UDC 518
DOI: 10.15587/1729-4061.2023.281986

A NOVEL APPROACH TO THE DEVELOPMENT OF NEURAL NETWORK ARCHITECTURE BASED ON METAHEURISTIC PROTIS APPROACH

Tengku Henny Febriana Harumy
Corresponding author
PhD Student*
E-mail: hennyharumy@usu.ac.id
Muhammad Zarlis
Professor of Computer Science
Department of Information System Management
Binus University
Kebon Jeruk Raya str., 27, Jakarta, Indonesia, 11530
Maya Silvi Lydia
PhD, Dean of Computer Science and Information Technology*
Syahril Efendi
PhD, Associate Professor*
*Department of Computer Science
Universitas Sumatera Utara
Dr. T. Mansur str., 9, Padang bulan, North Sumatera, Indonesia, 20222

Determining the best model for the neural network architecture and how to optimize the architecture with the metaheuristic Protis Approach is a subject of the study. A comprehensive investigation and utilization of metaheuristic methods are necessary. These methods aim to solve problems and adapt from the lifestyle of the amoeba protis. In this study, the proposed method modifies the life cycle of the amoeba, which consists of four phases: prophase, metaphase, anaphase, and telophase. These four phases are modified in the neural network architecture to optimize the appropriate number of hidden layers and produce an efficient architecture model. The results show that the protis approach optimized the neural network architecture, especially in generating hidden layers to improve the neural network model. Distinctive features of the results obtained are that the average range of degenerate neurons in the hidden layer is 0 to 35 neurons in each layer. The standard number of neurons makes it possible to solve the problem of determining the best model on the neural network architecture. The protis algorithm embedded in the protis recurrent neural network for categorical data measurements produces an average RMSE value, representing the difference between actual measurements and predictions, equal to 0.066.

Consequently, the developed model surpasses the current classical neural network model in terms of performance. Regarding accuracy, the protis algorithm embedded in the neural network for categorical and time series data achieves an average precision of 0.952 and a recall of 0.950. The protis convolutional neural network achieves an accuracy of 95.9 %. Therefore, from the three tested datasets, the protis convolutional neural network exhibits the highest accuracy value

Keywords: neural network, artificial intelligence, hidden layer optimization, deep neural network

Received date 10.05.2023
Accepted date 12.07.2023
Published date 31.08.2023

__How to Cite:__ Harumy, T. H. F., Zarlis, M., Lydia, M. S., Efendi, S. (2023). A novel approach to the development of neural network architecture based on metaheuristic protis approach. Eastern-European Journal of Enterprise Technologies, 4 (4 (124)), 46–59. doi: https://doi.org/10.15587/1729-4061.2023.281986

1. Introduction

The fields of metaheuristics and neural networks are experiencing rapid advancements within the realm of artificial intelligence. Within this framework, metaheuristics and swarm intelligence algorithms play integral roles as modern global optimization algorithms, contributing to computer intelligence and software development. These methods are capable of producing satisfactory solutions within an acceptable timeframe, but they do not guarantee the attainment of the absolute best solution [1, 2]. Metaheuristics employ a combination of rules and randomness to explore and find global optima through trial-and-error methods iteratively. These approaches possess fundamental characteristics that guide the search process effectively and have been successfully adapted to the domain of biocomputing for the purpose of finding optimal solutions [3].

The neural network architecture method faces certain weaknesses, mainly related to the challenge of determining the optimal combination of hidden layers that can produce effective and efficient results in the prediction and classification processes [4]. Previous relevant studies have provided some achievements in determining the best neural network architecture using a metaheuristic approach, but there are still deficiencies in determining the optimal number of neurons in the hidden layer of neural network architecture. In addition, previous research focused on an algorithmic approach [5, 6]. This research has highlighted the importance of striking the right balance in choosing the right number of neurons in the hidden layer and the dataset used. On the hidden layer problem, if too few layers are selected, the model may have difficulty capturing complex patterns in the data. On the other hand, an excessive number of layers can result in overfitting, in which the neural network model starts to catch irrelevant and random patterns from the training data, mistakenly considering them as fundamental patterns.

Previous studies have identified that one of the reasons for the slow training process of neural networks is the inappropriate number of hidden layers [7]. However, currently, the average standard number of neurons that are set to determine the optimal number of hidden layers in a model still varies [4, 8–10].

Consequently, the number of layers and neurons in the hidden layer is considered a hyperparameter and must be optimized using various approaches. In addition, the uncertain and statistically based nature of approaches, techniques, methods, and algorithms in neural networks poses another challenge. Also, determining the values of various model parameters and validating them using limited test data adds to the complexity of the problem. Nevertheless, the most crucial issue to address is determining the number of hidden layers and neurons, significantly affecting the prediction and classification results in artificial neural networks [11, 12].

To overcome this challenge, it is necessary to carry out in-depth analysis and develop new algorithms inspired by swarm intelligence metaheuristics to optimize the regeneration of the hidden layer and find the standard number of neurons in the hidden layer. Furthermore, the optimization of the dataset is used in this approach [13]. The protis approach, part of the metaheuristic intelligence and swarm intelligence algorithms, is currently being developed to optimize the architecture of artificial neural networks [14]. Previous literature on the development of the theory of protists in artificial neural networks [14, 15] highlights the potential of the protist amoeba adaptive lifestyle to enhance performance. Further exploration and exploitation of this approach are required to adapt it specifically to optimize the number of hidden layers in a neural network model [16].

Research on protist by [17, 18] revealed that protist has several phases, namely prophase, metaphase, and anaphase, which contribute to their strength and resilience. The in-depth approach involves studying and evaluating various techniques to determine their suitability for solving hidden layer optimization problems and choosing the most effective method. Standard tests should be carried out to assess efficiency and effectiveness, as well as to compare different algorithms and understand the strengths and weaknesses of each.

The next step involves fine-tuning the algorithms parameters to optimize performance, which may require adjusting settings such as population size and mutation rate. The goal is to identify the best combination of neural network architectures that produce a high-quality solution within a reasonable timeframe. In addition, problem representation design plays an important role in exploiting protist metaheuristics effectively. Well-designed representations can improve algorithm performance by facilitating an exploration of the problem space [10, 19–25]. Protis neural networks, as a metaheuristic method approach, are needed to optimize the neural network architecture to produce accurate models with reduced errors and increased performance. By generating optimal hidden layers, this approach aims to minimize lengthy training processes for categorical data, time series, and complex images, resulting in more effective and efficient prediction and classification models while minimizing errors. Therefore, research on the development of neural network architecture optimization is relevant to the research conducted by the authors [21–25].

2. Literature review and problem statement

In the paper [19], a novel method for determining the optimum number of hidden layers for FNNs was introduced and applied in financial data mining. It is shown that the removable nodes are identified by the delta values of hidden layers. It was observed that the sum of the delta values of the hidden layer shows a considerable correlation with the output error. Moreover, this method, based on mathematical arguments, suggests minor adjustments to the settings used in neural networks. Further, the modified architecture was obtained with very limited computations. At a time, 5–20 % can be removed from hidden layers without degrading the output performance. This approach in [26] focused on finding the optimal training parameters and number of hidden layer neurons in a two-layer perceptron for generalized object classification. However, this research allows us to argue that it is appropriate to conduct a study devoted to determining the standard neurons in the hidden layer that are appropriate for optimizing the neural network architecture.

The paper [27] assembles a precise prescient model using the Enhanced Deep Feed Forward Neural Network Model to predict the customer whittling down in the Banking Domain. The generalization abilities of feedforward neural networks with one and two hidden layers are compared. However, this study only discusses the comparison of classic machine learning. The resolved questions related to the postulation that identifying hidden layer neurons, which can minimize the error of the training cycle, would be significant for modeling hidden layer architecture, effectively proposing a method that allowed empirical comparisons between individual nodes in the hidden layers of these networks. The reason could be related to pruning the number of neurons, which makes related research impractical. Ten publicly available datasets for function approximation were used to evaluate the method. The results indicated that in nine out of ten instances, the network with two hidden layers performed better, although the extent of improvement varied across cases.

The study [28] introduced a technique called «transformative optimization» and used it to compare hidden nodes in single and two-layer feedforward networks. The findings consistently favored the two-hidden-layer feedforward networks (TLFNs) over single-hidden-layer ones (SLFNs) in most cases. The proposed method, in combination with binary sampling, offers a rapid means to determine the feasibility of employing two hidden layers in a specific problem. Although the presented data demonstrate the superiority of TLFNs over SLFNs, further research utilizing more challenging real-world datasets is required. Moreover, this technique has the potential to be applied in a variety of metaheuristic training. However, the amount of increase is highly case-dependent. All this allows us to argue that it is appropriate to study the proper number of hidden layer neurons to optimize the neural network architecture.

The paper [4] highlights that metaheuristics belong to a class of optimization techniques employed to discover approximate solutions for problems that are intricate or impossible to solve precisely. However, the resolved questions related to determining a flexible structure for a metal price forecasting model using an evolutionary-based ANN model are a costly part of the plan. A neural network (NN) model for metal price forecasting based on an evolutionary approach is proposed; however, this method is used only for time series metal price data, which makes the corresponding research inexpedient. Options for overcoming relevant difficulties can be researched by [20], namely a metaheuristic-based learning algorithm to build an ensemble system so that the training time becomes shorter. Furthermore, the training time is shorter. In our proposed method, a master-slave based metaheuristic algorithm is used in the optimization process to generate a heterogeneous group of feedforward neural networks. However, from the results of previous studies [23, 24], in which the paper states that random weights have the advantage of fast computation time in both training and testing, all this allows us to argue that it is appropriate to conduct a study devoted to neural architecture optimization network that can be used for various data and can minimize training time.

Further research in [10] highlights the numerous advantages of metaheuristics over traditional optimization methods, which render them valuable in tackling a wide range of problems. Computationally performed tests confirm this by showing that the approach proposed in this paper achieves competitive results to other algorithms developed lately, like the WWO algorithm, while also going one step further by finding the optimal count of hidden layers without user intervention, as to find the optimal architecture of the ANN for the given dataset. One significant advantage is their capability to handle complex problems that are challenging or impossible to solve precisely. However, for research questions related to structural design involving both the number of hidden layers and the number of neurons needed in each of these hidden layers, this approach requires more in-depth observations. The reason why this method is feasible is that progress has been made in adapting this methodology to more sophisticated algorithms and evaluating its performance. In addition, efforts are being made to minimize the breadth of the overall search space. This choice of difficulty is also relevant to the research conducted by [8]. However, further in-depth observations are still needed, and all of this allows us to conduct studies aimed at deepening neural network architectural models with metaheuristic and protis approaches.

The paper [3] shows that metaheuristics in dealing with non-linear constraints, indistinguishable functions, and noisy data provide a critical of existing wind power and solar power ML forecasters, namely artificial neural networks (ANNs), recurrent neural networks (RNNs), support vector machines (SVMs), and extreme learning machines (ELMs). Nevertheless, the resolved questions are only related to some critical Classical Neural Network Architecture, which makes related research be developed more deeply. Moreover, supporting evidence from [29] highlights that the ANN with 30 neurons performs best, but it cannot be visualized using a framework in research. It is emphasized that metaheuristics effectively handle uncertainty by incorporating randomization into the search process. Additionally, metaheuristics are well-suited for addressing large-scale problems characterized by many variables or extensive solution spaces. All this allows for future studies to focus on the development of Protis recurrent neural networks (RNNs).

In addition, metaheuristics can handle noisy or incomplete data, which are commonly encountered in real-world problems. They can also effectively deal with non-numeric categorical variables, making them applicable in scenarios where the variables are not continuous [30]. Developing a metaheuristic method typically involves two steps: exploration and exploitation. As mentioned in [9], exploring metaheuristic methods entails studying and evaluating various techniques to assess their suitability for solving a specific problem. Another approach to exploring metaheuristic methods involves experimenting with different variations of a particular algorithm.

Exploiting metaheuristics entails employing specific methods to obtain high-quality solutions for particular problems. This process often involves refining the algorithm's parameters and settings to optimize its performance in addressing the specific problem at hand. Additionally, it is crucial to consider the problem's unique characteristics and how they influence the algorithm's behavior. For instance, if the problem involves numerous constraints, incorporating constraint-handling mechanisms like penalty functions may be necessary. Furthermore, exploiting metaheuristics involves analyzing the obtained results and interpreting the solutions generated by the algorithm. This analysis helps identify patterns, trends, and insights that can be utilized to enhance the algorithm or gain a better understanding of the problem. In summary, exploiting metaheuristics is an iterative process that includes adjusting the algorithm's parameters and settings, designing an appropriate problem representation, and interpreting the obtained results [30, 31].

Previous studies [32] have focused on developing protis in artificial neural networks. These studies draw inspiration from how amoebas divide themselves to survive and become stronger, aiming to improve the performance and accuracy of neural network models. Building upon this concept, the current research aims to adapt it to the existing neural network model by introducing modifications that incorporate the phases of prophase, metaphase, and anaphase. These adjustments are intended to enhance the performance and accuracy of the neural network method. Moreover, by optimizing the principles of protis theory, this research contributes to the development of deep neural networks, where the concepts of division for survival, self-development, and hidden layers are inherited from one generation to another. The adjusted phases, namely the beginning, middle, and end, have shown promising results in improving the classification process. Differing from the classic neural network method, the protis neural network method introduces a distinct process for generating hidden layers in the neural network architecture [28, 33, 34]. The proposed method, in combination with binary sampling, offers a rapid means to determine the feasibility of employing two hidden layers in a specific problem. Although the presented data demonstrate the superiority of TLFNs over SLFNs, further research utilizing more challenging real-world datasets is required. Moreover, this technique holds potential for application in various training metaheuristics.

The paper in [16] explains that metaheuristics belong to a class of optimization techniques employed to discover approximate solutions for problems that are intricate or impossible to solve precisely. The research problem solved in this paper is about the regularized training problem in the RELU network, which makes this research continue to be developed. One significant advantage is their capability to handle complex problems that are challenging or impossible to solve precisely. Metaheuristics excel in dealing with non-linear constraints, indistinguishable functions, and noisy data. Moreover, supporting evidence from [30] emphasizes that metaheuristics effectively handle uncertainty by incorporating randomization into the search process. Additionally, metaheuristics are well-suited for addressing large-scale problems characterized by many variables or extensive solution spaces so that it is then appropriate to develop studies on data training development on neural network architectures.

In addition, metaheuristics can handle noisy or incomplete data, which are commonly encountered in real-world problems. They can also effectively deal with non-numeric categorical variables, making them applicable in scenarios where the variables are not continuous [35]. Developing a metaheuristic method typically involves two steps: exploration and exploitation. As mentioned in [36], exploring metaheuristic methods entails studying and evaluating various techniques to assess their suitability for solving a specific problem. Another approach to exploring metaheuristic methods involves experimenting with different variations of a particular algorithm.In [37], exploiting metaheuristics entails employing specific methods to obtain high-quality solutions for particular problems. This process often involves refining the algorithm's parameters and settings to optimize its performance in addressing the specific problem at hand. Additionally, it is crucial to take into account the problem's unique characteristics and how they influence the behavior of the algorithm. For instance, if the problem involves numerous constraints, incorporating constraint-handling mechanisms like penalty functions may be necessary. Furthermore, in the other paper, the approach used [38], exploiting metaheuristics involves analyzing the obtained results and interpreting the solutions generated by the algorithm. This analysis helps identify patterns, trends, and insights that can be utilized to enhance the algorithm or gain a better understanding of the problem. In summary, exploiting metaheuristics is an iterative process that includes adjusting the algorithm's parameters and settings, designing an appropriate problem representation, and interpreting the obtained results. So, it is possible to carry out further studies aimed at the development of metaheuristics for the development of neural network architectures.

Several previous studies [14] have focused on the development of protis in artificial neural networks. These studies draw inspiration from how amoebas divide themselves to survive and become stronger, aiming to improve the performance and accuracy of neural network models. Building upon this concept, the current research aims to adapt it to the existing neural network model by introducing modifications that incorporate the phases of prophase, metaphase, and anaphase. These adjustments are intended to enhance the performance and accuracy of the neural network method. This is an approach that is also used in [39].

Moreover, by optimizing the principles of protist theory, this research contributes to the development of deep neural networks, where the concepts of division for survival, self-development, and hidden layers are inherited from one generation to another. The adjusted phases, namely the beginning, middle, and end, have shown promising results in improving the classification process. Differing from the classic neural network method, the Protis neural network method introduces a distinct process for generating hidden layers in the neural network architecture.

3. The aim and objectives of the study

The aim of the study is to optimize the hidden layers in a neural network by combining a metaheuristic approach and protis theory. This optimization will improve the performance and accuracy of the neural network model.

To achieve this aim, the following objectives are accomplished:

– to deepen and understand the concept of Protis Theory and its implementation on neural network architecture;

– to adapt the concept of accurate pattern recognition and for searching neurons in the best hidden layer in neural networks adapted from Protis Theory;

– to take advantage of the robustness potential gained from the life processes of Protis Theory in enhancing the development of hidden layers for neural networks;

– to evaluate the Protis Neural Network Approach's performance to find the optimal number of neurons in hidden layer neural network architecture;

– to perform visualization in neural network modeling by providing insight into adapting protis life processes to better develop hidden layers.

4. Materials and methods

The object of research is a model for the neural network architecture and how to optimize the architecture.

The subject of research is to determine the best model for the neural network architecture and how to optimize the architecture with the metaheuristic Protis Approach.

The main hypothesis of the study is that the protis approach can optimize the neural network architecture, especially in generating hidden layers to improve the neural network model.

The Assumptions made in the work are as follows. The Hierarchical Representation Assumption assumes that the neural network architecture can describe a feature hierarchy that becomes increasingly complex as information passes through layers in the network. Each layer takes on more abstract and complex features than the previous input. Next is the Model Capacity Assumption, which assumes that more complex or significant architectures have a higher capacity to learn complex patterns in data. This assumption also considers the risk of overfitting when the model learns too much from the training data and cannot generalize well to unseen data. Next is the Assumption of Using the Activation Function, where this assumption assumes that using nonlinear activation functions at each layer is an essential requirement. Activation functions such as ReLU, sigmoid, or tanh introduce nonlinearity into the architecture and enable models to study complex relationships in data.

The Simplifications adopted in the work consist in optimizing and visualizing the neural network architecture by adopting the Metaheuristic Protis approach, namely taking inspiration from the concept of the protist way of life, which consists of several phases, namely metaphase, prophase, anaphase and telophase where efforts are made to optimize and generate a search for the best-hidden layer, determine the correct number of neurons in the hidden layer, and increase efficiency and effectiveness in neural network models.

The research flow follows the stages described below.

Deepening the Protis Neural Network approach by exploring and exploiting the way of life of an amoeba, which will later be embedded in the neural network. Furthermore, this research will be carried out according to the following research flow. The stages of Protis Neural Network research are as follows:

– Observation and preparation of categorical, time series, and image datasets.

This stage involves an in-depth analysis of existing neural networks, Metaheuristics, and swarm Intelligence methods in the context of dataset observation. Data preprocessing techniques such as cleansing, loss removal, and normalization are applied. The activation function, namely the sigmoid function, ensures the data is scaled from 0 to 1. It is crucial to consider the appropriate activation function for data normalization, such as Identity, binary step, sigmoid, Rectified Linear Unit (ReLU), Leaky ReLU, and softmax.

– Deepening protis approach to the neural network.

This stage delves into the protis Neural Network approach by exploring and leveraging the amoeba's way of life, which involves the following phases: prophase, metaphase, anaphase, and telophase. These phases mimic the process of amoeba reproduction through self-division. Additionally, modeling and mathematical analysis of the protist neural network algorithm flow are conducted, with attention to parameters such as epochs, activation function, architecture, and other parameters to enhance the training process.

– Design and estimation of the protis method in a neural network.

This stage continues the exploration and exploitation of the protis neural network approach by integrating the amoeba's way of life into the neural network. Starting from the input dataset, the process involves the prophase phase, followed by the metaphase phase, anaphase phase, and telophase phase, representing the reproduction process through self-division. Mathematical modeling and analysis of the protist neural network algorithm are performed, considering parameters such as epochs, activation function, architecture, and other relevant factors to improve the training process.

– Generating hidden layer/parameter on a neural network with protis approach.

The next step involves determining the tools, specifically Python and Matlab, for training and testing the protis model using a single computer and parallel computing (GPU). This allows for comparing running time, performance, and error in the model training architecture. The protis method is trained and tested with different hyperparameters and epochs, considering variations in the number of iterations and data division ratios, such as 70/30 %, 60/40 %, and 50/50 %. The dataset is divided into 70 % training and 30 % test data, with epochs set at 10, 100, 1000, 80, 120, and 200.

– Comprehensive evaluation models.

The protis neural network method is comprehensively evaluated by analyzing its advantages, strengths, and weaknesses. A comparison is made with existing classical neural network methods, such as feed forward neural networks (FFNN), Convolutional Neural Networks, and recurrent neural networks, particularly for time series, categorical, and image data. This comparison highlights the performance, efficiency, and level of each method. The results of the protis neural network method are visualized and displayed on a landing page website, facilitating user implementation and enabling further discussion and analysis.

5. Results of Neural Network Architecture Optimization Based on Protis Metaheuristic

5. 1. Deepening and understanding the concept of Protis Theory and its implementation on neural network architecture

The protis theory is currently being developed for various research purposes, including proposing an algorithm for pattern recognition. This algorithm aims to accurately track epithelial and endothelial cells in time-lapse image sequences with low contrast levels that gradually increase over time. Taking inspiration from the concept of the protis way of life, an effort will be made to adapt it to the search for the best, most optimal, efficient, and effective hidden layer in a neural network model. The inspiration for the theory is illustrated in Fig. 1.

The idea is that similar to how the phases in protists make their cells stronger, incorporating these phases into neural networks can potentially enhance their performance. To address the problem of determining the appropriate number of hidden layers for achieving a well-structured neural network architecture, a deeper exploration and exploitation of the protis way of life are considered necessary. This exploration involves studying the four phases: prophase, metaphase, and anaphase/telophase, and attempting to integrate them into developing hidden layers in neural networks [14].

5. 2. Adapting the concept of accurate pattern recognition and searching neurons in the best hidden layer in neural networks adapted from Protis Theory

The protis theory has been further developed and applied in the context of neural network architecture. The phases observed in protis have been incorporated into designing the neural network architecture. This application is depicted in Fig. 2.

To further advance the protis neural network approach, this research aims to explore and exploit the amoeba's way of life, which will be integrated into the neural network framework.

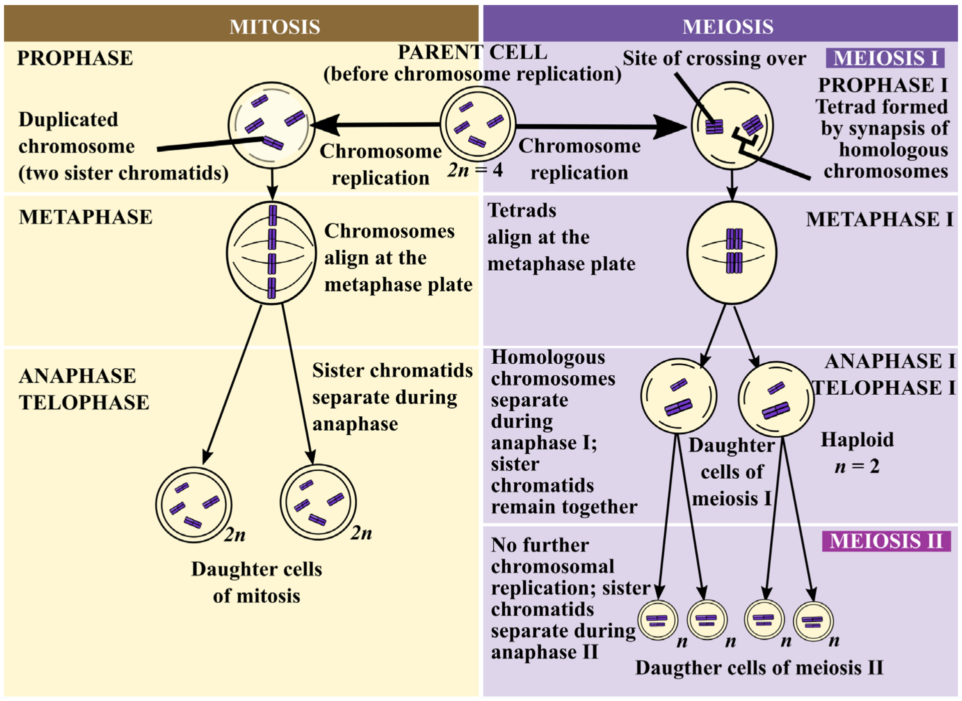

MITOSIS
MEIOSIS
PROPHASE
PARENT CELL
(before chromosome replication)
Site of crossing over
MEIOSIS I
PROPHASE I
Tetrad formed by synapsis of homologous chromosomes
Duplicated chromosome (two sister chromatids)
Chromosome replication
2n = 4
Chromosome replication
METAPHASE
Chromosomes align at the metaphase plate
Tetrads align at the metaphase plate
METAPHASE I
ANAPHASE TELOPHASE
Sister chromatids separate during anaphase
Homologous chromosomes separate during anaphase I; sister chromatids remain together
ANAPHASE I TELOPHASE I
Daughter cells of meiosis I
Haploid n = 2
2n
2n
Daughter cells of mitosis
MEIOSIS II
No further chromosomal replication; sister chromatids separate during anaphase II
n
n
n
n
Daughter cells of meiosis II

Fig. 1. Protis Neural Network Theory

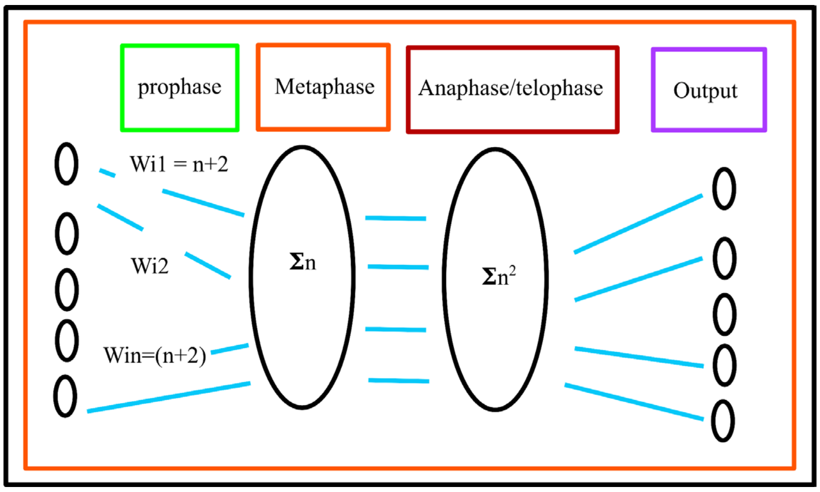


Fig. 2. Protis Neural Network Architecture

5. 3. Advantage of the robustness potential gained from the life processes of Protis Theory in enhancing the development of hidden layers for neural networks

To prepare the input data and target variables, they are replaced with values obtained from the collected data. The input data is then transformed into a matrix format to facilitate training. Categorical data is divided into several matrices: matrix A (6×60), matrix B (6×70), and matrix C (6×50), while the remaining data is used for testing, with matrix D (6×40), matrix E (6×30), and matrix F (6×50). Data normalization is performed for time series data, categorical data, CIFAR image data, and field observations.

In developing the life processes of Protist Theory in optimizing hidden layer neurons for neural network architecture, several processes were carried out, including:

1. Protis neural network architecture.

The sigmoid activation function is used for data normalization, mapping the original data to the range of 0 to 1. This is achieved by applying the binary sigmoid function to normalize the data from 0 to 1. By doing so, the input data values are adjusted to fit the sigmoid activation function. The initial input data ranges from 1 to 6, which is assumed based on the attribute weighting of the categorical data architecture (Fig. 3).

The Protis Feed Forward Neural Network algorithm network is employed in this case as the Artificial Protis Neural Network architecture, and it comprises the following: 6 nodes in the input layer ($x1, x2, x3, x4, x5, x6$). The hidden layer has a maximum of 4 nodes formed by the protist theory's phases, namely the Prophase Phase, Metaphase Phase, Anaphase Phase, and Telophase Phase, and up to 1,000 neurons can be selected in each node ($z1$, $z2$, $z3$, $z4$). The accuracy of categorizing disease spread in coastal areas categorical data (Y) is the output layer with one node.

2. Proposing a model design and estimation of the protis neural network.

The design and estimation of the protis neural network model follow the core structure of the algorithm. The steps involved are as follows:

Step 0: set all weights with small random integers.

Step 1: check if the termination condition is met. If not, proceed to Steps 2–8.

Step 2: repeat Steps 3–8 for each pair of training data.

Step 3: execute phase 1, which consists of Steps 3–5.

Each input unit receives a signal and sends it to the concealed unit above it.

Step 4: calculate the output values for all hidden units.

The core structure of the algorithm is the Neural Network primary approach in the architectural design of a Neural Network protist model for six inputs:

$$Zj(j=1,2,...,p)Z\_netj =$$

$$= \left(Vjo + \sum_{i=1}^{n} X_1 V_{ji}\right)^2, \qquad (1)$$

$$Zj = f(Z\_netj) = \frac{1}{1+\exp(-z\_netj)}. \qquad (2)$$

Step 5: modify the protis neural network within the neural network architecture.

Modification method prophase phase: increase variable weight and duplicate chromosomes:

$$(Wj1, Wj2, Wj3, Wjn)2Xn = (Random(1,0) = P < Prob). \qquad (3)$$

Modification method metaphase phase: optimization using the binary sigmoid activation function:

$$Yk = f(y\_net_k)\left(Y = f(x)\frac{1+\exp(-x)}{1+\exp(-y_{netk})}\right)^2, \qquad (4)$$

$$\left(y = f(x) = \frac{1}{1-\exp(-\delta x)}\right)$$

or

$$\left(y = f(x) = \frac{1+\exp(-x)}{1-\exp(-\delta x)}\right)^2. \qquad (5)$$

For each input chromosome, (Xi, i=1, 2, 3, ..., n), the input Xi is received and propagated to all chromosomes in the top layer (hidden layer). For the hidden chromosome (Zi, j=1, 2, 3, ..., p), the input values are calculated using the weight values:

$$z\_in_j = voj + \sum_{i=1}^{n} x_i v_{ij}. \qquad (6)$$

The output value is then determined using the activation function, which is a binary sigmoid function:

$$zj = f(z_{inj}). \qquad (7)$$

Modification method metaphase phase: binary sigmoid optimization.

The binary sigmoid function has two possible forms:

$$\left(y = f(x) = \frac{1}{1-\exp(-\delta x)}\right), \left(y = f(x) = \frac{1+\exp(-x)}{1-\exp(-\delta x)}\right)^2. \qquad (8)$$

The output value of each output chromosome (Yk, k=1, 2, 3, m) is then calculated using the input value, which is determined by the weighted sum of the hidden chromosome outputs:

$$y_{in}k = wok + \sum_{i=1}^{p} Z_i w_{jk}. \qquad (9)$$

Then the output value is calculated using the activation function:

$$y_k = f(y_{in}k). \qquad (10)$$

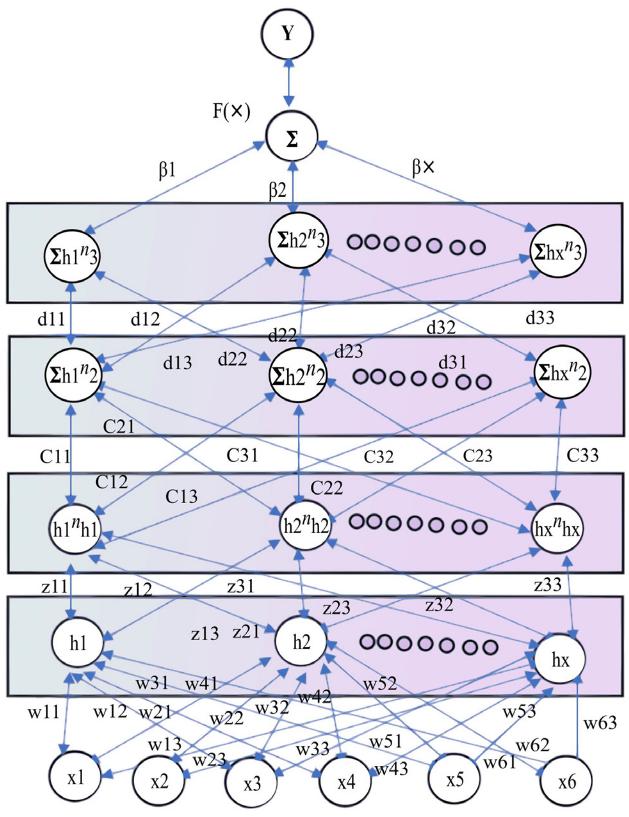


Fig. 3. Protis neural network architecture: x – input; w, z, c, d – the weights on the hidden layer; β – the weights on the output layer; Vn – weights on the output layer; Wb – bias in the hidden layer and output layer; Y – result output; $J=1$ to 6; $Yd=0$

Step 6: anaphase optimization using function:

$$\left(y=f(x)=\frac{1}{1+\exp(-\delta x)}\right)^2,$$

$$\left(Xn=Random(0,1)=P,Prob\right). \qquad (11)$$

Step 7: telophase optimization using function:

$$\left(Y=f(x)\frac{1+\exp(-x)}{1+\exp(-y_{netk})}\right)^2,$$

$$\left(Xn=Random(0,1)=P,Prob\right). \qquad (12)$$

Receive the target pattern that matches the input pattern for each chromosome output (Yk, $k=1, 2, 3, ..., m$) and calculate the error:

$$\Delta_k=\left((t_k-y_k)f'(y_{ink})\right)^2. \qquad (13)$$

The weight value modification is then computed and applied to the value w_{jk}:

$$\Delta w_{jk}=\left(\alpha\delta_k z_j\right)^2. \qquad (14)$$

Determine the bias value correction, which will be used to update the value w_{0k}:

$$\Delta w_{ok}=\alpha\delta_k. \qquad (15)$$

Then the value of δk is sent to the chromosome in the previous layer. For each hidden chromosome (Xn=(Random (0, 1)=$P<Prob$), (Zj, $j=1, 2, 3, ..., p$) calculated as the input delta of the chromosomes in the upper layer:

$$\delta\_inj=\sum_{k=1}^{m}\left(\delta_k w_{jk}\right)^2. \qquad (16)$$

The error information is then calculated by multiplying it by the activation function's derivative value:

$$\delta_j=\delta_{inj}f'\left(Z_{inj}\right)^2. \qquad (17)$$

Calculate the weight value correction, which is subsequently utilized to update v_{ij}:

$$\Delta v_{ij}=\left(\alpha\delta_j x_j\right)^2. \qquad (18)$$

Additionally, compute the bias adjustment value, which will be used to update the value ν_{oj}:

$$\Delta\nu_{oj}=(\alpha\delta_j)^2. \qquad (19)$$

Weight and bias values should be updated. For every bias and weighting value on the output chromosome (Yk, $k=1, 2, 3, ..., m$), update:

$$w_{jk}(new)=w_{jk}(before)+\Delta_{ij}, \qquad (20)$$

$$v_{i_j}(new)=w_{jk}(previously)+\Delta_{ij}, \qquad (21)$$

$$v_{ij}(new)=v_{ij}(previously)+\Delta_{ij}. \qquad (22)$$

Step 8: test whether the stop condition is met.

This stop condition is met if the resulting error value is less than the error value [14]. The following pseudocode is derived from the following algorithm design.

Design pseudocode:

```
PFFNN algorithm
Input:
Vector Input – $i$
Output:
Vector Output
Initialize the number of input and output neurons
Initialize generate the hidden layer
Initialize all weights with random values between 0 and 1
For each Max up 6 Network
Initialize max up to 1,000 neurons
Initialize Epoch – $n$
Initialize MaxError – $m$
While (Epoch≤$\|n\|$ Error value≥MaxError)
For each layer in the network
For each neuron in layer ($Z$)
Calculate the sum of weight $Vij$ and bias $V0j$ of each
input $Xi$ that goes to the hidden neuron
Apply the activation function to each neuron
(Metaphase Activation)
Apply the activation function to each neuron (Prophase
Activation)
For each anaphase/telophase: Optimization using
function
$Zj=f(Zinj)^2$
End
```

```
// forward propagation of each neuron in the layer
For each neuron, do the output layer
Calculation of the output error value ($yk$) against the
target ($tk$) $\delta\_j=((tk-Yk)\ Yk\ (1-Yk))^2)$
End.
```

The dataset used in this study consists of categorical and time series data that have been normalized using a sigmoid function, resulting in values ranging from 0 to 1. The protis neural network method is applied in this study to analyze categorical datasets and time series data. The data above is presented in Table 1.

Table 1

Sampling Dataset

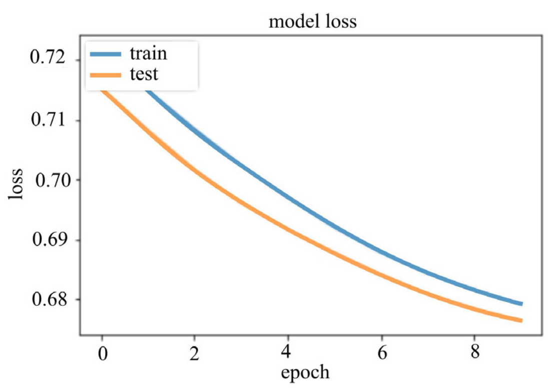
No.	($X1$)	($X2$)	($X3$)	($X4$)	($X5$)	($X6$)	(Y)
1	0.0000	0.3010	0.0000	0.3010	0.4771	0.0000	0.0000
2	0.3010	0.6021	0.3010	0.0000	0.3010	0.0000	0.3010
3	0.4771	0.0000	0.4771	0.3010	0.0000	0.3010	0.0000
4	0.0000	0.3010	0.6021	0.3010	0.3010	0.0000	0.4771
5	0.3010	0.4771	0.0000	0.3010	0.0000	0.3010	0.3010
6	0.4771	0.3010	0.3010	0.3010	0.4771	0.3010	0.0000
7	0.3010	0.6021	0.4771	0.0000	0.3010	0.0000	0.0000
8	0.4771	0.0000	0.6021	0.3010	0.4771	0.3010	0.4771
9	0.0000	0.3010	0.4771	0.3010	0.3010	0.3010	0.3010
10	0.3010	0.4771	0.6021	0.3010	0.0000	0.0000	0.3010

The categorical data consists of disease distribution data with 6 variables and 1 output variable with 3 labels. On the other hand, the time series data represents the number of dengue disease patients over 20 years. During the data trial, a comparison was conducted between the proposed protis neural network method and several other methods to assess their performance. The results obtained using the protis neural network methodology showed an area under the curve (AUC) of 0.997, classification accuracy value of 0.950, F1 value of 0.950, Precision value of 0.952, and Recall value of 0.950. These results indicate that the protis neural network approach is highly effective according to the data. Based on the findings of this study, the protis neural network method demonstrates superiority over traditional machine learning methods when dealing with categorical data situations.

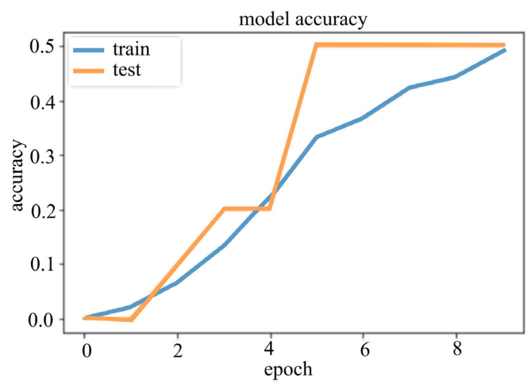
3. PFFNN (Implementation of Protis Feed Forward Neural Network method) for categorical data.

Based on the results of the tests carried out, it was obtained that the results of a comparison of the accuracy level of the Feed Forward Neural Network with epoch 10 were 0.4900, and the Protis Feed Forward Neural Network was 0.5900 where the results of PFNN are higher than FFNN. The results of comparing the model loss with epoch 10 are illustrated in Fig. 4, *a*, and the results of comparing the model accuracy are illustrated in Fig. 4, *b*.

Based on the results of the tests carried out, the result of comparing the accuracy level of the Feed Forward Neural Network with epoch 100 is 0.4490, and the Protis Feed Forward Neural Network is 0.6690, where the results of PFFNN are higher than FFNN. The results of comparing the model loss of the Protis Feed Forward Neural Network with epoch 100 are illustrated in Fig. 5, *a*, and the results of comparing the model accuracy of the Protis Feed Forward Neural Network are illustrated in Fig. 5, *b*.

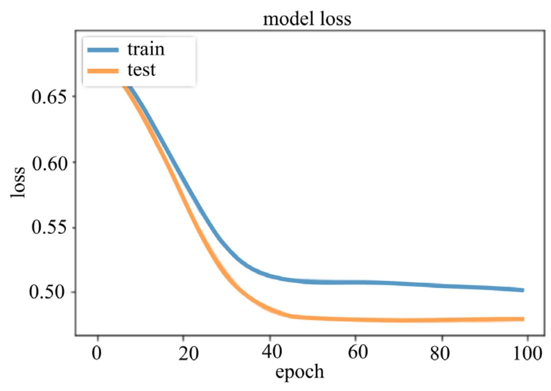


a

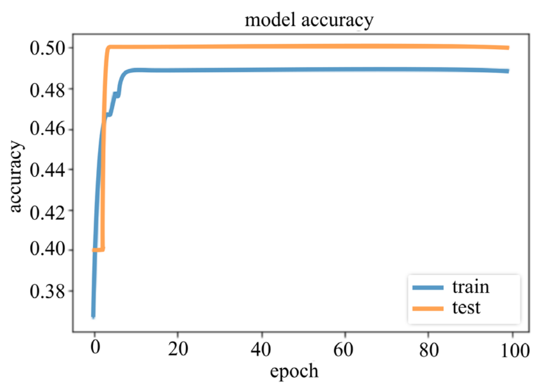


b

Fig. 4. Testing of the Protis neural network model on Feed Forward Neural Network Epoch 10: *a* – comparing model loss; *b* – comparing model accuracy



a

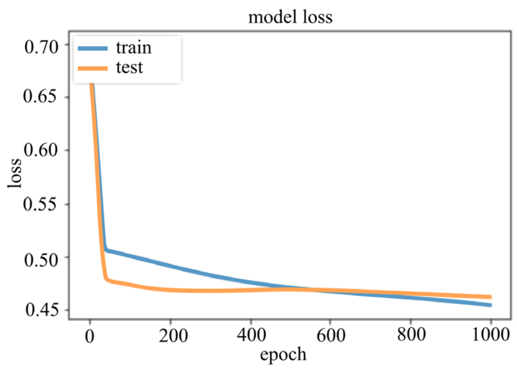


b

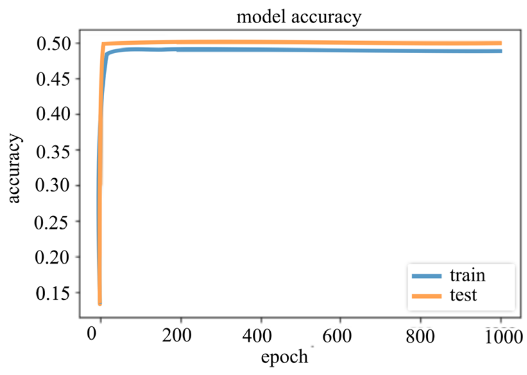
Fig. 5. Testing of the Protis neural network model on Feed Forward Neural Network epoch 100: *a* – comparing model loss; *b* – comparing model accuracy

Furthermore, the tests found that the result of comparing the accuracy level of the Feed Forward Neural Network with epoch 1000 for categorical data was 0.4900, and the Protis Feed Forward Neural Network was 0.4800, where the results of FFNN are higher than PFFNN. The results of comparing the model loss of the Protis Feed Forward Neural Network with epoch 1000 are illustrated in Fig. 6, *a*, and the results of comparing the model accuracy of the Protis Feed Forward Neural Network are illustrated in Fig. 6, *b*.

The accuracy results were evaluated during the training and testing trials of the FFNN and PFFNN methods for categorical data. It was observed that the best accuracy result was achieved for the testing data measurements when using an epoch of 100, which amounted to 0.6690 or 66.90 %. The above data is presented in Table 2.



a



b

Fig. 6. Testing of the Protis neural network model on Feed Forward Neural Network epoch 1000: *a* – comparing model loss; *b* – comparing model accuracy

Table 2

Comparison of FFNN Methods And PFFNN Categorical Data

Method	Feed Forward Neural Network	Protis Feed Forward Neural Network
Accuracy Training Epoch 10	0.6774	0.5324
Accuracy Testing Epoch 10	0.4900	0.5900
Accuracy Training Epoch 100	0.4993	0.5993
Accuracy Testing Epoch 100	0.4490	0.6690
Accuracy Training Epoch 1000	0.4540	0.4540
Accuracy Testing Epoch 1000	0.4900	0.4800

4. Implementation of the Protis Recurrent Neural Network Method for Categorical Data.

From the results of the training and testing carried out, it was found that the results of a comparison of the accuracy level between Recurrent Neural Network and Protis Recurrent Neural Network Method for Categorical Data with epoch 80 are 0.043, epoch 100 is 0.485, epoch 120 is 0.894, epoch 250 is 0.608 and epoch 400 is 0445. The results of PRNN are higher at 250 epoch measurements. In the training and testing trials of the RNN and PRNN methods for categorical data, it was found that the best accuracy results were for testing data measurements with an epoch of 120, namely 0.894 or 89.4 %. A comparison of RNN and PRNN Methods' categorical data is illustrated in Table 3.

Table 3

Comparison of RNN and PRNN Methods categorical data

Method	Recurrent Neural Network	Protis Recurrent Neural Network
RMSE Training Epoch 80	0.247	0.317
RMSE Testing Epoch 80	0.063	0.043
RMSE Training Epoch 100	0.115	0.275
RMSE Testing Epoch 100	0.445	0.485
RMSE Training Epoch 120	0.243	0.113
RMSE Testing Epoch 120	0.844	0.894
RMSE Training Epoch 250	0.215	0.115
RMSE Testing Epoch 250	0.708	0.608
RMSE Training Epoch 400	0.017	0.117
RMSE Testing Epoch 400	0.666	0.445

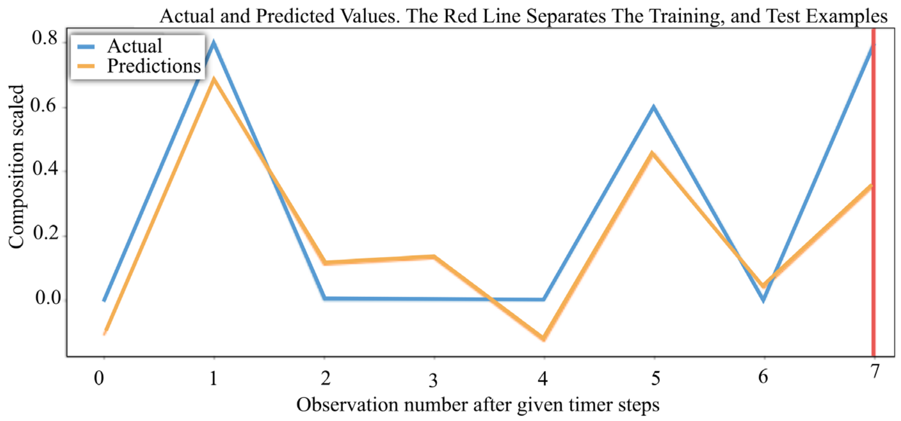
5. Implementation of the Protis recurrent neural network method for time series data.

From the training and testing results, it can be seen that the development of the Protis Reccurent Neural Network method for time series data has an improvement seen from the comparison where the RMSE Value of the Protis Reccurent Neural Network method at epoch 1000 is smaller than the Reccurent Neural Network. Furthermore, to make it easier to know the number of Hidden Neuron ranges, the display layer is made as a website to facilitate data visualization. The implementation of the Protis Recurrent Neural Network for time series for epoch 100 data is shown in Fig. 7, *a*, the implementation with epoch 120 is shown in Fig. 7, *b*. Then the implementation with 250 epochs is shown in Fig. 7, *c*, and the implementation with a total of 400 epochs is shown in Fig. 7, *d*.

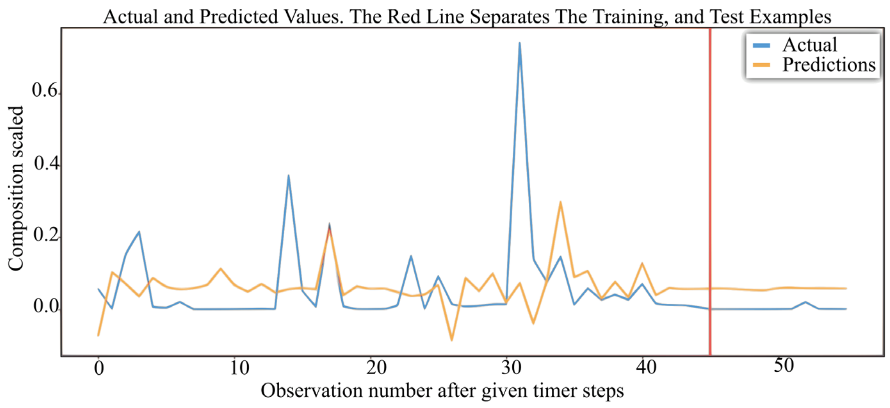
6. Implementation of the Protis convolutional neural network method for image data.

In implementing the Protis Convolutional Neural Network Method for image data, several datasets were used, including CIFAR image data, food product data, and rice disease data where the dataset was trained and tested with epochs 100 and 120 and then compared using the Convolutional Neural Network architecture where the results are given in Table 4.

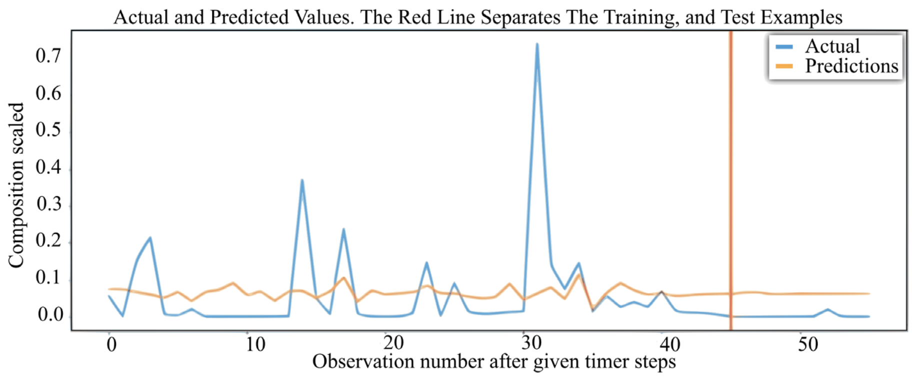
The protis convolutional neural network achieves an accuracy of 95.9 %. Therefore, from the three tested datasets, the protis convolutional neural network exhibits the highest accuracy value. So, from these results, it is found that the Protis neural network can be applied to several types of datasets, including categorical, image, and time series data, and can be used in several architectures, including Convolutional neural networks, Feedforward neural networks, and recurrent neural networks.



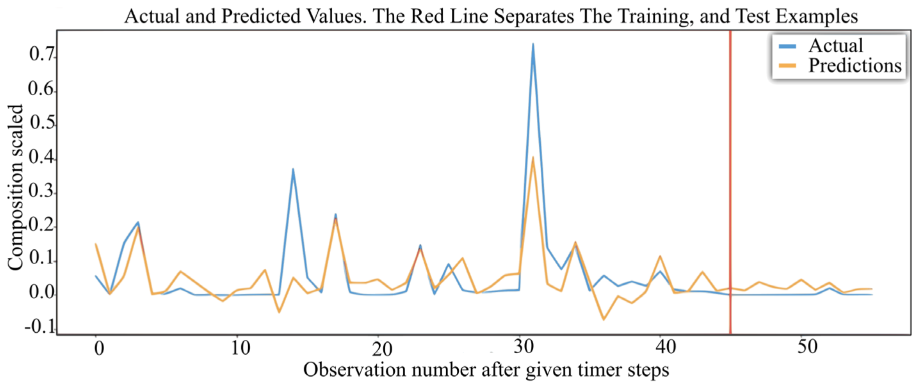
a



b



c



d

Fig. 7. Implementation of the Protis Recurrent Neural Network for Time series Data: *a* – epoch 100; *b* – epoch 120; *c* – epoch 250; *d* – epoch 400

Table 4

Comparison of CNN with PCNN for image data

Convolutional Neural Network					
Dataset Epoch 100	AUC	AUC	AUC	AUC	AUC
	58:42:00	60:40:00	70:30:00	80:20:00	90:10:00
CIFAR	80.47	75.00	80.93	70.95	67.99
Food product	64.80	54.80	50.30	60.80	43.80
Rice Diseases	80.47	86.47	77.47	75.47	82.47
Average AUC	75.25	72.09	69.57	69.07	64.75
Dataset Epoch 120	AUC	AUC	AUC	AUC	AUC
	58:42:00	60:40:00	70:30:00	80:20:00	90:10:00
CIFAR	80.47	95	93.3	95	99
Food product	54.80	57.80	60.80	64.80	65.80
Rice Diseases	81.67	90.55	89.00	93.67	92.56
Average AUC	72.31	81.12	81.03	84.49	85.79
Protis Convolutional Neural Network					
Dataset Epoch 100	AUC	AUC	AUC	AUC	AUC
	58:42:00	60:40:00	70:30:00	80:20:00	90:10:00
CIFAR	90.47	95.00	93.30	95.00	99.00
Food product	92.47	96.30	94.30	96.00	98.90
Rice Diseases	90.32	93.2	97.33	95	99
Average AUC	91.09	94.83	94.98	95.33	98.97
Dataset Epoch 120	AUC	AUC	AUC	AUC	AUC
	58:42:00	60:40:00	70:30:00	80:20:00	90:10:00
CIFAR	92.85	92.50	96.67	99.00	99.00
Food product	91.85	96.50	98.06	98.90	98.00
Rice Diseases	97.85	97.50	96.33	99.30	98.00
Average AUC	94.18	95.50	97.02	99.07	98.33

5. 4. Evaluating the performance of Protis, finding the optimal hidden layers

In applying the protist neural network method, categorical datasets and time-series data are used, with categorical data, namely disease distribution data consisting of 6 variables and 1 output consisting of 3 labels. For time series data, data on the number of dengue disease patients for 20 years are used. In this data trial, a comparison was also made with several other methods to see the performance of the proposed method approach results of classification assessments done with the Protis Neural Network methodology. The Protis neural network, which has an AUC of 0.997, CA value of 0.950, F1 value of 0.950, Precision value of 0.952, and Recall value of 0.950, is the most effective approach, according to the data. According to this study, the Protis Neural Network method is superior to traditional machine learning methods for categorical data situations.

In the final evaluation of determining the number of hidden layer neurons in the neural network architecture, it was found that from the results of the generating neuron model, Protis continued to search for the best architecture through 4 phases, namely anaphase, telophase, metaphase, and prophase. Finally, the best architectural standard was found from Protis Neural Network, Protis Recurrent Neural Network, and Protis Convolutional Neural Network, namely from the range of neurons 0–35 neurons in each layer. The final evaluation is shown in Table 5.

The standard neurons can be used as a reference for the formation of the neural network's architecture.

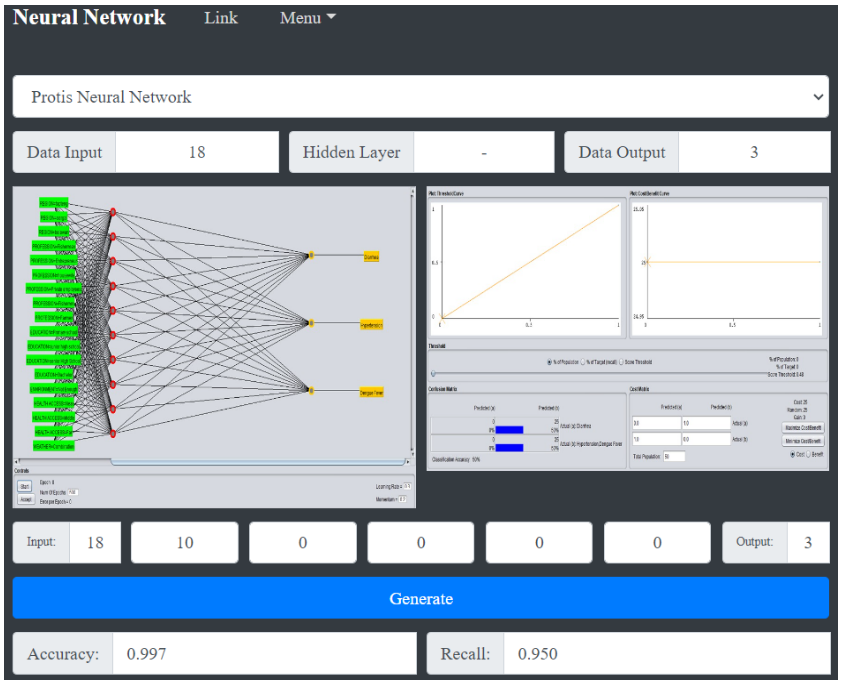
Table 5

Final Evaluation of Hidden Layer in Neural Networks

Algorithm	Input	Hidden Layer						Output	AUC
Neural Network	6	–	–	–	–	–	–	3	0.749
Protis Neural Network	6	12	0	35	32	8	0	3	0.997
Recurrent Neural Network	6	–	–	–	–	–	–	3	0.156
Protis Recurrent Neural Network	6	23	0	34	23	8	2	3	0.334
Protis Convolutional Neural Network	6	–	–	–	–	–	–	3	0.925
Protis Convolutional Neural Network	6	41	17	29	34	15	0	3	0.959

5. 5. Visualization in neural network modeling by providing insight into the adaptation of protis life processes for better development of hidden layers

To facilitate the standard determination of the number of hidden layer neurons in the neural network architecture, namely the Protis neural network, data visualization is then carried out using a website because the number of neurons in the neural network architecture is not visible, but with this framework, the number of neurons in the hidden layer can be seen. So far, the hidden layer used is still only 5 layers with a maximum number of neurons that can be formed up to 1000 neurons, but from several trials, it was found that the optimal number of neurons was 35 in each layer. The Framework website Generate Neuron Hidden Layer Protis Neural Network is shown in Fig. 8.



Protis Neural Network

Data Input 18 | Hidden Layer - | Data Output 3

Input: 18 | 10 | 0 | 0 | 0 | 0 | Output: 3

Accuracy: 0.997 | Recall: 0.950

Fig. 8. Display generate neuron hidden layer Protis Neural Network

In this framework, we can find an overview of the architecture formed and the results of accuracy and recall, input, and output.

6. Discussion of the results of optimization of the Protis Neural Network architecture for categorical, time series and image data

The results of studies from optimizing the Protis Neural Network architecture for categorical, time series, and image data types provide valuable insights into the approach's effectiveness. This study shows that the protis approach, which consists of several phases including prophase, metaphase, and anaphase/telophase found in Fig. 1 and Fig. 2, turns out to be effectively embedded in the classic Neural Network architecture where modification formulas are carried out in this architecture. Starting in the four phases, namely the modified modification method of the Prophase Phase by increasing the variable weights and doubling the chromosomes, modification method metaphase phase, and optimization using the activation function, modification method anaphase and telophase by generating each neuron by generating the number of neurons in multiples of two until the optimal number of neurons are found.

In this study, it was found that the average range optimization standard for the number of neurons in the hidden layer was found to be between 0 to 35 neurons. This provides a guideline for determining the appropriate size of the hidden layer in the neural network architecture, and a visualization generates the architecture in the form of a framework (Fig. 8) to visually show the optimal number of neurons in each layer. The test results of the Protis Algorithm Approach to the neural network are also shown in Table 5 regarding the Final Evaluation of Hidden Layers. Furthermore, testing the Protis Neural Network Architecture can optimize categorical data classification and prediction processes. The mean precision value of 0.952 indicates a high degree of accurate positive prediction, while the average recall value of 0.950 indicates the ability to capture a significant proportion of positive events. These results highlight the potential of the Protis algorithm in handling categorical data effectively and making good predictions.

The peculiarity of the proposed method lies in developing the Protis Neural network algorithm to optimize the neural network's architecture by optimizing the neurons contained in the hidden layer. This study shows that the metaheuristic approach optimizes the architecture for time series data. The information provided does not explicitly discuss the application of the Protis Neural Network Architecture to image data. However, given the versatility and adaptability of neural networks in dealing with image data, it can be concluded that this approach can potentially optimize the architecture for image classification and prediction tasks. Further research and experimentation will be needed to explore specific performance metrics and the ability of the Protis algorithm to optimize image data. Overall, the results obtained from optimizing the Protis Neural Network Architecture for categorical and time series data show its effectiveness in achieving high accuracy and precision. However, additional details and experimental evidence are needed to assess the performance of the approach on image data.

Further studies and comparisons with existing research can provide more comprehensive insights into the potential advantages and limitations of the Protis Neural Network Architecture in optimizing the architecture for different data types. This research is also relevant to research [19] methods to determine the optimal architecture by using a pruning technique. The unimportant neurons are identified using the delta values of hidden layers and research conducted by [30] regarding the modeling of artificial neural networks for silicon prediction in the cast iron production process, which also focuses on the optimal number of neurons and hidden layers. It is stated that the optimal number of neurons is 30 per hidden layer.

However, this research cannot be visualized through the framework. The limitation of this study is that the tests carried out only used categorical data, images, and time series but had not yet touched data such as anomaly and binomial data. The drawback of this research is that it is hoped to be discussed further with various types of datasets, such as anomaly and binomial data, as well as other datasets expected to be further developed.

7. Conclusions

1. Protis Neural Network Architecture can be implemented on Time Series, Categorical data types. The Protis Neural Network metaheuristic approach in this study can form a hidden layer neuron formation process that can optimize the Neural Network Architecture in the classification and prediction process.

2. The average range optimization standard for the number of neurons in the hidden layer is between 0 to 35 neurons. This provides a guideline for determining the appropriate size of the hidden layer in the neural network architecture.

3. Furthermore, when applying the Protis algorithm embedded in the Neural Network for categorical data and time series, impressive results are achieved. The average precision value of 0.952 indicates a high level of accurate positive predictions, while the average recall value of 0.950 demonstrates the ability to identify a significant proportion of the actual positive instances.

4. The Protis algorithm embedded in the Protis recurrent Neural Network for Categorical data measurements produces an average value of RMSE, or the difference between actual measurements and predictions, equal to 0.066. Overall, the Protis Neural Network Architecture and the embedded Protis algorithm show potential in optimizing the architecture and achieving high accuracy and precision in classification and prediction tasks for both time series and categorical data types.

5. Visualization of the model data is presented using a framework so that the number of neurons in the neural network architecture can be seen, which was previously difficult to know. This framework presents 5 layers with a maximum number of neurons that can form up to 1,000 neurons.

Conflict of interest

The authors declare that they have no conflict of interest in relation to this research, whether financial, personal, authorship or otherwise, that could affect the research and its results presented in this paper.

Funding

The study was conducted without financial support.

Data availability

All data are available in the main text of the manuscript.

References

1. Mladenović, N., Brimberg, J., Hansen, P., Moreno-Pérez, J. A. (2007). The p-median problem: A survey of metaheuristic approaches. European Journal of Operational Research, 179 (3), 927–939. doi: https://doi.org/10.1016/j.ejor.2005.05.034
2. Ren, P., Xiao, Y., Chang, X., Huang, P., Li, Z., Chen, X., Wang, X. (2021). A Comprehensive Survey of Neural Architecture Search. ACM Computing Surveys, 54 (4), 1–34. doi: https://doi.org/10.1145/3447582
3. Alkabbani, H., Ahmadian, A., Zhu, Q., Elkamel, A. (2021). Machine Learning and Metaheuristic Methods for Renewable Power Forecasting: A Recent Review. Frontiers in Chemical Engineering, 3. doi: https://doi.org/10.3389/fceng.2021.665415
4. Joshi, D., Chithaluru, P., Anand, D., Hajjej, F., Aggarwal, K., Torres, V. Y., Thompson, E. B. (2023). An Evolutionary Technique for Building Neural Network Models for Predicting Metal Prices. Mathematics, 11 (7), 1675. doi: https://doi.org/10.3390/math11071675
5. Panario, D. (2014). Open Problems for Polynomials over Finite Fields and Applications. Open Problems in Mathematics and Computational Science, 111–126. doi: https://doi.org/10.1007/978-3-319-10683-0_6
6. Panchal, G., Ganatra, A., Kosta, Y. P., Panchal, D. (2011). Behaviour Analysis of Multilayer Perceptronswith Multiple Hidden Neurons and Hidden Layers. International Journal of Computer Theory and Engineering, 3 (2), 332–337. doi: https://doi.org/10.7763/ijcte.2011.v3.328
7. Sengupta, S., Basak, S., Saikia, P., Paul, S., Tsalavoutis, V., Atiah, F. et al. (2020). A review of deep learning with special emphasis on architectures, applications and recent trends. Knowledge-Based Systems, 194, 105596. doi: https://doi.org/10.1016/j.knosys.2020.105596
8. Eskandar, H., Sadollah, A., Bahreininejad, A., Hamdi, M. (2012). Water cycle algorithm – A novel metaheuristic optimization method for solving constrained engineering optimization problems. Computers & Structures, 110-111, 151–166. doi: https://doi.org/10.1016/j.compstruc.2012.07.010
9. Alagoz, B. B., Simsek, O. I., Ari, D., Tepljakov, A., Petlenkov, E., Alimohammadi, H. (2022). An Evolutionary Field Theorem: Evolutionary Field Optimization in Training of Power-Weighted Multiplicative Neurons for Nitrogen Oxides-Sensitive Electronic Nose Applications. Sensors, 22 (10), 3836. doi: https://doi.org/10.3390/s22103836
10. Ebenezer M., A., Arya, A. (2022). An Atypical Metaheuristic Approach to Recognize an Optimal Architecture of a Neural Network. Proceedings of the 14th International Conference on Agents and Artificial Intelligence. doi: https://doi.org/10.5220/0010951600003116
11. Castellanos, J. L., Gomez, M. F., Adams, K. D. (2017). Using machine learning based on eye gaze to predict targets: An exploratory study. 2017 IEEE Symposium Series on Computational Intelligence (SSCI). doi: https://doi.org/10.1109/ssci.2017.8285207
12. Adhitya, E. K., Satria, R., Subagyo, H. (2015). Komparasi Metode Machine Learning dan Metode Non Machine Learning untuk Estimasi Usaha Perangkat Lunak. Journal of Software Engineering, 1 (2), 109–113. Available at: https://www.neliti.com/publications/90180/komparasi-metode-machine-learning-dan-metode-non-machine-learning-untuk-estimasi#cite
13. Demin, S. Yu., Berdieva, M. A., Podlipaeva, Yu. I., Yudin, A. L., Goodkov, A. V. (2017). Karyotyping of Amoeba proteus. Cell and Tissue Biology, 11 (4), 308–313. doi: https://doi.org/10.1134/s1990519x17040046
14. Harumy, T. H. F., Zarlis, M., Effendi, S., Lidya, M. S. (2021). Prediction Using A Neural Network Algorithm Approach (A Review). 2021 International Conference on Software Engineering & Computer Systems and 4th International Conference on Computational Science and Information Management (ICSECS-ICOCSIM). doi: https://doi.org/10.1109/icsecs52883.2021.00066
15. Harumy, T. H. F., Sitorus, J., Lubis, M. (2018). Sistem Informasi Absensi Pada Pt. Cospar Sentosa Jaya Menggunakan Bahasa Pemprograman Java. Jurnal Teknik dan Informatika, 5 (1), 63–70. Available at: https://jurnal.pancabudi.ac.id/index.php/Juti/article/view/95
16. Ergen, T., Pilanci, M. (2021). Convex geometry and duality of over-parameterized neural networks. Journal of Machine Learning Research, 22, 1–63. Available at: https://jmlr.org/papers/volume22/20-1447/20-1447.pdf
17. Harumy, T. H. F., Yustika Manik, F., Altaha (2021). Optimization Classification of Diseases Which is Dominant Suffered by Coastal Areas Using Neural Network. 2021 International Conference on Data Science, Artificial Intelligence, and Business Analytics (DATABIA). doi: https://doi.org/10.1109/databia53375.2021.9650223
18. Pomey, P. (2017). The Protis project (Marseilles, France). Ships And Maritime Landscapes, 484–489. doi: https://doi.org/10.2307/j.ctt20p56b6.86
19. Wagarachchi, N. M., Karunananda, A. S. (2013). Optimization of multi-layer artificial neural networks using delta values of hidden layers. 2013 IEEE Symposium on Computational Intelligence, Cognitive Algorithms, Mind, and Brain (CCMB). doi: https://doi.org/10.1109/ccmb.2013.6609169
20. Musikawan, P., Sunat, K., Kongsorot, Y., Horata, P., Chiewchanwattana, S. (2019). Parallelized Metaheuristic-Ensemble of Heterogeneous Feedforward Neural Networks for Regression Problems. IEEE Access, 7, 26909–26932. doi: https://doi.org/10.1109/access.2019.2900563

21. Guliyev, N. J., Ismailov, V. E. (2018). On the approximation by single hidden layer feedforward neural networks with fixed weights. Neural Networks, 98, 296–304. doi: https://doi.org/10.1016/j.neunet.2017.12.007
22. Qolomany, B., Maabreh, M., Al-Fuqaha, A., Gupta, A., Benhaddou, D. (2017). Parameters optimization of deep learning models using Particle swarm optimization. 2017 13th International Wireless Communications and Mobile Computing Conference (IWCMC). doi: https://doi.org/10.1109/iwcmc.2017.7986470
23. Henríquez, P. A., Ruz, G. A. (2018). A non-iterative method for pruning hidden neurons in neural networks with random weights. Applied Soft Computing, 70, 1109–1121. doi: https://doi.org/10.1016/j.asoc.2018.03.013
24. Balamurugan, P., Amudha, T., Satheeshkumar, J., Somam, M. (2021). Optimizing Neural Network Parameters For Effective Classification of Benign and Malicious Websites. Journal of Physics: Conference Series, 1998 (1), 012015. doi: https://doi.org/10.1088/1742-6596/1998/1/012015
25. Mohammed, A. J., Al-Majidi, S. D., Al-Nussairi, M. Kh., Abbod, M. F., Al-Raweshidy, H. S. (2022). Design of a Load Frequency Controller based on Artificial Neural Network for Single-Area Power System. 2022 57th International Universities Power Engineering Conference (UPEC). doi: https://doi.org/10.1109/upec55022.2022.9917853
26. Romanuke, V. (2015). Optimal Training Parameters and Hidden Layer Neuron Number of Two-Layer Perceptron for Generalised Scaled Object Classification Problem. Information Technology and Management Science, 18 (1). doi: https://doi.org/10.1515/itms-2015-0007
27. Hegde, S., Mundada, M. R. (2019). Enhanced Deep Feed Forward Neural Network Model for the Customer Attrition Analysis in Banking Sector. International Journal of Intelligent Systems and Applications, 11 (7), 10–19. doi: https://doi.org/10.5815/ijisa.2019.07.02
28. Thomas, A. J., Petridis, M., Walters, S. D., Gheytassi, S. M., Morgan, R. E. (2017). Two Hidden Layers are Usually Better than One. Communications in Computer and Information Science, 279–290. doi: https://doi.org/10.1007/978-3-319-65172-9_24
29. Cardoso, W., Di Felice, R., Dos Santos, B. N., Schitine, A. N., Pires Machado, T. A., Sousa Galdino, A. G. de, Morbach Dixini, P. V. (2022). Modeling of artificial neural networks for silicon prediction in the cast iron production process. IAES International Journal of Artificial Intelligence (IJ-AI), 11 (2), 530. doi: https://doi.org/10.11591/ijai.v11.i2.pp530-538
30. Zhou, Y., Niu, Y., Luo, Q., Jiang, M. (2020). Teaching learning-based whale optimization algorithm for multi-layer perceptron neural network training. Mathematical Biosciences and Engineering, 17 (5), 5987–6025. doi: https://doi.org/10.3934/mbe.2020319
31. Sadollah, A., Eskandar, H., Lee, H. M., Yoo, D. G., Kim, J. H. (2016). Water cycle algorithm: A detailed standard code. SoftwareX, 5, 37–43. doi: https://doi.org/10.1016/j.softx.2016.03.001
32. Zheng, Y.-J., Lu, X.-Q., Du, Y.-C., Xue, Y., Sheng, W.-G. (2019). Water wave optimization for combinatorial optimization: Design strategies and applications. Applied Soft Computing, 83, 105611. doi: https://doi.org/10.1016/j.asoc.2019.105611
33. Wang, N., Er, M. J., Han,M. (2015). Generalized Single-Hidden Layer Feedforward Networks for Regression Problems. IEEE Transactions on Neural Networks and Learning Systems, 26 (6), 1161–1176. doi: https://doi.org/10.1109/tnnls.2014.2334366
34. Geurts, A. M., Hackett, C. S., Bell, J. B., Bergemann, T. L., Collier, L. S., Carlson, C. M. et al. (2006). Structure-based prediction of insertion-site preferences of transposons into chromosomes. Nucleic Acids Research, 34 (9), 2803–2811. doi: https://doi.org/10.1093/nar/gkl301
35. Yang, X.-S. (2011). Metaheuristic Optimization: Algorithm Analysis and Open Problems. Lecture Notes in Computer Science, 21–32. doi: https://doi.org/10.1007/978-3-642-20662-7_2
36. Yang, X.-S., He, X. (2014). Swarm Intelligence and Evolutionary Computation: Overview and Analysis. Recent Advances in Swarm Intelligence and Evolutionary Computation, 1–23. doi: https://doi.org/10.1007/978-3-319-13826-8_1
37. Agrawal, P., Abutarboush, H. F., Ganesh, T., Mohamed, A. W. (2021). Metaheuristic Algorithms on Feature Selection: A Survey of One Decade of Research (2009-2019). IEEE Access, 9, 26766–26791. doi: https://doi.org/10.1109/access.2021.3056407
38. Ge, D. H., Li, H. S., Zhang, L., Liu, R. Y., Shen, P. Y., Miao, Q. G. (2020). Survey of Lightweight Neural Network. Journal of Software. doi: https://doi.org/10.13328/j.cnki.jos.005942
39. Hofmann, W., Sedlmeir-Hofmann, C., Ivandic', M., Ruth, D., Luppa, P. (2010). PROTIS: Use of Combined Biomarkers for Providing Diagnostic Information on Disease States. The Urinary Proteome, 123–142. doi: https://doi.org/10.1007/978-1-60761-711-2_8