IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Date of publication xxxx 00, 0000, date of current version xxxx 00, 0000.

Digital Object Identifier 10.1109/ACCESS.2017.DOI

Adaptive Stochastic Conjugate Gradient Optimization for Backpropagation Neural Networks

IBRAHIM ABAKER HASHEM [1], (Member, IEEE), FADELE AYOTUNDE ALABA [2], MUHAMMAD HARUNA JUMARE [2], ASHRAF OSMAN IBRAHIM [3], (Member, Senior IEEE), ANAS W. ABULFARAJ [4]

[1] Computer Science Department, University of Sharjah, Sharjah P.O. Box 27272, United Arab Emirates
[2] Department of Computer Science Education, Federal College of Education, Zaria, Kaduna, Nigeria
[3] Creative Advanced Machine Intelligence Research Centre, Faculty of Computing and Informatics, Universiti Malaysia Sabah, Malaysia
[4] Department of Information Systems, King Abdulaziz University, P.O. Box 344, Rabigh; 21911, Saudi Arabia

Corresponding authors: Ibrahim Abaker Hashem (ihashemf@sharjah.ac.ae)

ABSTRACT

Backpropagation neural networks are commonly utilized to solve complicated issues in various disciplines. However, optimizing their settings remains a significant task. Traditional gradient-based optimization methods, such as stochastic gradient descent (SGD), often exhibit slow convergence and hyperparameter sensitivity. An adaptive stochastic conjugate gradient (ASCG) optimization strategy for backpropagation neural networks is proposed in this research. ASCG combines the advantages of stochastic optimization and conjugate gradient techniques to increase training efficiency and convergence speed. Based on the observed gradients, the algorithm adaptively calculates the learning rate and search direction at each iteration, allowing for quicker convergence and greater generalization. Experimental findings on benchmark datasets show that ASCG optimization outperforms standard optimization techniques regarding convergence time and model performance. The proposed ASCG algorithm provides a viable method for improving the training process of backpropagation neural networks, making them more successful in tackling complicated problems across several domains. As a result, the information for initial seeds formed while the model is being trained grows. The coordinated efforts of ASCG's Conjugate Gradient and ASCG components improve learning and achieve global minima. Our results indicate that our ASCG algorithm achieves 21 percent higher accuracy on the HMT dataset and performs better than existing methods on other datasets(DIR-Lab dataset). The experimentation revealed that the conjugate gradient has an efficiency of 95 percent when utilizing the principal component analysis features, compared to 94 percent when using the correlation heatmap features selection approach with MSE of 0.0678.

INDEX TERMS

Adaptive stochastic conjugate gradient, Backpropagation, Neural networks, Stochastic gradient descent

I. INTRODUCTION

A neural network is a data-processing structure inspired by the neurons in the human brain. Each node has a certain output capability and uses its parameters to determine its output. These characteristics' relative importance allows the node capacity to be fine-tuned [1]. A neural network may be used as a learning tool for reliability modeling because of its performance in comparative design. The network consists of many neurons linked by weighted relationships. The network is primed with test input using an unsupervised learning technique, and its responses are monitored for a fixed amount of time. The training approach is accomplished when the network provides the desired and persuasive responses [2]. Layers of interwoven connections between neuron pieces determine the network's structure, which may be single or several layers. The standard neural network architecture consists of three layers: an active, visible output layer, a hidden, passive input layer, and a dynamic intermediate layer. The weights placed on the secret and output nodes control the neural network's behavior. Artificial neural networks have seen significant advancements in recent years, en-

abling the development of powerful machine-learning models capable of solving complex problems [3]. Backpropagation Neural Networks (BPNNs) are a class of neural networks that have shown remarkable success in various applications. However, training BPNNs involves optimizing many parameters, which can be computationally expensive and time-consuming [4]. To address this challenge, researchers have developed optimization algorithms that aim to improve the convergence speed and accuracy of BPNN training. One such algorithm is Adaptive Stochastic Conjugate Gradient (ASCG) optimization [5]. ASCG optimization combines the benefits of stochastic gradient descent with the efficiency of conjugate gradient methods, making it an effective approach for training BPNNs. The algorithm dynamically adjusts the learning rate and step size based on the gradient information, allowing it to navigate the complex parameter space more efficiently [6]. By adaptively updating the conjugate directions, ASCG avoids being trapped in local minima and accelerates the convergence of the optimization process [7]. The importance of ASCG optimization in training BPNNs lies in its ability to handle large-scale datasets and complex learning tasks. The adaptive nature of the algorithm ensures that it can effectively optimize the neural network parameters, even in the presence of noisy or sparse gradients [8]. This makes ASCG particularly suitable for solving real-world problems that involve nonlinear relationships and intricate patterns. One advantage of ASCG is its ability to handle non-convex and ill-conditioned optimization problems. The adaptive nature of the algorithm allows it to dynamically adjust the learning rate and step size, which helps overcome the challenges associated with complex and noisy data [6]. By incorporating adaptive learning strategies, ASCG ensures that the optimization process remains robust and efficient, even in noisy or sparse gradients. The application of ASCG in training BPNNs has shown promising results in various domains, such as image recognition, natural language processing, and financial prediction [1] [9]. The algorithm's adaptiveness enables it to effectively handle non-linear relationships and complex patterns inherent in these domains. Additionally, ASCG's efficiency makes it suitable for large-scale neural network architectures, where training time and computational resources are critical factors [10]. ASCG optimization is a powerful algorithm for training BPNNs but faces several challenges and limitations. These include sensitivity to initialization, computational complexity, hyperparameter tuning, overfitting and generalization, scalability, and interpretability. Initialization strategies must be carefully developed to ensure optimal convergence and avoid suboptimal solutions [11]. Computational complexity requires calculating gradients and storing historical gradient information, which can be costly and time-consuming when dealing with large-scale datasets and deep neural network architectures. Hyperparameter tuning involves careful tuning of the learning rate, momentum, and regularization parameters, which can be challenging and require extensive experimentation and cross-validation [6]. Overfitting occurs when the model becomes too complex or when the training dataset is insufficient. Scalability is another challenge, as it may face scalability issues when applied to extremely large-scale models with millions or billions of parameters [12]. Interpretability is another challenge, as neural networks trained using ASCG optimization are often considered black-box models, making it difficult to understand and interpret decision-making processes [13]. Addressing these challenges is essential to enhance the effectiveness and applicability of ASCG optimization for BPNNs. This paper delves into ASCG optimization for BPNNs, exploring its underlying principles and discussing its techniques. We examines how ASCG combines stochastic gradient descent with conjugate gradient methods to enhance the training process. Furthermore, we highlight the advantages of ASCG, such as its adaptiveness, efficiency, and ability to handle non-convex and ill-conditioned optimization problems. By understanding the strategic role of ASCG optimization in training BPNNs, we can appreciate its potential to improve the performance and capabilities of neural network models. Through this analysis, we aim to provide insights into the significance and impact of ASCG optimization in neural networks. By leveraging this advanced optimization algorithm, researchers and practitioners can enhance the training process of BPNNs, leading to more accurate and efficient models for a wide range of applications.

The main contributions of this work are summarized as follows:

- An optimized adaptive stochastic conjugate gradient algorithm has been developed to enhance backpropagation neural networks.
- The performance of the proposed method is tested on two benchmarked datasets. Namely Stroke prediction and Diabetes datasets.
- Differences in Mean Squared Error of our model is evaluated against recent and well-known models.
- We evaluated the difference in Mean Squared Error of our model against contemporary and established algorithms.
- Extensive experimentation has been undertaken to showcase the superior performance of our model in comparison to other competitive models.

The remainder of this paper is organized as follows; II provides an overview of related works. III presents a research methodology which involves backpropagation, methods for preparing data, scaling functions, selecting features, and the proposed model. IV highlights the experimental process, along with presenting results and discussions. V concludes the study.

II. RELATED WORK

A wide range of methods is being used in current projects, with results that may be classified as accurate to as low as possible. For example [14] published a paper titled

"Three learning stages and accuracy–efficiency tradeoff of restricted Boltzmann machines," which significantly contributed to the field of deep learning, particularly in training restricted Boltzmann machines (RBMs). The paper provides a comprehensive guide to RBM training, offering valuable insights into theoretical foundations, learning algorithms, and practical considerations. It introduces the contrastive divergence algorithm, simplifying gradient computation, and is widely adopted in RBM training. The paper discusses various preprocessing techniques, such as centering and scaling input data, weight initialization, and tuning hyperparameters. However, the article is limited to RBMs and does not cover other deep-learning models or architectures. It also lacks extensive experimental results and is published in the pre-deep learning era, which may not reflect the latest developments in the field. It is essential to supplement the paper with more recent research to understand training RBMs and their applications in deep learning comprehensively. [15] proposed a "Non-convex Stochastic Bregman Proximal Gradient Method with Application to Deep Learning" that introduces a stochastic conjugate gradient (SCG) algorithm for training large-scale deep belief networks (DBNs). The SCG algorithm combines the benefits of stochastic gradient descent and conjugate gradient methods, enabling faster convergence and better network weight optimization. It addresses computational challenges by reducing gradient evaluations and optimizing memory usage, allowing for more efficient training of deep networks with large layers and parameters. Experimental validation results show improved convergence rates and generalization capabilities. The SCG algorithm is designed for large-scale DBNs, making it suitable for handling complex datasets and models with high parameter counts. However, the paper lacks a comprehensive comparative analysis with a wide range of optimization techniques, focusing on DBNs and not extensively discussing other deep learning architectures. Additionally, the paper does not provide extensive discussions on hyperparameter tuning, which can significantly impact the algorithm's performance. The SCG algorithm significantly contributes by addressing computational challenges and demonstrating improved training efficiency. Further research and analysis are needed to fully understand the algorithm's capabilities, limitations, and applicability to different deep-learning architectures. Furthermore, [16] proposed "Accelerating stochastic sequential quadratic programming for equality constrained optimization using predictive variance reduction." The authors introduce predictive variance reduction (PVR) to accelerate the convergence of stochastic gradient descent (SGD) optimization algorithms. PVR leverages variance reduction to accelerate SGD algorithms, estimating the variance of stochastic gradient updates and adaptively adjusting the learning rate for each parameter, resulting in faster convergence. This method significantly reduces the number of iterations required to converge to a solution compared to traditional SGD algorithms, enabling faster convergence without compromising solution quality. Theoretical analysis of PVR's convergence properties establishes convergence bounds and shows faster convergence rates than standard SGD methods, especially in high-dimensional and sparse data scenarios. Experimental validation shows that PVR consistently outperforms standard SGD algorithms regarding convergence speed without compromising accuracy. Limitations include limited application domains, sensitivity to hyperparameters, complexity, and computational overhead, and the need for further research to explore the applicability of PVR in different environments, optimize hyperparameter settings, and address potential computational overhead concerns. [17] introduce an adaptive conjugate gradient (CG) algorithm specifically designed for deep neural networks. The algorithm leverages the CG method, known for its effectiveness in solving optimization problems, and adapts it to the training of deep neural networks. It aims to accelerate the convergence of training deep neural networks by efficiently utilizing the conjugate gradient direction to update model parameters. Experimental validation results show superior convergence speed and accuracy performance compared to other optimization techniques. The paper focuses on training deep neural networks on moderate-scale benchmark datasets, but its scalability to larger networks is not extensively explored. Further research is needed to evaluate the algorithm's performance and efficiency in handling large-scale networks and big data. Overall, the paper contributes to deep neural network training by introducing an adaptive CG algorithm specifically designed for deep networks. However, the article lacks an in-depth theoretical analysis of the convergence properties and guarantees of the adaptive CG algorithm. The performance of the adaptive CG algorithm is sensitive to hyperparameters, such as learning rate and regularization parameters. Further investigation and experimentation may be required to optimize hyperparameter settings for deep neural networks and datasets. [18] provides an adaptive backpropagation algorithm coupled with conjugate gradient (CG) optimization for training deep neural networks. The research advances deep neural network training by presenting an adaptive backpropagation method paired with CG optimization. During training, the technique adaptively modifies the learning rate and momentum parameters, allowing for improved convergence and optimization of the network weights. The findings reveal that the adaptive backpropagation algorithm with CG optimization delivers better convergence speed and accuracy than standard backpropagation and other optimization approaches. The performance of the suggested method, like that of many optimization algorithms, is sensitive to the choice of hyperparameters, such as learning rate, momentum parameter, and regularisation strength. The algorithm's scalability to bigger and more complicated networks, such as those employed in cutting-edge deep learning models, has not been fully

investigated. However, further study is required to analyze its theoretical features, optimize hyperparameters, and test its scalability to more expansive networks. [19] presents an approach to accelerate deep learning training using adaptive conjugate gradient (ACG) optimization. The paper introduces an ACG optimization algorithm for deep learning training, incorporating adaptive learning rates and conjugate gradient directions to update network weights. The authors present an efficient method for computing the Hessian-vector product, reducing computational complexity and memory requirements for large-scale deep learning models. Experimental validation results show that the ACG algorithm achieves faster convergence and comparable or better performance than other optimization methods like stochastic gradient descent (SGD) and Adam. However, the paper has limitations, including limited analysis of convergence properties, sensitivity to hyperparameters, and limited evaluation of specific deep-learning tasks. Further research is needed to analyze the algorithm's convergence properties, optimize hyperparameters, and evaluate its generalizability across various deep-learning studies and datasets. The paper contributes to deep learning optimization by introducing an ACG optimization algorithm tailored for deep learning training. Several limitations of currently used optimisation techniques have been highlighted by previous research. When the cost or availability of calculating the same gradient prevents using other methods, a stochastic gradient descent approach is often used. However, although using an approximation for the real angle might save computing time for each iteration, it may also slow convergence. The slowest gradient descent approach usually results in the most extensive calculations. While conjugate gradient converges quickly and uses minimal memory for big problems, it scales poorly for length and necessitates a more iterative inline search. The BFGS and LBFGS algorithms are two popular types of Quasi-Newton methods. BFGS and LBFGS take a long to complete a line search, but LBFGS is more efficient with memory management. For deterministic gradient descent, analytic or precise gradients are required. The conjugate gradient and adaptive stochastic gradient techniques work well together because they converge quickly, need few parameters, and require minimal gradient calculation time for each iteration. Stochastic estimates of the conjugate gradient of the cost function are used instead of ASGD's search direction in the main method. The step size of this ASCG approach is approximated using the Lipschitz constant of the gradient function, and it often converges more quickly than ASGD.

[20] re-evaluates the temporal difference (TD) learning algorithm for policy evaluation tasks in reinforcement learning. It notes the sensitivity of TD(0) and TD() to stepsizes, with TD(0) often suffering from slow convergence. Motivated by the connection between TD(0) and stochastic gradient methods, the paper introduces AdaTD(0), an adaptive projected variant of TD(0) with linear function approximation. AdaTD(0) is shown to be less sensitive to stepsizes and has provably convergent properties, with iteration complexity comparable to TD(0). [21] introduces an adaptive heavy ball momentum for gradient-based optimization algorithms, which eliminates the need for tuning a momentum-related hyperparameter. This adaptive momentum is inspired by the optimal choice for quadratic optimization and can improve the performance of stochastic gradient descent (SGD) and Adam. The new momentum leads to algorithms that are more robust to large learning rates, converge faster, and generalize better than baseline methods. The efficiency of SGD and Adam with the adaptive momentum is verified on various machine learning benchmarks, and convergence guarantees are provided. [22] revisits theoretical aspects of adaptive stochastic gradient descent methods widely used for training neural networks in computer vision and pattern recognition. It presents novel findings, including nonergodic convergence results in the general smooth case, wherein the expectation of gradients' norm converges rather than the minimum of past iterates. Additionally, the paper studies the methods' performance under the Polyak-Łojasiewicz property on the objective function, providing nonergodic convergence rates for the expectation of function values. The findings highlight the need for more substantial restrictions on steps to ensure nonergodic function values' convergence rates.

III. METHODOLOGY

This study aims to enhance the ASCG algorithm by optimizing the convergence time of the CG method, thereby reducing the utilization of computational resources. The present study has employed a conventional neural network architecture comprising an input layer, a single hidden layer, and an output layer. The optimization approach known as the quasi-newton method has replaced the more common gradient descent and conjugate gradient optimizers due to its quick convergence to the global minimum and dependability. When training the enhanced ASCG, we will employ the hyperbolic tangent function (tanh) between the input and hidden layer, and the widely-used sigmoid function between the hidden and output layers, as illustrated in equation 1

$$tanh(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}} \tag{1}$$

The following subsection presents the proposed approach of this study.

A. PROPOSED APPROACH

A secondary source was utilized to procure the dataset for this study. The datasets used may be found on Kaggle [23] and "https://www.kaggle.com/code/mathchi/diagnostic-a-patient-has-diabetes", a website hosted by the data-mining platform Kaggle. The first dataset comprises 5,000 instances. The item consists of the conventional set of twelve (12) characteristics, albeit in a partially

refined state, as illustrated in Table 1. The second dataset comprises 769 instances with 9 attributes.

1) Backpropagation

The backpropagation algorithm is a cornerstone of training neural networks and allows them to learn complex patterns and relationships in data [1]. It enables the adjustment of network parameters based on the calculated gradients, leading to improved accuracy and predictive capabilities. With the backpropagation algorithm, neural networks can adapt and optimize their internal representations, making them powerful tools in various domains, including image recognition, natural language processing, and pattern classification [18]. Backpropagation is a widely used method for adjusting weights in neural network models, but it is slow and unreliable due to manual learning rate determination [1]. The conjugate gradient method is proposed to address these issues, which has faster convergence than gradient descent due to its search along the conjugate direction. Neural networks are a type of model that map inputs to outputs using activation functions, such as tanh, sigmoid, or RELU. These networks consist of layers, such as input, hidden, and output layers, where input features are fed, hidden layers represent activation, and output layers perform prediction or classification. The input layer represents input features, the hidden layer represents network activation, and the output layer is the prediction layer. The input layer is fed with connection weights to the next layer, which applies activation functions, then passes to the output layer with their respective connection weights. The output layer is used for prediction or classification, and the error gradient updates the consequences for accurate predictions. Training neural networks involve optimizing weights and biases for every j layer, which involves forward and backward propagation, which requires computing gradients of loss function concerning weights and biases. The training model adopted for this research is backpropagation with a conjugate gradient method algorithm. The steps involved in the backpropagation algorithm are as follows:

1) *Compute the error or loss:* Calculate the difference between the predicted and target outputs using a suitable loss function, such as mean squared error or cross-entropy loss.

$$MSE = \frac{1}{n}\sum_{n}^{i=1}\left(Y_i - \hat{Y_i}\right)^2 \tag{2}$$

2) *Compute the output layer's gradients:* Calculate the error's partial derivatives concerning the activations and weights of the output layer. It involves applying the chain rule to determine how changes in the output layer affect the overall error.As such we apply MSE we calculated in the previous step and the output layer used an activiation function such as sigmoid function. The activiation of n neuron in the output layer is denoted by a_n^k, the input to the activiation function of the n-th neron in the output layer as x_n^k, the weigh between the m-th neuron in the previous layer and the n-th neuron in the output layer as w_{nm}^k. The partital derivative of the a_n^k is shown as:

$$\frac{\partial E}{\partial a_n^k} = a_n^k\left(1 - a_n^k\right) \tag{3}$$

To calcuate the partial of the weight w_{nm}^k, we computed the following formulate.

$$\frac{\partial MSE}{\partial w_{nm}^k} = \frac{\partial MSE}{a_n^k} * \frac{a_n^k}{x_n^k} * \frac{\partial x_n^k}{w_{nm}^k} \tag{4}$$

These derivatives can be computed for each neuron in the output layer and used in gradient-based optimization algorithms such as gradient descent to update the weights of the neural network during training.

3) *Backpropagate the gradients:* Propagate the gradients backward through the network, layer by layer. For each hidden layer, compute the gradients of the error concerning the activations and weights using the gradients from the subsequent layer. This process allows the error gradients to be efficiently propagated back to the input layer.

$$\frac{\partial a_l^k}{x_l^k} = a_l^k\left(1 - a_l^k\right) \tag{5}$$

where l-th the hidden layer of the actviation of k-th neuron. As such the gradient of the error concerning wrights w_{nm}^k where δ_l^k is the gradient of the input error to the activation function x_n^k.

$$\frac{\partial MSE}{w_{nm}^k} = \delta_l^k * a_{nm}^{l-1} \tag{6}$$

4) *Update the weights and biases:* Once the gradients have been computed for all layers, update them using an optimization algorithm, such as stochastic gradient descent (SGD) or its variants. The update rule involves subtracting a fraction of the gradient from the current weights and biases, scaled by a learning rate.
5) *Repeat the process:* Iterate the forward and backward propagation steps for a specified number of epochs or until convergence. Each iteration refines the network's weights and biases, gradually reducing errors and improving the network's performance.

2) Methods for Preparing Data

The dataset utilized in this study has previously been processed, except for a few tweaks, such as the label encoding of the target feature (title stroke) from strings (in the form present and absence) to numbers (1 and 0). Furthermore, several characteristics were removed to examine their impact on the accuracy of predictions.

Table 1: Description of Each Dataset Features.

	Stroke prediction	
#	Attribute	Values
1	Age	Continuous
2	gender	0 – Male, 1 – Female
3	hypertension	0- if the patient doesn't have hypertension, 1- if the patient has hypertension
4	stroke-disease	0 - if the patient doesn't have any stroke diseases, 1- if the patient has a stroke disease
5	ever-married	No or Yes
6	work-type	Private : 1, Self-employed : 2, Govt-job : 3, children : 4, Never-worked 5 :
7	Residence-type	Urban : 1, Rural : 0
8	avgglucose-level	Continuous value
9	BMI	
10	smoking-status	smokes- 1, never smoked- 2, Never-worked -3, formerly smoked -4, Unknown -5
11	stroke	1 if the patient had a stroke or 0 if not
	Diabetes	
1	Pregnancies	Continuous - Number of times pregnant
2	Glucose	Continuous- Plasma glucose concentration a 2 hours in an oral glucose tolerance test
3	BloodPressure	Continuous - Diastolic blood pressure (mm Hg)
4	SkinThickness	Continuous - Triceps skin fold thickness (mm)
5	Insulin	Continuous - 2-Hour serum insulin (mu U/ml)
6	BMI	Continuous - Body mass index (weight in kg/(height in m)^2)
7	DiabetesPedigreeFunction	Continuous
8	avgglucose-level	Continuous
9	BMI	
10	Age	Years
11	Outcome	Class variable (0 or 1)

3) Scaling Functions

Feature scaling, also known as data normalization, is a preprocessing technique used in machine learning to standardize a dataset's range and distribution of features [24]. It aims to bring all features to a similar scale or range, which can benefit certain algorithms and model training processes. Feature scaling is particularly important when the features have different units of measurement or varying magnitudes. The input characteristics were scaled using the Python Scikit learn module, ranging from -1 to 1. The in-built StandardScaler() function provides this unique capability.

4) Selecting Features

Feature selection is a crucial step in machine learning and data analysis, aimed at identifying and selecting the most relevant features from a given dataset. It involves determining the subset of features that contribute the most to the predictive power of a model while reducing redundancy and noise [25]. Feature selection aims to improve model performance, reduce overfitting, enhance interpretability, and reduce computational complexity. Attribute collinearity and percentage of explained variation were considered throughout the heat map correlation and principal component analysis (PCA) phases of reduction and selection.

5) Proposed Model and Neural Network Architecture

An optimisation method developed for the purpose of addressing optimisation issues is the Adaptive Stochastic Conjugate Gradient, or ASCG. Initialization, estimating the gradient norm, deciding on a sample size, creating random samples, calculating the stochastic gradient, changing the search direction, deciding on a step size, updating parameters, updating the iteration counter, and testing convergence are all necessary phases in the process. The algorithm estimates gradients using random samples and adjusts the search direction and step size dynamically. While dealing with objective functions that are both noisy and stochastic, it attempts to discover the best solution. Using stochastic sample gradient estimation and dynamic search direction and step size adaptation, the technique repeatedly updates the parameter vector x to minimise the objective function F(x). The pseudo code of the proposed adaptive stochastic conjugate gradient algorithm is explained in Algorithm 1.

The proposed adaptive stochastic conjugate gradient optimization model in backpropagation neural networks consists of a feedforward neural network architecture with multiple layers [21]. A feedforward neural network (FNN) architecture consists of an input layer, one or more hidden layers, and an output layer. Each layer contains neurons, which are interconnected with weighted connections.The network architecture can be customized based on the specific problem and data characteristics. With the architecture chosen, the network layers need to be defined, specifying the number of layers, types of neurons or filters, activation functions, and other relevant parameters. After defining the network, it is compiled by specifying the loss function, optimizer, and evaluation metrics. The

Algorithm 1 Pseudo code of the the proposed enhanced Adaptive Stochastic Conjugate Gradient (ASCG)

Step1: Initialization
Set x_0 randomly
Set $p_0 = 0$
Set t= p_0
while NOT converged **do**
  Step2:Estimate gradient norm:
  $|| \bigtriangledown F(x_t) ||$
  Determine sample size:
  $N_t = max(N_{min}, \alpha / || \bigtriangledown F(x_t) ||)$
  Draw $N_t i, i_d sample \varepsilon_t, 1, ..., \varepsilon_t, N_t$
  Estimate stochastic gradient: $\bigtriangledown f_t(x_t) = \frac{1}{N_t \times \sum \bigtriangledown f_t(x_t, \varepsilon_t, i)}$
  Step3: Compute search direction:
  $p_t = \bigtriangledown f_t(x_t) + \beta \times p_t - 1$
  Step3:Determine step size:
  $\alpha_t = LearnRate_t$
  Calculate X_1 (Eqs. **??**, **??**)
  Update d_1, d_2
  Update parameters:
  $x_t + 1 = x_t + \alpha x_t \times p_t$
  Update iteration counter:
  t = t + 1
end while

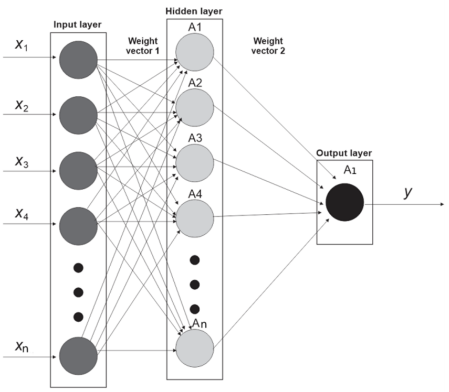

Figure 1: Neural Network's Architecture

model is then trained on the prepared data, adjusting its parameters through techniques our model. Figure 1 illustrate the Neural Network architecture design incorporates a multi-layer perceptron comprising three distinct layers: an input layer, a hidden layer, and an output layer. To further clarify the development of the proposed model, We have designed a flowchart, depicted in Figure 2, to illustrate the main components of the model. The flowchart begins with the preprocessing of the collected datasets. In the training phase, we utilized the Quasi-Newton method and CSG optimization approach. The developed algorithm employs a neural network, which iteratively updates weights, manages activation functions, and computes optimized cost errors.

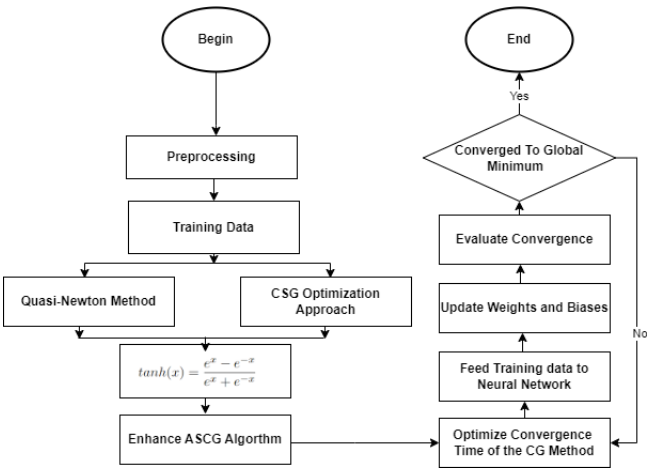

Figure 2: Flowchart Proposed Model

IV. EXPERIMENTATION PROCESS

This experiment employs a real-world dataset to evaluate the proposed enhanced conjugate gradient algorithms. Below is a description of the available datasets. The first dataset (HMT): The neural network model was trained and tested using data from all 5000 instances in the acquired dataset. The dataset was divided into a training dataset and a test dataset in the proportion of 70:30, which indicates that 70 % of the dataset was designated as the training dataset, and 30 % was designated as the test dataset. The correlation heat map method of feature selection was utilized to capture the characteristics of the ultimately employed dataset. Second Dataset (PCA): The neural network model was trained and validated using 5000 instances of the gathered dataset. The dataset was divided into a training dataset and a test dataset with a ratio of 70:30, indicating that 70 % of the dataset was used for training and 30 % for testing. Principal component analysis was performed to pick characteristics from the gathered dataset.

A. RESULT

This section addresses neural model testing and performance assessment. After being trained and verified, the model was evaluated on a test dataset to avoid overfitting or underfitting. To minimize the loss function, the conjugate gradient optimizer was utilized. Exploratory data analysis (EDA) was used to determine characteristics in the model's implementation; EDA used principal component analysis (PCA) and a correlation heat map. A graph showing the decrease in loss as the number of iterations increases.

1) Analysis

(a) We analyzed the HMT dataset, utilizing features selected based on a correlation heatmap. The results of the HMT dataset are provided in Figures 3,4,5 and 6. The figures illustrate the training process of a neural

network model using the Conjugate Gradient Technique on the HMT dataset. This process is tracked over multiple iterations.

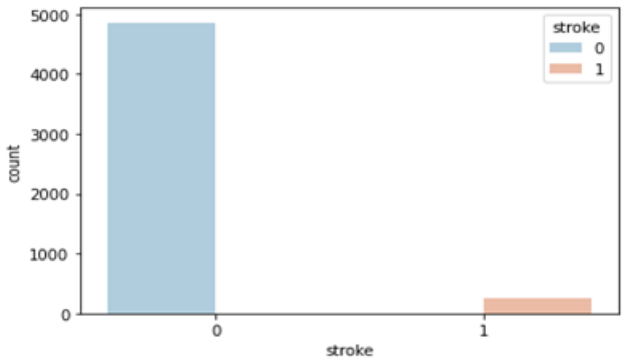

Figure 3: HMT Dataset Histogram Displaying the Distribution of Flaws

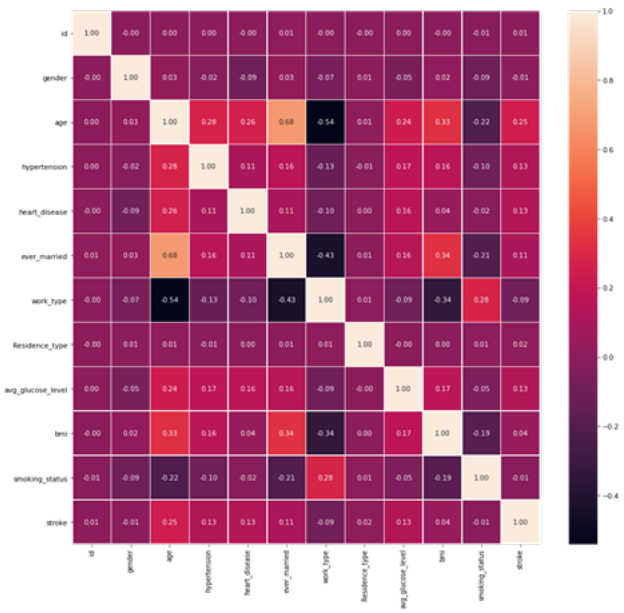

Figure 4: HMT Dataset Heat Map Depicting the Correlational Distribution

For example the blue curve in figure 5 represents the behavior of the cost function (mean squared error / cross-entropy) during training. At the start of training, the cost is relatively high, indicating that the model's initial predictions are far from the true values. As training progresses, the cost function steadily decreases. This decline suggests that the model is learning from the training data and improving its predictions. The curve eventually reaches a minimum point, indicating that the optimization process using the Conjugate Gradient Technique has converged. This is a crucial point because it signifies that the model has found the best possible parameters to minimize the training error. The red curve represents the validation loss, which measures how well the model generalizes to unseen data. Initially, as the model learns from the training data, the validation loss also decreases. This suggests that the model is becoming better at making predictions on data it hasn't seen during training. However, after a certain point, the validation loss may start to increase. Figure 5 demonstrates the trade-off between minimizing the training error (as shown by the cost function) and achieving good generalization (as measured by the validation loss). It emphasizes the importance of monitoring the validation loss to prevent overfitting and select the best model for deployment. The model's training appears to have converged successfully, with a low cost function value and good generalization performance at the minimum of the validation loss curve.

Figure 5: Cost Function Minimization Through the Conjugate Gradient Technique and Validation Loss by Iterative Training on the HMT Dataset

During the initial training phase with the HMT Dataset, the conjugate gradient method was implemented for 10,000 iterations. As a result, the error loss decreased from 0.250334 to 0.04680, while the validation loss decreased from 0.217654 to 0.040375.

Table 2 also presents the result derived from the confusion matrix analysis of the conjugate gradient method.

(b) We provided analysis of a dataset using Principal Component Analysis (PCA) with features selected by the same PCA method. The results of the PCA analysis are provided in Figures 7, 8, and 9. in Figure 7 shows the count ranging from 0 to 5000 which represent the number of iterations or epochs completed during the optimization process. Each iteration involves updating the model parameters based on our training data. In the the strokes represented the update of our in each alteration to minimizes the loss function. The result shows the majorly of our iteration does not require adjustment of the parameters. The main goal of this analysis is to understand the process of stochastic conjugate gradient algorithms during optimization.

Table 2: Result Derived from the Confusion Matrix Analysis of the Conjugate Gradient Method.

Performance Metrics	Quasi-Newton Method
True Positive	4
True Negative	76
False Positive	1506
False Negative	21
Accuracy	95%
Precision	92 %
Recall	95 %

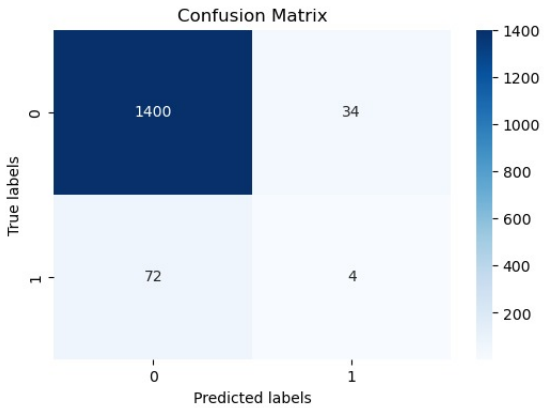

Figure 6: Confusion Matrix for the conjugate gradient Method based on HMT Dataset Experiment

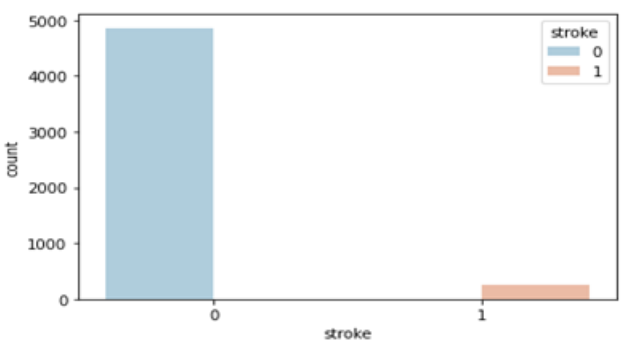

Figure 7: PCA Dataset Histogram Displaying the Distribution of Flaws

Figure 8 illustrates the process of minimizing the cost function using the Conjugate Gradient Technique and validating the loss through Iterative Training on the PCA Dataset. The training loss, represented by the blue line, initially starts at a high value and gradually decreases, converging to a minimum. In contrast, the validation loss, depicted by the orange line, begins at a high point and exhibits more fluctuation throughout the training process. However, it eventually reaches a minimum, albeit slightly higher than the minimum training loss. The gap between the training loss and the validation loss is an indication of how well the model is generalizing. A large gap suggests that the model is overfitting the training data and may not perform well on unseen data.

Figure 8: Cost Function Minimization Through the Conjugate Gradient Technique and Validation Loss by Iterative Training on the PCA Dataset

During the initial training phase with the PCA dataset, the error loss decreased from 0.240334 to 0.0680, while the validation loss decreased from 0.217654 to 0.043375 throughout 10,000 iterations using the conjugate gradient method.

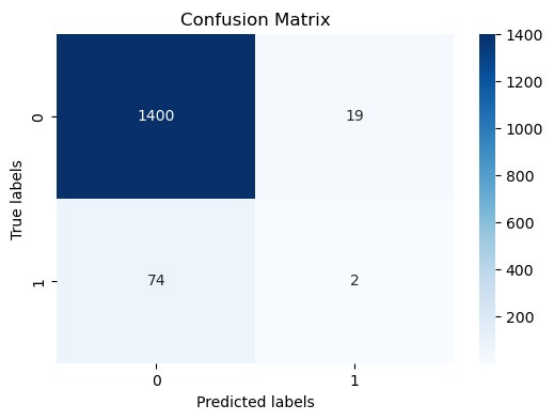

Figure 9: Confusion Matrix for the conjugate gradient Method based on PCA Dataset

In figure 9 we present confusion matrix which is a table used in machine learning to to visualize the performance of our algorithm. It compares the actual labels of the

data to the predicted labels by the model. As shown in the figure our model is preformed well in classifying the training data with a few miss-classification. Table 3 also presents the result derived from the confusion matrix analysis of the conjugate gradient method using the PCA dataset.

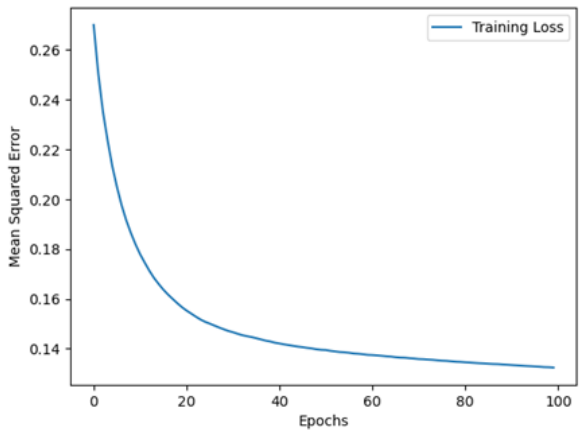

Figure 10: Cost Function Through Conjugate Gradient Technique - Diabetes dataset

We employed the Diabetes dataset to evaluate the model's performance based on Mean Squared Error (MSE). Figure 10 depicts the validation loss decreasing progressively over time, which indicates that the model is learning and improving. Initially positioned at approximately 0.26, the validation loss consistently diminishes throughout the 100 epochs, reaching approximately 0.16 by the end of the training process.

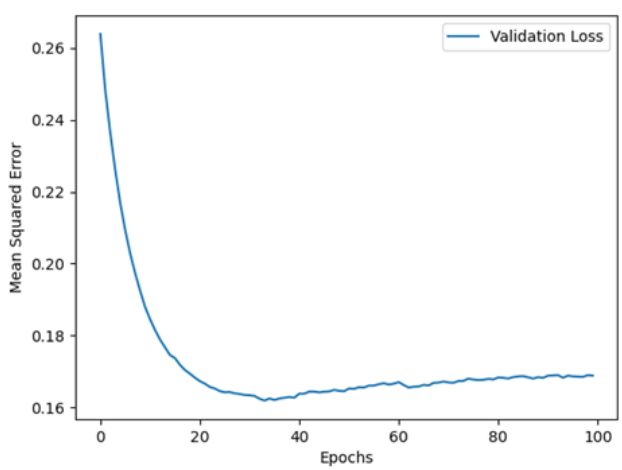

Figure 11: Validation Loss During Iterative Training - Diabetes dataset

Figure 11 shows the validation loss of our model during iterative training on a diabetes dataset. we represents both number of epochs, which are iterations of the training algorithm and the validation loss, which is a measure of how well the model performs on unseen data. the validation loss is decreasing over time, which indicates that the model is learning and improving.

B. DISCUSSIONS

The implementation revealed that the conjugate gradient has an efficiency of 95% when utilizing the principal component analysis features, compared to 94% when using the correlation heatmap features selection approach. According to the findings, the accuracy of the prediction will fall if the number of input characteristics is decreased; that is to say, the larger the number of input features, the greater the possibility of achieving a forecast with a higher level of accuracy will be achieved. Also, the conjugate gradient method algorithm is an effective optimizer compared to the commonly used gradient descent algorithm in some methods employed in earlier works. However, the efficiency of the conjugate gradient method compared with others is lower. This is because the conjugate gradient method algorithm optimizes specific techniques efficiently. The employment of the conjugate gradient technique as an optimizer in the training of neural networks for the prediction of stroke disease was investigated in this study. An accuracy of 94% was achieved using the correlation heatmap features selection approach, and 94% was performed using the principal component analysis and MSE of 0.0678. These results indicated that the conjugate gradient method algorithm is more efficient than the majority of the other methods used in terms of the area of MSE, as shown in Table 4.

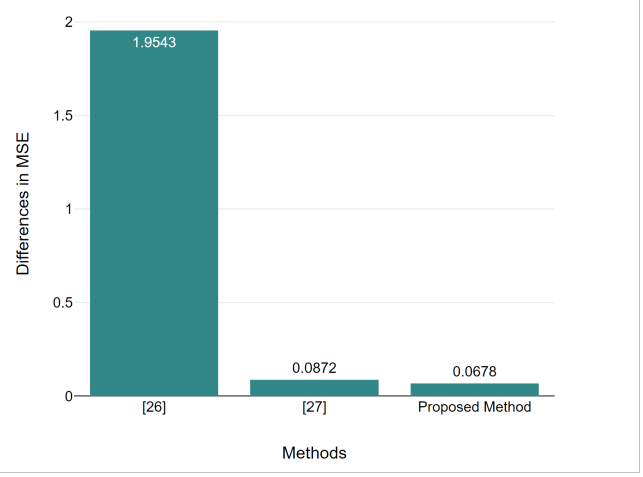

Figure 12: Differences in Mean Squared Error (MSE) values

The study by [26] introduced a new optimization algorithm based on correntropy and conjugate gradient for backpropagation neural networks. They achieved Mean Square error (MSE) of 1.9543 using their proposed method (CCG- PRP) with Lorenz Time Series Prediction. Correntropy measures statistical dependence that can handle outliers and non-linear relationships, making it

Table 3: Result Derived from the Confusion Matrix Analysis of the Conjugate Gradient Method.

Performance Metrics	Quasi-Newton Method
True Positive	5
True Negative	75
False Positive	1431
False Negative	35
Accuracy	94%
Precision	92 %
Recall	94 %
MSE	0.0678

Table 4: Comparison of the Proposed Algorithm with Contemporary Algorithms in a Graph and Table below

Study	Accuracy(%)	MSE	Method
[26]	-	1.9543	New Correntropy-Based Conjugate Gradient Backpropagation Algorithm with Lorenz Time Series Prediction)
[27]	-	0.0872	BR algorithm-trained ANN models using SCG Backpropagation
Our Study	94 %	0.0678	Adaptive Stochastic Conjugate Gradient Optimization for Backpropagation Neural Networks

suitable for optimizing neural networks. [27] conducted a study using ANN models trained with the BR (Bacterial Foraging Optimization with Recurrent Neural Network) algorithm. They achieved Mean Square error (MSE) of 0.0872 with their approach. The foraging behavior of bacteria inspires the BR algorithm and has been applied to solve optimization problems. In this study, it was employed to train the ANN models. As mentioned in the table 4 and figure 12, the present study achieved an impressive accuracy of 94 % and 0.0678 MSE using the proposed Adaptive Stochastic Conjugate Gradient (ASCG) optimization method for backpropagation neural networks. ASCG combines the benefits of adaptive learning rates and the conjugate gradient method to improve neural network training efficiency and convergence.

C. COMPARATIVE ANALYSIS

Based on the accuracy results provided in the table, it can be observed that the present study utilizing ASCG optimization outperformed the other two studies. The achieved accuracy of 94% with MSE 0.0678 indicates the effectiveness of ASCG in enhancing the learning process of backpropagation neural networks. Heravi and Hodtani (2018)'s correntropy-based conjugate gradient algorithm achieved an MSE 1.9543 , while Heng et al. (2022)'s approach utilizing the BR algorithm obtained an MSE of 0.0872. Although these studies demonstrated respectable MSE, the present study's ASCG optimization method exhibited a significantly lowe MSE. It is important to note that the performance of optimization algorithms can vary depending on various factors such as the dataset, problem complexity, network architecture, and hyperparameter settings. Therefore, further analysis and experiments would be necessary to draw definitive conclusions about the relative performance of these methods in different scenarios.

V. CONCLUSIONS AN FURTHER DIRECTIONS

The conjugate gradient approach was offered as a potential optimizer for training neural networks for predicting stroke disease in this body of work. The model that was created and put into action was then assessed using data from Kaggle. The model was trained in two stages. The first used a dataset that included features derived through exploratory data analysis methods such as correlation heatmap features selection and principal component analysis approaches. The algorithm for the conjugate gradient approach was used to train and test the suggested model on more than five thousand (5000) instances of the datasets. The model successfully determined whether or not a patient was suffering from a stroke (classifying the data). Compared to other methods developed by other researchers, the accuracy of our prediction system is comparatively greater by 10–15% in certain instances, but in some cases, it is lower by 2–5%. Accuracy levels of 95% for the correlation heatmap and 93% for the main component analysis were achieved using the quasi-Newton approach and two distinct feature selection strategies. It is evident from the findings that the conjugate gradient technique is effective in training a neural network for predicting stroke illness. Therefore, it is reasonable for us to conclude that the conjugate gradient method is an optimizer with superior capabilities than most other implementations. The research may be further improved by using genetic programming and group methods of data management-based ensemble models to forecast stroke

disease. The ensemble models take into consideration as component models the process of establishing genetic programming and group-based methods of data processing. Moreover we plan to extend our analysis to include a diversity of network structures, ranging from traditional models to state-of-the-art architectures, to provide a more comprehensive evaluation of our proposed method. We will expand our dataset to include diverse scenarios and domains, allowing us to assess the generalizability and robustness of our proposed method across different data distributions and characteristics. We will focus on the analysis by evaluating a wider range of network architectures, encompassing both traditional models and state-of-the-art techniques. These will contribute to a more comprehensive evaluation of our proposed method.

References

[1] Manogaran Madhiarasan, Mohamed Louzazni, et al. Analysis of artificial neural network: architecture, types, and forecasting applications. Journal of Electrical and Computer Engineering, 2022, 2022.

[2] Oludare Isaac Abiodun, Aman Jantan, Abiodun Esther Omolara, Kemi Victoria Dada, Nachaat AbdElatif Mohamed, and Humaira Arshad. State-of-the-art in artificial neural network applications: A survey. Heliyon, 4(11), 2018.

[3] Elham Kariri, Hassen Louati, Ali Louati, and Fatma Masmoudi. Exploring the advancements and future research directions of artificial neural networks: a text mining approach. Applied Sciences, 13(5):3186, 2023.

[4] Junxi Zhang and Shiru Qu. Optimization of backpropagation neural network under the adaptive genetic algorithm. Complexity, 2021:1–9, 2021.

[5] Yu Kobayashi and Hideaki Iiduka. Conjugate-gradient-based adam for stochastic optimization and its application to deep learning. arXiv preprint arXiv:2003.00231, 2020.

[6] Zhuang Yang and Li Ma. Adaptive stochastic gradient descent for large-scale learning problems. 2022.

[7] Debani Prasad Mishra, Kshirod Kumar Rout, Sivkumar Mishra, and Surender Reddy Salkuti. Various object detection algorithms and their comparison. Indonesian Journal of Electrical Engineering and Computer Science, 29(1):330–338, 2023.

[8] Susmita Ray. A quick review of machine learning algorithms. In 2019 International conference on machine learning, big data, cloud and parallel computing (COMITCon), pages 35–39. IEEE, 2019.

[9] Carina Schmidt. Prediction of the influenza virus propagation by using different epidemiological and machine learning models. PhD thesis, Hochschule Heilbronn, 2019.

[10] Laith Alzubaidi, Jinglan Zhang, Amjad J Humaidi, Ayad Al-Dujaili, Ye Duan, Omran Al-Shamma, José Santamaría, Mohammed A Fadhel, Muthana Al-Amidie, and Laith Farhan. Review of deep learning: Concepts, cnn architectures, challenges, applications, future directions. Journal of big Data, 8:1–74, 2021.

[11] Mohammad Mustafa Taye. Understanding of machine learning with deep learning: Architectures, workflow, applications and future directions. Computers, 12(5):91, 2023.

[12] Zhenda Xie, Zheng Zhang, Yue Cao, Yutong Lin, Yixuan Wei, Qi Dai, and Han Hu. On data scaling in masked image modeling. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 10365–10374, 2023.

[13] Iqbal H Sarker. Deep learning: A comprehensive overview on techniques, taxonomy, applications and research directions. sn computer science; 2. epub ahead of print 2021, 2021.

[14] Lennart Dabelow and Masahito Ueda. Three learning stages and accuracy–efficiency tradeoff of restricted boltzmann machines. Nature communications, 13(1):5474, 2022.

[15] Kuangyu Ding, Jingyang Li, and Kim-Chuan Toh. Nonconvex stochastic bregman proximal gradient method with application to deep learning. arXiv preprint arXiv:2306.14522, 2023.

[16] Albert S Berahas, Jiahao Shi, Zihong Yi, and Baoyu Zhou. Accelerating stochastic sequential quadratic programming for equality constrained optimization using predictive variance reduction. Computational Optimization and Applications, pages 1–38, 2023.

[17] Dimitris A Barkas, Stavros D Kaminaris, Konstantinos K Kalkanis, George Ch Ioannidis, and Constantinos S Psomopoulos. Condition assessment of power transformers through dga measurements evaluation using adaptive algorithms and deep learning. Energies, 16(1):54, 2022.

[18] Bhawesh Prasad, Raj Kumar, and Manmohan Singh. Performance analysis of various training algorithms of deep learning based controller. Engineering Research Express, 5(2):025038, 2023.

[19] Erik Schultheis and Rohit Babbar. Speeding-up one-versus-all training for extreme classification via mean-separating initialization. Machine Learning, 111(11):3953–3976, 2022.

[20] Tao Sun, Han Shen, Tianyi Chen, and Dongsheng Li. Adaptive temporal difference learning with linear function approximation. IEEE Transactions on Pattern Analysis and Machine Intelligence, 44(12):8812–8824, 2021.

[21] Tao Sun, Huaming Ling, Zuoqiang Shi, Dongsheng Li, and Bao Wang. Training deep neural networks with adaptive momentum inspired by the quadratic optimization. arXiv preprint arXiv:2110.09057, 2021.

[22] Tao Sun, Linbo Qiao, Qing Liao, and Dongsheng Li. Novel convergence results of adaptive stochastic gradient descents. IEEE Transactions on Image Processing, 30:1044–1056, 2020.

[23] F Soriano. Stroke prediction dataset, 2021.

[24] Yusuf Musa Malgwi, Ibrahim Goni, and Bamanga Mahmud Ahmad. Artificial neural network model for intrusion detection system. Mediterranean Journal of Basic and Applied Sciences (MJBAS) Volume, 6:20–26, 2022.

[25] Mochen Liao, Kai Lan, and Yuan Yao. Sustainability implications of artificial intelligence in the chemical industry: A conceptual framework. Journal of industrial ecology, 26(1):164–182, 2022.

[26] Ahmad Reza Heravi and Ghosheh Abed Hodtani. A new correntropy-based conjugate gradient backpropagation algorithm for improving training in neural networks. IEEE transactions on neural networks and learning systems, 29(12):6252–6263, 2018.

[27] Seah Yi Heng, Wanie M Ridwan, Pavitra Kumar, Ali Najah Ahmed, Chow Ming Fai, Ahmed Hussein Birima, and Ahmed El-Shafie. Artificial neural network model with different backpropagation algorithms and meteorological data for solar radiation prediction. Scientific reports, 12(1):10457, 2022.

Ibrahim Abaker Targio Hashem received his Ph.D. degree in Computer Science from the University of Malaya, Malaysia, in 2017. He received his M.S. degree in computing in 2012, Malaysia, and the B.E. degree in computer science in 2007, Sudan. Hashem obtained professional certificates from CISCO (CCNP, CCNA, and CCNA Security) and APMG Group (PRINCE2 Foundation, ITIL v3 Foundation, and OBASHI Foundation). He worked as a Tutor at CISCO Academy, University of Malaya. His main research interests include big data, cloud computing, distributed computing, and network.

Fadele Ayotunde Alaba works with the Department of Computer Science, Federal College of Education, Zaria. He is researching with several global awards in the IoT and security domain. He received a Bachelor of Computer Science (First Class Honours) from Nasarawa State University, Keffi, Nasarawa State, Nigeria in 2008. He received a Post Graduate Diploma in Education (PGDE) from Usman Danfodio University, Sokoto State, Nigeria 2012. He received a Master of Computer Science from Ahmadu Bello University, Zaria, Kaduna State, Nigeria in 2014. Finally, he received a Ph.D in Computer Science from the University of Malaya, Malaysia (QS WUR- 65). He is an active reviewer of several top journals including Future Generation Computer Systems (FGCS), Information Systems, Journal of Network and Computer Applications (JNCA), IEEE Wireless Communications, and International Journal of Environmental Science and Technology. He received awards including Best Survey Paper Award from Journal of Network and Computer Applications in 2019; Best presenter award at the Annual Faculty of Computer Science and Information Technology Postgraduate Symposium, University of Malaya, Malaysia in 2017. He has published several papers in top conferences and reputable journals. His current research interests include big data analytics, Blockchain, and IoT.

Muhammad Haruna Jumare holds a B.Sc. (Hons) degree in Computer Science from Baze University, Abuja, Nigeria, which he earned in 2016. Currently, he serves as a dedicated Lecturer in the esteemed Department of Computer Science at Federal College of Education, Zaria. With a passion for advancing technology, Jumare's research interests are centred around Machine Learning, Artificial Intelligence, and Software Engineering. His commitment to academic excellence and expertise in these domains contribute significantly to the academic community and the field of computer science.

ASHRAF OSMAN IBRAHIM (Senior Member, IEEE) received a B.Sc. degree in computer science from Al-Neelain University, an M.Sc degree in computer science from the University of Khartoum, and a Ph.D. in computer science from the Universiti Teknologi Malaysia (UTM). Currently, he is a member of the Advanced Machine Intelligence Research Group University Malaysia Sabah. Dr. Ibrahim has extensive experience in supervising and co-supervising postgraduate students, with over 20 postgraduate scholars successfully graduating under his guidance. In addition, Dr. Ibrahim serves as an external and internal examiner/evaluator for Ph.D. and Master's theses at numerous universities. His research interests include applications of intelligence systems, machine learning, artificial neural networks, deep learning in cyber security and different areas, evolutionary computation, data science, big data analytics in different applications, and multi-objective optimization. He is currently serving as a journal reviewer for many reputation journals such as IEEE Transactions on Neural Networks and Learning Systems, IEEE ACCESS, Expert Systems with Applications, Engineering Applications of Artificial Intelligence, Biomedical Signal Processing and Control, and Journal of King Saud University Computer and Information Sciences.

Anas Waleed Abulfaraj: received the B.S. degree in Information systems from King Abdulaziz university, Jeddah, Saudi Arabia and the M.S. and Ph.D. degrees in Human-Computer Interaction from DePaul University, Chicago, USA, in 2012, 2016, and 2021, respectively. Currently, he is an assistant professor at the Department of Information Systems, Faculty of Computing and Information Technology, King Abdulaziz University, Rabigh, Saudi Arabia. His research interests include Human-Computer Interaction, Machine Learning, Artificial Intelligence and Cybersecurity