www.ijecs.in
International Journal Of Engineering And Computer Science
Volume 13 Issue 09 September 2024, Page No. 26373-26390
ISSN: 2319-7242 DOI: 10.18535/ijecs/v13i08.4886

Optimizing Recurrent Neural Network with Bayesia Algorithm for Behavioural Authentication System

Taiwo Adigun [1]., Adedeji Adegbenle [2]., Oludele Awodele [3], Chibueze Ogbonna[4]

[1,] Software Engineering Department, School of Computing and Engineering Sciences, Babcock University, Ilishan-Remo, Ogun State Nigeria.

[2,3,4] Computer Science Department, School of Computing and Engineering Sciences, Babcock University, Ilishan-Remo, Ogun State Nigeria.

Abstract
Current studies on behavioural biometrics authentication have been focused on the use of deep learning and keystroke dynamics but the aspect of conscious optimization of the algorithm in order to obtain best outcome has not been considered. This study employed and incorporated Bayesian algorithm into Recurrent Neural Network to build a Keystroke Behavioural Biometric (KBB) authentication model used against social engineering attacks. The model begins with importing the keylogging dataset for data pre-processing, feature extraction, and RNN algorithm was used to build the KBB model. Hyperparameter tuning was done to achieve optimal results. A traditional optimizer called Adaptive Momentum Estimation (Adam) was used and evaluated so as to estimate the impact of optimization in model inferencing. RNN model result with Bayesian optimization technique shows a better performance than the result of RNN model with ADAM optimization. The essence of incorporating and evaluating the best optimization technique is to come up with an effective and accurate model for behavioural biometric authentication, that could mitigate effectively against social engineering attacks.

Key Words: Social Engineering Attacks, Keystroke Dynamics, Behavioural Biometrics, Optimization, Hyperparameter Tuning

1. Introduction

The challenges of preventing online identity theft, cyber fraud, malicious attacks, and phishing in Short Message Service (SMS), and man-in-the-middle attacks, are currently moving beyond configuration of Password-based authentication, 2-factor and multi-factor authentication methods because several of these attacks are based on behavioural manipulations of the victims. These attacks are known as social engineering attacks and the common authentication approaches are not built to detect and prevent social engineering [1]. Traditional authentication systems are now groping with sterner social engineering attacks. A vast majority of social attackers make use of email apps on mobile phones and voices to swindle their victims. The real-time opportunity of using mobile phones gives victims of social engineering attacks little time to detect and stop themselves from transferring funds to scammers [2]. In social engineering vishing attacks, scammers contact mobile phone users posing as representatives of legitimate businesses or government agencies to convince their victims to give them their sensitive information.

Social engineering attacks are a form of behavioural manipulations which easily bypass access control measures and existing physiological security prevention systems in place today [3]. In social engineering attacks, scammers impersonate trusted officials, like customer service representatives of a bank, to deceive and steal unsuspecting victims of millions of dollars every year [4]. Behavioural biometrics is currently being used in many authentication systems. This is made possible by using machine learning algorithms

that keep track of users' unique features and build on such unique features. This involves using user profiles and behaviour such as how the gadgets are helped, how the screen is swiped, and keyboard or gestural shortcuts [5]. However, preventing social engineering attacks using behavioural biometric approach has not received the required attention especially with focus on keystroke dynamics. This is because there were inconsistent and noisy dataset for effective keystroke attack mitigation, coupled with the use of ineffective and inefficient technique for identification and mitigation of the effect of attacks. Data needed to analyse keystroke dynamics is obtained by keystroke logging [6].

Several studies have employed behavioural biometrics especially keystroke dynamics for the purpose of authentication. [7] employed Parzen density estimation and a unimodal distribution as the statistical models for exploring identity authentication. The authors used human-computer interaction as the basis to describe a new behavioural biometric technique. By using a pointing device, they developed a system that captured users' interactions and used the acquired behavioural data to identify each user. [8] believed the aim of the US Defence Advanced Research Project Agency`s (DARPA) Active Authentication program was the continuous authentication of users of a system by using behavioural biometrics authentication systems that observed mouse movement, keystrokes and application usage in an office-like environment. According to [9], correlation analysis of a person's behavioural patterns could be used to ascertain the personality of a user of a system, and this could be used to develop the user's behaviour template that would be used to authenticate the user and grant his/her access or to reject the users based on deviations from the user's normal behaviour patterns.

The keystroke behavioural biometrics can leverage powerful statistical models and deep learning algorithms to spot the differences between a known user's gradual evolution and the unwanted presence of an entirely different user [10]. Besides being a potential mechanism for preventing social engineering attacks, it can guarantee a smooth user experience on mobile devices [11]. Behavioural biometrics can achieve a better user experience by collecting large amounts of user data or user parameters from a mobile device and using deep learning to resolve features to match each user. Current studies on behavioural biometrics authentication have been focused on the use of deep learning and keystroke dynamics [12-16]. Recurrent Neural Network (RNN) has not been reported among all the deep learning algorithms reported for behavioural biometrics authentication. RNN is known to be a fantastic algorithm in solving complex problems like image and speech recognition and believed to perform well in a noisy and non-linear data set like keylogging data. Also, it is important to know that a conscious optimization of deep learning algorithms improves their effectiveness and accuracy. This study aims to leverage the power of deep learning and optimization for more accurate and robust continuous authentication based on typing patterns.

The sections following are described as follow. Section 2 discusses the concept of optimization in details while section 3 explains the methodology employed for this study. The results are presented in section 4 and were equally discussed. Section 5 concludes this study.

2. Optimization

Optimization is a problem-solving technique that is used to get solution to a complex problem in an optimal manner. Optimization finds optimal solution to a problem by applying input to the objective function. The objective of optimization algorithm is to enhance the performance through the process of iteratively examining a set solution space and fine-tuning the parameters until the best result is achieved. The best solution could be in form of minimization or maximization of the objective function. Optimization also involve translation of real-world problem into mathematical model in order to carry out quantitative [17] analysis to select the best possible solution. Many fields of study make use of optimization techniques in decision making process [185]. Optimization algorithm is also the basis of some machine learning algorithm such as training artificial neural network and fitting logistic regression.

A. Types of Optimization Algorithm

Optimization algorithm can be categorized into two forms, differentiable objective function algorithm and non-differentiable objective function.

1. Differentiable Objective Function

Differentiable objective function is a type of optimization algorithm that deals with problem where the rate of change in the function can be calculated at all point. This rate of change is also known as the slope or derivative. The gradient in this approach helps to iteratively search for optimal solution by updating the parameters in a way that change near the objective function. Differentiable objective function algorithms include backtracking algorithm, first order algorithm and second-order algorithm.

i. Backtracking Algorithm

Backtracking Algorithm is an iterative procedure of solving search problem. This algorithm makes decisions based on feasibility of solution. Backtracking algorithm reverse any solution that go against the constraints or solution with no feasible completion. The typical usage of backtracking algorithm is for solving that relate to traversing through a distinct solution space [19]. Backtracking algorithm is very suitable for solving problem of optimization without time boundary. Figure 1 shows the diagrammatic representation of backtracking algorithm process.

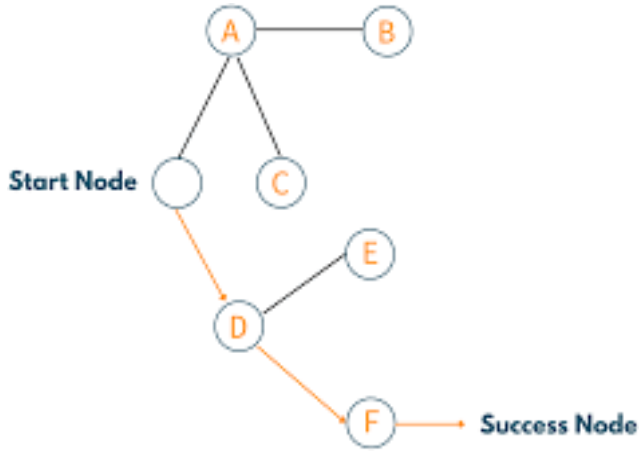

Figure 1: Backtracking Process [58]

ii. First-Order Algorithm

First-orders algorithms are an optimization algorithm mostly used in deep learning and machine learning. It is an iterative technique of getting optimal solution of an objective function making use of first derivative to get information of the objective function [20]. First-order algorithms are mostly used in solving intricate optimization problem. The algorithm provide medium accurate result [21]. Examples of First-Order algorithms are Gradient descent, AdaGrad, RMSProp and Adam.

Gradient Descent

The key task of the gradient descent algorithm is to discover the minimum value for a function [22]. The mathematical model for gradient descent algorithm is given below:

$$Y_{pred} = B0 + B1(x) \qquad \textbf{(1)}$$

where Ypred denote the output, B0 is the intercept, B1 is the slope and x represent the input value.

Adaptive Gradient

Adaptive Gradient (AdaGrad) is an optimization algorithm that utilizes the information of preceding gradients to bring up-to-date the learning rate in an adaptive manner[23]. AdaGrad is mostly used in Natural

Language Processing (NLP). In training large scale of neural nets, AdaGrad can be very useful [24]. AdaGrad is represented mathematically as:

$$\theta t+1,i=\theta t,i-\frac{y}{\sqrt{Gt,ii+s}}gt,i \quad \textbf{(2)}$$

Where θ is model weight, G is the updated element, η is the learning rate ε is stability constant, i and ii in the equation represent the index of the considered parameter at time t. the slope of loss function with respect to the index i is represented as gt, i.

Root Mean Square Propagation

Root Mean Square Propagation (RMSProp) is an advancement to AdaGrad Algorithm. It was proposed to overcome the shortcoming of the AdaGrad algorithm by adding decay factor so that the squared gradient cannot only increase but also shrink [24]. RMSProp is widely used in deep learning as it performs well in non-static setting.

$$[g^2](t)=\beta E[g^2](t-1)+(1-\beta)\left(\frac{\eth c}{\eth w_{ij}}\right)^2 \quad (3)$$

$$w_{ij}=w_{ij}(t-1)-\frac{y}{\sqrt{E[g^2]^{\delta_{ij}}}} \quad (4)$$

where $E[g^2]$ denote the moving average of squared gradients, 5 represent the learning rate, $\frac{\eth c}{\eth w}$ represent gradient cost function with respect to weight, and β represent moving average parameter. w_{ij} denotes the current parameter at position i, j. the preceding value of the parameter is represented as $w_{ij}(t-1)$.

Adaptive Moment Algorithm

Adaptive Moment Algorithm (Adam) is an integration of RMSProp and Momentum. Adam stores the previous average of decaying gradient squares as Momentum [25]. Adam performs excellently in deep learning. Adam algorithm structure is defined using the following equations:

$$w_{t+1}=w_t-\frac{\alpha_t}{(v_t+s)^{1/2}}*\left[\frac{\eth L}{\eth w}\right]_t \quad (5)$$

$$v_t=\beta v_{t-1}+(1-\beta)*\left[\frac{\eth L}{\eth w_t}\right]^2 \quad (6)$$

In the equations above, w_t symbolize the present value of parameter w, w_{t+1} denotes the restructured value at time t+1, a_t represents the learning rate at time t, v_t represents moving average of squared gradient at time t, s is an additional small constant for numerical constancy, $\frac{\eth L}{\eth w_t}$ denote gradient of loss function with respect to parameter at time t. Q determine the quantity of weight given to past parameter. It is usually between the range of 0 and 1.

iii. Second-order Algorithm

Second-order algorithm is an improvement of the first-order algorithm. It provides mechanism to reduce the training repetition process in order to cut down the time consumption of the algorithm and reduce the need for hyper-parameter tuning in neural network [26]. There are several approaches in second-order optimization algorithm. The most simplest and common approach is the Newton's approach [27]. Other approaches are Conjugate Gradient Method, Quasi-Newton Method, Gauss-Newton Method, etc. second-order algorithm could be single variable or multi-variable. The equation below represents the multi-variable of second-order algorithm, where:

$$f(X)=f(x_1,x_{2,\ldots}x_n) \quad (7)$$

$$Af(X)=\left[\frac{6f(x)}{6x_1},\frac{6f(x)}{6x_2}\ldots\frac{6f(x)}{6x_n}\right] \quad (8)$$

f(x) represent function with input of *X* with n components, Af(X) represents the gradient of the function *f(X)* with respect to the vector $X.\frac{6f^{(x)}}{6x_n}$ represents the partial derivative of the function f with respect to the nth component x_n of *X*.

1. Non-differentiable Objective Function

Complex problem cannot be optimized using differentiable objective function. Non-differentiable objective function can be used to optimize problem that are complex [28], that is, where the problem does not have derivative. Non-differentiable objective function algorithms include direct algorithm, stochastic algorithm, and Population algorithm.

i. Direct Algorithm

Direct algorithm which is also referred to as black-box optimization [29]. It stands as substitute to Gradient-based methods. Direct algorithm is best suited to solve problem which is neither too complex nor too simple [30], but between range of complex and simple project. Cyclic Coordinate Search, particle Swarm optimization, genetic algorithm etc. are all examples of Direct Algorithm.

$$x_{\{k+1\}} = x_k - H^{\{-1\}}(x_k)\, Af(x_k) \quad (9)$$

Where $x_{\{k+1\}}$ represent the Hessian matrix at point x_k, $Af(x_k)$ denotes the gradient of the objective function f(x). It is usually the second derivative function.

ii. Stochastic Algorithm

Stochastic Algorithm is an optimization algorithm for solving complex problem. It traverses the search space in a randomized manner, in contrast to deterministic approach that uses a fixed value. Stochastic Algorithm uses a comprehensive [31] method that can easily be understood. Stochastic algorithm are mostly useful in Natural language processing (NLP), Computer Vision etc. [32]. Examples of Stochastic Algorithm are Cross-entropy and evolution strategy.

$$f(x) \rightarrow E[f\gamma(x)] \quad (10)$$

Where E represents the expectation over the random variable γ and fγ(x) is a function of x and.

iii. Population-Based Algorithm

Population-based algorithm is based on collection of candidate solutions. These solutions maintained by population-based algorithm is mostly denoted as population or swarm. The optimal solution is achieved through iteratively traversing through the swarm. This algorithm uses the basis of social behavior [33].

$$v_i(t+1) = w * v_i(t) + c1 * rand() * (pbest_i - x_i(t)) + c2 * rand() * (gbest - x_i(t)) \quad (11)$$

$$x_i(t+1) = x_i(t) + v_i(t+1) \quad (12)$$

where, w is the inertia weight, c1 and c2 are acceleration constants, vi(t) represents the velocity of particle i at iteration t, xi(t) represents the position of particle i at iteration t, pbesti represents the personal best position found by particle i, and gbest represents the global best position found by any particle in the swarm. A random number between 0 and 1 is produced using rand ().

B. Hyper Parameter Tuning

Hyperparameter is a substantial factor in machine learning and deep learning. It controls performance, configuration and function of a model. The efficiency of training a model directly depends on the choice of hyperparameter. Hyperparameters are usually gotten before the training process. Some examples of

hyperparameter are learning rate of training neural network, the k in k-nearest neighbour, and the L1 or L2 regularization in logistic regression.

Hyperparameter tuning also known as hyperparameter optimization is an imperative process in machine learning that involve the process of amending model performance in order to achieve an optimal output [34]. Hyperparameter tuning is very important in model building especially when the model's objective function is expensive to define. Hyperparameter tuning contribute to the complexity and time consumption of building an well-organized machine learning model [35], therefore it requires careful selection. The strength required to carry out hyperparameter tuning rely on the weight of the neural network [36].

Several methods are employed to carry out the process of optimizing hyperparameter. Some of the approaches are random search, grid search, Bayesian optimization, tree structure-parzen estimator.

1. Grid Search

Grid search is one of the earliest methods in performing hyperparameter tuning. It involve thorough searching through manually defined subset of the learning algorithm space [37]. Grid search construct a network of possible hyperparameter and iteratively combine hyperparameters in a specified order. The performance is recorded for each iteration. Then the best model with the optimal hyperparameter. Grid search consumes a lot of time since it searches through all the possible hyperparameters in the solution space. Figure 2 gives the diagrammatic representation of grid search. When dealing with model with less dimension, grid search tends to be more reliable [43].

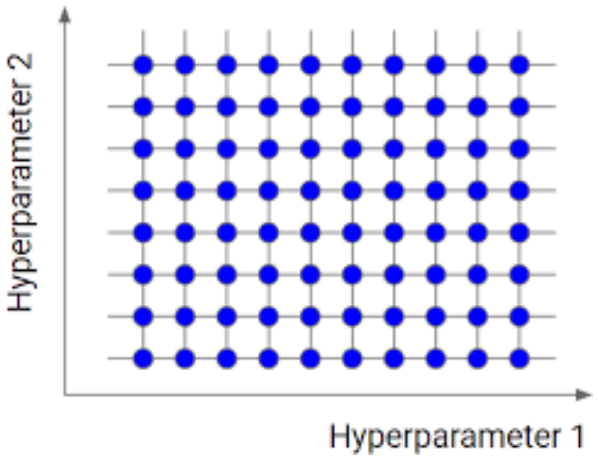

Figure 2: Grid Search Method [75]

2. Random Search

Grid search consumes more time due to the fact that it performs exhaustive search through the solution space. Random search reduces the time consumption by searching through arbitrarily selected point of the solution space. The combination of point that result at the optimal solution will be finally selected. Random search is not always reliable in achieving the optimal solution [34]. Figure 3 represent random search method in a diagrammatic form.

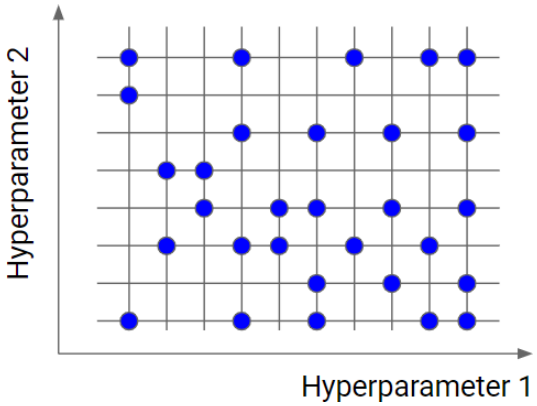

Figure 3: Random Search Method [37]

3. Bayesian Optimization

Bayesian optimization is used in tuning hyperparameter of objective functions that are complex, costly to evaluate, and noisy. It applies the theory of Bayes to search for optimal objective function, either minimum or maximum. It is mostly used in machine learning and deep learning to maximize the objective function such as getting the efficiency of industrial process or getting the efficiency of Deep Learning model [38]. This method best suit problem with multiple objective optimizations, due to the fact that it can search through complex models**.**

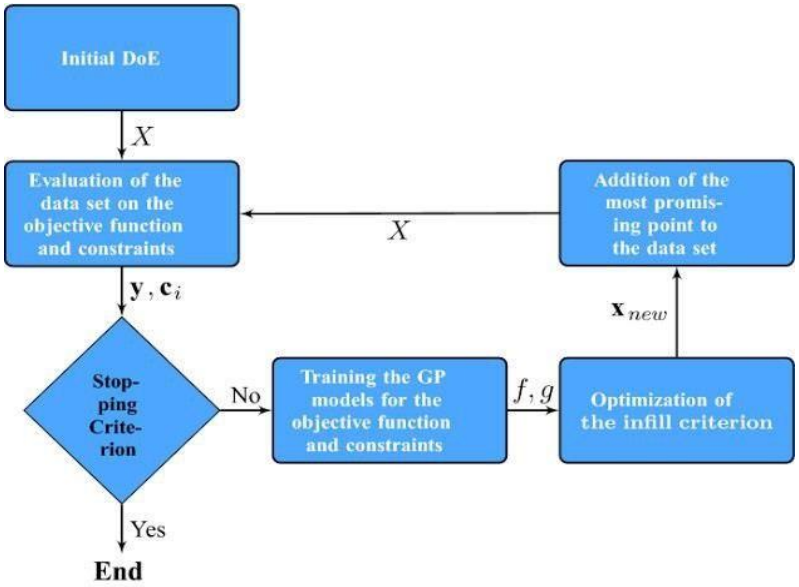

Figure 4: Process model for Bayesian optimization Algorithm [39]

4. Tree-Structured Parzen Estimator

Tree-structured Parzen Estimator (TPE) is a variant of Bayesian Optimization that create a tee like structure to guide through the search of optimal hyperparameter. The process in TPE is an iterative development that utilizes historical data of estimated hyperparameters to suggest subsequent set of hyperparameters [40]. TPE is widely used due to its flexibility and firm performance [41]. TPE make use of alternative model to determine the next configuration likewise Acquisition function (AF). The only drawback of TPE is that it does not accommodate interaction between hyperparameters. Figure 5 represents the flowchart for TPE process. Some of the advantages of using TPE are:

i. It requires lesser time to perform tuning
ii. It is error-tolerant and has potential for improvement
iii. TPE can accommodate extensive variation of variables in parameter search.

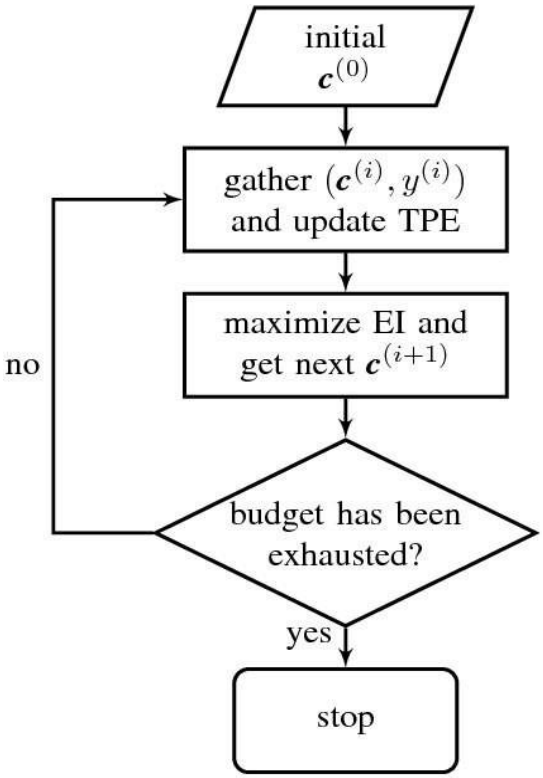

Figure 5: Flowchart for Tree-Structured Parzen Estimator Process [42]

3. Methodology

Building a RNN model for behavioural biometric authentication and obtaining a performance employed a traditional optimizer called Adaptive Momentum Estimation (Adam) and Bayesian algorithm as the optimizers so as to estimate the impact of optimization in model inferencing. The study develops a model to prevent social engineering attack which, utilized optimized Recurrent Neural Network. The model begins with importing the keylogging dataset for data pre-processing and feature extraction. RNN algorithm was used to build the Keystroke Behavioural Biometric (KBB) model. Hyperparameter tuning was done to achieve optimal results. Figure 6 describes the KBB authentication model for preventing social engineering attacks emphasizing the importance of optimization. This significance of optimization was assess using Adam and Bayesian techniques. This study begins by processing the keystroke dataset, extracting the essence of users' behaviour, building RNN model and obtaining the performance evaluation metrics results with the optimization techniques. The essence of incorporating and evaluating the best optimization technique is to come up with an effective and accurate model for behavioural biometric authentication.

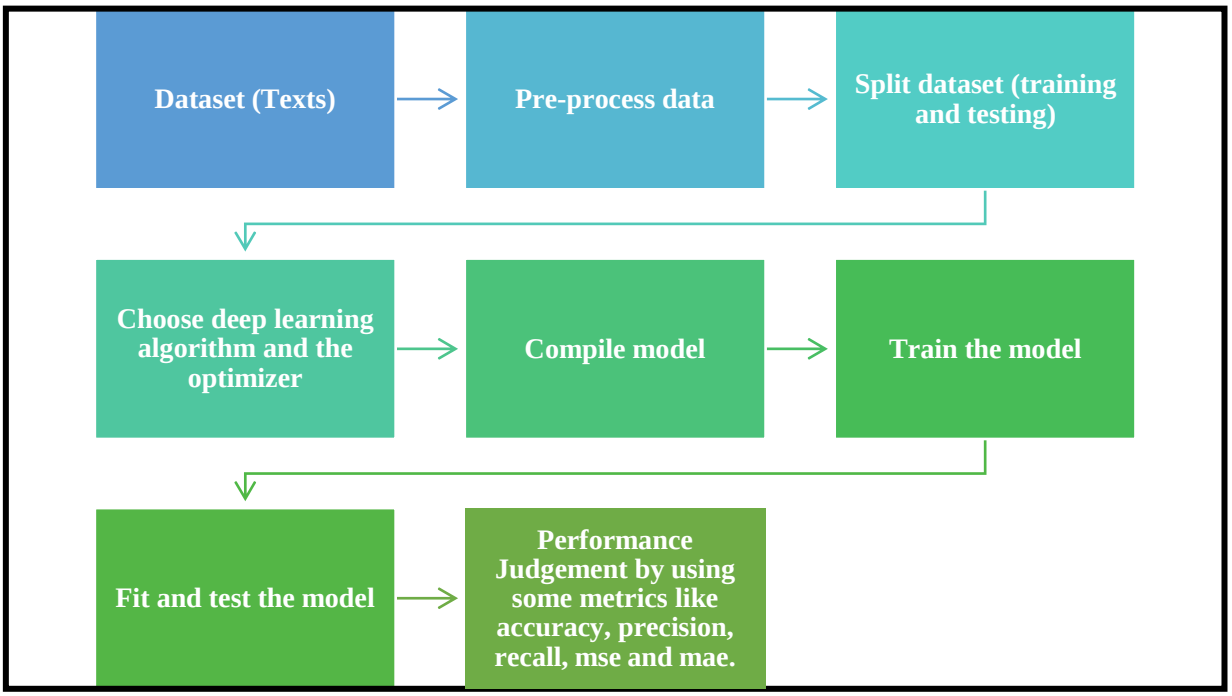

Figure 6: Sequence Diagram for Building Deep Learning Model

A. Dataset Collection

The keylogging dataset was downloaded from UNSW-NB Cyber Range Laboratory of the Australian Centre for Cybersecurity (ACC). The dataset contains 1633 instances. The total number of normal records in the dataset is 1469. The dataset also contains 164 keylogging records and 24 number of columns. Table 1 gives the summary of dataset information.

Table 1: Keylogging Data Description

S/N	Data	Values
1	Number of columns	24
2	Number of normal records	1,469
3	Number of keylogging records	164
4	Total number of records	1,633

B. Data Pre-processing

In order to ensure building of effective model for the classification, the dataset was cleaned to remove irrelevant or noisy data points. Missing value was handled on the dataset by filling the necessary fields using the median of the dataset. Other preprocessing activities include scaling, and encoding.

Scaling

Scaling was used to transform the numerical data into specific range so that the algorithm can work effectively with the data. The study utilizes MinMax Scaler approach to scale the data. The equation for min-max scaling is given in 13.

$$X_{scaled} = \frac{(K - K_{min})}{(K_{max} - K_{min})} \quad (13)$$

Missing Value Handling

Handling missing value is very important in data pre-processing. The data is checked to ensure that all the values are complete for each instance of the dataset. The data is checked to ensure there are no error in the data entries, empty field. The structure of the dataset is observed to get idea of the values, giving the idea of the proportion of the missing value. Then, the value of unimportant columns are dropped.

Encoding

This process involves transforming the character field in the dataset to corresponding numerical value so that the deep learning model can carry out effective classification. The protocol names such as TCP, UDP, ARP, IGMP, and RARP in the data set are converted to corresponding numerical values.

C. Feature Extraction

The next step after the pre-processing is the extraction of plausible features from the dataset. Dimensionality reduction approach was used to extract the features from the dataset. This study utilizes the Principal Component Analysis technique to perform dimensionality reduction on the data set. PCA transform high dimensional data into low dimensional data, at the same time preserving the most important features of the data. The formula for PCA is given thus.

$$\mathbf{Cov(X,Y)} = \frac{1}{n}\sum_{i=1}^{n}(x - x')(y - y') \qquad (14)$$

D. Recurrent Neural Network (RNN)

Recurrent Neural Networks (RNNs) stand as enchanting pillar in the realm of deep learning, offering a unique approach to process chronological data. Unlike traditional feed forward neural networks, RNNs possess a fascinating ability to retain information from previous time steps, making them adept at tasks involving sequences. At the heart of an RNN lies its recurrent nature, where each hidden layer not only receives input from the current time step but also from its own output in the previous time step. This intricate feedback loop allows RNNs to capture dependencies and patterns that extend across time, creating a sense of memory that empowers them to perceive context and context changes dynamically. The formula for RNN is illustrated in 13.

$$h=\sigma(UX+Wh\_(-1)+B \qquad (15)$$

$$Y=O(Vh+C) \qquad (16)$$

RNN is used in this study based on its ability to operates and work well with sequences or time series dataset which is the type of dataset used in this study. The dataset was split into training and testing in the model design phase and some parameters such as dense, activation, batch size, sigmoid, filament, input size and epoch were used for building the Recurrent Neural Network. The RNN model is used to generate the performance metrics for evaluation such as log loss, accuracy, and recall, precision, MAE.

E. Hyper parameter Tuning

The major focus of this study is to perform hyperparameter optimization to improve the model in order to yield optimal solution. ADAM and Bayesian optimizers are used and compared in order to reinforce the significance of optimization in model building.

Adam optimizer is one of the traditional optimizers used in deep learning framework. It is an efficient optimization method that adjusts the learning rates of each parameter in the model, leading to faster convergence and better performance. The equation for Adam optimizer is given thus.

$$w_{t+1} = w_t - \frac{\alpha_t}{(v_t+s)^{1/2}} * \left[\frac{\partial L}{\partial w}\right]_t \qquad (17)$$

$$v_t = \beta v_{t-1} + (1-\beta) * \left[\frac{\partial L}{\partial w_t}\right]^2 \qquad (18)$$

In the equations above, w_t symbolize the present value of parameter w, w_{t+1} denotes the restructured value at time t+1, a_t represents the learning rate at time t, v_t represents moving average of squared gradient at time t, s is an additional small constants for numerical constancy, $\frac{\partial L}{\partial w_t}$ denote gradient of loss function with

respect to parameter at time t. Q determine the quantity of weight given to past parameter. It is usually between the range of 0 and 1.

Bayesian optimization method was also adopted and incorporated for the hyperparameter optimization process in this study due to its ability to modify each epoch's weights during deep learning model optimizers training as well as it brought about minimization of loss function. This algorithm is used to adjusts neural network attributes for example; learning rates, number of epochs, number of dense layer units and activation function. Thus, helping in accuracy improvement as well as overall loss reduction. Bayesian optimizer is well appropriate algorithm for classification or regression hyperparameters model optimization. The hyperparameters that were optimized in this study are learning rate, direction, batch size, epoch, and activation.

F. Evaluation

Subsequently, after the model building to generate the performance metrics. The metrics that are used to evaluate the model include accuracy, precision, recall, and F1-score. Costs metrics such as MAE, MSE as well as RMSE is used in order to know the cost or error value associated with the algorithm used.

4. Results and Discussion

A. Dimensionality Reduction

Dimensionality reduction is done on the dataset using principal component analysis (PCA). PCA is used on the dataset to rank the columns present in the dataset based on their important and influence on the deep learning model performance. From the analysis, 15 most important column were selected based on PCA results to carry out the modeling operation. Table 2 shows the PCA results on the all columns.

Table 2: PCA Results on the 23 Columns in the Dataset

S/N	Column Name	PCA Value
1.	Sport	8.984483636
2.	Stime	5.601938036
3.	Bytes	5.500149479
4.	Pkts	4.904953085
5.	Seq	3.42865005
6.	Ltime	1.251295417
7.	Dport	1.021077818
8.	Mean	0.91161932
9.	Dpkts	0.387231068
10.	Sbytes	0.303931881
11.	Max	0.141405754
12.	Dbytes	0.015518912
13.	Srate	2.01409E-15
14.	Drate	-4.23652E-16
15.	Rate	-0.001631127
16.	Spkts	-0.35145568
17.	Sum	-0.760994446
18.	State	-0.851771308
19.	Proto	-1.415025276
20.	Dur	-1.521098736
21.	Flgs	-2.415818592
22.	Stddev	-3.647112133
23.	Min	-3.954203717

From the PCA result shown in Table 2 above, 15 most important columns based on the ranking result is used to carry out deep learning operations in this study. The most important fifteen columns are shown in Table 3

Table 3: The Most Important 15 Columns

S/N	Columns	PCA Results
1.	Sport	8.984483636
2.	Stime	5.601938036
3.	Bytes	5.500149479
4.	Pkts	4.904953085
5.	Seq	3.42865005
6.	Ltime	1.251295417
7.	Dport	1.021077818
8.	Mean	0.91161932
9.	Dpkts	0.387231068
10.	sbytes	0.303931881
11.	Max	0.141405754
12.	dbytes	0.015518912
13.	Srate	2.01409E-15
14.	drate	-4.23652E-16
15.	Rate	-0.001631127

B. Hyper parameter Tuning Setting

The major contribution of this study is hybridizing Bayesian optimization technique in order to come up with a robust and an accurate authentication system. We used one of the traditional optimizers with RNN before using Bayesian optimizer with it. The traditional optimizer is called ADAM optimizer and its parameter and the corresponding values are set as shown in Table 4. The set of parameters of Bayesian optimizer and the corresponding values as shown in Table 5. These values were set after several trials of different combinations of these values so as to get a setting that will improve the result gotten from RNN parameter optimization by ADAM and Bayesian algorithm.

Table 4: Adam Hyper parameter Tuning Settings

Optimizer	Parameter	Values
Adam	Number of dense layer unit	1
	Dropout	0.5
	Activation function	Sigmoid
	Learning rate	0.01
	Number of epochs	10
	Input shape	15, 1
	Batch size	32

Table 5: Bayesian Hyper parameter Tuning Settings

Optimizer	Parameter	Values
Bayesian Optimizer	Number of dense layer unit	1
	Dropout	0.5
	Activation function	Sigmoid
	Learning rate	0.001
	Number of epochs	10
	Input shape	15, 1
	Batch size	32
	Number of trials	15
	Number of iterations	60

	Minimum Hyperparameter units	96
	Maximum Hyperparameter units	480

C. Recurrent Neural Network (RNN) Using Adam Optimizer

Performance evaluation analysis done on RNN using some machine learning metrics which are log loss, accuracy, precision, recall, RMSE, MAE as well as MSE based on Adam parameter tuning optimization which is given in Table 6 below. From the analysis log loss result is 0.0186, accuracy of 99.69, precision of 0.9966, recall of 1.0000, RMSE of 0.0563, MSE of 0.0032 as well as MAE of 0.0108.

Table 6: RNN Metrics Result based on Adam Optimizer

S/N	Machine learning Metrics	Result
1.	Log loss	0.0186
2.	Accuracy	99.69
3.	Precision	0.9966
4.	Recall	1.0000
5.	RMSE	0.0563
6.	MAE	0.0032
7.	MSE	0.0108

D. Recurrent Neural Network (RNN) Using Bayesian Optimizer

The results of RNN with optimization is presented in Table 7 below give the result gotten from RNN operations where log loss result is 0.0036, accuracy of 100, precision of 1.00, recall of 1.00, RMSE of 0.0006, MSE of 0.0032 as well as MAE of 0.0032.

Table 7: RNN Metrics Result based on Bayesian Optimizer

S/N	Machine learning Metrics	Result
1.	Log loss	0.0036
2.	Accuracy	100
3.	Precision	1.00
4.	Recall	1.00
5.	RMSE	0.0006
6.	MAE	0.0032
7.	MSE	0.0032

Figure 7 below give the graphical representation of training loss and validation loss gotten from 10 epoch iterations where line red denote training loss and the blue line represent validation loss and Figure 8 illustrates training and validation accuracy result with the same number of epochs. The two graphs describe a balanced Bias-Variance tradeoff.

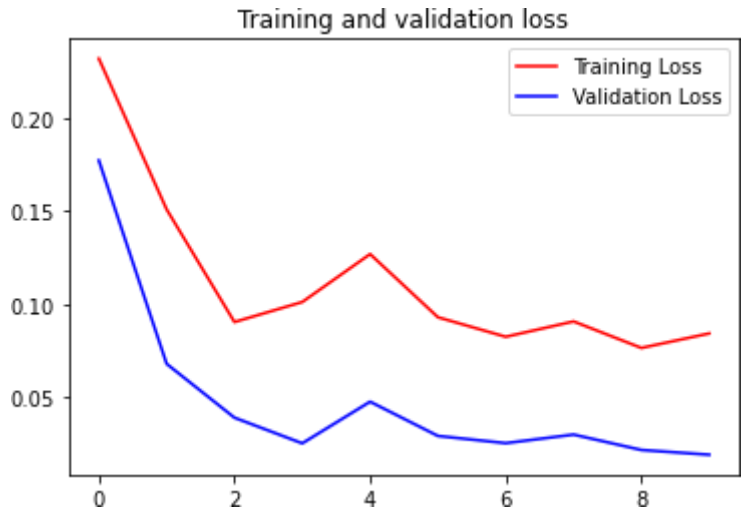

Figure 7: RNN Training and Validation Loss

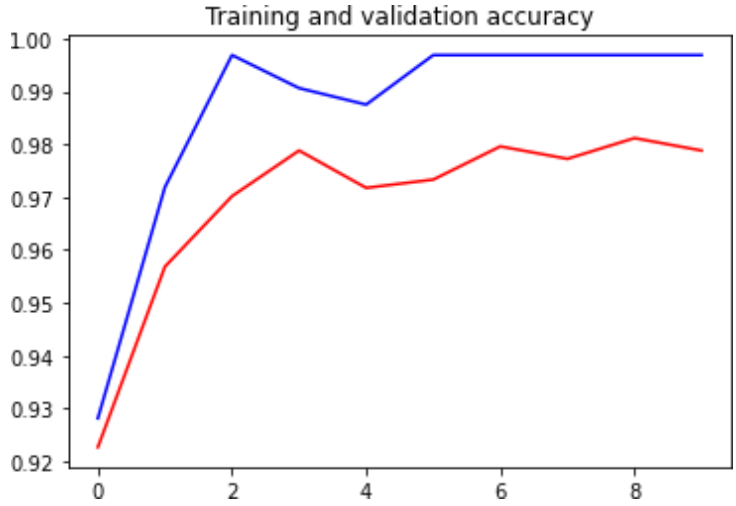

Figure 8: RNN Training and Validation Accuracy

Looking at Figure7 and Figure 8, it is also important to know that the prediction was highly accurate with less error. It shows the results of training and the testing phases. Figure 7 shows that the loss of RNN model during validation is lesser than the loss during the training. And Figure 8 shows that the accuracy of RNN model during validation is higher than the its accuracy during the training. So, the model is highly accurate with Bayesian optimization.

E. Comparative Analysis of RNN Performance Evaluation Metrics Result with and Without Bayesian Optimization

The comparative analysis is done using tables and graphs for the visualization of the difference in the implementation of RNN with two difference optimization techniques. the metrics result. Table 8 shows the comparative analysis showing the various metrics values with ADAM optimization and with Bayesian optimization.

Table 8: RNN Metrics Result with and without Bayesian Optimization

S/N	Machine learning Metrics	With Adam Optimization	With Bayesian Optimization
1.	Log loss	0.0186	0.0036
2.	Accuracy	99.69	100
3.	Precision	0.9966	1.00
4.	Recall	1.0000	1.00

5.	RMSE	0.0563	0.0006
6.	MAE	0.0032	0.0032
7.	MSE	0.0108	0.0032

Figure 9 shows the graphical representation of the comparative analysis of log loss, precision, recall, RMSE, MAE and MSE result while Figure 10 shows the comparison of accuracy result with respect to the two optimization techniques.

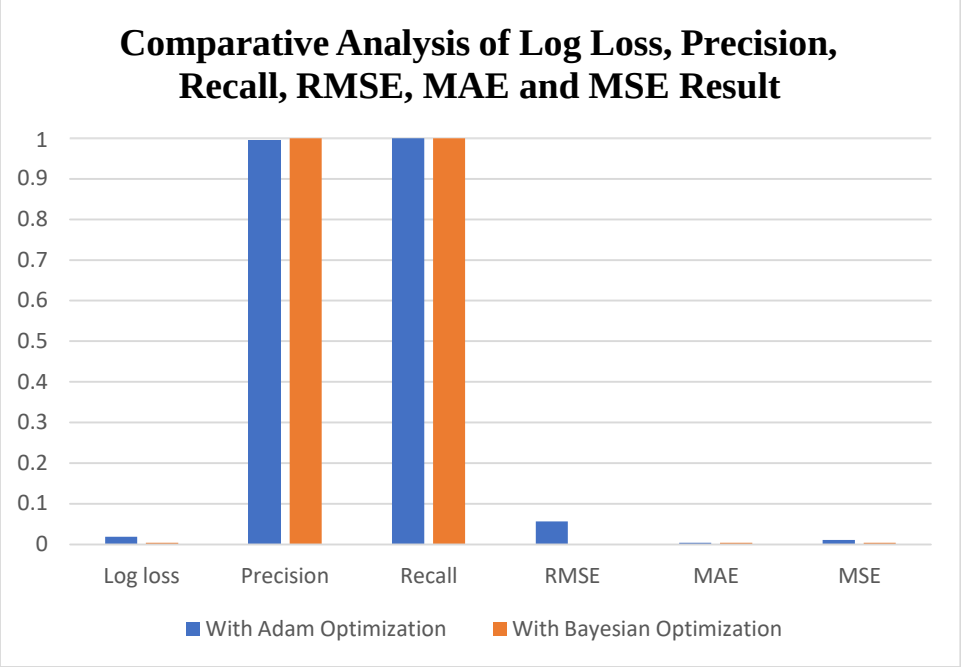

Figure 9: Comparative Analysis for Log Loss, Precision, Recall, RMSE, MAE and MSE Results

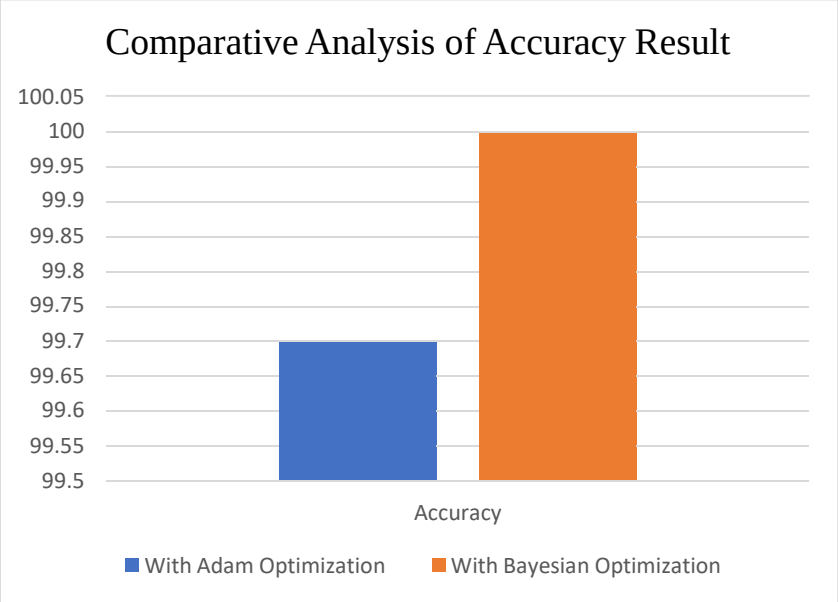

Figure 10: Comparative Analysis for Accuracy Result Comparative Analysis

5. Discussion

By considering the evaluation metrics results, it will be discovered that the choice of RNN algorithm with ADAM optimizer is not a bad idea at all with a considerable high accuracy, precision and recall. Even the metrics showing the errors, that is, the difference between a statistical model's predicted values and the actual values are considerable low. These are Log Loss, Root Mean Square Error, Mean Absolute Error, and Mean Square Error. But looking at the result we have got with Bayesian optimization algorithm, we can see

that there is a great improvement. That is, the accuracy and the precision of the model got increased and the cost metrics got reduced, that is, the measures of errors. The comparison shown justifies the choice of this behavioural authentication framework for mitigating against social engineering attacks.

6. Conclusion

This study has established the importance and significance of optimizing deep learning algorithms especially in addressing complex problems that involves non-linear and noisy data. Appropriate use of an optimization algorithm makes a robust, effective and accurate deep leaning model especially where we used the approach to analyze a keystroke dynamics dataset and predict an attack. Hence, we have developed a behavioural authentication model to mitigate against social engineering attacks.

References

1. A. Henricks and H. Kettani, "On Data Protection Using Multi-Factor Authentication," pp. 1–4, 2019.
2. L. Razaq, T. Ahmad, S. Ibtasam, U. Ramzan, and S. Mare, "Understanding Mobile-based Fraud Through Victims' Experiences," Proc. ACM Human-Computer Interact., vol. 5, no. CSCW1, 2021, doi: 10.1145/3449115.
3. L. Razaq, T. Ahmad, S. Ibtasam, U. Ramzan, and S. Mare, "Understanding Mobile-based Frauds Through Victims' Experience," no. April, 2021, doi: 10.1145/3449115.
4. F. Mouton, M. Malan, L. Leenen, and H. Venter, "Social Engineering Attack Framework," no. August, 2014, doi: 10.1109/ISSA.2014.6950510.
5. T. Eude and C. Chang, "One-class SVM for biometric authentication by," no. February, pp. 1–16, 2017, doi: 10.1111/coin.12122.
6. K. Corpus, R. Joseph, D. Gonzales, L. Vea, and A. Morada, "Mobile User Identification through Authentication using Keystroke Dynamics and Accelerometer Biometrics," pp. 11–12, 2016.
7. H. Gamboa and A. Fred, "A Behavioural Biometric System Based on Human Computer Interaction," Biometric Technol. Hum. Identif., vol. 5404, pp. 381–392, 2004, doi: 10.1117/12.542625.
8. M. Frank, R. Biedert, E. Ma, I. Martinovic, and D. Song, "Touchalytics : On the Applicability of Touchscreen Input as a Behavioral Biometric for Continuous Authentication," pp. 1–20, 2012.
9. B. Schouten, A. Salah, and R. van Kranenburg, "Behavioural Biometrics and Human Identity BT - Second Generation Biometrics: The Ethical, Legal and Social Context," pp. 195–214, 2012, doi: 10.1007/978-94-007-3892-8_9.
10. L. Leonard, Web-Based Behavioral Modeling for Continuous User Authentication (CUA), 1st ed., vol. 105. Elsevier Inc., 2017.
11. E. Ellavarason, R. Guest, and F. Deravi, "Touch-dynamics based Behavioural Biometrics on Mobile Devices – A Review from a Usability and Performance," vol. 53, no. 6, 2020.
12. A. T. Princy and M. K. Preetha, "Active Behavioural Biometric Authentication using CAT Swarm Optimization Variants with Deep Learning," Indian J. Comput. Sci. Eng., vol. 13, no. 3, pp. 653–668, 2022.
13. M. Lansley, N. Polatidis, S. Kapetanakis, K. Amin, G. Samakovitis, and M. Petridis, "Detecting social engineering attacks using case-based reasoning and deep learning," CEUR Workshop Proc., vol. 2567, pp. 39–48, 2019.
14. N. Ryabchuk et al., "Artificial intelligence technologies using in social engineering attacks," CEUR Workshop Proc., vol. 2654, pp. 546–555, 2020.
15. A. Acien, A. Morales, J. V. Monaco, R. Vera-Rodriguez, and J. Fierrez, "TypeNet: Deep Learning Keystroke Biometrics," IEEE Trans. Biometrics, Behav. Identity Sci., vol. 4, no. 1, pp. 57–70, 2022, doi: 10.1109/TBIOM.2021.3112540.

16. V. Gurcinas, J. Dautartas, J. Janulevicius, N. Goranin, and A. Cenys, “A Deep-Learning-Based Approach to Keystroke-Injection,” Electronics, vol. 14, no. 13, pp. 1–29, 2023.
17. S. D. Boyles, “Basic Optimization Concepts What is an optimization problem,” pp. 1–11, 2015.
18. M. Dehghani and P. Trojovský, “A new optimization algorithm based on average and subtraction of the best and worst members of the population for solving various optimization problems,” pp. 1–29, 2022, doi: 10.7717/peerj-cs.910.
19. H. A. Priestley and M. P. Ward, “A multipurpose backtracking algorithm,” J. Symb. Comput., vol. 18, no. 1, pp. 1–40, 1994, doi: 10.1006/jsco.1994.1035.
20. H. H. Tan and K. H. Lim, “Review of second-order optimization techniques in artificial neural networks backpropagation,” IOP Conf. Ser. Mater. Sci. Eng., vol. 495, no. 1, 2019, doi: 10.1088/1757-899X/495/1/012003.
21. J. Bolte, S. Sabach, M. Teboulle, and Y. Vaisbourd, “First order methods beyond convexity and lipschitz gradient continuity with applications to quadratic inverse problems,” SIAM J. Optim., vol. 28, no. 3, pp. 2131–2151, 2018, doi: 10.1137/17M1138558.
22. P. K. Chaurasia, “Gradient Descent Algorithm in Machine Learning.” 2018, [Online]. Available: https://mgcub.ac.in/pdf/material/202004290203577d596f1ec8.pdf.
23. K. Chakrabarti and N. Chopra, “Generalized AdaGrad (G-AdaGrad) and Adam: A State-Space Perspective,” Proc. IEEE Conf. Decis. Control, vol. 2021-Decem, pp. 1496–1501, 2021, doi: 10.1109/CDC45484.2021.9682994.
24. S. Ruder, “An overview of gradient descent optimization,” arxiv, vol. 02, pp. 1–14, 2017.
25. A. Mustapha, “Comparative study of optimization techniques in deep learning : Application in the ophthalmology Comparative study of optimization techniques in deep learning : Application in the ophthalmology field .,” Phys. Conf. Ser. 1743 012002, 2021, doi: 10.1088/1742-6596/1743/1/012002.
26. H. H. Tan and K. H. Lim, “Review of second-order optimization techniques in artificial neural networks backpropagation,” IOP Conf. Ser. Mater. Sci. Eng., vol. 495, no. 1, 2019, doi: 10.1088/1757-899X/495/1/012003.
27. J. Frost and R. Lavatt, “Comparison of Second Order Optimization Algorithms in Neural Networks Applied on Large-Scale Problems,” 2020.
28. M. L. Hanel and C. Schonlieb, “Efficient Global Optimization of Non-Differentiable, Symmetric Objectives for Multi Camera Placement,” IEEE Sens. J., vol. 22, no. 6, pp. 5278–5287, 2022, doi: 10.1109/JSEN.2021.3086037.
29. D. R. Jones and J. A. Martins, “The DIRECT algorithm: 25 years Later,” J. Glob. Optim., vol. 79, no. 3, pp. 521–566, 2021, doi: 10.1007/s10898-020-00952-6.
30. A. Lovison and K. Miettinen, “On the Extension of the DIRECT Algorithm to Multiple Objectives,” J. Glob. Optim., vol. 79, no. 2, pp. 387–412, 2021, doi: 10.1007/s10898-020-00942-8.
31. L. Abualigah, A. Diabat, and R. Zitar, “Orthogonal Learning Rosenbrock’s Direct Rotation with the Gazelle Optimization Algorithm for Global Optimization,” Mathematics, vol. 10, no. 23, pp. 1–42, 2022, doi: 10.3390/math10234509.
32. S. Allassonnière, “Special Issue : Stochastic Algorithms and Their Applications,” pp. 15–16, 2022.
33. J. Sienz, “Population-Based Methods: PARTICLE SWARM OPTIMIZATION-Development of a General-Purpose Optimizer and Applications-PART I,” 2006.
34. R. Hossain and D. Timmer, “Machine Learning Model Optimization with Hyper Parameter Tuning Approach,” vol. 21, no. 2, 2021.
35. Y. A. Ali, E. Awwad, M. Al-Razgan, and A. Maarouf, “Hyperparameter Search for Machine Learning Algorithms for Optimizing the Computational Complexity,” Processes, vol. 11, no. 2, 2023, doi: 10.3390/pr11020349.

36. W. Lichao, G. Perin, and S. Picek, “I Choose You : Automated Hyperparameter Tuning for Deep Learning-based Side-channel Analysis,” pp. 1–23, 2020.

37. D. M. Belete and M. D. Huchaiah, “Grid search in hyperparameter optimization of machine learning models for prediction of HIV/AIDS test results,” Int. J. Comput. Appl., vol. 44, no. 9, pp. 875–886, 2022, doi: 10.1080/1206212X.2021.1974663.

38. V. Hoang Tu, H. Ngoc, and L. Quach, “An Approach to Hyperparameter Tuning in Transfer Learning for Driver Drowsiness Detection Based on Bayesian Optimization and Random Search,” Int. J. Adv. Comput. Sci. Appl., vol. 14, no. 4, pp. 828–837, 2023, doi: 10.14569/IJACSA.2023.0140492.

39. A. Hebbal, L. Brevault, M. Balesdent, E. Talbi, and N. Melab, Bayesian optimization using deep Gaussian processes with applications to aerospace system design, vol. 22, no. 1. Springer US, 2021.

40. S. Watanabe, “Tree-Structured Parzen Estimator: Understanding Its Algorithm Components and Their Roles for Better Empirical Performance,” pp. 1–74, 2023, [Online]. Available: http://arxiv.org/abs/2304.11127.

41. S. Watanabe and F. Hutter, “c-TPE : Generalizing Tree-structured Parzen Estimator with Inequality Constraints for Continuous and Categorical Hyperparameter Optimization,” no. 2014, pp. 1–29, 2022.

42. Y. Xu, W. Gao, F. Qian, and Y. Li, “Potential Analysis of the Attention-Based LSTM Model in Ultra-Short-Term Forecasting of Building HVAC Energy Consumption,” vol. 9, no. August, pp. 1–14, 2021, doi: 10.3389/fenrg.2021.730640.

43. J. Bergstra and Y. Bengio, “Random search for hyper-parameter optimization,” J. Mach. Learn. Res., vol. 13, pp. 281–305, 2012