Prediction of Angiotensin-Converting-Enzyme (ACE) Inhibitors as Antihypertensive Agents Using Artificial Neural Network Optimized by Artificial Bee Colony

1st Atilla Fejril
School of Computing
Telkom University
Bandung, Indonesia
atillafejril@student.telkomuniversity.ac.id

2nd Isman Kurniawan
School of Computing
Telkom University
Bandung, Indonesia
ismankrn@telkomuniversity.ac.id

***Abstract*—This study uses a dataset from the ChEMBL database to predict the activity of Angiotensin-Converting Enzyme (ACE) inhibitors as antihypertensive medicines by use of the Artificial Bee Colony (ABC) algorithm and Artificial Neural Network (ANN). Although they are time-consuming, expensive, and prone to uncertainty, traditional methods include wet-lab testing are often utilized for ACE inhibitor identification. This study intends to increase predictive performance by including systematic optimization through the ABC algorithm with ANN. Architectural and hyperparameters of ANN models were optimized using the ABC algorithm. With an R^2 of 0.683 on the test set, the model with 10 population size showed the greatest performance among the five models evaluated, therefore proving its efficacy in reflecting the general data pattern. These findings show the possibilities of integrating the ABC algorithm with ANN for precise prediction of ACE inhibitor activity, therefore offering a faster and less expensive substitute for conventional approaches. Larger and more varied datasets are required for more thorough investigation to confirm the generalizability of this method for several drug discovery uses.**

***Index Terms*—artificial bee colony, artificial neural network, ACE inhibitor, hypertension**

I. Introduction

Antihypertensive medications can usually be used to treat hypertension, which is a major cause of death worldwide[1, 2]. However, approximately half of those receiving therapy have uncontrolled hypertension because over 97% of persons with high blood pressure do not take medication[3]. High blood pressure, which is controlled by the renin-angiotensin system and in which the Angiotensin-Converting Enzyme (ACE) is essential, is one of the main causes of hypertension. Hence, hypertension can be avoided by inhibiting ACE.[4]. One of the most popular cardiovascular medications for treating hypertension and associated disorders like heart failure is an ACE inhibitor[5].

ACE inhibitors are antihypertensive drugs with many modes of action[6]. They prevent the formation of angiotensin II by inhibiting the ACE enzyme[6]. The ACE enzyme converts angiotensin I into angiotensin II, a potent vasoconstrictor and aldosterone secretion stimulant[6]. Because this enzyme causes vasodilation and decreases vascular resistance, it lowers blood pressure[7].

One method used to discover ACE inhibitors is the traditional wet-lab test. This method focuses on laboratory experiments using chemicals and reagents based on liquids or solutions. However, this method has disadvantages related to time, cost, and uncertainty issues[8]. Therefore, it is important to develop in silico methods, such as a Machine Learning approach that can be done in more efficient time.

Recently, several studies have been conducted to predict ACE Inhibitor activity using a variety of machine learning techniques. In 2011, He R. and co-workers used multivariate statistical regression techniques and amino acid descriptors to build different QSAR models to predict ACE inhibitor peptides. QSAR models for ACE inhibitor peptides have been constructed using Artificial Neural Networks (ANN), which have demonstrated a good correlation coefficient of 0.928 in predicting peptide action[9]. In 2017, Liang Y. and co-workers predicted how ACE drugs would interact with their receptors using the Support Vector Machine (SVM) method. With an accuracy rate of 88.74, the data demonstrated that SVM is a good predictor[10].

Machine learning has improved the prediction of ACE inhibitors in recent research. In 2020, Wang F. and co-workers employed molecular docking and 3D-QSAR models to obtaining predictive correlation coefficients (R^2_{pred}) of 0.6257 and 0.6969 using ComFa and ComSiA models[11]. In 2021, Wang L. and co-workers discovered that XGBoost performed better than RF, SVM, and kNN, attaining 94.11% AUC and 86.5% accuracy[12]. In 2023, Yu T. et al. predicted bioactivity using Random Forest and Mordred descriptors, with test accuracy of 0.823 and 0.745[13].

Deep learning techniques, as more advanced methods of machine learning, are quite promising to obtain better results. Unfortunately, the utilization of deep learning techniques, such as Artificial Neural Networks (ANN) is still quite rare.

Also, one of the challenges in utilizing deep learning is the determination of architecture and hyperparameters. This issue can be addressed by implementing meta-heuristic optimization methods, such as Artificial Bee Colony (ABC). The algorithm has been proven to be effective in optimizing a wide range of cases.

This study aims to implement ANN that is optimized by using ABC to perform ACE inhibitor prediction. The ABC method effectively explores and exploits the parameter search space, overcoming constraints in manual optimization such as a confined search space and increased complexity with greater feature dimensions. It was inspired by the foraging activity of bees. ABC performs better in complicated, multimodal issues by striking a balance between local and global search procedures[14, 15]. ANN is used because, even under a variety of circumstances, it can generate predictions that are both valid and extremely accurate while also closely matching experimental data[16, 17].

II. Materials and methods

A. Dataset

This study used a chemical structure dataset of 255 compounds with ACE inhibitory effect from Rattus norvegicus. This data came from the ChEMBL database, which contains a wide range of heterocyclic chemical types. The compounds' inhibitory activity was expressed as IC_{50} (nM) and converted to molar units prior to calculating pIC_{50} ($-log_{10}IC_{50}$) for QSAR analysis. Using OpenBabel, SMILES notations of all compounds were obtained from ChEMBL and transformed into SDF format considering 3D geometry optimization. Afterwards, the chemical structures were energy optimized in Marvin from ChemAxon using the MMFF94 force field. Before the dataset is processed, a cleaning process will be carried out so that the data used is relevant data and can provide optimal results. The dataset is also divided into train data and test data with a ratio of 70:30.

B. Artificial Neural Network

The Artificial Neural Network (ANN), a data processing system that effectively processes information, is inspired by a biological neural network system, such as the brain. Artificial neural networks (ANNs) are composed of interconnected processing units called neurons that, like humans, learn by doing in order to solve problems. Input, hidden, and output are three or more interconnected levels that are frequently used to organize networks. The input layer receives information from external sources, processes it in the hidden layers, and then produces the final result in the output layer[18].

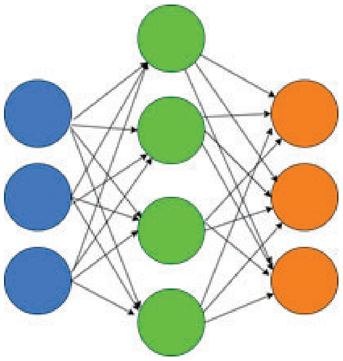

Fig. 1: The basic architecture of Artificial Neural Network.

Nodes, which stand for separate units that process input in the network, are also known as neurons or processing elements in artificial neural networks (ANNs). The layers of nodes include input nodes that take in information from the outside world, hidden nodes that perform calculations, and output nodes that generate the final output. Edges in an ANN represent connections between nodes, through which information flows during the calculation process. These connections, or edges, are associated with weights that determine the strength of the connection between nodes, affecting the impact of one node's output on another[18]. A node receives input from the previous layer and passes it to the next layer after processing it.

$$Y_k(x) = f\{\sum_{i=1}^{n} w_{ki}x_i + b_k\} \tag{1}$$

Equation 1 shows the mathematical operations performed by nodes[19]. Where Y_k is the output of the k^{th} node, w_{ki} is the i^{th} element of the pre-trained weight matrix of the k^{th} node in the l^{th} layer, x_i is the i^{th} input of the node and b_k is the bias of the node and f is the activation function[19].

C. Model Optimization

Inspired by the foraging behavior of honeybees, the Artificial Bee Colony (ABC) algorithm represents the solution in a multi-dimensional search space as a food source (nectar) and maintains a population of three different bee species (scout, employed, and spectator) to find the best food source. The method iteratively improves the solution by having employed bees exploit the food source, spectator bees select the food source based on the knowledge provided by the employed bees, and scout bees discover new food sources through random search. ABC has been successfully applied in neural network training to maximize feed-forward neural network link weights for classification applications[20].

Inspired by honeybee foraging, the ABC algorithm's stochastic optimization approach makes it a potent tool for optimization problems by effectively exploring and exploiting solutions in a multi-dimensional search space[20].

In general, the position of the i^{th} food is represented as $S_i = (S_{i1}, S_{i2}, ..., S_{iD})$. The worker bees share information when they return to the hive. The other bees go to the food source region explored by the worker bees in S_i based on the probability P_i defined as

$$P_i = \frac{fit_i}{\sum_{k=1}^{FS} fit_k} \tag{2}$$

Where FS is the total of Food Sources. The fitness value fit_i is calculated using the equation

$$fit_i = \frac{1}{1 + f(S_i)} \tag{3}$$

Where $f(S_i)$ denotes the objective function under consideration. The spectator bees find their food source in region S_i by using the equation

$$S_{new} = S_{ij} + r * (S_{ij} - S_{kj}) \tag{4}$$

Where S_{new} is the new food source explored by the spectator bee, and k is the solution in neighborhood i, r^* is a random number in the range -1 to +1, and j is the dimension of the problem under consideration. The bee moves to the new food source leaving the old one if the new fitness value surpasses than the fitness value attained thus far; else, it keeps the old one. Once all the other worker bees have finished, information is given to the spectator bees. Each spectator bee selects its food source according to the probabilities given above. Therefore, a good food source is well accommodated by the spectator bees. Each bee searches for food for a certain number of cycles. If it doesn't find food, it becomes a scout bee[21].

The number of hidden layers, the number of nodes in each hidden layer, the activation function in each hidden layer, and the model's optimizer type are among the hyperparameters that the Artificial Bee Colony Algorithm will optimize, as listed in II. The Neural Network Fixed Parameters, including validation split, epochs, and callbacks, were shown in III.

TABLE I: Artificial Bee Colony Algorithm Parameters

Parameter	Value	Description
population_size	[10, 20, 30, 40, 50]	implementation of population individual that is a bee
limit	100	The iteration limit at which scout bees take the place of unimproved food sources because they are deemed ineffective.

TABLE II: Artificial Neural Network Optimization Parameters

Hyperparameter	Value Range
Hidden Layer	[3, 4, 5]
Hidden Node	[8, 9, ..., 127, 128]
Activation	[relu, tanh, gelu]
Optimizer	[adam, sgd, rmsprop]

TABLE III: Artificial Neural Network Fixed Parameters

Hyperparameter	Value Range
Validation Split	0.3
Epochs	100
Callbacks	Early stopping [monitor = val_loss, patience = 7, mode = min, restore_best_weights = true]

TABLE IV: ABC Algorithm Parameter Scheme

Scheme	Population_size
pop10	10
pop20	20
pop30	30
pop40	40
pop50	50

D. Model Validation

The developed ANN model will go through internal and external validation. The coefficient of determination (R^2_{train}) will be computed using training data for internal validation. This demonstrates that the model can effectively detect patterns in the inputs and ensures that it performs well with the data it was trained on. To do external validation in the interim, the coefficient of determination (R^2_{test}) is calculated using test data. This step evaluates the model's generalizability by analyzing how well it performs on unknown data to ensure the model's practical usefulness. If the R^2 value is more than 0.6, the model can be accepted since it indicates that the expected accuracy and dependability satisfy the established acceptance criteria. The following formula is used to calculate the validation parameters.

$$R^2 = 1 - \frac{\Sigma\left(y_{test} - \hat{y}_{test}\right)^2}{\Sigma\left(y_{test} - \bar{y}_{train}\right)^2} \tag{5}$$

$$k = \frac{\Sigma\left(y \times \hat{y}\right)}{\Sigma\left(\hat{y}\right)^2} \tag{6}$$

$$r_0^2 = 1 - \frac{\sum(y - k \times \hat{y})^2}{\sum(y - \bar{y})^2} \tag{7}$$

$$r_0'^2 = 1 - \frac{\sum(\hat{y} - k' \times y)^2}{\sum(\hat{y} - \bar{\hat{y}})^2} \tag{8}$$

$$r_m^2 = r^2\left(1 - \sqrt{r^2 - r_0^2}\right) \tag{9}$$

$$r_m'^2 = r^2\left(1 - \sqrt{r^2 - r_0'^2}\right) \tag{10}$$

$$\overline{r_m^2} = \frac{\left(r_m^2 + r_m'^2\right)}{2} \tag{11}$$

$$\Delta r_m^2 = \left|r_m^2 - r_m'^2\right| \tag{12}$$

$$r^2 = \frac{\left[\Sigma(y - \bar{y})(\hat{y} - \bar{\hat{y}})\right]^2}{\Sigma(y - \bar{y})^2 \times \Sigma(\hat{y} - \bar{\hat{y}})^2} \tag{13}$$

where the experimental and expected values of pIC_{50} are denoted by y and $\hat{y}$, respectively, and the average of the experimental and predicted values is denoted by $\bar{y}$ and $\bar{\hat{y}}$. You can use this parameter to make sure the model isn't overfitting. The following standards were used to evaluate the model's acceptability

$$R^2 > 0.6$$

$$0.85 \leq k \leq 1.15 \text{ or } 0.85 \leq k' \leq 1.15$$

$$\frac{(r^2 - r_0^2)}{r^2} \leq 0.1 \text{ or } \frac{(r^2 - r_0'^2)}{r^2} \leq 0.1$$

$$\left| r_0^2 - r_0'^2 \right| < 0.3$$

$$\overline{r_m^2} > 0.5$$

$$\Delta r_m^2 < 0.2$$

III. Result and discussion

A. Architecture Optimization

Table V summarizes the hyperparameter tuning results for five neural network models. Model pop10, pop40, and pop50 has four hidden layer with different node, activation function, and optimizer configurations: [53, 90, 112, 104], [tanh, tanh, relu, tanh], and Adam for model pop10, [104, 65, 59, 32], [relu, relu, tanh, tanh], Adam for model pop40, and [112, 49, 92, 114], [tanh, tanh, tanh, relu], Adam for model pop50. While Model pop20 and pop30 has 3 hidden layer with different node activation function, and optimizer configurations: [72, 24, 86], [tanh, tanh, relu], SGD for Model pop20 and [84,77, 93], [tanh, tanh, tanh], RMSprop for Model pop30. All models employ linear activation for output layers to predict continuous values in regression task.

TABLE V: Hyperparameter Tuning

Model	HL	HN	Activation	Opt
pop 10	4	[53, 90, 112, 104]	[tanh, tanh, relu, tanh]	Adam
pop 20	3	[72, 24, 86]	[tanh, tanh, relu]	SGD
pop 30	3	[84, 77, 93]	[tanh, tanh, tanh]	RMSprop
pop 40	4	[104, 65, 59, 32]	[relu, relu, tanh, tanh]	Adam
pop 50	4	[112, 49, 92, 114]	[tanh, tanh, tanh, relu]	Adam

The Sample of learning curve for Loss in the created ANN model is shown in Fig. 2 It can be concluded that, It can be assumed that, the trained model can develop well.

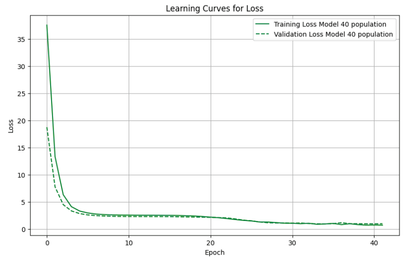

Fig. 2: Sample of Learning Curve.

Fig. 3 shows that each model with different populations (10, 20, 30, 40, and 50) has unique convergence characteristics. pop10 has the highest initial value and shows slow progress at the beginning before finally showing significant progress, although finally getting a suboptimal final value. pop20 does not demonstrate much improvement in the end, but it does exhibit a quicker pattern of steady decrease and stability than pop10. Early prototypes of pop30 showed significant research and underwent significant modifications before rapidly convergent to a relatively low value that was better than Pop40 but not as excellent as pop50. pop40 showed little exploration because the final value was higher than pop50, even though it remained stable from the start of the iterations with no noticeable changes. pop50 outperformed all the models with the lowest initial value, the highest stability, and the most optimized final value. In general, larger populations yield better answers and faster convergence, and the pop50 is the most effective population for getting the greatest results.

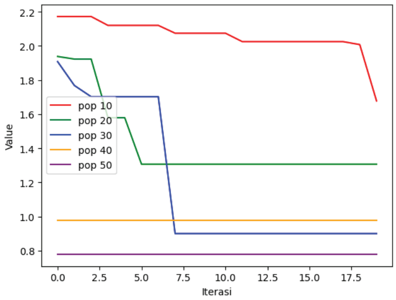

Fig. 3: Convergent Plot.

TABLE VI: Calculated Statistical Parameter Training

Model	R^2	k	$\frac{(r^2-r_0^2)}{r^2}$	$\left\|r_0^2 - r_0'^2\right\|$	$\overline{r_m^2}$	Δr_m^2
pop10	0.763	**0.870**	**0.001**	0.056	**0.671**	0.176
pop20	0.702	0.832	0.006	**0.041**	0.62	**0.106**
pop30	0.740	0.866	0.003	0.072	0.652	0.193
pop40	0.661	0.730	0.002	0.126	0.557	0.239
pop50	**0.771**	0.850	0.002	0.043	0.537	0.146

TABLE VII: Calculated Statistical Parameter Test

Model	R^2	k	$\frac{(r^2-r_0^2)}{r^2}$	$\left\|r_0^2 - r_0'^2\right\|$	$\overline{r_m^2}$	Δr_m^2
pop10	0.683	**0.921**	**0.001**	**0.116**	**0.568**	0.233
pop20	0.529	0.825	0.029	0.119	0.407	**0.131**
pop30	0.652	0.851	0.002	0.169	0.521	0.252
pop40	0.668	0.754	0.002	0.167	0.522	0.251
pop50	**0.684**	0.88	0.003	0.157	0.537	0.244

To evaluate the performance of the models, a range of statistical parameters was assessed, as shown in Table VI and Table VII. The models under consideration differ by population size (10, 20, 30, 40, and 50) and activation functions (GELU, ReLU, and Tanh), with a variety of configurations in terms of the number of hidden layers and the number of nodes in each layer.

Number of hidden layers (*HL*), neurons per layer (*HN*), activation function, and optimizers impact model performance on many metrics. pop10 performed best for parameters k,

$\frac{(r^2-r_0^2)}{r^2}$, and $\overline{r_m^2}$. The model's architecture: 4 hidden layers with varying numbers of neurons ($[53, 90, 112, 104]$ and activation functions tanh and relu help explain this. The tanh function captures complicated data patterns with non-linearity, while the Adam optimizer provides convergence efficiency. The reduced population size enables more precise parameter exploration, leading to optimal solutions for measures like k.

The pop50 model explains data fluctuations best on R^2. This model's complicated architecture with 4 hidden layers [112, 49, 92, 114] and activation functions tanh for most layers captures non-linear correlations well. Last layer relu improves model data generalization. Adam's optimizer, utilized here, is stable for complex models. This model performs well because to its vast population size and parameter exploration possibilities. However pop10 has a very small difference in R^2, but pop10 can outperform pop50. So pop10 can be said to be better overall.

The pop20 performs best on parameters Δr_m^2, demonstrating prediction stability and consistency. Three hidden layers [72, 24, 86] with activation functions tanh and relu and an SGD optimizer help this model focus on simple generalization patterns. The moderate population size also ensures a balance between exploration and exploitation, resulting in more stable predictions with low variation.

Based on the analysis results, the selection of population size in the model setup has a significant impact on the performance of the statistical parameters tested. Smaller populations, such as pop10, tend to perform best on local and exploratory accuracy-related metrics, such as k and $\overline{r_m^2}$, as they are able to explore the solution space more purposefully. In contrast, larger populations, such as pop50, excel at global metrics, such as R^2, which indicates the ability to better capture the overall pattern of the data. This is supported by the complex model architecture with a combination of activation functions tanh and relu, as well as the convergence stability obtained through the use of the Adam optimizer. Models with medium population, such as pop20, show stable performance on metrics that measure consistency, such as Δr_m^2, thanks to a simpler architecture and the use of the SGD optimizer. These results indicate that population size, model architecture, and optimization algorithm should be chosen based on the metrics to be optimized to provide the optimal performance for the application. These findings inform optimization algorithm and model setup design in numerous sectors.

IV. Conclusion

This study aimed to optimize the predictive performance of Artificial Neural Network (ANN) for ACE inhibitor activity by employing the Artificial Bee Colony (ABC) algorithm. Five models with populations of 10, 20, 30, 40, and 50 were evaluated using statistical parameters. Due to efficient parameter tuning and focused exploration, smaller populations, such pop10, perform better in localized metrics like k, $\frac{(r^2-r_0^2)}{r^2}$, and $\overline{r_m^2}$. In contrast, the model with the biggest population size (pop50) had the highest R^2, indicating better data pattern capture. Pop20, the medium population model, performed better on Δr_m^2 with consistent forecasts and little variation.

References

[1] M. H. Strauss, A. S. Hall, and K. Narkiewicz, "The Combination of Beta-Blockers and ACE Inhibitors Across the Spectrum of Cardiovascular Diseases," Cardiovascular Drugs and Therapy, vol. 37, no. 4. 2023. doi: 10.1007/s10557-021-07248-1.

[2] NICE, "Recommendations | Hypertension in adults: diagnosis and management | Guidance | NICE," NICE Guideance, 2019.

[3] S. A. Beshah et al., "Cost analysis of the WHO-HEARTS program for hypertension control and CVD prevention in primary health facilities in Ethiopia," Public Health in Practice, vol. 6, 2023, doi: 10.1016/j.puhip.2023.100423.

[4] Y. Wang et al., "Interaction of ACE genotype and salt intake on hypertension among Chinese Kazakhs: Results from a population-based cross-sectional study," BMJ Open, vol. 7, no. 5, 2017, doi: 10.1136/bmjopen-2016-014246.

[5] G. Iaccarino et al., "Renin-Angiotensin System Inhibition in Cardiovascular Patients at the Time of COVID19: Much Ado for Nothing? A Statement of Activity from the Directors of the Board and the Scientific Directors of the Italian Society of Hypertension," High Blood Pressure and Cardiovascular Prevention, vol. 27, no. 2, pp. 105–108, 2020, doi: 10.1007/s40292-020-00380-3.

[6] A. R. Y. Eff, H. Z. Huri, M. Radji, A. Mun'im, F. D. Suyatna, and Y. Eden, "Angiotensin converting enzyme (ACE) inhibitors activity from purified compounds Fructus Phaleria macrocarpa (Scheff) Boerl," BMC Complement Med Ther, vol. 23, no. 1, 2023, doi: 10.1186/s12906-023-03889-x.

[7] L. B. Arendse et al., "Novel therapeutic approaches targeting the renin-angiotensin system and associated peptides in hypertension and heart failure," Pharmacol Rev, vol. 71, no. 4, pp. 539–570, 2019, doi: 10.1124/pr.118.017129.

[8] Y. Wang, "Identification of Cardiovascular Diseases Based on Machine Learning," ACM International Conference Proceeding Series, vol. 1, no. 1, pp. 531–536, 2022, doi: 10.1145/3570773.3570855.

[9] R. He et al., "Modeling the QSAR of ACE-inhibitory peptides with ANN and its applied illustration," Int J Pept, vol. 2012, 2012, doi: 10.1155/2012/620609.

[10] Y. Liang, D. Qin, Y. Zhang, W. Liu, and G. Liang, "Comprehensive Interactions of ACE Inhibitors With Their Receptor by a Support Vector Machine Model and Molecular Docking," Journal of the Chinese Chemical Society, vol. 64, no. 3, pp. 310–320, 2017, doi: 10.1002/jccs.201600803.

[11] F. Wang and B. Zhou, "Investigation of angiotensin-I-converting enzyme (ACE) inhibitory tri-peptides: A combination of 3D-QSAR and molecular docking sim-

ulations," RSC Adv, vol. 10, no. 59, pp. 35811–35819, 2020, doi: 10.1039/d0ra05119e.

[12] L. Wang, D. Niu, X. Wang, J. Khan, Q. Shen, and Y. Xue, "A novel machine learning strategy for the prediction of antihypertensive peptides derived from food with high efficiency," Foods, vol. 10, no. 3, 2021, doi: 10.3390/foods10030550.

[13] T. Yu, C. Nantasenamat, N. Anuwongcharoen, and T. Piacham, "Machine Learning Approaches to Investigate the Structure-Activity Relationship of Angiotensin-Converting Enzyme Inhibitors," ACS Omega, vol. 8, no. 46, pp. 43500–43510, 2023, doi: 10.1021/acsomega.3c03225.

[14] D. Shootha, P. Thathola, and K. Dasila, "Artificial Bee Colony for Optimization of Process Parameters for Various Enzyme Productions," in Optimization of Sustainable Enzymes Production, 2022. doi: 10.1201/9781003292333-9.

[15] D. Karaboga and B. Akay, "A comparative study of Artificial Bee Colony algorithm," Appl Math Comput, vol. 214, no. 1, pp. 108–132, 2009, doi: 10.1016/j.amc.2009.03.090.

[16] S. Kim, Y. M. Seo, S. Y. Yoon, S. Kim, S. Pandey, and M. Y. Ha, "Thermo-hydraulic performance prediction for offset-strip fin heat exchangers using artificial neural networks," Journal of Mechanical Science and Technology, vol. 37, no. 5, 2023, doi: 10.1007/s12206-023-0436-3.

[17] C. F. Rodríguez-Hernández, M. Musso, E. Kyndt, and E. Cascallar, "Artificial neural networks in academic performance prediction: Systematic implementation and predictor evaluation," Computers and Education: Artificial Intelligence, vol. 2, 2021, doi: 10.1016/j.caeai.2021.100018.

[18] M. Thorat, S. Pandit, and S. Balote, "Artificial Neural Network: A brief study," Asian Journal of Convergence in Technology, vol. 8, no. 3, pp. 12–16, 2022, doi: 10.33130/ajct.2022v08i03.003.

[19] N. B. Gaikwad, V. Tiwari, A. Keskar, and N. C. Shivaprakash, "Efficient FPGA Implementation of Multilayer Perceptron for Real-Time Human Activity Classification," IEEE Access, vol. 7, 2019, doi: 10.1109/ACCESS.2019.2900084.

[20] J. A. Bullinaria and K. Alyahya, "Artificial bee colony training of neural networks," Studies in Computational Intelligence, vol. 512, pp. 191–201, 2014, doi: 10.1007/978-3-319-01692-4_15.

[21] K. Balasubramani and K. Marcus, "A Comprehensive review of Artificial Bee Colony Algorithm," International Journal of Computers & Technology, vol. 5, no. 1, pp. 15–28, 2013, doi: 10.24297/ijct.v5i1.4382.