites, and
- optionally automatically extract and apply some heuristic rules.

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, in general, belong to one of the following groups: Address Bar based features, Abnormal based features, HTML and JavaScript-based features, and Domain-based features. In table 1 are presented some of the most important features of each group.

Table 1: Some of the most important features, used to detect the phishing websites

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
	...

The constructed features, shown in Table 1 can take in general a binary or natural 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. In the case of the application of the heuristic rules is also common to construct the features which take ternary values. Those hold, in addition to the binary ones, the value "suspicious", where the existence of the feature in a specific ratio determines the value assigned for that feature.

After an in-depth study of the existing phishing datasets, we came to the conclusion, that the majority of such datasets is more or less outdated and not quite

large. In order to properly benchmark our proposed method variants on various kinds and sizes of datasets, we were lacking larger dataset with the higher number of features. On those bases we prepared two new, larger phishing website dataset variants.

We conducted the experiment on total four phishing datasets: The Phishing Websites dataset by Mohhamad [41], The Phishing Websites dataset by Abdelhamid [37] and two variants of our new dataset - The Phishing Websites dataset by Vrbančič, which are accessible on the following address [42]. Each of them is characterized in the following subsections.

4.1.1. *The Phishing Websites dataset by Mohammad*

The heuristic-based approach has been applied in the case of preparing the Phishing Websites dataset by Mohammad [41]. 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 [43]. 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 [44], 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.

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 [37,41]. 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:

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

Similar heuristic rules have been determined for all other features as well. The basic information about the Phishing Websites dataset by Mohammad is presented in Table 2.

4.1.2. *The Phishing Websites dataset by Abdelhamid*

The Phishing websites dataset by Abdelhamid is in general prepared in the same manner as the previously described phishing dataset by Mohammad. The authors collected over 1,350 websites from different sources. The phishing websites authors

Table 2: The basic information about the Phishing Websites dataset by Mohammad

Parameter	Value
Number of features	30
Number of instances	11,055
Number of classes	2 classes
Distribution of classes	4,898 phishing websites
	6,157 legitimate websites

collected from the Millersmiles [45] and Phishtank data archives, which are free community sites for sharing phishing data while the legitimate websites were collected from Yahoo directory utilizing authors own developed PHP script [37].

In their study, after performing the frequency analysis against collected website URLs, the authors have identified sixteen different features plus the target class. The authors have also performed some association rule mining and applied heuristic rules against the obtained dataset. Also, all of the collected features were assessed in order to identify the smallest significant set of features. Finally, the authors have applied the Chi-Square methodology to further assess the relevance of extracted features, which has shown that nine out of ten features have a correlation with the class attribute values. The final outcome of the performed study is the phishing websites dataset which basic information is presented in Table 3.

Table 3: The basic information about the Phishing Websites dataset by Abdelhamid

Parameter	Value
Number of features	9
Number of instances	1,353
Number of classes	3 classes
Distribution of classes	702 phishing websites
	103 suspicious websites
	548 legitimate websites

4.1.3. *The Phishing Websites dataset by Vrbančič*

In order to evaluate the performance of our proposed method variants against larger datasets with many features we prepared two variants of new phishing websites datasets, a balanced and an imbalanced one. Following the common recipe for preparing such datasets, we firstly collected a list of confirmed phishing URLs (30,647) from the Phishtank website. The list of legitimate URLs was obtained from Alexa ranking website from which we gathered 58,000 legitimate website URLs. We also obtained the list of community labeled and organized URLs (27,998) which are containing the objectively reported news and are in that manner also legitimate.

From those gathered URLs we extracted 111 features, in our case mostly ad-

dress bar based and domain based features. For extracting the features which are categorized in the group of address bar based features, we performed a counting of special characters - symbols on different parts of URL (whole URL, domain, folder, file, parameters) as presented in Table 4. Categorized under domain based features we extracted features such as search time response domain, time (in days) of domain activation, the number of resolved name servers, time-to-live value associated with the hostname, etc. All values of the address bar based features are holding the natural numbers including zero ($\mathbb{N} \cup \{0\}$), while the missing values are presented as -1 value.

Table 4: List of symbols used for extracting the address bar based features.

Symbols	
dot (.)	space ( )
hyphen (-)	tilde (~)
underline (_)	comma (,)
slash (/)	plus (+)
question mark (?)	asterisk (*)
equal (=)	hashtag (#)
at (@)	dollar ($)
and (&)	percent (%)
exclamation (!)	

In contrast to the other two datasets, we did not perform any additional heuristics or apply any rules against the extracted features, so the values of features are as they were collected. From the collections of phishy and legitimate websites, we prepared two variants of the dataset, one with more balanced data, named Vrbančič-small and one with unbalanced data named Vrbančič. The Vrbančič-small dataset consists instances of extracted features from Phishtank URLs and instances of extracted features from community labeled and organized URLs representing legitimate ones, resulting in a dataset with quite equal distribution between classes. On the other hand, the Vrbančič dataset consist from all of the instances from the Vrbančič-small dataset and the additional instances of extracted features from Alexa top sites URL list, so that the latter dataset variant is quite unbalanced in favor of legitimate instances. The basic information of dataset variants is presented in Table 5.

The prepared variants of datasets allows us to perform experiments and validate the performance of our proposed method variants against datasets of various sizes, which enables us gather more generalized view on the methods performance.

4.2. *Baseline and compared methods*

In order to assess the performance of the proposed approach and its applicability to the identification of phishing websites, a series of tests has been carried out, in which we measured website classification performance.

Table 5: The basic information about the Phishing Websites dataset variants by Vrbančič

Parameter \ Dataset	Vrbančič-small	Vrbančič
Number of features	111	111
Number of instances	58,645	88,647
Number of classes	2 classes	2 classes
Distribution of classes	30,647 phishing websites 27,998 legitimate websites	30,647 phishing websites 58,000 legitimate websites

The first classification method that we used in our comparison was a logistic regression (LR), which have its origin in statistics and share many similarities with artificial neural networks (ANN). LR and ANNs have common roots in statistical pattern recognition, and the ANNs can be seen as a generalization of LR [46]. The second classification method, which served as a baseline, we used a multilayer feed-forward neural network with handpicked training parameters (base), as will be described below. To assess the proposed meta-heuristic approach PSDNN, we used three different swarm intelligence approaches for setting the parameters of deep learning neural networks: a bat algorithm (PSDNN.BA), a hybrid bat algorithm (PSDNN.HBA), and a firefly algorithm (PSDNN.FA).

4.2.1. *Baseline neural network*

As a baseline classification method in the experiment, we defined a feed-forward NN with two fully connected hidden layers and a fully-connected output layer. The width (the number of neurons) of layers is set dynamically, based on the parameters passed in the initialization phase and is matching the number of features in a given dataset resulting in the baseline neural network architectures presented in Table 6.

Table 6: Properties of baseline neural network architectures.

Dataset	Nr. of neurons in hidden layers	Nr. of neurons in output layer
Phishing Websites dataset by Mohammad	30	2
Phishing Websites dataset by Abdelhamid	9	3
Phishing Websites dataset variants by Vrbančič	111	2

The number of neurons in the output layer is set to match the number of target classes in the given dataset. Both hidden layers are using a non-linear activation function ReLU, while the output layer is using softmax. As proposed in Section 3.3, we used ADAM as the optimizer function.

4.2.2. *Bat, hybrid bat and firefly algorithm parameters*

For initializing the parameters of the bat, hybrid bat and firefly algorithms, we used the values presented in Table 7.

Table 7: Used parameter values for bat, hybrid bat, and firefly algorithms

Parameter	BA	HBA	FA
Dimension of the problem	4	4	4
Population size	40	40	40
Number of function evaluations	400	400	400
Lower bound	0.0	0.0	0.0
Upper bound	1.0	1.0	1.0
Loudness	0.5	0.5	–
Pulse rate	0.5	0.5	–
Min. frequency	0.0	0.0	–
Max. frequency	2.0	2.0	–
F (Scaling factor)	–	0.5	–
CR (Crossover probability)	–	0.5	–
Alpha	–	–	0.5
Betamin	–	–	0.2
Gamma	–	–	1.0

Initialized with the given parameters, the three swarm intelligence algorithms are used to search for the optimal values for the following parameters: a number of epochs, batch size, learning rate and the number of neurons in the first hidden layer of NN.

4.2.3. *Learning parameters*

Based on [34] and our previous experience in machine learning, we handpicked the learning parameters as optimal as possible. For batch size we choose 32, for learning rate 10^{-3} and for the number of epochs 150.

The performance of baseline NN with these parameters should give us a good starting point in comparison of performance between conventional methods (LR, base) and our proposed methods (PSDNN.BA, PSDNN.HBA, PSDNN.FA).

4.2.4. *Computing environment*

For the proposed deep learning approach the experiments were implemented with Python programming language using the following libraries: NiaPy [47], Keras [48], NumPy [49], Pandas and scikit-learn [50]. All classification models were used with their default settings.

All of the conducted experiments were performed using the 8-core Intel Core i7-6700 CPU with clock speed at 3.4GHz and 16GB of DDR4 memory.

4.3. *Evaluation method and metrics used*

Considering the imbalance of the datasets, we performed the evaluation using two predictive performance measures – the accuracy and the $F1$-macro score. While accuracy measures the ratio of correctly classified websites among all tested regardless of their class (whether a website is phishy or not), the $F1$-macro score is a harmonic mean of the precision and recall, averaged over all decision classes. The $F1$ score is computed for each class within the dataset and then the average is obtained over all classes. In this way, equal weight is assigned to each class regardless of the class frequency.

To objectively evaluate the performance of the proposed classification approach, and to compare it with the existing classification algorithms, we adopted the 10-fold cross-validation approach, where a dataset is divided into train and test sets in a ratio of 90:10. In this manner, within one fold, we trained the classification model using the instances (websites) from 9 out of 10 folds and then tested the model on the remaining fold (the remaining websites, which were not used for training).

In the case of swarm intelligence approaches for parameter settings of DNN learning process, which are iterative in nature, all of the used datasets were initially divided into two subsets in ratio 80:20. The smaller subset (20%) was used for searching the best parameter settings utilizing our proposed method variants PSDNN.BA/PSDNN.HBA/PSDNN.FA, while the larger subset (80%) was used to perform ten-fold cross-validation, in order to keep the performance evaluation as fair as possible. Folds were made using the stratification method, which splits sets in such a way that the distribution of classes remains the same across all folds.

All the results reported are the averaged accuracy and $F1$-macro scores, obtained on the test websites over all runs for each of the 10 folds, if not specified otherwise.

5. Results

Using the described experiment setup, we obtained the classification results for five classification methods (LR, base, PSDNN.BA, PSDNN.HBA, PSDNN.FA) on four different datasets. In this section, the obtained results are presented, compared and discussed in detail.

5.1. *Results on the Mohammad dataset*

The classification results (accuracy and $F1$-macro score), obtained on the Mohammad dataset, are presented in Table 2. The best results, for both accuracy and $F1$-macro score and for each fold, are presented in bold. We can see, that the firefly method achieved the best overall accuracy and $F1$-macro score. All three swarm approaches turned out to outperform the two compared baseline methods, LR and baseline NN, in all single folds, while the results of LR and base NN are almost identical. Interestingly, the accuracy and $F1$-macro results are very similar and show practically no difference.

To further compare all the methods, we calculated their ranks frequencies (Fig. 3). The calculated trends of ranks frequencies suggest that the best method is firefly (with an average rank of 4.8 out of 5 methods), followed by a bat (4.0), hybrid bat (3.2), LR (1.5), and finally the base NN (1.3). The firefly method turned out to be superior to others in 8 out of 10 folds, while in the remaining two folds it achieved the second best rank (being outperformed only by the bat method).

Table 8: The comparison of predictive performance on Mohammad dataset

	accuracy					F1-macro				
	LR	base	bat	hbat	firefly	LR	base	bat	hbat	firefly
$fold_1$	93.79	93.91	96.50	96.28	**96.73**	93.69	93.81	96.45	96.24	**96.68**
$fold_2$	93.67	93.79	96.05	96.16	**96.72**	93.59	93.70	95.98	96.12	**96.68**
$fold_3$	91.30	91.30	93.33	92.99	**96.84**	91.20	91.20	93.28	92.91	**96.79**
$fold_4$	91.63	91.52	95.14	94.80	**95.93**	91.47	91.36	95.05	94.73	**95.86**
$fold_5$	92.19	91.97	94.91	92.31	**96.38**	92.09	91.87	94.82	92.07	**96.34**
$fold_6$	92.53	92.42	**96.38**	95.59	96.27	92.44	92.34	**96.31**	95.53	96.23
$fold_7$	93.55	93.44	96.83	94.91	**96.95**	93.46	93.35	96.79	94.86	**96.91**
$fold_8$	92.08	92.08	95.25	94.68	**96.38**	91.97	91.96	95.20	94.63	**96.33**
$fold_9$	93.78	93.21	**96.83**	95.48	96.72	93.66	93.09	**96.80**	95.44	96.67
$fold_{10}$	93.67	94.00	96.38	96.83	**97.62**	93.57	93.92	96.33	96.79	**97.59**
avg	92.82	92.76	95.76	95.00	**96.65**	92.71	92.66	95.70	94.93	**96.61**

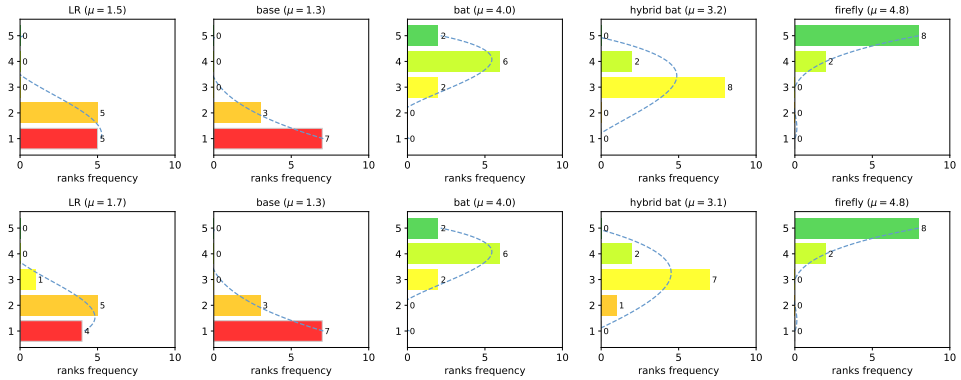

Fig. 3: The comparison of achieved ranks on the Mohammad dataset regarding accuracy (top) and f1-macro score (bottom).

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, regarding both accuracy and $F1$-macro score, as was expected already from observing Fig. 3.

The post-hoc Wilcoxon signed ranks test of accuracy ranks with Holm-Bonferroni correction revealed that both LR (p=0.0051 when compared to bat, p=0.005 when compared to hybrid bat, and p=0.005 when compared to firefly) and base NN (p=0.005, p=0.0051, p=0.005) were significantly outperformed by all three swarm methods. On the other hand, base NN (with handpicked parameters) was not significantly different from LR (p=0.2603), and even achieved slightly worse ranks. Among the three swarm methods, firefly significantly outperformed both bat (p=0.0163) and hybrid bat (p=0.0051), while bat outperformed its hybrid variant (p=0.0284). The post-hoc analysis of $F1$-macro ranks is almost identical, with the same statistical significances and almost the same p values.

5.2. *Results on the Abdelhamid dataset*

The classification results on the Abdelhamid dataset are presented in Table 9. As this dataset has three decision classes (legitimate, suspicious, and phishing websites), the accuracy and $F1$-macro results, in this case, are quite different. While firefly is again the best method regarding overall accuracy, the hybrid bat achieved the best overall $F1$-macro score among all the compared methods. The results of the base NN on this dataset are by far the worse among all methods and lag also far behind those of the LR. Again, all three proposed swarm approaches outperformed the existing two methods.

When looking at the ranks frequencies (Fig. 4), we can see that the base NN is undoubtedly the worse method here, while LR being the second worst. The differences among the three swarm methods seem quite low. The Friedman test confirmed a significant difference among the five methods (p<0.001), both regarding accuracy and $F1$-macro score.

The post-hoc Wilcoxon signed ranks test of accuracy confirmed that the base NN is significantly outperformed by all other methods, and the LR is outperformed by all three swarm methods, both regarding accuracy and $F1$-macro. The difference among the three swarm methods on the Abdelhamid dataset are not significant, neither regarding accuracy nor $F1$-macro score. The most best ranks regarding accuracy were obtained with the firefly method (4 straight wins and 2 tied wins), and regarding $F1$-macro with the hybrid bat method (7 wins).

5.3. *Results on the Vrbančič-small dataset*

The classification results on the Vrbančič-small dataset are presented in Table 10. This dataset is, besides the Mohammad dataset, the most balanced one (having a very similar frequency of all decision classes), which results in almost the same results regarding accuracy and $F1$-macro score. The best overall accuracy as well as $F1$-macro score is again achieved by the firefly method, while the results of the remaining four methods are very similar.

When looking at the ranks frequencies (Fig. 5), we can see that the firefly is undoubtedly the best method (with the rank 4.6), achieving 9 out of 10 best ranks

Table 9: The comparison of predictive performance on Abdelhamid dataset

	accuracy					F1-macro				
	LR	base	bat	hbat	firefly	LR	base	bat	hbat	firefly
$fold_1$	80.91	51.82	**87.27**	84.55	85.45	56.35	22.75	**85.07**	83.77	83.57
$fold_2$	83.49	52.29	87.16	**88.07**	77.98	57.59	22.89	71.88	**79.72**	68.63
$fold_3$	86.11	51.85	**89.81**	84.26	85.19	59.60	22.76	**85.34**	76.43	78.37
$fold_4$	82.41	51.85	82.41	**84.26**	**84.26**	57.15	22.76	74.97	**81.37**	76.30
$fold_5$	81.48	51.85	81.48	87.96	**89.81**	56.31	22.76	72.64	**85.54**	85.54
$fold_6$	81.48	51.85	**86.11**	85.19	**86.11**	56.17	22.76	77.65	**80.11**	65.79
$fold_7$	82.41	51.85	87.04	90.74	**91.67**	56.72	22.76	82.11	**88.32**	83.32
$fold_8$	84.26	51.85	82.41	**87.04**	81.48	58.24	22.76	73.82	**81.10**	74.25
$fold_9$	80.56	75.93	85.19	84.26	**88.89**	62.80	52.47	80.77	**85.47**	84.89
$fold_{10}$	82.24	76.64	85.98	84.11	**89.72**	56.65	52.25	66.05	75.30	**85.48**
avg	82.53	56.78	85.49	86.04	**86.06**	57.76	28.70	77.03	**81.71**	78.61

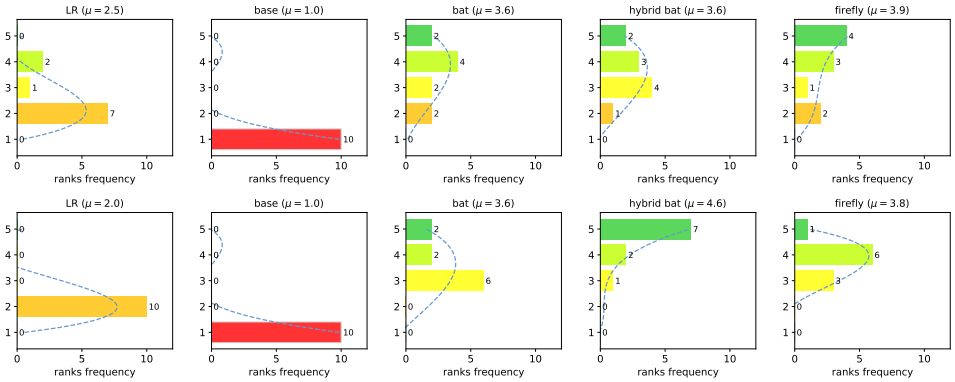

Fig. 4: The comparison of achieved ranks on the Abdelhamid dataset regarding accuracy (top) and f1-macro score (bottom).

both regarding the accuracy and $F1$-macro score. It is interesting, though, that on the remaining fold, the firefly method performed the worst. The second best rank achieved the base method (rank 3.4), followed by the bat (rank 3.0), hybrid bat (rank 2.1), and LR (rank 1.9).

The Friedman test confirmed significant differences (p=0.008) in both accuracy and $F1$-macro score. The post-hoc Wilcoxon test of accuracy results showed that the firefly method is significantly better than base (p=0.0367), bat (p=0.0469), hybrid bat (p=0.0069), and LR (p=0.0069). The post-hoc analysis of $F1$-macro score showed that firefly significantly outperformed base (p=0.0469), hybrid bat (p=0.0069), and LR (p=0.0069), while the differences between firefly and bat do not seem to be significant (p=0.0593), although by a very narrow margin.

Table 10: The comparison of predictive performance on Vrbančič-small dataset

	accuracy					F1-macro				
	LR	base	bat	hbat	firefly	LR	base	bat	hbat	firefly
$fold_1$	88.09	89.62	86.98	90.30	**91.20**	88.05	89.60	86.98	90.27	**91.16**
$fold_2$	88.04	90.09	88.90	87.62	**91.16**	87.96	90.04	88.80	87.61	**91.11**
$fold_3$	87.62	89.24	**89.77**	87.38	86.66	87.53	89.23	**89.75**	87.22	86.40
$fold_4$	87.21	87.57	88.55	88.34	**90.75**	87.13	87.43	88.53	88.27	**90.68**
$fold_5$	87.77	89.66	88.83	88.30	**90.17**	87.68	89.66	88.78	88.19	**90.13**
$fold_6$	86.91	88.83	88.36	67.21	**89.28**	86.83	88.75	88.30	63.37	**89.18**
$fold_7$	87.98	88.98	87.40	87.68	**90.26**	87.90	88.88	87.29	87.68	**90.24**
$fold_8$	87.98	89.75	89.17	89.02	**91.32**	87.90	89.69	89.14	88.93	**91.27**
$fold_9$	87.29	86.57	88.77	88.49	**90.13**	87.23	86.34	88.70	88.38	**90.12**
$fold_{10}$	87.34	88.21	87.57	86.27	**90.81**	87.27	88.10	87.48	85.96	**90.78**
avg	87.62	88.85	88.43	86.06	**90.17**	87.55	88.77	88.37	85.59	**90.11**

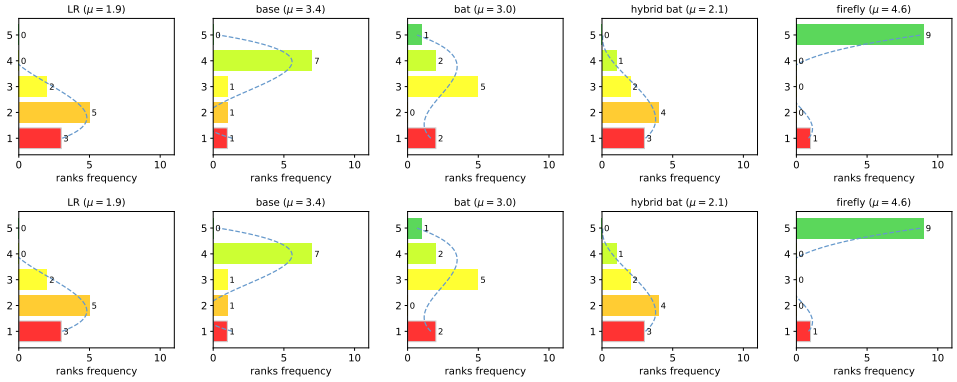

Fig. 5: The comparison of achieved ranks on the Vrbančič-small dataset regarding accuracy (top) and f1-macro score (bottom).

5.4. *Results on the Vrbančič dataset*

Finally, the classification results on the Vrbančič dataset are presented in Table 11. In contrast to Vrbančič-small, this dataset is not balanced. However, this fact does not seem to have a big influence on the $F1$-macro score results, which are very similar to accuracy. Once more, the best overall accuracy, as well as $F1$-macro score, are achieved by the firefly method, following by both bat methods, then the base, while LR performed the worst, both regarding accuracy and $F1$-macro score.

On this dataset, the methods seem to be very consistent in achieving the same rank (Fig. 6). The best overall rank was achieved by the firefly method (rank 4.6), followed by a hybrid bat (4.1), bat (3.3), base (1.9), and LR (1.1).

The Friedman test confirmed the significant differences among methods ($p<0.001$) in both accuracy and $F1$-macro score. The post-hoc analysis confirmed that LR is significantly worse than all other methods (with $p<0.009$) and the base

is significantly worse than all three swarm methods (with p=0.005 or p=0.0051). Among the swarm methods, however, the firefly significantly outperformed bat regarding accuracy (p=0.0218) and $F1$-macro score (p=0.0284), while other differences were not significant.

Table 11: The comparison of predictive performance on Vrbančič dataset

	accuracy					F1-macro				
	LR	base	bat	hbat	firefly	LR	base	bat	hbat	firefly
$fold_1$	91.57	92.23	93.64	**94.15**	94.02	90.82	91.17	93.09	**93.59**	93.50
$fold_2$	92.22	92.68	93.98	94.53	**94.77**	91.52	92.10	93.38	94.01	**94.27**
$fold_3$	92.12	91.95	94.14	94.18	**94.67**	91.40	90.85	93.59	93.60	**94.12**
$fold_4$	91.34	92.19	93.98	94.06	**94.51**	90.50	91.55	93.36	93.45	**93.99**
$fold_5$	91.28	93.22	93.99	93.53	**94.20**	90.52	92.47	93.45	92.73	**93.65**
$fold_6$	91.79	92.51	93.75	93.92	**94.11**	91.08	91.92	93.22	93.34	**93.59**
$fold_7$	91.76	93.15	94.09	94.18	**94.80**	91.02	92.41	93.48	93.62	**94.29**
$fold_8$	92.31	93.33	94.43	95.28	**95.46**	91.58	92.74	93.83	94.78	**94.96**
$fold_9$	91.30	92.62	93.57	**94.08**	93.63	90.51	91.85	92.84	**93.49**	92.87
$fold_{10}$	91.51	92.79	**94.20**	93.96	93.78	90.71	92.11	**93.63**	93.36	93.07
avg	91.72	92.67	93.98	94.19	**94.39**	90.97	91.92	93.39	93.60	**93.83**

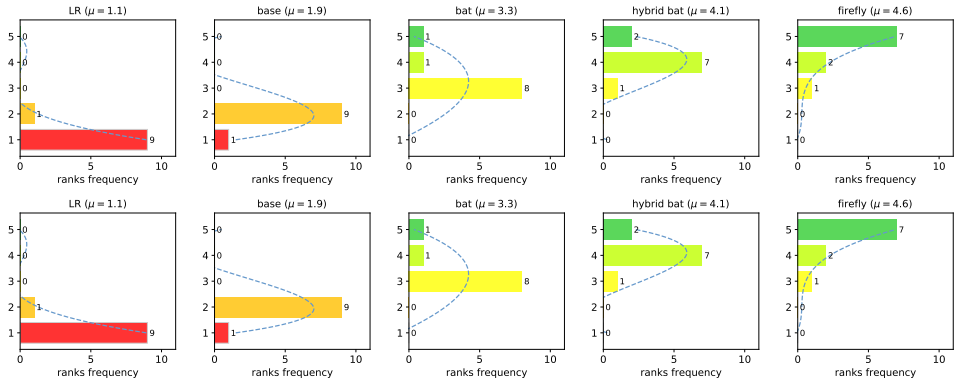

Fig. 6: The comparison of achieved ranks on the Vrbančič dataset regarding accuracy (top) and f1-macro score (bottom).

5.5. *The resulting parameter settings*

The performed analysis of empirical results shows the advantage of swarm methods when compared to LR and base NN. For this purpose, the calculated parameter settings of their best solutions are presented in Table 12; for comparison, also the handpicked values for the base NN are presented.

We can see that all three swarm methods resulted in a higher number of neurons than handpicked, with the only exception being the Vrbančič-small dataset. Espe-

Table 12: The parameter settings of the best solutions, as obtained by different swarm approaches, and compared to handpicked values for base NN

	Mohammad dataset				Abdelhamid dataset			
	base	bat	hbat	firefly	base	bat	hbat	firefly
num. of epochs	150	155	135	192	150	134	132	171
batch size	32	44	101	37	32	46	62	24
learning rate	.001	.018471	.046237	.005338	.001	.061724	.048576	.060051
num. of neurons	30	50	42	50	9	10	10	16

	Vrbančič-small dataset				Vrbančič dataset			
	base	bat	hbat	firefly	base	bat	hbat	firefly
num. of epochs	150	140	176	153	150	199	100	175
batch size	32	61	65	69	32	1	27	15
learning rate	.001	.000497	.002134	.001371	.001	.000036	.000264	.000194
num. of neurons	111	4	49	57	111	196	221	186

cially the firefly method, which generally performs the best, produced the highest number on neurons, which suggests that a higher number of neurons allows for better predictive performance. Interestingly though, the numbers of neurons of all three swarm methods on the Vrbančič-small dataset were a much smaller than handpicked.

On all four datasets, the firefly method used a higher number of epochs than handpicked. On the one hand, this confirms the common idea that more learning results in better models. On the other hand, however, this result suggests that the learning process here is not very prone to overfitting.

The differences between handpicked and optimized values regarding the batch size and the learning rate, especially when compared with the firefly method, are much smaller, suggesting that those two parameters were picked appropriately.

It is also interesting to compare the calculated parameter values for Vrbančič-small and Vrbančič datasets. The differences, especially regarding the number of neurons and batch size, are substantial. We must consider here, that Vrbančič-small is a balanced version of Vrbančič dataset, with the same features but with fewer instances. Considering that, it seems that additional instances, although making a dataset imbalanced, contribute more to the predictive performance (see Table 11) when compared to a balanced, undersampled version of the same (see Table 10).

Finally, in general, it seems to be beneficial to select a higher number of neurons and epochs, while the batch size and the learning rate should be smaller.

6. Threats to validity

We are aware of possible threats to the validity of the presented results and implications they bring. In machine learning approaches, the threats often relate to the diversity, quality, and quantity of the data. To reduce the construct validity regarding the data used to train and test the proposed method, we used four different

datasets, prepared by different authors from different data sources. To minimize the threat to external validity, we chose two publicly available phishing websites datasets and two datasets that we prepared ourselves. Nevertheless, our results may not be simply generalized to all possible specific situations.

The datasets for training and testing that we used rely crucially on how the websites were pre-classified. In general, for all the datasets some community sites for sharing phishing data were used to determine the suspicious websites, while the non-suspicious websites were collected from trusted directories. In such a way, the human errors and personal bias have been reduced as much as possible. Besides, our approach depends on the correctness of the underlying tools we utilize. To mitigate this risk, we used tools that are commonly used in machine learning community, such as scikit-learn[b] and Keras[c], which contains the classification methods used in our approach.

The potential threat to internal validity could be either splitting the data between train and test sets, or the non-deterministic nature of methods used. For the first part, we adopted a well-known 10-fold cross-validation approach, while for the second part, each result that is based on non-deterministic method has been obtained as an average of several algorithm runs.

7. Conclusions

In this paper, we presented a swarm intelligence based approach to parameter setting for learning deep neural networks. The proposed approach has been implemented using three different swarm intelligence algorithms, namely bat algorithm, hybrid bat algorithm, and firefly algorithm, and applied to the problem of phishing websites classification. The results, obtained from the conducted empirical experiments, have proven the proposed approach to be very promising. The predictive performance of the resulting deep neural networks, trained using the parameters values as optimized with the help of the proposed swarm intelligence based methods, improved significantly when compared to the manually tuned neural network. In general, the proposed firefly method showed the best results in classifying phishing websites, which was also statistically confirmed.

In the future, we would like to continue our work towards including additional learning parameters into the optimization process, possibly using different swarm intelligence algorithms (e.g., Cuckoo search algorithm) and expanding our experiments to other datasets. One step further from the research, presented in this paper, which we also intend to pursue in the future, is to use computational intelligence approaches to optimize the self-construction of different neural network topologies with various depths, widths and types of layers. Additionally, we plan to analyze the obtained solutions in order to possibly discover some insights into how to optimally construct a deep neural network for a specific purpose.

[b]https://scikit-learn.org/
[c]https://keras.io/

Acknowledgments

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

References

1. A. Krizhevsky, I. Sutskever and G. E. Hinton, Imagenet classification with deep convolutional neural networks, in *Advances in neural information processing systems*2012, pp. 1097–1105.
2. C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, A. Rabinovich *et al.*, Going deeper with convolutions Cvpr2015.
3. K. He, X. Zhang, S. Ren and J. Sun, Identity mappings in deep residual networks, in *European Conference on Computer Vision* Springer2016, pp. 630–645.
4. G. Hinton, L. Deng, D. Yu, G. E. Dahl, A.-r. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. N. Sainath *et al.*, Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups, *IEEE Signal Processing Magazine* **29**(6) (2012) 82–97.
5. Y. Goldberg, A primer on neural network models for natural language processing, *Journal of Artificial Intelligence Research* **57** (2016) 345–420.
6. F. H.-F. Leung, H.-K. Lam, S.-H. Ling and P. K.-S. Tam, Tuning of the structure and parameters of a neural network using an improved genetic algorithm, *IEEE Transactions on Neural networks* **14**(1) (2003) 79–88.
7. J.-T. Tsai, J.-H. Chou and T.-K. Liu, Tuning the structure and parameters of a neural network by using hybrid taguchi-genetic algorithm, *IEEE Transactions on Neural Networks* **17**(1) (2006) 69–80.
8. D.-S. Huang and J.-X. Du, A constructive hybrid structure optimization methodology for radial basis probabilistic neural networks, *IEEE Transactions on neural networks* **19**(12) (2008) 2099–2115.
9. R. Miikkulainen, J. Liang, E. Meyerson, A. Rawal, D. Fink, O. Francon, B. Raju, H. Shahrzad, A. Navruzyan, N. Duffy *et al.*, Evolving deep neural networks, *arXiv preprint arXiv:1703.00548* (2017).
10. T. Salimans, J. Ho, X. Chen, S. Sidor and I. Sutskever, Evolution strategies as a scalable alternative to reinforcement learning, *arXiv preprint arXiv:1703.03864* (2017).
11. B. Wang, Y. Sun, B. Xue and M. Zhang, A hybrid differential evolution approach to designing deep convolutional neural networks for image classification (2018).
12. A. P. Engelbrecht, *Fundamentals of computational swarm intelligence* (John Wiley & Sons, 2006).
13. A. E. Hassanien and E. Emary, *Swarm intelligence: principles, advances, and applications* (CRC Press, 2016).
14. X.-S. Yang and M. Karamanoglu, Swarm intelligence and bio-inspired computation: an overview, in *Swarm Intelligence and Bio-Inspired Computation* (Elsevier, 2013) pp. 3–23.
15. X. S. Yang, A new metaheuristic Bat-inspired Algorithm, *Studies in Computational Intelligence* **284** (2010) 65–74.
16. X.-S. Yang, Firefly algorithm, stochastic test functions and design optimisation, *arXiv preprint arXiv:1003.1409* (2010).
17. D. W. Boeringer and D. H. Werner, Particle swarm optimization versus genetic algorithms for phased array synthesis, *IEEE Transactions on antennas and propagation* **52**(3) (2004) 771–779.
18. A. P. Piotrowski, M. J. Napiorkowski, J. J. Napiorkowski and P. M. Rowinski, Swarm

intelligence and evolutionary algorithms: performance versus speed, *Information Sciences* **384** (2017) 34–85.
19. G. Vrbančič, I. Fister, Jr. and V. Podgorelec, Swarm intelligence approaches for parameter setting of deep learning neural network: Case study on phishing websites classification, in *Proceedings of the 8th International Conference on Web Intelligence, Mining and Semantics WIMS '18*, (ACM, New York, NY, USA, 2018), pp. 9:1–9:8.
20. G. Vrbančič, I. Fister, Jr. and V. Podgorelec, Designing deep neural network topologies with population-based metaheuristics, in *Proceedings of the Central European Conference on Information and Intelligent Systems, Varaždin, Croatia, 19. - 21. september*2018, pp. 163–170.
21. D. Lacey, P. Salmon and P. Glancy, Taking the Bait: A Systems Analysis of Phishing Attacks, *Procedia Manufacturing* **3** (2015) 1109–1116.
22. R. Gowtham and I. Krishnamurthi, A comprehensive and efficacious architecture for detecting phishing webpages, *Computers and Security* **40** (feb 2014) 23–37.
23. I. Fister, D. Fister and X. S. Yang, A hybrid bat algorithm, *Elektrotehniški Vestnik/Electrotechnical Review* **80**(1-2) (2013) 1–7.
24. I. Fister, I. Fister Jr, X.-S. Yang and J. Brest, A comprehensive review of firefly algorithms, *Swarm and Evolutionary Computation* **13** (2013) 34–46.
25. I. Fister, P. N. Suganthan, I. Fister, S. M. Kamal, F. M. Al-Marzouki, M. Perc and D. Strnad, Artificial neural network regression as a local search heuristic for ensemble strategies in differential evolution, *Nonlinear Dynamics* **84**(2) (2015) 895–914.
26. J. Schmidhuber, Deep Learning in neural networks: An overview, *Neural Networks* **61** (2015) 85–117.
27. W. Liu, Z. Wang, X. Liu, N. Zeng, Y. Liu and F. E. Alsaadi, A survey of deep neural network architectures and their applications, *Neurocomputing* **234** (2017) 11–26.
28. V. Nair and G. E. Hinton, Rectified linear units improve restricted boltzmann machines, in *Proceedings of the 27th international conference on machine learning (ICML-10)*2010, pp. 807–814.
29. X. Glorot, A. Bordes and Y. Bengio, Deep sparse rectifier neural networks, *AISTATS '11: Proceedings of the 14th International Conference on Artificial Intelligence and Statistics* **15** (2011) 315–323.
30. Y. Lecun, Y. Bengio and G. Hinton, Deep learning (may 2015).
31. R. M. Mohammad, F. Thabtah and L. McCluskey, Predicting phishing websites based on self-structuring neural network, *Neural Computing and Applications* **25**(2) (2014) 443–458.
32. Y. Bengio, Practical recommendations for gradient-based training of deep architectures, *CoRR* **abs/1206.5533** (2012).
33. L. Bottou, Large-Scale Machine Learning with Stochastic Gradient Descent, in *Proceedings of COMPSTAT'2010*, eds. Y. Lechevallier and G. Saporta (Physica-Verlag HD, Heidelberg, 2010), pp. 177–186.
34. D. P. Kingma and J. L. Ba, Adam: A Method for Stochastic Optimization, *ICLR* (2015).
35. J. Duchi, E. Hazan and Y. Singer, Adaptive Subgradient Methods for Online Learning and Stochastic Optimization, *Journal of Machine Learning Research* **12** (2011) 2121–2159.
36. T. Tieleman and G. Hinton, 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.
37. N. Abdelhamid, A. Ayesh and F. Thabtah, Phishing detection based Associative Classification data mining (oct 2014).

38. G. Aaron and R. Manning, APWG phishing reports. apwg. 2014.
39. I. Fette, N. Sadeh and A. Tomasic, Learning to detect phishing emails, in *Proceedings of the 16th international conference on World Wide Web* ACM2007, pp. 649–656.
40. D. Miyamoto, H. Hazeyama and Y. Kadobayashi, 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)* **5506 LNCS**, (Springer, Berlin, Heidelberg, nov 2008), pp. 539–546.
41. R. M. Mohammad, F. Thabtah and L. McCluskey, An assessment of features related to phishing websites using an automated technique, in *Internet Technology And Secured Transactions, 2012 International Conference for* IEEE2012, pp. 492–497.
42. G. Vrbančič, Phishing dataset (2019), Available at https://github.com/GregaVrbancic/Phishing-Dataset, Accessed: 2019-05-23.
43. R. B. Basnet, A. H. Sung and Q. Liu, Rule-based phishing attack detection, in *International Conference on Security and Management (SAM 2011), Las Vegas, NV* 2011.
44. OpenDNS, PhishTank data archives Available at https://www.phishtank.com/, Accessed: 2018-01-17.
45. Mat Bright, Phishing scams and spoof emails at MillerSmiles.co.uk (2003).
46. S. Dreiseitl and L. Ohno-Machado, Logistic regression and artificial neural network classification models: a methodology review, *Journal of Biomedical Informatics* **35**(5) (2002) 352 – 359.
47. G. Vrbančič, L. Brezočnik, U. Mlakar, D. Fister and I. Fister Jr., NiaPy: Python microframework for building nature-inspired algorithms, *Journal of Open Source Software* **3** (2018).
48. F. Chollet *et al.*, Keras (2015), Available at https://keras.io, Accessed: 2018-01-23.
49. T. E. Oliphant, *A guide to NumPy* (Trelgol Publishing USA, 2006).
50. 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, Scikit-learn: Machine learning in Python, *Journal of Machine Learning Research* **12** (2011) 2825–2830.