Fingerprint-based Side Effect Prediction using Artificial Neural Network optimized by Bat Algorithm: Case Study Metabolism and Nutrition Disorders

Sydney Salma Nur Henny
School of Computing
Telkom University
Bandung, Indonesia
salmasydneey@student.telkomuniversity.ac.id

Jondri
School of Computing
Telkom University
Bandung, Indonesia
jondri@telkomuniversity.ac.id

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

***Abstract*— This study employs the Bat Algorithm and Artificial Neural Network (ANN) to predict drug side effects associated with disorders of nutrition and metabolism, utilizing a dataset from the SIDER database. The conventional reliance on clinical trials or post-market surveillance for side effect identification has limitations, leading to late or missed detections. Recognizing the need for robust strategies, machine learning methodologies, particularly deep learning, are incorporated to enable a more nuanced analysis of data. Despite recent advancements, deep learning is underutilized, and manual tuning prevails. The Bat Algorithm, known for its efficiency, is employed for architectural optimization of the ANN models. Three different architectures are optimized, and results indicate that the best-performing model achieves an accuracy value of 0.8160. The study highlights the potential of combining the Bat Algorithm and ANN for early and efficient prediction of drug side effects, thereby reducing costs and time associated with drug development. Further validation on diverse datasets and real-world scenarios is essential for assessing the generalizability of the proposed models and their implications in advancing drug side effect prediction.**

Keywords—Bat Algorithm. Artificial Neural Network, Side Effect, Fingerprint-Based.

I. Introduction

Side effects are phenotypic responses of the human organism to drug treatment with potential negative impacts on health. [1]. Severe side effects can lead to the failure of drug development or the withdrawal of a drug from the market [2]. Shockingly, statistics indicate that serious drug side effects rank as the fourth leading cause of deaths in the US, causing 10,000 deaths annually [3]. There are many cases where drug side effects are detected too late or not even detected in clinical trials [4]. Detecting drug side effects early is crucial to prevent delayed identification, ultimately saving lives and reducing the ever-increasing costs and time in drug development. [5].

Currently, the identification of side effects relied on conventional methods, primarily through clinical trial or post-market surveillance [2]. However, these approaches were not without limitation. Clinical trials, for instance, faced challenges such as small sample sizes and susceptibility to distortions from uncontrolled external factors [2]. Recognizing the need for more robust strategies, an alternative approach emerged. This involved the incorporation of machine learning methodologies into the process, enabling a more comprehensive and nuanced analysis of data. By leveraging computational algorithms this shift aimed to address the shortcoming of traditional methods, providing a more sophisticated means of identifying and understanding potential side effects.

Recent advancements in machine learning have led to the development of diverse methods for predicting drug side effects using various data types. Notably, researchers often rely on the chemical structure and protein targets of drugs [2]. In 2019, Ding et al introduced the Maximize the Cosine Similarity-based Multiple Kernel Learning (MCS-MKL) algorithm, aiming to enhance prediction performance through information integration. They utilized the Weighted K Nearest Known Neighbors (WKNKN) algorithm to process known drug associations and side effects, contributing to performance improvement [6]. Chandak et al (2020), developed AwareDX, a machine learning algorithm that predicts sex-specific risks of adverse drug effects by leveraging a public database of adverse event reports. The algorithm combines multiple data similarities, including drug chemical structure and protein targets, using node attribute-based and network structure-based similarity measures [7].

Another approach, proposed by Liang et al (2022) involves a novel machine learning method for predicting drug side effects by treating it as a multi-label learning problem and incorporating sparse structure learning. Their approach, validated through cross-validation, significantly outperformed existing methods, providing an interpretable model for integrating chemical and biological data to enhance drug safety predictions [8]. Muhammad et al (2022) proposed the "DLMSE" deep learning model for predicting side effects of multiple drugs based on drug chemical structures. They achieved a high accuracy of 0.9494 using their multi-table classification approach [9]. In a similar vein, Nicolo et al (2022) employed the DruGNN method and an Unsupervised Algorithm to predict drug side effects based on Neural Fingerprint (NF). Their dataset included a single graph with 9222 nodes (1341 drug nodes, 7881 gene nodes) and 331,623 edges (12,002 gene-gene interaction rules, 314,369 DPI, and 5252 similarity medicines) [10].

Despite these advancements, deep learning is still underutilized for predicting drug side effects, and manual tuning is prevalent [11]. To overcome this, one alternative for architectural optimization is auto tuning. Auto tuning handles automatic code generation and optimization using various scenarios and architectures. Automated tuning involves the development of a method to optimize various parameters automatically, with the goal of maximizing or minimizing the satisfaction level of an objective function [12]. Based on the literature survey, auto tuning in deep learning in predicting drug side effects is still lacking.

This study aims to identify drug side effects using fingerprints by applying Artificial Neural Network and Bat Algorithm for architectures optimization, focusing on metabolism and nutrition disorders. The Bat Algorithm, capable of handling both continuous and discrete optimization problems, provides flexibility in modeling data [13]. This study utilizes an Artificial Neural Network, drawing inspiration from early models of sensory processing in the brain [14].

II. Materials and Methods

A. Dataset

The dataset for this study is derived from the SIDER website and focuses on case studies related to Metabolism and Nutrition Disorders [15]. SIDER is a database that includes details about approved drugs and their associated adverse reactions. The fourth edition of SIDER encompasses 1430 drugs and encompasses 5880 terms related to side effects [15]. In this study, 1080 drugs from SIDER were successfully linked to the DrugBank database, augmenting the dataset with additional drug-related biological features [15]. The dataset, known as SIDER 4, is utilized for independent experiments.

The dataset comprises molecular structures represented in SMILES format. In the initial preprocessing step, these structures are converted into binary data, with 0 indicating “Normal” and 1 indicating “Side Effect”. The chosen feature is a PubChem fingerprint computed through a PADEL descriptor. Molecular fingerprints encode structural information about a compound, capturing features such as atom types, bond types, and molecular connectivity patterns. The dataset originally contains 1121 records, with 33 duplicates that are removed. The remaining data is divided into training and testing sets in an 80:20 ratio, yielding 870 samples for training and 218 samples for testing.

Fig. 1 illustrates the complete schematic of the dataset distribution from the initial stage to its readiness for use. It can be seen in the train set that there is a data imbalance between labels 0 and 1, so it is necessary to oversample using a Random Oversampler. The result is there are 597 labels 0 and 1. Meanwhile, Fig. 2 visually captures the distribution of the Principal Component Analysis (PCA), offering insights into the reduced-dimensional representation of the dataset.

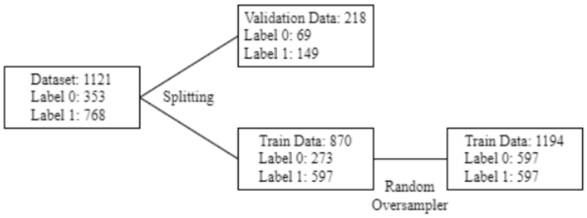
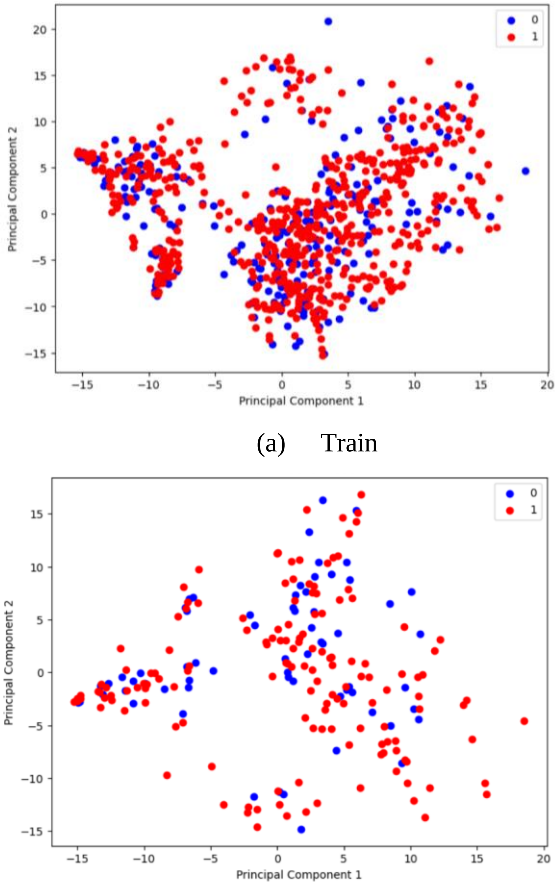


Fig. 1. RandomOverSampler Flow



(a) Train

(b) Test

Fig. 2. Distribution of Dataset for (a) Train and (b) Test

B. Artificial Neural Network

Artificial Neural Networks (ANNs) emulate early models of brain sensory processing on computers, drawing from disciplines like neuroscience, mathematics, and computer science [14]. With applications in diverse fields, ANNs possess the crucial ability to learn from input data, either with or without a teacher.

Similar to Biological Neural Networks, ANNs consist of interconnected nodes, resembling neurons. Every neural network is characterized by three essential elements: node character, network topology, and learning rules. Node character govern signal processing within a node, encompassing factors like input and output quantities and the corresponding weights. Network topology defines the arrangement and connections of nodes, and learning rules prescribe the initiation and adjustment of weights [20].

Defining the architecture of an Artificial Neural Network involves specifying the quantity of input, output, and hidden neurons, along with the number of hidden layers. According to the universal approximation theorem, a feed-forward network with a single hidden layer containing a finite number of neurons can approximate a continuous function on a compact subset of R^n, where n represents the number of inputs [21].

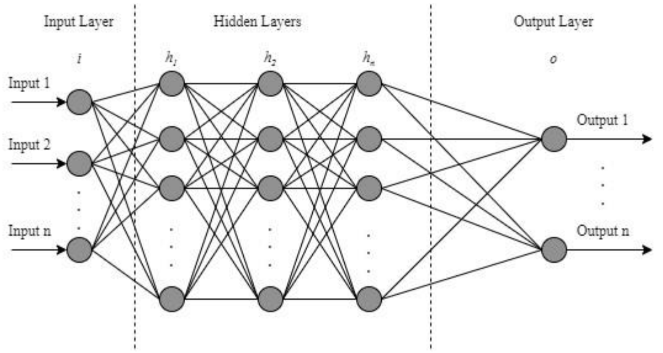


Fig. 3. Artificial Neural Network Structures

The perceptron takes input from the preceding layer, conducts certain mathematical operations, and then transmits the output to the subsequent layer. The mathematical operations are depicted in Equation (1) [22].

$$Y_k(x) = f\left\{\sum_{i=1}^{n}(w_{ki}x_i \ + \ b_{k)}\right\} \qquad (1)$$

where Y_k represent the output of the k^{th} perceptron, w_{ki} denotes i^{th} element of the pre-trained weight matrix of the k^{th} perceptron in the l^{th} layer, x_i stands for the i^{th} input of the perceptron, b_k is the bias of the perceptron and f represents the activation function [22].

Equation (2) illustrates the output matrix from the hidden layer.

$$X^2 = \ f_1\{W^2 \times \ X^1 \ + \ B^2\} \qquad (2)$$

where W^2 is the weight matrix of the hidden layer, X^1 is the output matrix from the input layer, B^2 is the bias matrix from the hidden layer and f_1 is the sigmoidal approach [22].

Equation (3) computes the output matrix for this output layer.

$$Y \ = \ f_2\{W^3 \ \times \ X^2 \ + \ B^3\} \qquad (3)$$

where W^3 is the weight matrix for the output layer, X^2 is the hidden layer for the output matrix, B^3 is the bias matrix for the output layer and f_2 is a pure linear activation function [22].

C. Model Optimization

In this case, the model is optimized using the Bat Algorithm, a highly efficient bio-inspired method developed by Yang in 2010 [16]. Widely applied in diverse engineering fields, the algorithm's simplicity facilitates computer implementation and delivers reliable results for low-dimensional problems [17]. Global search capabilities are influenced by parameters such as loudness and frequency. To improve performance, Wang et al. (2019) introduced mixBA, a superior version demonstrated through comparisons with the CEC2013 benchmark test [18].

The Bat Algorithm outperforms other algorithms due to its three key features: automatic zoom, frequency tuning, and parameter control [19]. These attributes contribute to the enhanced efficiency of the Bat Algorithm. The accompanying flowchart illustrates the sequence of steps involved in the implementation of the algorithm.

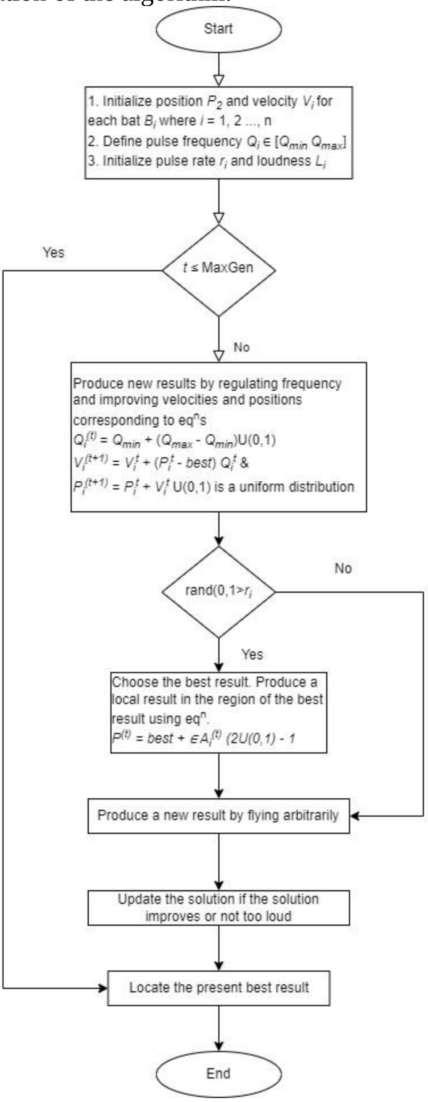

Start
1. Initialize position P_2 and velocity V_i for each bat B_i where i = 1, 2 ..., n
2. Define pulse frequency $Q_i \in [Q_{min}\ Q_{max}]$
3. Initialize pulse rate r_i and loudness L_i
Yes
$t \leq$ MaxGen
No
Produce new results by regulating frequency and improving velocities and positions corresponding to eq^{n}s
$Q_i^{(t)} = Q_{min} + (Q_{max} - Q_{min})U(0,1)$
$V_i^{(t+1)} = V_i^t + (P_i^t - best)\ Q_i^t$ &
$P_i^{(t+1)} = P_i^t + V_i^t\ U(0,1)$ is a uniform distribution
rand(0,1>r_i
No
Yes
Choose the best result. Produce a local result in the region of the best result using eq^n.
$P^{(t)} = best + \epsilon A_i^{(t)}\ (2U(0,1) - 1$
Produce a new result by flying arbitrarily
Update the solution if the solution improves or not too loud
Locate the present best result
End

Fig. 4. Bat Algorithm Flowchart

In each iteration (t), bats are characterized by their speed v_i^t, and location x_i^t, in the d-space. The best solution among all bats is represented by x*. The rules mentioned can be translated into equation for x_i^t and v_i^t:

$$f_i = f_{min} + (f_{max} - \ f_{min})\beta, \qquad (1)$$

$$v_i^{t-1} = \ v_i^{t-1} + (x_i^{t-1} - \ x_*)fi, \qquad (2)$$

$$x_i^t = x_i^{t-1} + \ v_i^t, \qquad (3)$$

where β selected from a uniform distribution in the range [1,0], is a random vector [16].

To establish an efficient system for managing exploration and transitioning to the exploitation stage as needed, it is essential to adjust both the loudness A_i and pulse emission rate r_i. Based on these considerations, the following conditions are derived:

$$A_i^{t+1} = \alpha A_i^t, r_i^{t+1} = r_i^0[1 - \exp(-\gamma t)] \qquad (4)$$

$$A_i^t \rightarrow 0, r_i^t \rightarrow r_i^0, as\ t \rightarrow \infty. \qquad (5)$$

in the context, α and γ are constant. Essentially α plays a role analogous to the cooling factor found in the cooling schedule of simulated annealing [16].

In the Bat Algorithm, the effectiveness of global search is influenced by the parameters of loudness and pulse rate, both set value of 1.0. The population rate employed in this algorithm is 40. As detailed in the Table I.

The Bat Algorithm will optimize the hyperparameters, including the number of hidden layers, the number of nodes in each hidden layer, the activation function in each hidden layer, the model's optimizer type, the learning rate value, the dropout rate value, and the adaptive parameter, as outlined in Table 2.

TABLE I. BAT ALGORITHM

Parameter	Description	Value
Population Rate	The proportion of Bat populations in a specific iteration relative to the total desired number of Bat populations.	40
Loudness	The magnitude of the amplitude or strength of the ultrasonic signal produced by Bat in the process of finding a solution.	1,0
Pulse Rate	Pulse frequency or Bat Movement in the process of finding a solution.	1,0

TABLE II. NEURAL NETWORK OPTIMIZATION PARAMETERS

Hyperparameter	Value Range
Hidden Layer	[1, 2, 3, 4, 5]
Hidden Node	[500, 501, …, 1000]
Activation	[tanh, relu, sigmoid]
Optimizer	[sgd, adam, rmsprop]

TABLE III. NEURAL NETWORK FIXED PARAMETERS

Hyperparameter	Value Range
Validation Split	0.2
Batch Size	32
Shuffle	True
Epochs	125
Callbacks	Early stopping [monitor = val_loss, patience = 5, restore_best_weights = true]

D. Model Validation

TABLE IV. CONFUSION MATRIX

Actual	Predicted	
	Positive	Negative
Positive	TP (True Positive)	FN (False Negative
Negative	FP (False Positive)	TN (True Negative)

The classification algorithm's performance is boosted by metrics such as F1-score, recall, precision, and accuracy, which evaluate results based on the confusion matrix. This matrix compares the system's classification outcomes with the expected results [23]. Refer to Table 4 for the Confusion Matrix.

From Table 4 the following equation can be determined:

$$AC = \frac{TP+TN}{TP+FP+TN+FN} \times 100\% \qquad (1)$$

$$F1 - Score = 2 \times \frac{Precision*Recall}{Precision+Recall} \qquad (2)$$

$$RE = \frac{TP}{TP+FN} \qquad (3)$$

$$PRE = \frac{TP}{TP+FP} \qquad (4)$$

where accuracy is the ratio of correctly predicted instances (1). F1-Score (2) is the average of recall (3) and precision (4). Accuracy measures the system’s precision in classifying data [23].

III. RESULT AND DISCUSSION

A. Architecture Optimization

Fig. 5 depicts the convergence of fitness loss data for optimized models, showing convergence in all three methods. The first model converges from 0.52 to approximately 0.48. The second model maintains stability around 0.51, with a notable drop in the 10th iteration. The third model fluctuates, starting at 0.52 and gradually decreasing to around 0.49. Overall, the third model exhibits more erratic behavior compared to the other two.

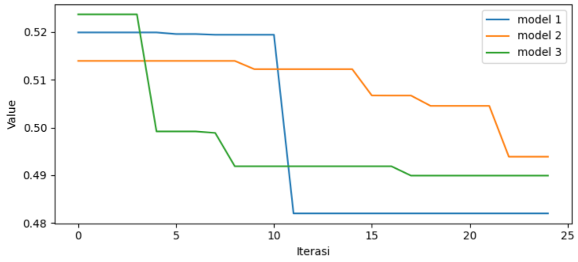


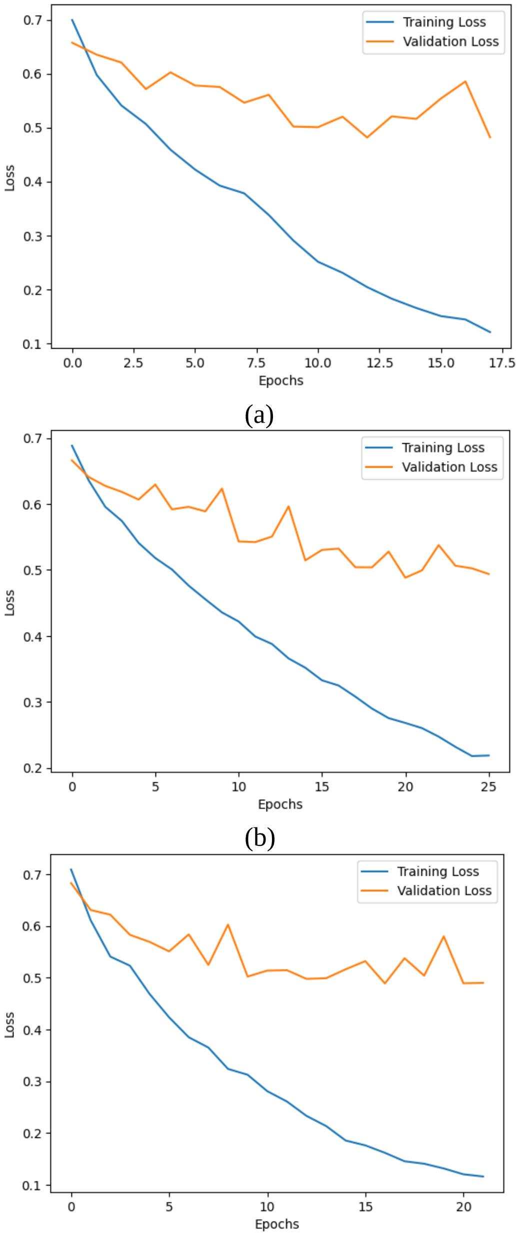
Fig. 5. Convergence Data

Table V summarizes the hyperparameter tuning results for three neural network models. Model 1 has a single hidden layer with node [288], while Models 2 and 3 also use one hidden layer with different node configurations: [54] for Model 2 and [186] for Model 3. All models employ relu activation for hidden layers to introduce non-linearity. Model 1 and Model 3 use the Adam optimizer, while Model 2 utilizes RMSprop.

TABLE V. HYPERPARAMETER TUNNING RESULTS

Model	**HL**	**HN**	**Act**	**Opt**
Model 1	1	[288]	relu	Adam
Model 2	1	[54]	relu	RMSprop
Model 3	1	[186]	relu	Adam

Fig. 6 illustrates the learning curves of Artificial Neural Networks (ANNs) incorporating Model 1, Model 2, and Model 3. Model 1 demonstrates effective learning without overfitting, Model 2 exhibits sensitivity to data distribution, indicating challenges with diverse patterns, and Model 3 consistently improves during training, showcasing its efficacy in handling intricate data patterns with robust performance.



(a)

(b)

(c)

Fig. 6. The learning curve of ANN training optimize by (a) Model 1, (b) Model 2, and (c) Model 3

Model performance is assessed through validation parameters, including confusion matrix, accuracy, and others, detailed in Table VI and Table VII for both the train and test sets. Within the training set, Baseline 1 excels in correctly identifying positives instances but reveals a limitation in mitigating false negatives. Model 3 distinguishes itself by maintaining a consistently low count of false positives. Transitioning to the test set, Model 3 continues to shine with its exceptional performance in minimizing false positives. Furthermore, in both the training and test sets, Model 2 demonstrates a trade-off between true and false positives. It maintains a balance between correctly identifying positive instances (TP) and instances incorrectly identified as positive (FP).

TABLE VI. CONFUSION MATRIX

Model	**TP**	**FP**	**TN**	**FN**
Baseline Train				
Baseline 1	**422**	32	151	**290**
Baseline 2	251	**203**	**329**	112
Baseline 3	399	55	198	243
Model 1	425	29	**430**	11
Model 2	420	**34**	418	**23**
Model 3	**443**	11	429	12
Test				
Baseline 1	**59**	10	50	**99**
Baseline 2	24	**45**	**113**	36
Baseline 3	50	19	67	82
Model 1	120	23	**124**	32
Model 2	114	**29**	117	**39**
Model 3	**124**	19	119	37

Regarding the evaluation metrics, we computed four parameters to evaluate the performance of the model, as outlined in Table VII. In the initial baseline training stage, Baseline 3 emerges with the highest accuracy at 0.6670 among the baselines, while Baseline 2 showcases higher F-1 Score at 0.6763 and recall at 0.7460. in the subsequent testing phase, Baseline 2 surpasses other baseline across metrics, particularly excelling with an accuracy of 0.6284, along with impressive F1-Score, recall, and precision values. Transitioning to the advanced models, Model 3stands out during training with the highest accuracy at 0.9743 and precision at 0.9750, whereas Model 1 excels in recall at 0.9750, however, in the testing phase, Model 1 takes the lead with the highest accuracy at 0.8160 and F1-Score at 0.8184, showcasing a well-balanced performance across various metrics. Model 2 and Model 3 also deliver commendable results, with Model 3 achieving the highest precision at 0.8623.

TABLE VII. VALIDATION PARAMETER CALCULATION RESULT

Model	Accuracy	F1-Score	Recall	Precision
Train				
Baseline 1	0.6402	0.4840	0.3424	**0.8251**
Baseline 2	0.6480	**0.6763**	**0.7460**	0.6184
Baseline 3	**0.6670**	0.5706	0.4490	0.7826
Model 1	0.9553	0.9555	**0.9750**	0.9368
Model 2	0.9363	0.9361	0.9478	0.9247
Model 3	**0.9743**	**0.9738**	0.9727	**0.9750**
Test				
Baseline 1	0.5000	0.4785	0.3356	**0.8333**
Baseline 2	**0.6284**	**0.7362**	**0.7584**	0.7152
Baseline 3	0.5367	0.5702	0.4497	0.7791
Model 1	**0.8160**	**0.8184**	**0.7948**	0.8435
Model 2	0.7725	0.7748	0.7500	0.8013
Model 3	0.8127	0.8095	0.7628	**0.8623**

In comparison to prior research efforts, our study introduces a novel approach to predicting drug side effects by leveraging the Bat Algorithm for the optimization of Artificial Neural Network (ANN) architecture. Previous studies have predominantly relied on conventional methods or various machine learning techniques, with deep learning often being underutilized and manual tuning prevalent [11]. Our work addresses this gap by combining the Bat Algorithm with ANN, demonstrating its potential in enhancing the prediction accuracy of drug side effects. Continued research and validation efforts will refine our approach, paving the way for more accurate and efficient drug side effect predictions in the future.

IV. CONCLUSION

In summary, this study applied the Bat Algorithm to optimize Artificial Neural Network (ANN) architecture for predicting drug side effects related to metabolism and nutrition disorders. The SIDER database was preprocessed, and three ANN models were fine-tuned. Notably, on the test set, Model 1 led with an accuracy of 0.8160 and a balanced F1-Score, indicating promising potential for advancing drug side effect prediction. Further validation on diverse datasets and real-world scenarios is crucial for assessing model generalizability, potentially reducing costs and time in drug development. While the study establishes a foundation for using the Bat Algorithm and ANN in drug side effect prediction, ongoing research and collaboration are essential to fully realize their potential in transforming drug development and safety assessment.

REFERENCES

[1] Michael Kuhn, Monica Campillos, Ivica Letunic, Lars Juhl Jensen, and Peer Bork, “A side effect resource to capture phenotypic effects of drugs.” January 2010.

[2] Xujun Liang, Ying Fu, Lingzhi Qu, Pengfei Zhang, and Yongheng Chen, “Prediction of drug side effects with transductive matrix co-completion.,” Bioinformatics (Oxford, England), 01 Jan 2023.

[3] Fei Wang, Ping Zhang, Nan Cao, Jianying Hu, and Robert Sorrentino, “Exploring the associations between drug side-effects and therapeutic indications.”, Vol.1, pp. 15-23 (2014).

[4] Satya Katragadda, Harika Karnati, Murali Pusala, Vijay Raghavan, and Ryan Benton, “Detecting adverse drug effects using link classification on twitter data.”, 2015 IEEE.

[5] Y. Raoul Frijters, Stefan Verhoeven, Wynand Alkema, René van Schaik & Jan Polman, “Literature-based compound profiling,”, 8(11):1521-34 (2007).

[6] Yijie Ding, Jijun Tang, Fei Guo, ” Identification of Drug-side Effect Association via Semoiyi-supervised Model and Multiple Kernel Learning.”, IEEE J Biomed, 23(6):2619-2632 (2018).

[7] Payal Chandak, Nicholas P. Tatonetti, “Using Machine Learning to Identify Adverse Drug Effects Posing Increased Risk to Women.”, Patterns 1, 100108 October 9, 2020

[8] Xujun Liang, Jun Li, Ying Fu, Lingzhi Qu, Yuying tan, Pengfei Zhang, “A novel machine learning model based on sparse structure learning with adaptive graph regularization for predicting drug side effects.”, 1532-0464/© 2022 Elsevier Inc.

[9] Muhammad Asad Arshed, Muhammad Asad Arshed, Omer Riaz, Waqas Sharif, Dr Saima Abdullah, Imran Rana Imran, “A Deep Learning Framework for Multi Drug Side Effects Prediction with Drug Chemical Substructure.”, International Journal of Innovations in Science and Technology 4(1):19-31, January 2022.

[10] Niccolò Pancino, Yohann Perron, Pietro Bongini and Franco Scarselli, “Drug Side Effect Prediction with Deep Learning Molecular Embedding in a Graph-of-Graphs Domain.”, Vol. 10, 2022.

[11] Basheer Qolomany, Majdi Maabreh, Ala Al-Fuqaha, Ajay Gupta, and Driss Benhaddou, “Parameters Optimization of Deep Learning Models using Particle Swarm Optimization.”, IWCMC 2017.

[12] Amir h. Ashouri, William killian, John cavazos, Gianluca palermo, Cristina silvano, “A Survey on Compiler Autotuning using Machine Learning.”, 3 Sep 2018.

[13] Iztok Fister jr., Iztok Fister, Xin-She Yang, Simon Fong, Yanan Zhuang, “Bat algorithm: Recent advances.” January 2014.

[14] Anders Krogh, “What are artificial neural networks?”, Vol. 26, pp. 195-197 (2008).

[15] Wen Zhang, Feng Liu, Longqiang Luo4 and Jingxia Zhang, “Predicting drug side effects by multi-label learning and ensemble learning.” BMC Bioinformatics (2015).

[16] Xin-She Yang, Bat algorithm: literature review and applications, Int. J. Bio-Inspired Computation, Vol. 5, No. 3, pp. 141–149 (2013).

[17] Asma Chakri, Haroun Ragueb & Xin-She Yang, “Bat Algorithm and Directional Bat Algorithm with Case Studies.”. pp. 189–216, January 2018.

[18] Yechuang Wang, Penghong Wang, Jiangjiang Zhang, Zhihua Cui, Xingjuan Cai, Wensheng Zhang and Jinjun Chen, “A Novel Bat Algorithm with Multiple Strategies Coupling for Numerical Optimization.”, 2019.

[19] Preeti & Dinesh Kumar, “Feature selection for face recognition using DCT-PCA and Bat algorithm.” Bharati Vidyapeeth’s Institute of Computer Applications and Management 2017.

[20] Jinming Zou , Yi Han , and Sung-Sau So, “Overview of Artificial Neural Networks.”, January 2009, Methods in molecular biology (Clifton, N.J.) 458:14-22.

[21] Facundo Bre , Juan M. Gimenez , Víctor D. Fachinotti, “Prediction of wind pressure coefficients on building surfaces using artificial neural networks.”, Vol.158, 1 January 2018, pp. 1429-1441.

[22] Nikhil B. Gaikwad, Varun Tiwari, Avinash Keskar, N. C. Shivaprakash, “Efficient FPGA Implementation of Multilayer Perceptron for Real-Time Human Activity Classification.”, Vol. 7 (2019).

[23] Crisanadenta Wintang Kencana, Erwin Budi Setiawan, Isman Kurniawan, “Hoax Detection on Twitter using Feed-forward and Back-propagation Neural Networks Method.”, Vol. 4 No. 4 (2020) 648 – 654.