Hindawi
Security and Communication Networks
Volume 2023, Article ID 9864694, 1 page
https://doi.org/10.1155/2023/9864694

Retraction

Retracted: Bio-Optimization of Deep Learning Network Architectures

Security and Communication Networks

Received 5 December 2023; Accepted 5 December 2023; Published 6 December 2023

This article has been retracted by Hindawi, as publisher, following an investigation undertaken by the publisher [1]. This investigation has uncovered evidence of systematic manipulation of the publication and peer-review process. We cannot, therefore, vouch for the reliability or integrity of this article.

Please note that this notice is intended solely to alert readers that the peer-review process of this article has been compromised.

Wiley and Hindawi regret that the usual quality checks did not identify these issues before publication and have since put additional measures in place to safeguard research integrity.

We wish to credit our Research Integrity and Research Publishing teams and anonymous and named external researchers and research integrity experts for contributing to this investigation.

The corresponding author, as the representative of all authors, has been given the opportunity to register their agreement or disagreement to this retraction. We have kept a record of any response received.

References

[1] P. Shanmugavadivu, M. Mary Shanthi Rani, P. Chitra, and S. Lakshmanan, "Bio-Optimization of Deep Learning Network Architectures," *Security and Communication Networks*, vol. 2022, Article ID 3718340, 11 pages, 2022.

Hindawi
Security and Communication Networks
Volume 2022, Article ID 3718340, 11 pages
https://doi.org/10.1155/2022/3718340

Research Article

Bio-Optimization of Deep Learning Network Architectures

Shanmugavadivu P,[1] Mary Shanthi Rani M ,[1] Chitra P ,[1] Lakshmanan S ,[1] Nagaraja P,[1] and Vignesh U [2]

[1]*Gandhigram Rural Institute, Dindigul, India*
[2]*Information and Communication Technology Department, Manipal Institute of Technology, Manipal Academy of Higher Education, Manipal, Karnataka, India*

Correspondence should be addressed to Vignesh U; u.vignesh@manipal.edu

Received 4 June 2022; Accepted 24 August 2022; Published 20 September 2022

Academic Editor: C. Venkatesan

Deep learning is reaching new heights as a result of its cutting-edge performance in a variety of fields, including computer vision, natural language processing, time series analysis, and healthcare. Deep learning is implemented using batch and stochastic gradient descent methods, as well as a few optimizers; however, this led to subpar model performance. However, there is now a lot of effort being done to improve deep learning's performance using gradient optimization methods. The suggested work analyses convolutional neural networks (CNN) and deep neural networks (DNN) using several cutting-edge optimizers to enhance the performance of architectures. This work uses specific optimizers (SGD, RMSprop, Adam, Adadelta, etc.) to enhance the performance of designs using different types of datasets for result matching. A thorough report on the optimizers' performance across a variety of architectures and datasets finishes the study effort. This research will be helpful to researchers in developing their framework and appropriate architecture optimizers. The proposed work involves eight new optimizers using four CNN and DNN architectures. The experimental results exploit breakthrough results for improving the efficiency of CNN and DNN architectures using various datasets.

1. Introduction

The current technology-driven application focuses on AI-based ways to realize practical problems for varied exercises. Deep learning could be a crucial technology for various applications with extensive information for the process. Deep learning methods imply an optimization fashion for enhancing their performance [1, 2]. CNN is concerned as a category of deep neural networks (DNN), which will fete and cluster-specific components from photos and are loosely utilized for visual activity photos. Their applications vary from image and video acknowledgment, image arrangement, clinical image examination, laptop vision, and traditional language handling [3–5].

Generally, AI-based uses neural configuration to faux the bumps achieving advanced delicacy with bottom time quality. Neural networks are tangled as artificial neural networks or dissembled neural networks. It is also a set of machine accomplishments and the core of deep accomplishment algorithms. The human brain evokes the neural network structure, which will parade the neuron's functions and gestures to at least one another. The architecture of the neural network is shown in Figure 1.

The process of neural networks has some attributes in their methodologies. The crucial options are deduced from the following:

(i) Input: it is the set of options fed into the model for the accomplishment method. For illustration, the input in object discovery may be an array of constituent values concerning a picture.

(ii) Weight: it is the main operation to provide significance to those options contributing to accomplishment. It introduces scalar addition between the input price and also the weight matrix. A negative word would impact the choice of the sentiment analysis model more than a brace of neutral words.

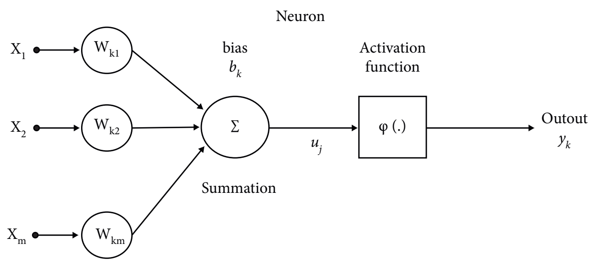

Figure 1: The architecture of the neural network.

(iii) Transfer function: the job of the transfer function is to mix multiple inputs into one affair price, so the activation function may be applied. It is done by accessible information to the transfer function.

(iv) Activation Function: it introduces nonlinearity within the operating of the perceptron to contemplate variable one-dimensionality with the inputs. While not this, the output would be a linear combination of input values and would not be appropriate to introduce nonlinearity within the network.

Deep learning systems are large, complex, and frequently involve numerous layers and nonlinearity, which makes them difficult to optimize. Optimizers must be forced to stir up a complex system that is difficult to understand. Some deep learning systems only provide a small number of parameters that may be modified, which reduces their usefulness. Deep learning models can still be improved and created more easily in some rational ways.

1.1. Overview of Optimizers. Optimizers are techniques or algorithms used to reduce a loss function (error function) or increase production efficiency. Optimizers are mathematical operations that depend on the weights and biases. The features of neural networks, such as weights and learning rate, are modified using optimization algorithms and techniques, which lower the losses that occur during their operation. Typically, optimizers are used to split up optimization tasks by minimizing the function. The weight is initialized using several starting procedures and is optimized each time:

$$W_{\text{new}} = W_{\text{old}} - lr * (\nabla_W L)_{W_{\text{old}}}. \quad (1)$$

The above equation updates the weights to reach the most accurate result. The stylish effect can be achieved using optimization strategies or algorithms called optimizers. Colorful optimizers have been examined with their advantages and disadvantages. The model's literacy parameters, such as weights and impulses, are used to define optimizers, which are defined as fine functions.

1.1.1. Loss Function. The foundation of machine learning algorithms is the loss function. The model assessment system determines whether it is useful for forecasting. The performance of the model is also improved through algorithmic adjustments, and the loss function determines whether this improvement was successful or not. By performing their values, the loss function is utilized to determine the total loss in the dataset.

1.1.2. The Learning Rate. The score of weights by adding and abating too much can hamper the loss function. There is no longer to jump for an optimal value for a given weight. This term is defined as the literacy rate medium. This process can apply to a small number like 0.001 that can multiply the slants by spanning them.

1.1.3. Regularization Process. Experimenters in machine literacy are constantly terrified of overfitting problems. Overfitting occurs when a model performs well on the data used to train it but poorly on fresh data that arise in the actual world. This is only possible if one parameter dominates the formula and is counted excessively. To prevent this, regularization is a phrase that has been introduced to the optimization process. The loss function has an additional component that penalizes high weight values during regularization. If the predictions are accurate, penalties for having accurate predictions with high weight values are obtained. This ensures that weights remain on the lower side, improving their ability to generalize the new data.

1.2. Types of Optimizers. The various fundamental optimizers to reduce the loss function are described as follows.

1.2.1. Gradient Descent (GD) Optimizer. The most fundamental optimizer is gradient descent, which is a smooth process. This could reduce the loss by using the derivatives of the loss operation and learning rate. Once the effective parameters have been shared among all of the different layers, this method will borrow the backpropagation in neural networks. While the gradient is calculated for the dataset, slowing down the algorithm, the weights are effective. To create a resource-empty method, a considerable amount of RAM is required. If this algorithmic rule needs to be adjusted, the overall strategy is found better.

1.2.2. Stochastic Gradient Descent (SGD). A modified interpretation of the GD system where the model parameters are efficient on each replication is called stochastic gradient descent. The loss operation is tested after every coaching sample, proving that the model is effective. These regular updates allow for faster minimum compliance. The model bridge will be created in the necessary place but at the cost of increased variance. The advantage of this approach is that it uses less memory than the previous one because it is not necessary to retain the most recent values of the loss functions. In the convolution setting, SGD-based optimizers that employ various hyperactive parameters are regarded as competitive species similar to the complement of the optimizers.

1.2.3. Minibatch Gradient Descent. Another form of this GD method is known as minibatch, where the model parameters are still useful for tiny batch sizes. To ensure that the model is paced towards minima gradually and to prevent frequent derailments, it is indicated that the model parameters are updated every '*n*' batches. This leads to low variation within the model and decreased memory usage.

1.2.4. Momentum-Based Gradient Descent. The parameters supplied by the first-order outgrowth of the loss function are being updated by backpropagating the system. The number of updates inside the parameters is sometimes overlooked, even though the frequency of updates is frequently replicated for every batch or every time. The term "initiation" in this optimizer refers to the inclusion of this historical component in later updates, which will speed up the overall process.

1.2.5. Nesterov Accelerated Gradient (NAG). The instigation-based largely GD is currently very widespread, down to the lowest levels. The system trials fluctuate, enter the minimum boundary, and add to the total number of times. The next technique is also not up to standard GD. But this problem also needs an exhausted NAG repair. The strategy adopted was to first develop the history component before creating the parameters update. Calculations are made to the outgrowth, which could cause it to advance or regress. This is known as the "look-ahead strategy," and it makes even more sense because the wind is blowing almost at the minimum.

1.2.6. RMSProp. RMSProp frequently enhances the Adagrad optimizer. This optimizer uses an exponential traditional of the slants to reduce the acquisition rate. Acquiring rate reconciliation is still comprehensive because classic can manage various acquisition rates under settings with more small updates and a lesser rate under extremely complex update conditions.

1.2.7. Adam. The RME optimizer combines the RMSprop and instigation-primarily based on GD methodologies. The possibility for stimulus in Adam optimizers to recover the data from history results in balancing acquisition rate gain from the RMSprop. The technique demonstrates the importance of the Adam optimizer. Two hyperactive settings are introduced in this optimizer to fit the use case.

1.2.8. Adagrad-Reconciling Gradient Formula. Adagrad is a reconciling grade optimizer that updates the higher price (high acquisition rates) for parameters with infrequent options and modifies the acquisition rate to a lower price for parameters associated with rush of options circumstances, particularly the justification for dealing with distributed information. Although the intended work is what the model parameters are primarily focused on, they also have an impact on our coaching because they are assigned consistent prices for the duration of the coaching. The learning rate is a similar crucial dynamic component, and varying it may change the tutoring tempo. A complex learning rate for a dispersed purpose input is observed where the maximum of the values is zero to increase the fading gradient acting from these lightweight options.

1.2.9. AdaDelta. By addressing the issues of losing the acquisition rate due to the monotonously increasing add of the court of slants, AdaDelta adheres to the broad interpretation of AdaGrad. AdaDelta compiles the total number of once gradients; however, it only takes a few once slopes into account rather than all angles. Another method, like AdaDelta, to restore AdaGrad's declining learning rate is RMSProp.

1.2.10. Adamax. The resolving movement estimation optimization algorithm has been extended by the Adamax formula. It is an expansion of the gradient descent optimization formula, which is used a lot in astronomy. The formula was defined by Jimmy Lei Ba and Diederik Kingma.

1.2.11. NAdam. The reconciling movement estimation optimization is extended to include Nesterov's accelerated grade (Horse), also known as Nesterov instigation, which is a complex type of momentum.

1.2.12. FTRL. To estimate click-through rates, Google created "Follow the Regularized Leader" (FTRL) in the early 2010s. According to McMahan, the shallow models work better for large dispersed areas.

2. Related Works

This section presents a review of recent works of literature based on the various optimizers and their performance using CNN and DNN architectures.

The authors proposed a frame using the DNN-based optimization strategy for prognosticating the true optimum. The ways are proposed to discover operations in the early stages of aerospace design [6]. In [7], the authors described a random multimodal deep learning (RMDL) which is an ensemble system to break the problem of finding a stylish deep learning structure. Principally, RMDL takes the multiple aimlessly generated model for training using the deep neural network (DNN), convolutional neural network (CNN), and intermittent neural network (INN) for achieving better results. A double algorithm is defined as an optimizer for mongrel anomaly discovery for intrusion discovery evaluation. The exploration work conducts the trial with the anomaly bracket of IDS using DNN [8].

AdaSwarm is a gradient based on outgrowth-based optimization. It implies on functions a differentiable sphere. AdaSwarm includes an exponentially weighted momentum flyspeck swarm optimizer (EMPSO) for making effective analysis [9]. The authors discovered an ATMO (AdapTive Meta Optimizers) which integrates two different optimizers for importing the benefactions and produces the result with a single optimizer [10]. In [11], the authors determined an intertwined model as EO-ELM in a deep neural network using R-R modeling. The efficacy of a model is estimated using query analysis and two-tagged *t*-tests. The authors described the Identifier-Actor-Optimizer (IAO) policy learning armature for applying a real-time optimum control for nonstop-time and nonlinear systems [12]. The authors presented a learning frame using evolutionary-based optimizers using DNN armature with generated samples. In this approach, the authors used a simulation of evolutionary-based combinatorial optimizers [13].

In [14], the authors proposed a system based on optimization analysis using the previous electrode mock two-dimensional (P2D) lithium-ion battery model. The model DeepChess is described for the confluence of optimization, and an inheritable algorithm is included for maximizes the folding of the optimization brace. The design of a combination of DNNOpt using underpinning literacy inspired a deep neural network-based black-box optimization frame for enforcing analog circuit sizing [15]. A population-based evolutionary stochastic gradient descent (ESGD) frame for optimizing deep neural networks. ESGD combines SGD and grade-free evolutionary algorithms as reciprocal algorithms in one frame in which the optimization alternates between the SGD step and elaboration step to ameliorate the average fitness of the population [16]. The authors described the layerwise literacy-based stochastic grade descent system (LLb-SGD) for grade-based optimization of objective functions in deep literacy, which is simple and computationally effective [17].

As the nearest processing unit to the sensors, the authors proposed a deep maker frame that intends to automatically create several mainly reliable DNN infrastructures for eliminating bias [18]. The authors explored the CIFAR-10 datasets hyperparameter hunt approaches using vibrant optimization techniques [19]. The original hunt system combined with the mongrel system of inheritable algorithms optimizes both network architecture and network training. Most reviews and analyses have been performed utilizing studies that standardize the use of DNN infrastructures for bracket and discovery using ML and DL algorithms [20–22]. The authors describe the detection of malaria disease using the CNN technique with SGD, RMSprop, and Adam optimizers [23]. The authors present an analysis of various optimizers on the deep convolutional neural network model in the application of hyperspectral remote sensing image classification [16]. The authors propose the performance analysis of different optimizers for deep learning-based image recognition [22]. The review has been assessed by using various kinds of techniques for CNN and DNN architectures. The existing research works demonstrated the performance according to their selection of optimizers and architectures. The proposed work focuses on CNN and DNN architectures with various kinds of optimizers on a trial-and-error basis [10, 24–27].

The existing research works demonstrated the performance according to their selection of optimizers and architectures. The proposed work focuses on CNN and DNN architectures with various kinds of optimizers on a trial-and-error basis.

3. Methodology

This section presents the ways which are included in this proposed work. This proposed work uses the CNN and DNN architecture using eight new optimizers to accelerate the architecture performance. This study reveals different results using different optimizers. Each optimizer has demonstrated using their dataset and architecture. During the trial, the optimizers were tested with different learning rates for tuning better results.

Optimizers guide modifying the neural network's weights and learning rate to minimize losses. The weights for each epoch are adjusted during deep learning model training and reduce the loss function. An optimizer is a procedure or method that alters neural network properties like weights and learning rates. As a result, it aids in decreasing total loss and raising precision. A deep learning model typically has millions of parameters, making the task of selecting the proper weights for the model challenging. It highlights the importance to select an optimization algorithm that is appropriate for your application. Therefore, before delving deeply into the subject, it is vital to comprehend these algorithms.

Different optimizers are used in the proposed work to adjust your weights and learning rate. The optimal optimizer to use, though, depends on the application. The major limitation is to try every possibility and pick the one that yields the best results. This might not seem like a big deal at first, but when working with hundreds of gigabytes of data, even one epoch can take a while. The proposed CNN and DNN architecture with various optimizers is shown in Figures 2 and 3, respectively.

3.1. Convolutional Neural Network. In CNN, the word "convolution" refers to the fine capability of confusion, a remarkable type of direct action in which two capabilities are duplicated to produce a third capability that communicates the condition of one capability is altered by the other. Two images that can be used as lattices are copied to provide a problem that is used to assess the picture's key features. The basic architecture of CNN is shown in Figure 4. There are primarily two ways to access CNN engineering:

(i) A confusing device known as point birth separates and identifies the picture's brightest components for inspection

(ii) A related subcaste that guesses the class of the picture based on the elements removed in earlier phases using the problem from the complexity cycle

Convolutional layers, pooling layers, and fully associated layers are the three types of layers that make up the CNN. A CNN engineering will take shape once these layers are stacked. In addition to these three levels, the dropout layer and the enactment capability, which are described below, are two other important limits.

3.1.1. Convolutional Layer. The primary layer for removing the various elements from the input images is this

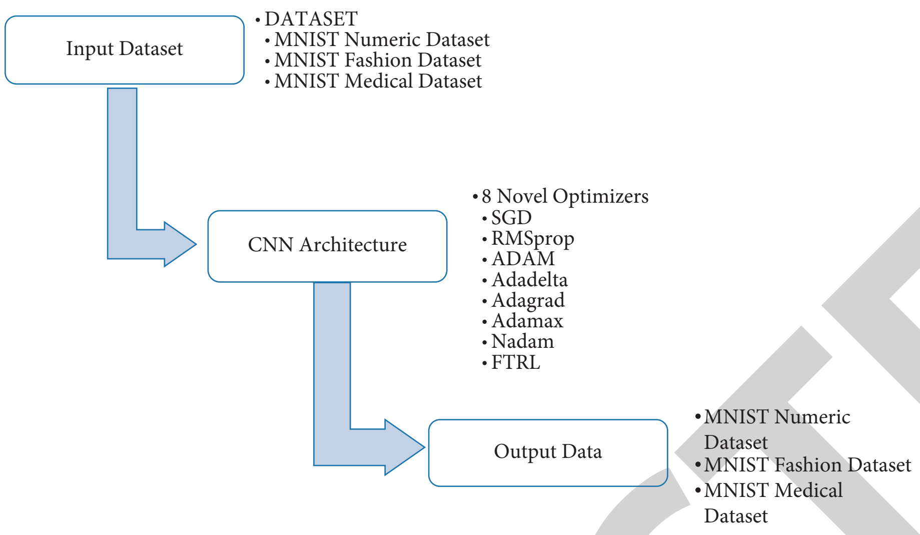

Figure 2: CNN architecture with various optimizers.

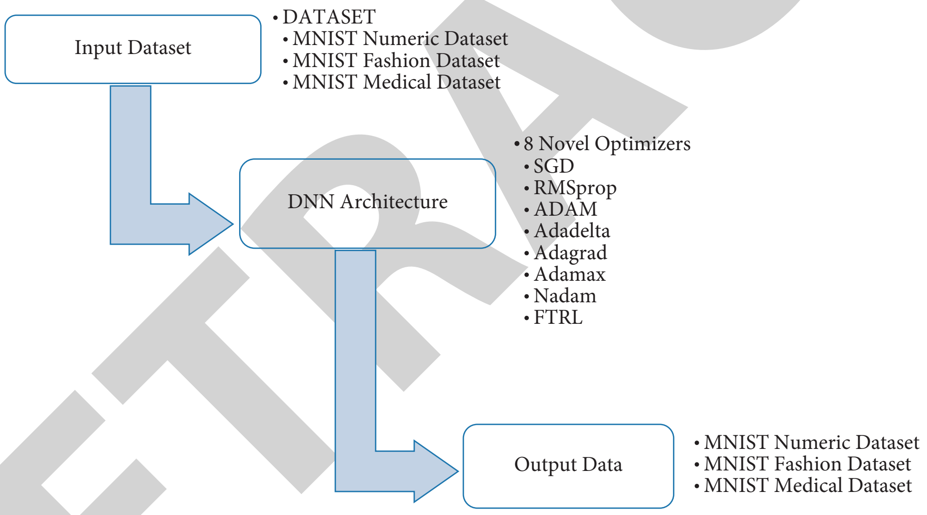

Figure 3: DNN architecture with various optimizers.

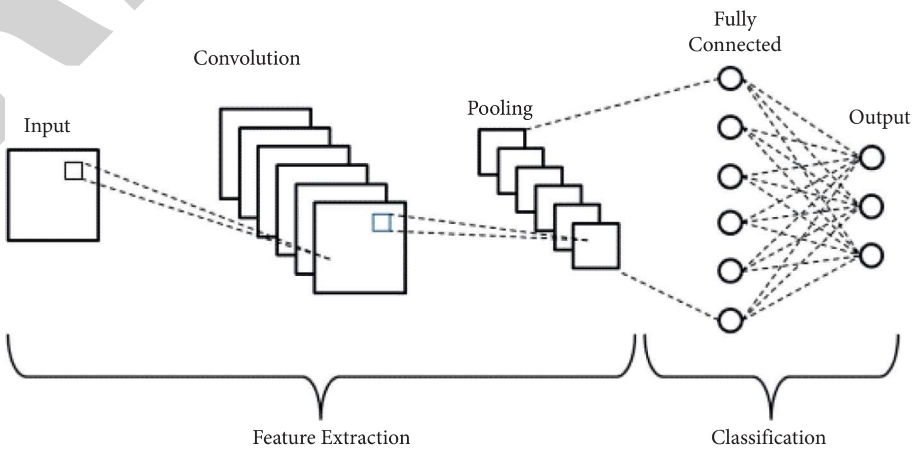

Figure 4: Architecture of CNN.

convolutional layer. This layer involves performing the proper confusion activity between the information image and a channel of a chosen size $M \times M$. The speck item is taken between the medium and the knowledge picture passageway with the muck's dimensions by sliding the pipeline over the knowledge image ($M \times M$).

3.1.2. Pooling Layer. A convolutional layer is typically followed by a pooling layer. The crucial step in this layer is to reduce the convolved direct diagram's size to save on computational costs. This is accomplished by reducing the linkages between layers and working separately on each element map. There are many types of pooling jobs depending on the framework used. The most important element of max pooling is derived from the highlighted map. Common pooling operates outside the bounds of the fundamentals in a measured image segment that has been predefined. Sum pooling figures the total quantity of the essentials in the predefined detail. Most of the time, the pooling subcaste acts as a ground between the convolutional layer and the FC layer.

3.1.3. Fully Connected Layer. The Fully connected (FC) layer connects the neurons between two different layers by combining the loads and incentives with the neurons. These layers often sit before the problem subcaste and help to build the final several layers of a CNN architecture. This smoothes and takes care of the information picture from the preceding layers down to the FC subrank. The smoothed vector also passes through numerous further FC levels, where the majority of the advanced capability jobs take place. The arranging cycle begins to take place at this point.

3.1.4. Dropout. In general, the preparation dataset can be overfitted when every highlight is connected to the FC layer. Overfitting occurs when a certain model performs well on the preparation data but has negative effects on the model's presentation when applied to other details. A dropout layer, which reduces the size of the model by removing a large number of neurons from the brain network during preparation, is employed to solve this problem. Thirty of the knocks are randomly removed from the brain organization after passing a dropout of 0.3.

3.1.5. Activation Functions. The CNN model's actuation capacity represents one of its primary long-term limits. They are used to identify and investigate any kind of ongoing and intricate relationship between organizational constituent parts. In other words, it establishes which model data should be fired in the forward direction. It gives the organization more nonlinearity. There are just a few commonly used initiating capabilities, such as ReLU, softmax, tanH, and sigmoid capabilities. There is a specific activity for each of these abilities. The sigmoid and softmax capabilities are preferred for a CNN model with two groups such as multiclass order and softmax.

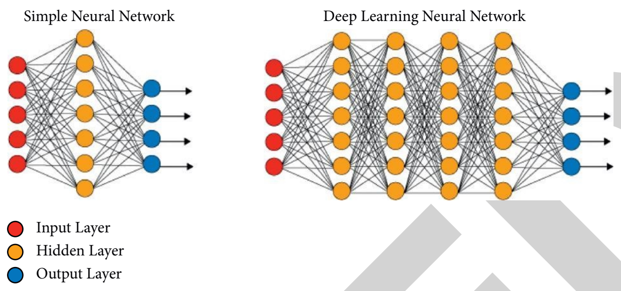

Figure 5: Architecture of simple NN and DNN.

3.2. Deep Neural Network. To integrate AI into the daily activities of self-driving cars, smartphones, games, drones, etc., deep neural networks (DNNs) have emerged as a promising solution. Most often, DNNs were accelerated by a boy with several computing devices, like a GPU, but current technological advancements call for energy-efficient DNN acceleration as the most advanced operations moved down to mobile computing devices. Neural processing unit (NPU) infrastructures focused on accelerating DNN with minimal energy consumption become necessary. Numerous experiments have shown that exercising lower bit perfection is sufficient for a conclusion with minimal power consumption, even if the training phase of DNN demands precise number representations.

DNNs outperform the more traditional ANN with numerous layers in terms of performance. Due to their exceptional ability to learn both the initial structure of the input data vectors as well as the nonlinear input—affair mapping, DNN models are currently becoming rather popular. The majority of DNNs are feed forward networks (FFNNs), in which data go from the input layer to the output layer without going backward 3 and the links between the layers are only ever in the forward direction and never drop a loop again. Through backpropagation, supervised learning is used to complete the tasks using datasets with certain information. The architecture of simple NN and DNN is shown in Figure 5.

3.3. Dataset Used

(i) MNIST is a collection of manually written integers. The collection consists of test photos and training examples. The images' boundaries are grayscale and 28×28 in size. The handwritten numbers on the photos range from 0 to 9, totaling 10 classes.

(ii) Fashion MNIST: a dataset of fashion-connected pictures. The dataset consists of coaching exemplifications and checks images. The reach of the photograph is square measure 28×28 and square measure grayscale. The views contain coaching and check particulars of a jersey, trousers, pullover, dress, coat, sandal, shirt, sneakers, bag, and mortise joint boot has developed the style MNIST dataset.

(iii) Medical MNIST dataset was used to evaluate the performance of the opposing dataset. The topics covered include binary/multiclass, multilabel, and

Table 1: The performance of CNN architecture using the Fashion MNIST dataset.

			CNN architecture			
Dataset	Epochs	Batch size	Optimizers	Training accuracy	Testing accuracy	Loss
Fashion MNIST dataset	5	32	SGD	93.42	91.15	0.042
			RMSprop	93.37	90.94	0.040
			Adam	94.68	91.25	0.076
			Adadelta	93.37	91.73	0.048
			Adagrad	91.22	90.52	0.046
			Adamax	93.28	91.63	0.050
			NAdam	94.72	91.07	0.058
			Ftrl	89.88	89.75	0.045

Table 2: The performance of DNN architecture using Fashion MNIST dataset.

			DNN architecture			
Dataset	Epochs	Batch size	Optimizers	Training accuracy	Testing accuracy	Loss
Fashion MNIST dataset	5	32	SGD	88.43	76.34	0.32
			RMSprop	87.34	77.32	0.41
			Adam	84.69	78.63	0.48
			Adadelta	83.54	74.64	0.30
			Adagrad	85.28	75.85	0.34
			Adamax	86.37	74.73	0.38
			NAdam	86.35	76.84	0.37
			Ftrl	89.55	79.68	0.52

ordinal regression. The dataset sizes range from 100 to 100,000. It is as varied as possible since the VDD and MSD fairly evaluate the performance of generalizable machine learning algorithms across a range of contexts. However, real-time and three-dimensional medical specialist images are offered. It primarily focuses on machine learning rather than the end-to-end system like AN MNIST-like dataset assortment to do classification jobs on small photos. The 2828 (2D) or 282828 (3D) modest size is ideal for testing machine learning techniques. Medical specialty image analysis as a knowledge domain analysis space is challenging for researchers from various communities since it requires baseline knowledge.

4. Results and Discussion

This research work has been conducted using two different data sets: Fashion MNIST and MNIST, testing eight novel optimizers. Python language is used for developing a model using eight novel optimizers. The proposed work has achieved 16 results for using CNN and DNN architecture for each dataset. Overall performance has demonstrated the efficiency of the optimizers. Without optimizers, the result will go down, and the loss may be increased. This approach could be a promising method to set a goal of better accuracy for different kinds of datasets and architectures. The ideology behind this proposed work aims to elevate the typical results to be higher. Comparative analysis of various optimizers shows the variety of improvements that may change depending on architectures and datasets. The performance comparison uses eight novel optimizers with CNN and DNN architectures. The comparative analysis is performed using training and testing accuracy. Moreover, the loss value describes the qualitative result of the proposed work.

Table 1 exhibits the overall performance using CNN architecture using eight novel optimizers. This result shows the comparative analysis between different optimizers using the Fashion MNIST dataset. The performance report reveals better results for using all the optimizers. Through the observation from Table 1, Adadelta achieves higher accuracy of 91.732% among the other optimizers. The next better accuracy of 91.628% gives the Adamax optimizer. The Ftrl optimizer has obtained minimum accuracy among the other optimizers. The results of Table 2 show that the efficiency of the proposed work achieves higher accuracy for the Ftrl optimizer. The other optimizers also reached higher accuracy with slight differences. Also, the testing accuracy slightly reduces their accuracy compared with the training accuracy.

Table 3 shows the result using CNN architecture with eight novel optimizers. The experimental result demonstrates the higher accuracy of using all the optimizers. Especially for SGD optimizer has obtained better accuracy for MNIST dataset than the other optimizers. So, the SGD optimizer is well suited for the MNIST dataset. The next priority will be given to Adamax, RMSprop, and Adadelta optimizers because these optimizers reach similar results for their dataset.

Table 4 shows the efficiency of the method which has improved the level of accuracy of the CNN architecture. The overall report shows that the DNN architecture gives a better result than the CNN architecture. Table 5 shows the result using CNN architecture with eight novel optimizers using the Medical MNIST dataset. The experimental result demonstrates overall accuracy has been improved using all optimizers.

Table 3: The performance of CNN architecture using the MNIST dataset.

			CNN architecture			
Dataset	Epochs	Batch size	Optimizers	Training accuracy	Testing accuracy	Loss
MNIST dataset	5	32	SGD	99.62	98.68	0.062
			RMSprop	99.45	98.55	0.060
			Adam	99.67	98.43	0.085
			Adadelta	99.30	98.53	0.059
			Adagrad	99.65	98.26	0.061
			Adamax	99.50	98.58	0.060
			NAdam	99.75	98.40	0.078
			Ftrl	98.96	98.11	0.052

Table 4: The performance of DNN architecture using the MNIST dataset.

			DNN architecture			
Dataset	Epochs	Batch size	Optimizers	Training accuracy	Testing accuracy	Loss
MNIST dataset	5	32	SGD	99.86	99.27	0.046
			RMSprop	99.75	99.34	0.047
			Adam	99.67	99.31	0.046
			Adadelta	99.73	99.23	0.043
			Adagrad	99.67	99.17	0.041
			Adamax	99.52	99.19	0.044
			NAdam	99.25	99.02	0.038
			Ftrl	99.79	99.57	0.056

Table 5: The performance of CNN architecture using the Medical MNIST dataset.

			CNN architecture			
Dataset	Epochs	Batch size	Optimizers	Training accuracy	Testing accuracy	Loss
Medical MNIST dataset	5	32	SGD	99.65	99.53	0.034
			RMSprop	99.58	99.46	0.032
			Adam	99.78	99.47	0.031
			Adadelta	99.43	99.39	0.029
			Adagrad	99.63	99.56	0.040
			Adamax	99.64	99.57	0.042
			NAdam	99.49	99.42	0.039
			Ftrl	99.02	98.89	0.029

Table 6: The performance of DNN architecture using the Medical MNIST dataset.

			DNN architecture			
Dataset	Epochs	Batch size	Optimizers	Training accuracy	Testing accuracy	Loss
Medical MNIST dataset	5	32	SGD	99.86	99.43	0.031
			RMSprop	99.55	99.36	0.028
			Adam	99.75	99.48	0.027
			Adadelta	99.45	99.26	0.026
			Adagrad	99.68	99.42	0.035
			Adamax	99.53	99.16	0.033
			NAdam	99.47	99.28	0.028
			Ftrl	99.12	98.76	0.039

Especially for SGD optimizer has obtained better accuracy for Medical MNIST dataset than the other optimizers.

The performance evaluation shows that the SGD optimizer is well suited for the Medical MNIST dataset. The next priority will be given to Adamax, RMSprop, and Adadelta optimizers because these optimizers reach similar results for the dataset. The analysis of the results depicts the performance of the proposed work exhibiting better results compared to training and testing accuracy. Moreover, the results bring the efficacy of the outcomes compared with the existing architecture performance. The visualization report demonstrates the overall performance of the architectures

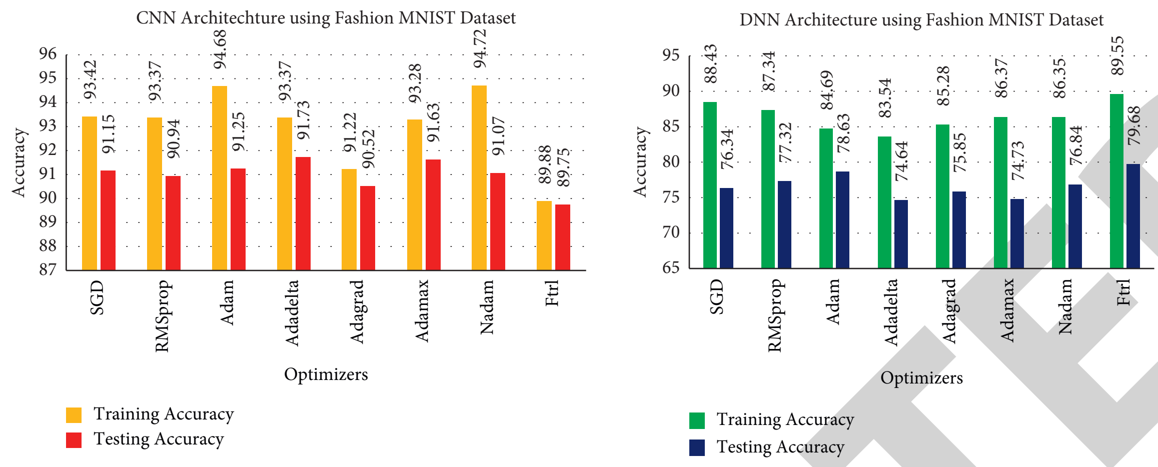

FIGURE 6: Visualization of CNN and DNN using optimizers for Fashion MNIST dataset.

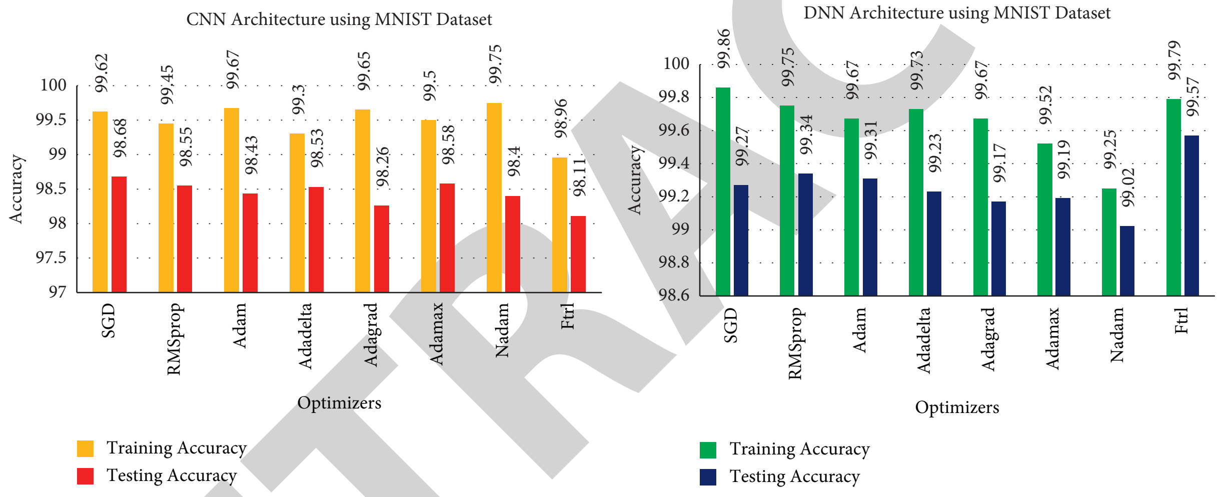

FIGURE 7: Visualization of CNN and DNN using optimizers for MNIST Numerical dataset.

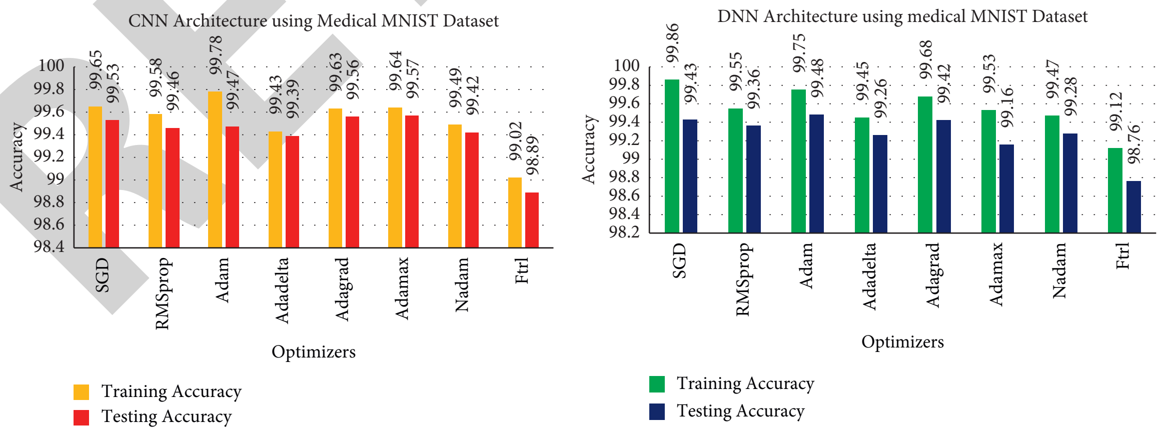

FIGURE 8: Visualization of CNN and DNN using optimizers for MNIST Medical dataset.

using various optimizers, and the level can be improved accordingly. Table 6 shows the efficiency of the method which has improved the level of accuracy of the CNN architecture. The overall report shows that the DNN architecture gives a better result than the CNN architecture. The observation of Table 6 reveals high accuracy using the Adam optimizer for the Medical MNIST dataset than the other optimizers. It is well suited for the Medical MNIST dataset. The performance report has been compared with each optimizer and tested using various trials to find the best-suited optimizers for DNN architectures.

Overall result analysis presents that the comparative performance analysis for CNN and DNN architecture is presented from Figures 6 to 8.

Through the observation, various optimizers are tested using different datasets, and it is noted that each optimizer has unique attributes. The results included parameters like the number of epochs, batch size, and learning rate. Finally, the epochs will be fixed as 5, batch size will be 32, and the learning rate as 0.01 has been taken for the higher accuracy value.

5. Conclusion

The proposed method presents an analysis of various novel optimizers used for fine-tuning the performance of CNN and DNN architecture. The performance of the proposed method has been evaluated using measures to display the accuracy and loss value. This novel approach has been evident in achieving comparable results in various datasets. Each phase of implementation of the dataset and architectures of CNN and DNN reveals different results accordingly. The overall performance of this proposed work has been evaluated using such parameters as batch size, the number of epochs, and learning rate. This research work is a nutshell to compare the various optimizers for the different architectures and the datasets. The comprehensive report has been constructed using multiple components to improve its accuracy. The proposed work is extended to use the other datasets and architectures to test a comparable accuracy range.

Data Availability

The data are available upon request.

Conflicts of Interest

The authors declare that they have no conflicts of interest to report regarding the study.

Acknowledgments

The proposed research trials were conducted in "The Advanced Image Processing DST-FIST Laboratory," Department of Computer Science and Applications, the Gandhigram Rural Institute (Deemed to be University), Dindigul, Tamil Nadu, India.

References

[1] I. H. Sarker, "Deep learning: a comprehensive overview on techniques, taxonomy, applications and research directions," *SN Computer Science*, vol. 2, no. 6, pp. 420–20, 2021.

[2] T. J. Sejnowski, "The unreasonable effectiveness of deep learning in artificial intelligence," *Proceedings of the National Academy of Sciences*, vol. 117, no. 48, pp. 30033–30038, 2020.

[3] L. Alzubaidi, J. Zhang, A. J. Humaidi et al., "Review of deep learning: concepts, CNN architectures, challenges, applications, future directions," *Journal of Big Data*, vol. 8, no. 1, pp. 53–74, 2021.

[4] S. Indolia, A. K. Goswami, S. P. Mishra, and P. Asopa, "Conceptual understanding of convolutional neural network- a deep learning approach," *Procedia Computer Science*, vol. 132, pp. 679–688, 2018.

[5] S. R. Dubey, S. Chakraborty, S. K. Roy, S. Mukherjee, S. K. Singh, and B. B. Chaudhuri, "DiffGrad: an optimization method for convolutional neural networks," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 31, no. 11, pp. 4500–4511, 2020.

[6] R. Mohapatra, S. Saha, C. A. C. Coello, A. Bhattacharya, S. S. Dhavala, and S. Saha, "AdaSwarm: augmenting gradient-based optimizers in deep learning with swarm intelligence," *IEEE Transactions on Emerging Topics in Computational Intelligence*, vol. 6, no. 2, pp. 329–340, 2022.

[7] N. Landro, I. Gallo, and R. L. Grassa, "Combining optimization methods using an adaptive Meta optimizer," *Algorithms*, vol. 14, no. 6, p. 186, 2021.

[8] B. Roy, M. P. Singh, M. R. Kaloop et al., "Data-driven approach for rainfall-runoff modelling using equilibrium optimizer coupled extreme learning machine and deep neural network," *Applied Sciences*, vol. 11, no. 13, p. 6238, 2021.

[9] L. Cheng, Z. Wang, F. Jiang, and J. Li, "An identifier-actor-optimizer policy learning architecture for optimal control of continuous-time nonlinear systems," *Science China Physics, Mechanics & Astronomy*, vol. 63, no. 6, pp. 264511-264512, 2020.

[10] M. Yaqub, J. Feng, M. S. Zia et al., "State-of-the-art CNN optimizer for brain tumor segmentation in magnetic resonance images," *Brain Sciences*, vol. 10, no. 7, p. 427, 2020.

[11] N. Dawson-Elli, S. Kolluri, K. Mitra, and V. R. Subramanian, "On the creation of a chess-AI-inspired problem-specific optimizer for the pseudo two-dimensional battery model using neural networks," *Journal of the Electrochemical Society*, vol. 166, no. 6, pp. A886–A896, 2019.

[12] X. Cui, X. ., W. Zhang, Z. Tuske, and M. Picheny, "Evolutionary stochastic gradient descent for optimization of deep neural networks," *Advances in Neural Information Processing Systems*, vol. 31, 2018.

[13] Q. Zheng, X. Tian, N. Jiang, and M. Yang, "Layer-wise learning based stochastic gradient descent method for the optimization of deep convolutional neural network," *Journal of Intelligent and Fuzzy Systems*, vol. 37, no. 4, pp. 5641–5654, 2019.

[14] M. Loni, S. Sinaei, A. Zoljodi, M. Daneshtalab, M. Sjodin, and D. Maker, "DeepMaker: a multi-objective optimization framework for deep neural networks in embedded systems," *Microprocessors and Microsystems*, vol. 73, Article ID 102989, 2020.

[15] N. M. Aszemi and P. D. D. Dominic, "Hyperparameter optimization in convolutional neural network using genetic algorithms," *International Journal of Advanced Computer Science and Applications*, vol. 10, no. 6, pp. 269–278, 2019.

[16] S. Bera and V. K. Shrivastava, "Analysis of various optimizers on deep convolutional neural network model in the application of hyperspectral remote sensing image classification," *International Journal of Remote Sensing*, vol. 41, no. 7, pp. 2664–2683, 2020.

[17] Q. Zheng, D. Fu, Y. Wang, H. Chen, and H. Zhang, "A study on global optimization and deep neural network modeling method in performance-seeking control," *Proceedings of the Institution of Mechanical Engineers - Part I: Journal of Systems & Control Engineering*, vol. 234, no. 1, pp. 46–59, 2020.

[18] I. Kandel, M. Castelli, and A. Popovic, "Comparative study of first order optimizers for image classification using convolutional neural networks on histopathology images," *Journal of Imaging*, vol. 6, no. 9, p. 92, 2020.

[19] E. Zitzler, L. Thiele, M. Laumanns, C. M. Fonseca, and V. G. Da Fonseca, "Performance assessment of multiobjective optimizers: an analysis and review," *IEEE Transactions on Evolutionary Computation*, vol. 7, no. 2, pp. 117–132, 2003.

[20] V. Pavan, "Types of Optimizers in Deep Learning Every AI Engineer Should Know," 2020, https://www.upgrad.com/blog/types-of-optimizers-in-deep-learning.

[21] Q. Berthet, M. Blondel, O. Teboul, M. Cuturi, J. P. Vert, and F. Bach, "Learning with differentiable pertubed optimizers," *Advances in Neural Information Processing Systems*, vol. 33, pp. 9508–9519, 2020.

[22] S. Postalcıoğlu, "Performance analysis of different optimizers for deep learning-based image recognition," *International Journal of Pattern Recognition and Artificial Intelligence*, vol. 34, no. 02, Article ID 2051003, 2020.

[23] A. Kumar, S. Sarkar, and C. Pradhan, "Malaria disease detection using CNN technique with sgd, rmsprop and Adam optimizers," in *Proceedings of the Deep Learning Techniques for Biomedical and Health Informatics*, pp. 211–230, Springer, Cham, 2020.

[24] A. M. Taqi, A. Ahmed, F. Al-Azzo, and M. Milanova, "The impact of multi-optimizers and data augmentation on TensorFlow convolutional neural network performance," in *Proceedings of the IEEE Conference on Multimedia Information Processing and Retrieval (MIPR)*, pp. 140–145, IEEE, Miami, FL, USA, April 2018.

[25] M. Fradi, L. Khriji, and M. Machhout, "Real-time arrhythmia heart disease detection system using CNN architecture based various optimizers-networks," *Multimedia Tools and Applications*, pp. 1–22. In press, 2021.

[26] M. Agarwal, A. Rajak, and A. Kumar Shrivastava, "Assessment of optimizers impact on image recognition with convolutional neural network to adversarial datasets," in *Journal of Physics: Conference Series*vol. 1998, no. 1, IOP Publishing, Article ID 012008, 2021.

[27] A. Shaf, T. Ali, W. Farooq, S. Javaid, U. Draz, and S. Yasin, "Two classes classification using different optimizers in convolutional neural network," in *Proceedings of the IEEE 21st International Multi-Topic Conference (INMIC)*, pp. 1–6, IEEE, Karachi, Pakistan, November 2018.