International Journal of Multidisciplinary Studies and Innovative Technologies

e-ISSN: 2602-4888
dergipark.org.tr/en/pub/ijmsit
Research Article

2024, 8 (2), 52-58
DOI: 10.36287/ijmsit.8.2.1
Received: July 3, 2024; Accepted: August 30, 2024

Comparison of Different Optimization Algorithms in the Fashion MNIST Dataset

Umut Saray [1*], Uğur Çavdar [2]

[1*] *Department of Electronic Automation, Turhal Vocational School, Tokat Gaziosmanpaşa University, Tokat, Türkiye,(umutsaray@gmail.com) (ORCID: 0000-0003-3339-6876)*
[2] *Department of Mechanical Engineering, Faculty of Engineering, Izmir Democracy University, İzmir, Türkiye,(ugur.cavdar@idu.edu.tr) (ORCID:0000-0002-3434-6670)*

Abstract – This study examines the effects of various optimization algorithms used in deep learning models to classify fashion-oriented clothing items. The Fashion MNIST dataset has been chosen as a rich data source. Models developed using Convolutional Neural Networks (CNN) have been trained with various optimization algorithms such as Nadam, Adadelta, Adamax, Adam, Adagrad, SGD, and RMSprop. Understanding the impact of these algorithms on the model's performance during the training process forms the basis of the study. The findings of the research reveal that optimization algorithms have a significant effect on the accuracy rates of the model. While the Nadam and Adadelta algorithms achieved the highest accuracy rates, the RMSprop algorithm displayed relatively lower performance. These results indicate that different optimization techniques can significantly influence the performance of deep learning-based classification systems.

Keywords *– Convolutional Neural Networks (CNN),Fashion MNIST, Optimization Algorithms, Adam, RMSprop*

Citation: Saray, U., Çavdar, U. (2024). Comparison of Different Optimization Algorithms in the Fashion MNIST Dataset. International Journal of Multidisciplinary Studies and Innovative Technologies, 8(2): 52-58.

I. INTRODUCTION

Deep learning has become one of the most attractive areas in artificial intelligence and machine learning over the past decade, undergoing significant evolution. This method offers the ability to learn from comprehensive and voluminous datasets through models composed of multi-layered neural networks. Particularly, revolutionary results have been achieved in fields such as visual and auditory recognition, forecasting apps, natural language processing, and various pattern recognitions [1],[2]-[5].

Optimization algorithms play an indispensable role in the training process of these models [6]. The success of a deep learning model is largely dependent on the effectiveness of the chosen optimization algorithm. Stochastic Gradient Descent (SGD) [7-8] and its variants enable the model to demonstrate superior performance on the dataset by adjusting its weights and bias values. In literature, comparing different optimization algorithms and the identification of the most suitable one have become important research topics, especially for models operating on large and complex datasets [9].

Recent studies on deep learning and optimization algorithms have examined the performance of various algorithms on different datasets. For instance, Ö. Dolma [1] classified COVID-19 and non-COVID-19 lung CT scan images using deep convolutional neural networks. E. Avuçlu [2-3] evaluated the classification performance of COVID-19 images using deep learning methods. In another study, M. C. Bingöl and G. Bilgin [4] investigated the prediction of chicken diseases using transfer learning methods. Comparing optimization algorithms, especially for large and complex datasets, is crucial to determining which algorithm is more suitable. Stochastic Gradient Descent (SGD) and its variants enable the model to adjust its weights and biases to perform optimally on the dataset [6], [7-8]. Algorithms with adaptive learning rates, such as Adam [9], [10-11], Nadam [12], [13-14], RMSprop [15], [16-17], and Adagrad [18], [19-20], are widely used to achieve strong results in the training process of deep learning models. In this context, the Fashion MNIST dataset is frequently preferred as a rich data source for classifying fashion-oriented clothing items. For example, R. Sirisha and colleagues [23] compared the performance of different optimization algorithms on the Fashion MNIST dataset; A. S. Henrique and his team [24] developed CNN models using this dataset. Khanday and colleagues [25] examined the effect of filter sizes on classification accuracy. Other studies include those by Tang et al. [26], Kayed et al. [27], Zhu et al. [28], and Hur et al. [29], who have all utilized the Fashion MNIST dataset for various purposes, such as optimizing deep residual networks, using CNN LeNet-5 architecture, space-efficient optical computing, and quantum convolutional neural networks, respectively. These studies examined the impact of different optimization algorithms on the accuracy rates of deep learning-based classification models and identified the most effective algorithms [21], [22-29].

Researchers and practitioners have closely examined the algorithms used in optimizing deep learning models in recent years. Among these algorithms, methods with adaptive learning rates such as Adam, Nadam, RMSprop, and Adagrad have become popular for achieving strong results in the training process of deep learning models [21].

This study aims to examine the effects of these algorithms on the Fashion MNIST dataset [22], [23-29], a widely used dataset for training and testing contemporary artificial intelligence and machine learning systems. This dataset contains grayscale images of various clothing items and offers an excellent test ground for algorithmic classification [30].

The purpose of this research is to understand the impact of different optimization algorithms on the accuracy rates of deep learning-based classification models and to use this knowledge to enhance the effectiveness of classification systems. The findings highlight the importance of selecting optimization strategies in AI applications and guide future research in this direction.

II. Materials and Method

Convolutional Neural Networks (CNNs)[31] are frequently utilized in image processing and visual recognition tasks. Essentially, they employ convolutional layers to detect local features in an image, such as edges, textures, and shapes. These layers, through a specific learning process, automatically learn to extract useful features from different parts of the image. CNNs are capable of recognizing complex visual patterns by combining and interpreting these features in subsequent layers.

The fundamental components of CNNs include convolutional layers, activation functions, pooling layers, and fully connected layers. Convolutional layers apply filters to the input image to create feature maps, effectively extracting information from different sections of the image to identify important features.

Activation functions enhance the network's non-linear learning capability. One of the most commonly used activation functions is ReLU, which speeds up the model's training process by setting negative values to zero and helping to address the gradient vanishing problem.

Pooling layers reduce the dimensionality of feature maps, lightening the network's computational load. This is achieved by taking the maximum or average value of certain sections of the image. Pooling ensures the network's robustness against translational variances, such as changes in the position of an object within the image [32].

Fully connected layers are located at the end of the network and use the learned features to perform tasks such as classification or regression, producing the final output. These layers associate each input with probabilities for each class in the output.

Due to their ability to successfully recognize complex visual patterns, CNNs are effectively used in various application areas such as face recognition, vehicle license plate recognition, medical image analysis, and object detection from satellite images. Recent advancements in deep learning have further improved the performance and applicability of CNNs, making them an indispensable component of artificial intelligence applications [33].

In this study, Convolutional Neural Networks (CNNs) were used. Various optimization algorithms within the CNN have been compared for their success rates on the MNIST dataset. The optimization algorithms used are explained in sequence. The algorithms employed include Stochastic Gradient Descent (SGD), Adagrad, RMSprop, Adadelta, Adamax, Nadam, and Adam.

A. Stochastic Gradient Descent (SGD)

Stochastic Gradient Descent (SGD) is a method that calculates the gradient using a single training example at each step to update the model parameters. This approach enables quick parameter updates based on randomly selected samples, eliminating the need to process the entire dataset in each iteration. This efficiency makes SGD particularly effective for large datasets. However, the optimization path of SGD can be somewhat erratic, leading towards the target through a fluctuating route, which necessitates precise hyperparameter tuning for optimal performance.

The core of SGD's methodology is encapsulated in its update rule, where the parameter θ at any given iteration t+1 is adjusted according to the formula:

$$\theta_{t+1} = \theta_t - \eta \nabla_\theta L(\theta_t; x_i, y_i) \quad (1)$$

In this equation, θ_t represents the parameter vector at iteration t, η denotes the learning rate, and $\nabla_\theta L(\theta_t; x_i, y_i)$ signifies the gradient of the loss function L with respect to θ, evaluated for the i th training example at the t th iteration. This process underscores the iterative nature of SGD, where each step is calculated to steer the parameters closer to the optimum by leveraging the gradient information from a single, randomly selected training example [7-8].

B. Adagrad

Adagrad is an optimization algorithm that adaptively adjusts the learning rate for each parameter, making it particularly well-suited for dealing with sparse datasets. Unlike conventional methods that use a single learning rate for all parameters throughout the training process, Adagrad modifies the learning rate individually for each parameter based on the historical gradient information. This approach lowers the learning rate for parameters corresponding to frequently occurring features, while ensuring a higher learning rate for rare features. As a result, Adagrad can significantly improve the efficiency of model training, especially in scenarios where the data is sparse.The key to Adagrad's adaptive learning rate adjustment lies in its update rule, which is mathematically formulated as follows:

For each parameter θ_t, the update at iteration t is given by;

$$\theta_{i,t+1} = \theta_{i,t} - \frac{\eta}{\sqrt{G_{i,t}+\epsilon}} \cdot g_{i,t} \quad (2)$$

Here $g_{i,t}$, represents the gradient of the loss with respect to the parameter θ_i at iteration t, $G_{i,t}$ is the sum of the squares of the past gradients with respect to θ_i up to time t, η is a global learning rate, and ϵ is a smoothing term added to improve numerical stability (often set to a small constant like 1^{e-8}), preventing division by zero.

This formula ensures that parameters with large gradients have their learning rate decreased over time, which helps in honing in on the minimum more efficiently. However, a notable downside of Adagrad is its tendency for the learning rate to decrease continually throughout training, potentially leading to premature convergence and the model stopping early in long training processes. Despite this limitation, Adagrad's ability to adapt the learning rate to the parameters has made it a foundational algorithm for further developments in adaptive learning rate techniques[18], [34-35].

C. RMSprop

RMSprop, short for Root Mean Square Propagation, is an optimization algorithm designed to overcome the challenge of the excessively decreasing learning rate that Adagrad faces. By focusing on the magnitude of gradients in only the most recent iterations, RMSprop dynamically adjusts the learning rate. This method ensures that the learning rate does not diminish too quickly, maintaining a level that is conducive to continued learning and optimization over time. RMSprop is particularly effective in scenarios involving recurrent neural networks and non-stationary targets, where the landscape of the optimization problem changes over time.

The mathematical foundation of RMSprop is expressed through its update rule, which modifies the learning rate for each parameter based on the recent gradients. The update for a parameter θ at iteration t is given by:

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{v_t + \epsilon}} . g_t \quad (3)$$

In this equation, g_t is the gradient of the loss with respect to the parameter θ at iteration t, η is the initial learning rate, and ϵ is a small constant (like 1^{e-8}) to prevent division by zero. The term v_t represents the exponentially weighted moving average of the squares of the gradients, calculated as:

$$v_t = \beta v_{t-1} + (1 + \beta)\, g_t^2 \quad (4)$$

Here, β is a decay rate that determines the extent to which the moving average considers the most recent gradient magnitudes, typically set to a value like 0.9. This mechanism of adjusting $v_t$ensures that RMSprop considers the magnitude of recent gradients, enabling adaptive learning rates that respond to the current state of the optimization process.

By employing this strategy, RMSprop effectively prevents the learning rate from dropping too low, a significant improvement over Adagrad's approach. This adaptability makes RMSprop a robust choice for training deep neural networks, particularly in the challenging environments presented by recurrent neural networks and tasks with non-stationary objectives. [36],[18].

D. Adadelta

Adadelta is an optimization algorithm that extends the principles of RMSprop to enhance stability in the learning rate throughout the training process. It achieves this by employing a unit measure for weight updates, which allows for continuous model improvement without the explicit need to adjust the learning rate manually. This approach addresses one of the key challenges in optimization algorithms - the sensitivity to the choice of learning rate. By eliminating the dependence on a global learning rate, Adadelta simplifies the optimization process, making it more robust and easier to use, especially in environments where parameter tuning can be laborious.

The foundation of Adadelta is grounded in the modification of the RMSprop update rule, incorporating the use of the moving average of squared gradients to adjust the learning rate dynamically, but it also introduces the concept of accumulating updates over time to determine the step size. The update rule in Adadelta for a parameter θ at iteration t can be expressed as follows:

$$\Delta\theta_{t+1} = -\frac{\sqrt{\sum_{i=1}^{t-1} \Delta\theta_{i-1}^2 + \epsilon}}{\sqrt{E[g^2]_t + \epsilon}} . g_t \quad (5)$$

Here, g_t represents the gradient of the loss with respect to the parameter θ at iteration *t*, $E[g^2]_t$ is the exponentially decaying average of squared gradients up to time *t*, and ϵ is a small constant (similar to RMSprop) added for numerical stability. The term $\Delta\theta_t$denotes the change in θ at iteration t, and the numerator $\sqrt{\sum_{i=1}^{t-1} \Delta\theta_{i-1}^2 + \epsilon}$ represents the root mean square of previous parameter updates, which serves to scale the gradient in proportion to the historical update magnitudes.

The key innovation of Adadelta is that it does not require an explicit learning rate. Instead, it adapts the parameter updates based on the moving averages of the squared gradients and the squared updates, thus regulating the step size based on the history of changes. This self-adjusting mechanism ensures more stable and consistent learning progress, mitigating the risk of drastic updates that could potentially derail the optimization process.

By combining the adaptive gradient approach of RMSprop with the innovative update adjustment mechanism, Adadelta offers a sophisticated solution to the challenge of learning rate selection and stability, making it an attractive choice for training deep neural networks where tuning hyperparameters can be particularly challenging [36-37].

E. Adamax

Adamax is a variation of the Adam optimization algorithm, designed to enhance stability in scenarios characterized by extreme gradient values. While Adam employs adaptive moment estimation to adjust learning rates based on the first and second moments of gradients (mean and uncentered variance), Adamax introduces an alternative approach by utilizing a different norm, making it potentially more robust in the face of extreme updates. This characteristic of Adamax stems from its adaptation of the ∞-norm, which provides a theoretical upper bound on the updates, hence its name. The update rules for Adamax at iteration t for a parameter θ can be summarized as follows:

Update the first moment (the mean) of the gradient:

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1)\, g_t \quad (6)$$

where $g_t$is the gradient of the loss with respect to θ at iteration t, and m_t is the first moment vector.

Update the ∞-norm of the gradients rather than the second moment:

$$u_t = \max(\beta_2\, u_{t-1}, |g_t|) \quad (7)$$

Here, u_t represents the ∞-norm of the gradients, which is updated to be the maximum of the previous ∞-norm scaled by β_2 and the absolute value of the current gradient. This replaces the second moment estimation used in the original Adam.

Compute the parameter update using the first moment and the ∞-norm:

$$\theta_{t+1} = \theta_t - \frac{\eta}{u_t + \epsilon}\, m_t \quad (8)$$

In this equation, η is the step size (learning rate), and ϵ is a small constant added for numerical stability.The introduction of the ∞-norm in Adamax, as opposed to the squared gradients norm in Adam, aims to provide a more stable and less aggressive adaptation of the learning rates, especially in the presence of large gradients. This makes Adamax an appealing alternative for optimization in machine learning tasks where gradients can vary significantly in magnitude, potentially

leading to more consistent and reliable convergence over the course of training [38-39].

F. Nadam

Nadam, short for Nesterov-accelerated Adaptive Moment Estimation, merges the Adam optimization algorithm with Nesterov momentum, harnessing the strengths of both methodologies to achieve more efficient optimization. By integrating Adam's adaptive learning rate features with the anticipatory updates of Nesterov momentum, Nadam facilitates faster convergence and improved performance, particularly in the context of deep learning and complex optimization tasks. This combination allows Nadam to navigate the optimization landscape more effectively, making it a powerful tool for training neural networks.

The mathematical formulation of Nadam incorporates elements from both Adam and Nesterov momentum, resulting in an update rule that looks as follows:

Update the first moment (mean) and the second moment (uncentered variance) of the gradients, similar to Adam:

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t \ , v_t = \beta v_{t-1} + (1 + \beta)\, g_t^2 \quad (9)$$

where g_t is the gradient of the loss with respect to the parameter θ at iteration *t*, m_t is the first moment vector, and v_t is the second moment vector.

Incorporate Nesterov momentum into the moment update by adjusting the first moment before the parameter update:

$$\hat{m}_t = \frac{m_t}{1-\beta_1^t} + \frac{(1-\beta_1) g_t}{(1-\beta_1^t)(1-\beta_1)} \quad (10)$$

$$\hat{v}_t = \frac{v_t}{(1-\beta_2^t)} \quad (11)$$

Compute the parameter update using the adjusted first moment and the second moment:

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{\hat{v}_t} + \epsilon} \hat{m}_t \quad (12)$$

In this equation, η is the learning rate, and ϵ is a small constant added for numerical stability.

By leveraging the lookahead nature of Nesterov momentum, which essentially incorporates information about the future gradient, Nadam ensures that each update is more informed and precise. This results in a more aggressive and effective approach to finding the minimum of the loss function, reducing the number of iterations needed to achieve convergence. Nadam's unique blend of Adam's adaptiveness and Nesterov's accelerated updates provides a significant advantage in training deep learning models, offering a balance between speed and accuracy in the optimization process [38-39].

G. Adam

The Adam optimization algorithm, standing for Adaptive Moment Estimation, is widely recognized for its ability to adaptively adjust both the learning rate and momentum for each parameter, making it a popular and effective method for deep learning tasks. By calculating exponential moving averages of both the gradients and the squared gradients, Adam maintains separate learning rates for each parameter, which are adjusted as learning progresses. This adaptability allows Adam to perform well across a wide range of deep-learning tasks, from simple to complex models. The mathematical formulation of Adam involves several key steps, as outlined below:

First and Second Moment Estimation: For each parameter θ, Adam computes the first moment (the mean) and the second moment (the uncentered variance) of the gradients:

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t \ , v_t = \beta v_{t-1} + (1 + \beta_2)\, g_t^2 \quad (13)$$

Here, g_t represents the gradient of the loss with respect to θ at iteration t, m_t and v_t are the estimates of the first and second moments respectively, and β_1 and β_2 are the decay rates for these moments.

Bias Correction: To counteract the biases introduced by initializing the moments as zeros, Adam applies bias corrections:

$$\hat{m}_t = \frac{m_t}{1-\beta_1^t} \ , \ \hat{v}_t = \frac{v_t}{(1-\beta_2^t)} \quad (14)$$

This step ensures that the moment estimates are unbiased towards zero at the start of optimization.

Parameter Update: The parameters are updated using the bias-corrected moments:

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{\hat{v}_t} + \epsilon} \hat{m}_t \quad (15)$$

In this equation, η is the step size (learning rate), and ϵ is a small constant added for numerical stability. Adam's approach to adjusting the learning rate based on the first and second moments of the gradients allows for more effective and efficient optimization, especially in the context of deep learning. This adaptability, combined with its straightforward implementation and robust performance across various tasks, has cemented Adam's status as a go-to optimization algorithm for many deep learning practitioners.[9],[40].

III.IMPLEMENTATION

Dataset and Preprocessing; The Fashion MNIST dataset consists of 60,000 training and 10,000 test images. Each image is a grayscale image of a garment with a resolution of 28x28 pixels.

In terms of data preprocessing steps, the images have been normalized between 0 and 1 and reshaped to the appropriate input size for the model (28x28x1), facilitating more effective learning by the model.

Model Architecture; First Convolutional Layer: Equipped with 32 filters, each having a kernel size of (3,3), and utilizes the ReLU activation function. This layer accepts input data of 28x28 pixel resolution with 1 color channel (grayscale images).

First Max Pooling Layer: Has a pool size of 2x2, aiming to halve the spatial dimensions.

Second Convolutional Layer: Contains 64 filters, also using the ReLU activation function, with each filter having a kernel size of (3,3).

Second Max Pooling Layer: Applies a 2x2 pooling operation again to further reduce spatial dimensions.

Flatten Layer: Transforms the outputs from the convolutional and pooling layers into a single, long feature vector.

First Dense Layer: Comprises 128 neurons and employs the ReLU activation function.

Output Dense Layer: Contains 10 neurons corresponding to the ten different garment classes in the dataset, using the

softmax activation function to output probability distributions for classification.

This study aims to compare the performance of different optimization algorithms: SGD, Adagrad, RMSprop, Adadelta, Adamax, Nadam, and Adam. Each algorithm is evaluated separately using the same model architecture.

The models are trained over twenty epochs, and the accuracy rates of each optimization algorithm are assessed on the test set. Performance evaluations are conducted using loss and accuracy metrics on the test set. This methodology allows the research to be conducted within a concrete and reproducible framework, contributing to the reliability of the results obtained.

IV. Results

This research has yielded significant findings by examining the impact of different optimization algorithms on the Fashion MNIST dataset. The study reveals that optimization algorithms have substantial effects on the training process and accuracy rates of the model.

The graphs depicted in the figures demonstrate the variations in loss and accuracy values during the training process for different optimization algorithms. Figure 1 presents the comparative training and validation loss across training epochs, while Figure 2 illustrates the training and validation accuracy throughout the training iterations for various optimization algorithms on the Fashion MNIST dataset.

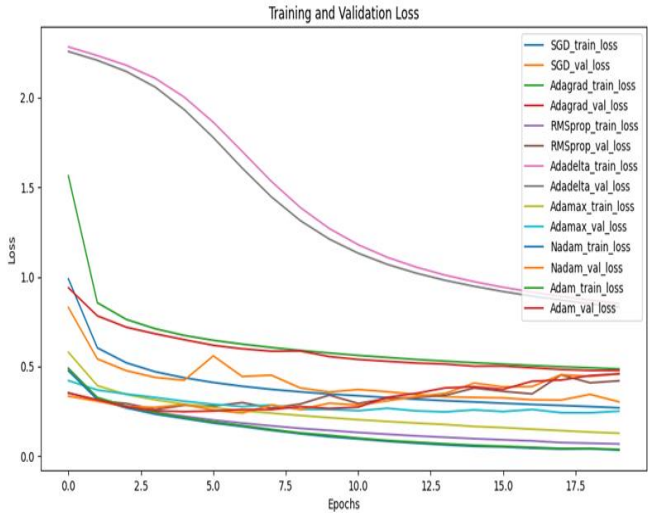

Figure 1. Optimization Algorithm Comparison: Training and Validation Loss Across Epochs for Fashion MNIST Dataset

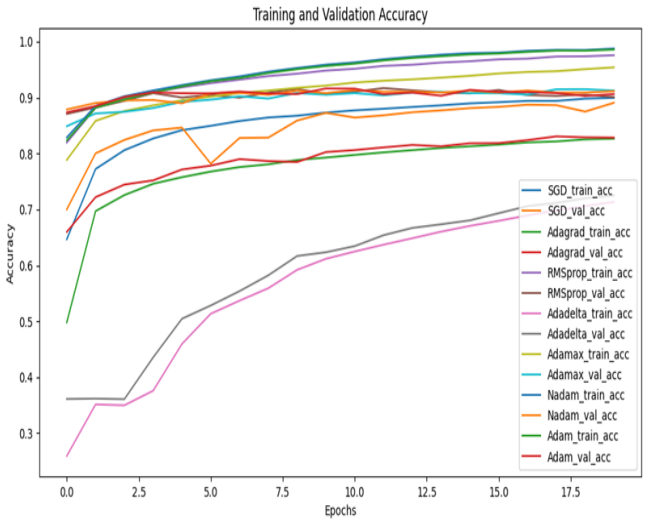

Figure 2. Comparative Analysis of Training and Validation Accuracy Over Epochs Among Various Optimization Algorithms on Fashion MNIST Dataset

Overall, all algorithms manage to reduce training loss, but some are notably more effective in reducing validation loss. The Nadam and Adam algorithms, in particular, maintain low and stable validation losses, indicating their strong generalization capabilities. On the other hand, the Adadelta algorithm, despite rapidly decreasing high initial loss values, appears to be less effective than other algorithms. The validation losses for SGD and Adagrad are also observed to be higher, which could suggest inferior generalization performance.

As seen in Figure 2, Nadam and Adam algorithms also display high validation accuracy, signifying robust performance. This high accuracy suggests that the model is well-generalized to new data. In contrast, SGD and Adagrad have lower validation accuracy, hinting that they may be less effective for training the model on this dataset. RMSprop, although exhibiting high training accuracy, has a validation accuracy that falls short of expectations, a possible indication of overfitting.

These insights reveal that the efficacy of an algorithm can vary significantly based on the dataset and specific problem at hand. They also highlight the critical role of model selection and hyperparameter tuning in machine learning. When selecting the best model, the performance on the validation set should be carefully considered.

Table.1. Performance Metrics of Different Optimization Algorithms on Fashion MNIST Dataset

Optimization Algorithm	Final Training Accuracy	Final Training Loss	Final Validation Accuracy	Final Validation Loss
SGD	0.9019	0.2651	0.8765	0.3398
Adagrad	0.8325	0.4681	0.8352	0.4603
RMSprop	0.9751	0.0678	0.8998	0.5287
Adadelta	0.7214	0.8193	0.7305	0.8004
Adamax	0.9557	0.1269	0.9115	0.2569
Nadam	0.9876	0.0331	0.9040	0.4831
Adam	0.9839	0.0436	0.9098	0.4788

According to Table 1; SGD has shown lower training accuracy and higher training loss compared to other algorithms. These findings suggest that SGD is less applicable to this particular problem than other alternatives.

Adagrad Optimization Algorithm: Adagrad has shown a moderate performance similar to SGD. Its accuracy and loss values were found to be at an average level. While Adagrad's dynamic adjustment of the learning rate can be advantageous in some problems, it has not achieved the best performance in this study.

RMSprop Optimization Algorithm: The RMSprop algorithm achieved high training accuracy and low training loss but maintained a high validation loss. This could indicate that the model did not generalize well to the validation data and might be an indication of overfitting.

Adadelta Optimization Algorithm: Adadelta's performance was lower compared to other algorithms. Both its training and validation accuracy, as well as loss values, were found to be high, indicating that Adadelta's generalization capability and training performance are lower than other alternatives.

Adamax Optimization Algorithm: Adamax exhibited good performance with high training and validation accuracy. Its low validation loss indicates that the model generalizes well to new data.

Nadam Optimization Algorithm: Nadam overall showed the best performance with the highest validation accuracy and the lowest validation loss. This indicates that the model generalizes very well to new data and that this optimization algorithm is effective for the chosen dataset.

Adam Optimization Algorithm: Adam showed a performance very close to Nadam. Its high training and validation accuracy and low validation loss indicate that this algorithm generalizes effectively.

V. DISCUSSION

This study evaluates the effectiveness of various optimization algorithms on the Fashion MNIST dataset, revealing significant differences in overall model performance. Specifically, the Nadam and Adam algorithms demonstrate superior generalization capabilities with their low validation losses and high validation accuracies, indicating their resilience against overfitting due to adaptive learning rates. On the other hand, the weaker performance exhibited by algorithms such as SGD and Adagrad, particularly in terms of high training and validation losses, highlights their limitations in developing effective learning strategies for high-dimensional datasets. These findings emphasize the critical importance of selecting and tuning optimization algorithms in machine learning projects and underscore the significance of further research to improve these algorithms. Additionally, a better understanding of performance variations among algorithms can enhance the applicability of models across broader datasets and ensure more successful implementations in practical applications.

VI. CONCLUSION

This study has demonstrated that the most effective optimization algorithms for the Fashion MNIST dataset are Nadam and Adam. The performance of other alternatives was found to be lower compared to these two algorithms. However, since the performance of each algorithm can vary depending on the dataset and the problem, conducting more comprehensive tests such as cross-validation is recommended to select the most suitable algorithm.

Statement of Conflicts of Interest

There is no conflict of interest between the authors.

Statement of Research and Publication Ethics

The authors declare that this study complies with Research and Publication Ethics

REFERENCES

[1] Ö. Dolma, "COVID-19 and Non-COVID-19 Classification from Lung CT-Scan Images Using Deep Convolutional Neural Networks," *Int. J. Multidiscip. Stud. Innov. Technol.*, vol. 7, no. 2, p. 53, 2023, doi: 10.36287/ijmsit.7.2.3.

[2] E. Avuçlu, "Examining The Effect of Pre-processed Covid-19 Images On Classification Performance Using Deep Learning Method," *Int. Sci. Vocat. Stud. J.*, vol. 7, no. 2, pp. 94–102, Dec. 2023, doi: 10.47897/bilmes.1359954.

[3] E. Avuçlu, "Classification of Pistachio Images Using VGG16 and VGG19 Deep Learning Models," *Int. Sci. Vocat. Stud. J.*, vol. 7, no. 2, pp. 79–86, Dec. 2023, doi: 10.47897/bilmes.1328313.

[4] M. C. Bıngol and G. Bilgin, "Prediction of Chicken Diseases by Transfer Learning Method," *Int. Sci. Vocat. Stud. J.*, vol. 7, no. 2, pp. 170–175, Dec. 2023, doi: 10.47897/bilmes.1396890.

[5] Y. Durgun, "Classification of Starch Adulteration in Milk Using Spectroscopic Data and Machine Learning," *Int. J. Eng. Res. Dev*., vol. 16, no. 1, pp. 221-226, 2024, doi: 10.29137/umagd.1379171.

[6] A. Williams, N. Walton, A. Maryanski, S. Bogetic, W. Hines, and V. Sobes, "Stochastic gradient descent for optimization for nuclear systems," *Sci. Rep.*, vol. 13, no. 1, p. 8474, May 2023, doi: 10.1038/s41598-023-32112-7.

[7] S. Nagendram *et al.*, "Stochastic gradient descent optimisation for convolutional neural network for medical image segmentation," *Open Life Sci.*, vol. 18, no. 1, Aug. 2023, doi: 10.1515/biol-2022-0665.

[8] C. Song, A. Pons, and K. Yen, "AG-SGD: Angle-Based Stochastic Gradient Descent," *IEEE Access*, vol. 9, pp. 23007–23024, 2021, doi: 10.1109/ACCESS.2021.3055993.

[9] C. Milovic *et al.*, "Comparison of parameter optimization methods for quantitative susceptibility mapping," *Magn. Reson. Med.*, vol. 85, no. 1, pp. 480–494, Jan. 2021, doi: 10.1002/mrm.28435.

[10] M. Reyad, A. M. Sarhan, and M. Arafa, "A modified Adam algorithm for deep neural network optimization," *Neural Comput. Appl.*, vol. 35, no. 23, pp. 17095–17112, 2023, doi: 10.1007/s00521-023-08568-z.

[11] I. K. M. Jais, A. R. Ismail, and S. Q. Nisa, "Adam Optimization Algorithm for Wide and Deep Neural Network," *Knowl. Eng. Data Sci.*, vol. 2, no. 1, p. 41, 2019, doi: 10.17977/um018v2i12019p41-46.

[12] B. Cortiñas-Lorenzo and F. Pérez-González, "Adam and the Ants: On the Influence of the Optimization Algorithm on the Detectability of DNN Watermarks," *Entropy*, vol. 22, no. 12, p. 1379, Dec. 2020, doi: 10.3390/e22121379.

[13] P. Ramachandran, T. Eswarlal, M. Lehman, and Z. Colbert, "Assessment of optimizers and their performance in autosegmenting lung tumors," *J. Med. Phys.*, vol. 48, no. 2, pp. 129–135, 2023, doi: 10.4103/jmp.jmp_54_23.

[14] P. Podder *et al.*, "LDDNet: A Deep Learning Framework for the Diagnosis of Infectious Lung Diseases," *Sensors*, vol. 23, no. 1, 2023, doi: 10.3390/s23010480.

[15] C. Annamalai, C. Vijayakumaran, V. Ponnusamy, and H. Kim, "Optimal ElGamal Encryption with Hybrid Deep-Learning-Based Classification on Secure Internet of Things Environment," *Sensors*, vol. 23, no. 12, p. 5596, Jun. 2023, doi: 10.3390/s23125596.

[16] R. Elshamy, O. Abu-Elnasr, M. Elhoseny, and S. Elmougy, "Improving the efficiency of RMSProp optimizer by utilizing Nestrove in deep learning," *Sci. Rep.*, vol. 13, no. 1, p. 8814, May 2023, doi: 10.1038/s41598-023-35663-x.

[17] X. Jiang, B. Hu, S. Chandra Satapathy, S. H. Wang, and Y. D. Zhang, "Fingerspelling Identification for Chinese Sign Language via AlexNet-Based Transfer Learning and Adam Optimizer," *Sci. Program.*, vol. 2020, 2020, doi: 10.1155/2020/3291426.

[18] A. Daneshvar, M. Ebrahimi, F. Salahi, M. Rahmaty, and M. Homayounfar, "Brent Crude Oil Price Forecast Utilizing Deep Neural Network Architectures," *Comput. Intell. Neurosci.*, vol. 2022, pp. 1–13, May 2022, doi: 10.1155/2022/6140796.

[19] V. Ojha and G. Nicosia, "Backpropagation Neural Tree," *Neural*

Networks, vol. 149, pp. 66–83, May 2022, doi: 10.1016/j.neunet.2022.02.003.

[20] B. Zhu, Y. Shi, J. Hao, and G. Fu, "Prediction of Coal Mine Pressure Hazard Based on Logistic Regression and Adagrad Algorithm—A Case Study of C Coal Mine," *Appl. Sci.*, vol. 13, no. 22, 2023, doi: 10.3390/app132212227.

[21] F. Aamir, I. Aslam, M. Arshad, and H. Omer, "Accelerated Diffusion-Weighted MR Image Reconstruction Using Deep Neural Networks," *J. Digit. Imaging*, vol. 36, no. 1, pp. 276–288, Nov. 2022, doi: 10.1007/s10278-022-00709-5.

[22] G. Ayana, J. Park, J.-W. Jeong, and S. Choe, "A Novel Multistage Transfer Learning for Ultrasound Breast Cancer Image Classification," *Diagnostics*, vol. 12, no. 1, p. 135, Jan. 2022, doi: 10.3390/diagnostics12010135.

[23] R. Sirisha, N. Anjum, and K. Vaidehi, "INDIAN JOURNAL OF SCIENCE AND TECHNOLOGY Implementation of CNN and ANN for Fashion-MNIST-Dataset using Different Optimizers," *Indian J. Sci. Technol.*, vol. 15, no. 47, pp. 2639–2645, 2022, [Online]. Available: https://www.indjst.org/

[24] A. S. Henrique *et al.*, "Classifying Garments from Fashion-MNIST Dataset Through CNNs," *Adv. Sci. Technol. Eng. Syst. J.*, vol. 6, no. 1, pp. 989–994, 2021, doi: 10.25046/aj0601109.

[25] O. M. Khanday, S. Dadvandipour, and M. A. Lone, "Effect of filter sizes on image classification in CNN: A case study on CFIR10 and fashion-MNIST datasets," *IAES Int. J. Artif. Intell.*, vol. 10, no. 4, pp. 872–878, 2021, doi: 10.11591/ijai.v10.i4.pp872-878.

[26] Y. Tang, H. Cui, and S. Liu, "Optimal Design of Deep Residual Network Based on Image Classification of Fashion-MNIST Dataset," *J. Phys. Conf. Ser.*, vol. 1624, no. 5, pp. 0–7, 2020, doi: 10.1088/1742-6596/1624/5/052011.

[27] M. Kayed, A. Anter, and H. Mohamed, "Classification of Garments from Fashion MNIST Dataset Using CNN LeNet-5 Architecture," *Proc. 2020 Int. Conf. Innov. Trends Commun. Comput. Eng. ITCE 2020*, no. June, pp. 238–243, 2020, doi: 10.1109/ITCE48509.2020.9047776.

[28] H. H. Zhu *et al.*, "Space-efficient optical computing with an integrated chip diffractive neural network," *Nat. Commun.*, vol. 13, no. 1, pp. 1–9, 2022, doi: 10.1038/s41467-022-28702-0.

[29] T. Hur, L. Kim, and D. K. Park, "Quantum convolutional neural network for classical data classification," *Quantum Mach. Intell.*, vol. 4, no. 1, pp. 1–18, 2022, doi: 10.1007/s42484-021-00061-x.

[30] O. Nocentini, J. Kim, M. Z. Bashir, and F. Cavallo, "Image Classification Using Multiple Convolutional Neural Networks on the Fashion-MNIST Dataset," *Sensors*, vol. 22, no. 23, p. 9544, Dec. 2022, doi: 10.3390/s22239544.

[31] S. Yang, S. Hoque, and F. Deravi, "Adaptive Template Reconstruction for Effective Pattern Classification," *Sensors*, vol. 23, no. 15, p. 6707, Jul. 2023, doi: 10.3390/s23156707.

[32] S. Coleman, D. Kerr, and Y. Zhang, "Image Sensing and Processing with Convolutional Neural Networks," *Sensors*, vol. 22, no. 10, p. 3612, May 2022, doi: 10.3390/s22103612.

[33] V. Terziyan, D. Malyk, M. Golovianko, and V. Branytskyi, "Hyper-flexible Convolutional Neural Networks based on Generalized Lehmer and Power Means," *Neural Networks*, vol. 155, pp. 177–203, Nov. 2022, doi: 10.1016/j.neunet.2022.08.017.

[34] K. Wang, C. Xu, G. Li, Y. Zhang, Y. Zheng, and C. Sun, "Combining convolutional neural networks and self-attention for fundus diseases identification," *Sci. Rep.*, vol. 13, no. 1, p. 76, Jan. 2023, doi: 10.1038/s41598-022-27358-6.

[35] E. Chu, D. Li, and Y. Tong, "Optimized federated learning based on Adagrad algorithm and algorithm optimization," *Appl. Comput. Eng.*, vol. 19, no. 1, pp. 9–17, Oct. 2023, doi: 10.54254/2755-2721/19/20231000.

[36] I. Naseer, S. Akram, T. Masood, A. Jaffar, M. A. Khan, and A. Mosavi, "Performance Analysis of State-of-the-Art CNN Architectures for LUNA16," *Sensors*, vol. 22, no. 12, p. 4426, Jun. 2022, doi: 10.3390/s22124426.

[37] Y. S. Saboo, S. Kapse, and P. Prasanna, "Convolutional Neural Networks (CNNs) for Pneumonia Classification on Pediatric Chest Radiographs," *Cureus*, Aug. 2023, doi: 10.7759/cureus.44130.

[38] M. Uppal *et al.*, "Enhancing accuracy in brain stroke detection: Multi-layer perceptron with Adadelta, RMSProp and AdaMax optimizers," *Front. Bioeng. Biotechnol.*, vol. 11, Sep. 2023, doi: 10.3389/fbioe.2023.1257591.

[39] R. Liang, X. Chang, P. Jia, and C. Xu, "Mine Gas Concentration Forecasting Model Based on an Optimized BiGRU Network," *ACS Omega*, vol. 5, no. 44, pp. 28579–28586, Nov. 2020, doi: 10.1021/acsomega.0c03417.

[40] S. B. ud din Tahir, A. Jalal, and K. Kim, "Wearable Inertial Sensors for Daily Activity Analysis Based on Adam Optimization and the Maximum Entropy Markov Model," *Entropy*, vol. 22, no. 5, p. 579, May 2020, doi: 10.3390/e22050579.