Performance analysis of optimization algorithms for convolutional neural network-based handwritten digit recognition

Abdulhakeem Qusay Albayati[1], Sarmad A. Jameel Altaie[1], Wasseem N. Ibrahem Al-Obaydy[2], Farah Flayyeh Alkhalid[3]
[1]Department of Computer Engineering, University of Technology-Iraq, Baghdad, Iraq
[2]Department of Computer Engineering, College of Engineering, Al-Iraqia University, Baghdad, Iraq
[3]Department of Control and Systems Engineering, University of Technology-Iraq, Baghdad, Iraq

Article Info

Article history:

Received Feb 4, 2023
Revised May 13, 2023
Accepted May 21, 2023

Keywords:

Convolutional neural network
Deep learning
Handwritten digit recognition
Optimization algorithms

ABSTRACT

Handwritten digit recognition has been widely researched by the recognition society during the last decades. Deep convolutional neural networks (CNN) have been exploited to propose efficient handwritten digit recognition approaches. However, the CNN model may need an optimization algorithm to achieve satisfactory performance. In this work, a performance evaluation of seven optimization methods applied in a straightforward CNN architecture is presented. The inspected algorithms are stochastic gradient descent (SGD), adaptive gradient (AdaGrad), adaptive delta (AdaDelta), adaptive moment estimation (ADAM), maximum adaptive moment estimation (AdaMax), nesterov-accelerated adaptive moment estimation (Nadam), and root mean square propagation (RMSprop). Experiments have been carried out on two standard digit datasets, namely Modified National Institute of Standards and Technology (MNIST) and Extended MNIST (EMNIST). The results have shown the superior performance of RMSprop and Adam algorithms over the peer methods, respectively.

Corresponding Author:

Wasseem N. Ibrahem Al-Obaydy
Department of Computer Engineering, College of Engineering, Al-Iraqia University
Baghdad, Iraq
Email: wasseem.nahi@aliraqia.edu.iq

1. INTRODUCTION

Although visual recognition tasks come naturally to humans, they remain challenging and sophisticated for machine autonomy [1]. Handwritten digit recognition is one of the automated image classification and recognition problems that received considerable attention by researchers over the past few years [2]. The importance of this technology stems from its numerous applications, for example optical character recognition (OCR), signature verification, text interpretation, manipulation, and many more [3]. Nowadays, deep learning techniques play a significant role in solving handwritten digit recognition and other image recognition problems, such as medical imaging, especially when the dataset size is large [4]-[6]. However, deep neural networks are very complex models that often require the estimation of a large number of parameters. Assigning inappropriate values to these parameters may negatively affect the network's convergence speed and accuracy. Optimization algorithms have been proposed to assign typical values to the network parameters and improve the overall performance [7]. The following paragraphs review the most widespread optimization algorithms.

The well-known stochastic gradient descent (SGD) [8], [9] is reasonably both effective and efficient since computing 1^{st} partial derivative w.r.t. each parameter has equivalent processing difficulty to function evaluation. Nevertheless, the learning rate initialization must be adjusted for SGD. The convergence will slow down or vibrate and the training algorithm may be diverged as a consequence of inadequate learning rate [10].

Another popular optimization algorithm is the adaptive learning rate method (AdaGrad) [11]. By adapting learning rate to parameters, AdaGrad updates unusual parameters more than recurring parameters. Although AdaGrad avoids the necessity for a manual learning rate adjustment, the denominator's gradients' squares accumulation may decrease learning rate as well as delay the convergence speed.

To remedy AdaGrad's flaw, AdaDelta and Adam methods were proposed [12], [13]. The adaptive delta (AdaDelta) is an SGD version based on the adaptive learning rate per dimension. It tackles two limitations: the necessity for a manually chosen global learning rate and the continuous decline in learning rates during training. Instead of accumulating all prior gradients in AdaGrad, AdaDelta adjusts the learning rates according to a changing window of gradient updates [12], [14].

Adam refers to the adaptive moment estimation algorithm that aims to calculate the rates of adaptive learning for each parameter. It uses training data rather than the classical stochastic gradient descent procedure to iteratively update the network weights [13]. In order to converge faster, Adam uses momentum and adaptive learning rates. The maximum adaptive moment estimation (AdaMax) is an extension of Adam by generalizing to infinity norm. It is effective in terms of computing and is ideally suited to situations with large datasets [15]. The Nesterov-accelerated adaptive moment estimation (Nadam) algorithm is another improved version of Adam which alters the Adam's moment component by Nesterov´s accelerated gradient (NAG) to speed up the convergence process and enhance the learned models [16].

Instead of using cumulative or accumulative sum of squared gradients like Adagrad, the root mean square propagation (RMSProp) method uses exponentially decaying average of squared gradient and does not consider the history from the extreme past. As a result, the algorithm converges rapidly once it finds the locally convex bowl [17]. During an update, Chandra and Sharma [18] also applied the Laplacian score approach in conjunction with an adaptive learning rate to modify the weights in mini-batches.

Despite the abundance of optimization algorithms, existing deep learning-based handwritten digit recognition approaches have not investigated their performance when different optimization algorithm is applied. Hence, the reported results may not reflect the optimal performance of these approaches. The main contribution of this paper is an empirical examination of the performance of seven optimization algorithms applied in a convolutional neural network (CNN)-based handwritten digit recognition approach. To the best of our knowledge, this study represents the first attempt to highlight the effect of these algorithms on CNN's accuracy, loss, error, and processing time in the handwritten digit recognition problem. The remaining part of this article is structured as follows. Section 2 describes the CNN architecture. Section 3 elaborates the optimization algorithms used in this evaluation. The standard handwritten digit datasets are explained in section 4. Section 5 reports the experimental results. Finally, section 6 outlines the conclusion of this work.

2. CONVOLUTIONAL NEURAL NETWORK (CNN)

This section provides a detailed explanation of the convolutional neural network used in our analytical study. The network consists of input (convolutional) layer, pooling layer, dropout (regularization) layer, flatten layer, fully connected layer, and output layer as shown in Figure 1 [19], [20]. The convolutional layer receives the input images and is composed of thirty two five-by-five feature maps with a rectifier transfer function. The convolution process is demonstrated as follows:

– Each layer is denoted by l^{th} where $l = 1$ is the first layer and $l = L$ is the last layer.
– The input image is denoted by x of size $H \times W$ with indices i and j.
– The filter or kernel is denoted by w of size $k_1 \times k_2$ with indices m and n.
– The weight matrix that connects neurons of layer l with those of layer $l - 1$ is denoted by $w^l_{m,n}$.
– At layer l, the bias unit is denoted by b^l.
– At layer l, the convolved input vector $x^l_{i,j}$ is defined by:

$$x^l_{i,j} = \sum_{m}^{k_1} \sum_{n}^{k_2} w^l_{m,n} o^{l-1}_{i+m,j+n} + b^l \tag{1}$$

– At layer l, the output vector $o^l_{i,j}$ is defined by applying the activation function $f(.)$ to the convolved input vector:

$$o^l_{i,j} = f(x^l_{i,j}) \tag{2}$$

The pooling layer is designed with a pool size of 2×2 and it takes the max in the pool. The purpose of dropout layer is to decrease the network overfitting by excluding 20% of the layer's neurons. The flatten layer aims at transforming the 2D matrix data to a flatten vector that can be processed by the subsequent fully connected layer which consists of one hundred twenty-two neurons and a rectifier transfer function. The output layer is comprised of ten neurons for ten classes and employs a softmax transfer function to give a probability score for each class prediction. The CNN model is trained using each of the seven optimization algorithms described in the next section.

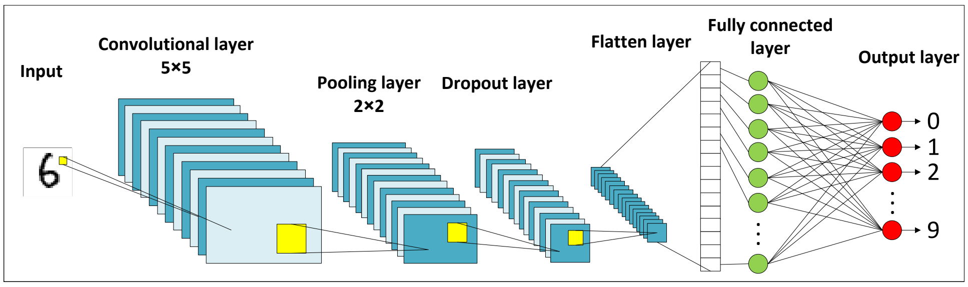


Figure 1. Convolutional neural network

3. OPTIMIZATION ALGORITHMS

This section presents a thorough description of the seven optimization algorithms used in this work. The gradient descent (GD) optimization algorithm is the one that is most frequently applied to address classification issues. The performance of CNN was enhanced using a variety of GD algorithm optimizers, including SGD, AdaGrad, AdaDelta, Adam, AdaMax, Nadam, and RMSprop. The following subsections elaborate each algorithm in more detail.

3.1. Stochastic gradient descent (SGD)

The most prevailing algorithm used to train deep neural networks is SGD. This technique has shown a well performance in many applications. It moves the parameters of the model in the opposite direction of the minibatch-evaluated gradient to achieve a reduced loss [21]. Its fast work and prevention of redundant computation have made the SGD algorithm the preferred choice in deep learning techniques. In this method, the training data and learning rate are denoted by x_i, y_i and η, respectively. The term $\nabla_{\theta i} E(\theta_i)$ refers to a gradient of error (loss) function $E(\theta_i)$ with respect to the parameter θ_i, and momentum factor (m). Then (3),

$$\theta_{i+1} = \theta_i - \eta_i \nabla_{\theta i} E(\theta_i; x_i, y_i) \tag{3}$$

is used to update the learning parameters for each training pair, until the stopping condition is met, which yields the result θ_{i+1} [22]. However, the SGD method suffers from the oscillation in the gradient direction due to two reasons. Firstly, the extra noise resulted from the arbitrary selection. Secondly, the procedure of blind search in the solution space. Moreover, the gradients' variance and the movement direction in SGD are large and biased, respectively [7].

3.2. Adaptive gradient (AdaGrad)

Another gradient-based optimization algorithm that individually adjusts the learning rates of model parameters is AdaGrad. The primary concept of this technique is adopting a lower learning rate for parameters that correspond to recurrent attributes and a higher learning rate for parameters that correspond to rare traits. This is performed by using all the historical squared gradient values [23]. The Adagrad optimizer formulates the update equation as defined in (4).

$$\theta_{t+1,i} = \theta_{t,i} - \eta \frac{1}{\sqrt{G_{t,ii} + \varepsilon}} g_{t,i} \tag{4}$$

Where $g_{t,i}$ stands for the loss function gradient for parameter $\theta_{t,i}$ at time step t. It represents a smoothing term considered to prevent the division by 0. The diagonal matrix denoted by $G_{t,ii}$ represents the sum of squares of gradient for parameter $\theta_{t,i}$ at time step t. This method eliminates the necessity for adjusting the learning rate manually. However, the learning rate may shrink, and the convergence may decelerate due to accumulating the squares of gradients in the denominator [22].

3.3. Adaptive delta (AdaDelta)

AdaDelta is a stochastic optimization technique that was introduced to alleviate the limitation of AdaGrad, through reducing the accumulation in the denominator. It is fully automatic and does not need a default value of learning rate. This optimizer limits the number of previously gathered gradients to a defined size ω rather than aggregating all previously squared gradients [24]. Instead of inefficiently storing gradients, the sum of gradients is repeatedly calculated as a falling average of all ω squares of gradients before it. Then, the running average $E[g^2]_t$ at time step t is dependent solely on the current gradient and the past average [12], as defined by (5). Typically, δ is set at about 0.9. The update equation of SGD is rewritten concerning the update vector parameter as defined by (6) and (7).

$$E[g^2]_t = \delta E[g^2]_{t-1} + (1-\delta)g_t^2 \tag{5}$$

$$\Delta\theta_t = -\eta g_{t,i} \tag{6}$$

$$\theta_{t+1} = \theta_t + \Delta\theta_t \tag{7}$$

Then AdaDelta is defined by (8).

$$\Delta\theta_t = -\eta \frac{1}{\sqrt{E[g^2]_t + \varepsilon}} g_t \tag{8}$$

3.4. Adaptive moment estimation (ADAM)

Adam is a method for optimizing stochastic objective functions. It determines individual adaptive learning rates for each parameter using estimates of the gradients' first and second moments. It is easy to apply, efficient in terms of computation and low-memory, resistent to gradients' diagonal rescaling, and suitable for tasks involving a significant amount of input parameters and data [13]. As in AdaDelta, Adam retains an exponentially decaying average of both preceding gradients m_t and previous squared gradients ν_t as shown in (9) and (10) [23]. Computing the bias-corrected first and second-moment estimations eliminates the biases to produce equations (11) and (12). Then, the parameters are updated as defined by (13). Adam is an effective technique when used to solve problems with extremely noisy and sparse gradients as well as non-stationary objectives [25].

$$m_t = \beta_1 m_{t-1} + (1-\beta_1)g_t \tag{9}$$

$$\nu_t = \beta_2 \nu_{t-1} + (1-\beta_2)g_t^2 \tag{10}$$

$$\widehat{m}_t = \frac{m_t}{1-\beta_1^t} \tag{11}$$

$$\widehat{\nu}_t = \frac{\nu_t}{1-\beta_2^t} \tag{12}$$

$$\theta_t = \theta_{t-1} - \eta \frac{1}{\sqrt{\widehat{\nu}_t} + \varepsilon} \widehat{m}_{t-1} \tag{13}$$

3.5. Maximum adaptive moment estimation (AdaMax)

AdaMax is an Adam's variation that is based on the norm of infinity. It is a type of adaptive SGD. The main advantage of AdaMax over SGD is that it has a very little sensitivity against the choice of hyper-parameters. The Adam's second momentum element is fully utilized in the AdaMax equation. This provides a more consistent solution [25]. The velocity parameter of the approach modifies the gradient across the ν_t and current gradient $|g_t|^2$ terms in inverse proportion to the previous gradients' L^2 norm:

$$\nu_t = \beta_2 \nu_{t-1} + (1-\beta_2)|g_t|^2 \tag{14}$$

the gradient's convergence with L^{∞} norm is defined by (15).

$$\nu_t = \beta_2^{\infty}\nu_{t-1} + (1-\beta_2^{\infty})|g_t|^{\infty} = max(\beta_2\nu_{t-1}, |g_t|) \tag{15}$$

Furthermore, the AdaMax update parameter is described by (16), where $\nu_t = max(\beta_2\nu_{t-1}, |g_t|)$ with initial value $\nu_0 = 0$. For machine learning experiments, good default values of β_1, β_2 and η are 0.9, 0.999 and 0.002, respectively [24].

$$\theta_t = \theta_{t-1} - \eta\frac{\widehat{m}_t}{\nu_t} \tag{16}$$

3.6. Nesterov-accelerated adaptive moment estimation (Nadam)

The momentum and the adaptive learning rate component are the two primary parts of Adam's optimization algorithm. A comparable approach called Nesterov's accelerated gradient (NAG) outperformed the conventional momentum [25]. Nadam modifies Adam's momentum component $\widehat{m}_t$ defined by (17) by updating the gradient g_t in addition to updating the parameters θ_{t-1} rather than the twice addition of the momentum component. Furthermore, the Nadam new update rule is expressed by (18). This modification is to benefit from the NAG, which enhances the speed of convergence and the learned models quality [26].

$$\widehat{m}_t = \beta_{t+1}m_t + (1-\beta_t)g_t \tag{17}$$

$$\theta_t = \theta_{t-1} - \eta\frac{1}{\sqrt{\nu_t + \varepsilon}}\widehat{m}_t \tag{18}$$

3.7. RMSprop

RMSprop is a popular adaptive stochastic algorithm for deep neural network training. It alters AdaGrad such that the gradient accumulation is optimized to operate in the non-convex environment. An exponentially weighted moving average is created by aggregating gradients. In addition, current knowledge of the gradient is preserved by RMSProp, discarding the past [27]. The RMSProp method divides the learning rate using an average of squared gradients with exponential decay [28], [13], and it is described by (19) and (20), where the suitable values of γ and learning rate η are 0.9, 0.001, respectively. The RMSProp performs effectively in both offline and online environments, and it demands less tuning in comparison with the SGD [29].

$$E[g^2]_t = \gamma E[g^2]_{t-1} + (1-\gamma)g_t^2 \tag{19}$$

$$\theta_t = \theta_{t-1} - \eta\frac{1}{\sqrt{E[g^2]_t + \varepsilon}}g_t \tag{20}$$

4. DATASETS

Over the recent years, a variety of different handwritten image databases have been disclosed to facilitate the handwritten image recognition research. Two most popular datasets namely, Modified National Institute of Standards and Technology (MNIST) [30] and Extended MNIST (EMNIST) [31] were used in our experiments. These datasets were developed and released by National Institute of Standards and Technology (NIST) to evaluate the performance of handwritten digit recognition approaches. The next two subsections describe each dataset in more detail.

4.1. Modified National Institute of Standards and Technology (MNIST)

The MNIST dataset [30] consists of seventy thousand instances from which $60,000$ are used for training and $10,000$ are used for testing. This dataset was gathered from two sources: NIST's Special Database 1 and NIST's Special Database 3. The former was gathered from high school students, while the latter was retrieved from Census Bureau staff. The choice of the training and testing sets was made to avoid having the same author contribute to both sets. The training set contains writing examples from more than 250 different authors [32]. Preprocessing was applied to the original images. First, photos must be normalized to fit within a 20×20 pixels box while preserving the aspect ratio. The black and white pictures were then converted to grayscale using an anti-aliasing filter. To enlarge the images into a 28×28 pixels box and to ensure that the center of the digit's mass matched its center, blank padding was finally applied [33]. Figure 2 shows samples of the MNIST dataset.

4.2. Extended Modified National Institute of Standards and Technology (EMNIST)

The EMNIST database introduced in April 2017 by Cohen *et al.* [31] provides handwritten digits and characters constructed from the NIST Special Database 19, which contains an entire library of training resources for character recognition and handwritten documents [34]. The size of EMNIST is considerably larger than that of MNIST, and the images were collected from a different source. The photos were also converted to a 28×28 pixels format similar to MNIST [33]. Figure 3 demonstrates samples of the EMNIST dataset. The EMNIST digit dataset contains $280,000$ characters in 10 balanced classes. It comprises $235,000$ training images, $40,000$ testing images, and 5000 validation set images. It presents the data in two different forms, all of which include identical information. It is offered in MATLAB format in the first format. In addition, the second form gives it in a binary format similar to that of the original MNIST dataset [35].

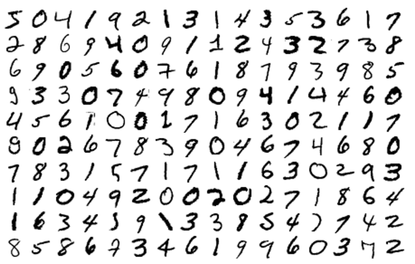


Figure 2. A hundred and sixty samples from the MNIST

Figure 3. EMNIST dataset digits and letters samples

5. EXPERIMENTS AND RESULTS

This section provides an experimental performance assessment of the previously mentioned optimization algorithms. The experiments were conducted using the benchmark MNIST and EMNIST digit datasets. Various performance metrics including processing time, error rate, loss, accuracy, validation loss, and validation accuracy were adopted to demonstrate the effectiveness of competing algorithms. The experiments were implemented using the Keras Python library in the Jupyter notebook platform on a Windows 10 MSI laptop with Intel Core i7-11800H CPU 2.3 GHz and 16 GB RAM.

Table 1 shows the resulting outcomes of MNIST experiments. As evident by the results highlighted in bold, the Adamax method showed the fastest performance. However, the RMSprop algorithm showed an outstanding performance outperforming the peer algorithms in terms of CNN error, loss, accuracy, validation loss, and validation accuracy.

Table 1. Results on MNIST dataset

Optimizer	Processing time (msec)	CNN Error (%)	Loss	Accuracy	Val_Loss	Val_Accuracy
SGD	41.801	4.43	0.1941	0.9434	0.1591	0.9557
Adagrad	41.744	8.63	0.3616	0.8967	0.3145	0.9137
Adadelta	43.284	23.71	1.4997	0.7166	1.4084	0.7629
Adam	43.918	1.10	0.0147	0.9952	0.0386	0.9890
Adamax	**34.900**	1.36	0.0330	0.9903	0.0371	0.9864
Nadam	36.586	1.18	0.0146	0.9952	0.0388	0.9882
RMSprop	45.059	**1.06**	**0.0102**	**0.9969**	**0.0347**	**0.9894**

The results of EMNIST digit dataset experiments are reported in Table 2. The scores highlighted in bold indicate that the SGD method achieved the fastest performance, while the Adamax attained the lowest validation loss. However, the Adam algorithm outperformed its counterparts according to CNN error, loss, accuracy, and validation accuracy.

The chart in Figure 4 compares the accuracy rates of each algorithm in both datasets. As can be seen in the figure, every algorithm, except RMSprop, produced a higher accuracy rate in EMNIST than that in MNIST. One possible reason to justify this performance is that the size of training data in EMNIST is much larger than that in MNIST. It should also be mentioned that the Adam, Adamax, Nadam, and RMSprop methods produced higher accuracy rates than that of SGD, Adagrad, and Adadelta algorithms in both datasets.

Table 2. Results on EMNIST digit dataset

Optimizer	Processing time (msec)	CNN Error (%)	Loss	Accuracy	Val_Loss	Val_Accuracy
SGD	**134.96**	1.85	0.0757	0.9779	0.0623	0.9815
Adagrad	140.17	4.53	0.1747	0.9510	0.1573	0.9547
Adadelta	144.32	9.48	0.3677	0.8996	0.3440	0.9052
Adam	137.76	**0.50**	**0.0071**	**0.9978**	0.0208	**0.9950**
Adamax	148.07	0.62	0.0164	0.9951	**0.0207**	0.9938
Nadam	147.71	0.59	0.0076	0.9976	0.0242	0.9941
RMSprop	141.03	0.53	0.0111	0.9969	0.0230	0.9947

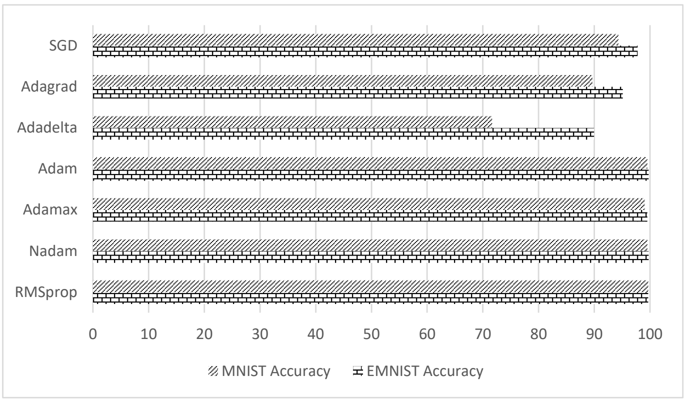


Figure 4. Accuracy of each algorithm in both datasets

6. CONCLUSION

This paper has presented a performance evaluation of seven optimization algorithms for handwritten digit recognition using a convolutional neural network. These algorithms include SGD, AdaGrad, AdaDelta, ADAM, AdaMax, Nadam, and RMSprop. We used two benchmark datasets namely, MNIST and EMNIST to conduct the experiments. The performance was assessed using several standard metrics including processing time, error rate, loss, accuracy, validation loss, and validation accuracy. The experimental results exposed two preferred methods. The overall highest scores of MNIST experiments proved that the RMSprop method is the best choice for CNN-based handwritten digit recognition. On the other hand, the Adam method was proved as the best approach as demonstrated by the highest scores of EMNIST experiments. Although the two methods showed excellent performance results, they still cannot reach 100% accuracy. This can be considered as a limitation of the two optimization algorithms, which requires further investigation.

REFERENCES

[1] Z. Li, L. Ma, X. Ke, and Y.Wang, “Handwritten digit recognition via active belief decision trees,” in *2016 35th Chinese Control Conference (CCC)*, 2016, pp. 7021–7025, doi: 10.1109/ChiCC.2016.7554464.

[2] P. Kuang, W.-N. Cao, and Q. Wu, “Preview on structures and algorithms of deep learning,” in *2014 11th International Computer Conference on Wavelet Actiev Media Technology and Information Processing (ICCWAMTIP)*, 2014, pp. 176–179, doi: 10.1109/ICCWAMTIP.2014.7073385.

[3] C.-L. Liu, K. Nakashima, H. Sako, and H. Fujisawa, “Handwritten digit recognition: benchmarking of state-of-the-art techniques,” *Pattern recognition*, vol. 36, no. 10, pp. 2271–2285, 2003, doi: 10.1016/S0031-3203(03)00085-2.

[4] F. F. Alkhalid, A. Q. Albayati, and A. A. Alhammad, “Expansion dataset covid-19 chest x-ray using data augmentation and histogram equalization,” *International Journal of Electrical and Computer Engineering*, vol. 12, no. 2, pp. 1904–1909, 2022, doi: 10.11591/ijece.v12i2.pp1904-1909.

[5] N. Sharma, R. Sharma, and N. Jindal, “Machine learning and deep learning applications-a vision,” *Global Transitions Proceedings*, vol. 2, no. 1, pp. 24–28, 2021, doi: 10.1016/j.gltp.2021.01.004.

[6] M. M. A. Ghosh and A. Y. Maghari, “A comparative study on handwriting digit recognition using neural networks,” in *2017 international conference on promising electronic technologies (ICPET)*, 2017, pp. 77–81, doi: 10.1109/ICPET.2017.20.

[7] S. Sun, Z. Cao, H. Zhu, and J. Zhao, “A survey of optimization methods from a machine learning perspective,” *IEEE transactions on cybernetics*, vol. 50, no. 8, pp. 3668–3681, 2019, doi: 10.1109/TCYB.2019.2950779.

[8] G. Hinton *et al.*, “Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups,” *IEEE Signal processing magazine*, vol. 29, no. 6, pp. 82–97, 2012, doi: 10.1109/MSP.2012.2205597.

[9] L. Deng *et al.*, “Recent advances in deep learning for speech research at Microsoft,” in *2013 IEEE international conference on acoustics, speech and signal processing*, 2013, pp. 8604–8608, doi: 10.1109/ICASSP.2013.6639345.

[10] L. Bottou, “Stochastic gradient descent tricks,” in *Neural networks: Tricks of the trade*, Berlin, Heidelberg: Springer, 2012, pp. 421–436, doi: 10.1007/978-3-642-35289-8_25.

[11] J. Duchi, E. Hazan, and Y. Singer, “Adaptive subgradient methods for online learning and stochastic optimization,” *Journal of machine learning research*, vol. 12, no. 7, 2011, [Online]. Available: http://jmlr.org/papers/v12/duchi11a.html.

[12] M. D. Zeiler, “Adadelta: an adaptive learning rate method,” *arXiv preprint arXiv:1212.5701*, 2012, doi: 10.48550/arXiv.1212.5701.

[13] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” *arXiv preprint arXiv:1412.6980*, 2014, doi: 10.48550/ARXIV.1412.6980.

[14] F. F. Alkhalid, “The effect of optimizers in fingerprint classification model utilizing deep learning,” *Indonesian Journal of Electrical Engineering and Computer Science*, vol. 20, no. 2, pp. 1098–1102, 2020, doi: 10.11591/ijeecs.v20.i2.pp1098-1102.

[15] M. Sohaib and J.-M. Kim, “Fault diagnosis of rotary machine bearings under inconsistent working conditions,” *IEEE Transactions on Instrumentation and Measurement*, vol. 69, no. 6, pp. 3334–3347, 2019, doi: 10.1109/TIM.2019.2933342.

[16] I. Sutskever, J. Martens, G. Dahl, and G. Hinton, “On the importance of initialization and momentum in deep learning,” in *International conference on machine learning*, 2013, pp. 1139–1147, [Online]. Available: http://proceedings.mlr.press/v28/sutskever13.html.

[17] P. Bahar, T. Alkhouli, P. Jan-Thorsten, C. J.-S. Brix, and H. Ney, “Empirical investigation of optimization algorithms in neural machine translation,” *The Prague Bulletin of Mathematical Linguistics*, vol. 108, no. 1, p. 13, 2017, doi: 10.1515/pralin-2017-0005.

[18] B. Chandra and R. K. Sharma, “Deep learning with adaptive learning rate using laplacian score,” *Expert Systems with Applications*, vol. 63, pp. 1–7, 2016, doi: 10.1016/j.eswa.2016.05.022.

[19] H. H. Aghdam and E. J. Heravi, *Guide to convolutional neural networks*. New York, NY, USA: Springer, 2017.

[20] W. S. Ahmed and A. a. A. Karim, “Motion classification using CNN based on image difference,” in *2020 5th International Conference on Innovative Technologies in Intelligent Systems and Industrial Applications (CITISIA)*, 2020, pp. 1–6, doi: 10.1109/CITISIA50690.2020.9371835.

[21] M. Tang, Z. Huang, Y. Yuan, C. Wang, and Y. Peng, “A bounded scheduling method for adaptive gradient methods,” *Applied Sciences*, vol. 9, no. 17, p. 3569, 2019, doi: 10.3390/app9173569.

[22] S. Ahlawat, A. Choudhary, A. Nayyar, S. Singh, and B. Yoon, “Improved handwritten digit recognition using convolutional neural networks (CNN),” *Sensors*, vol. 20, no. 12, p. 3344, 2020, doi: 10.3390/s20123344.

[23] H. Zhao, F. Liu, H. Zhang, and Z. Liang, “Research on a learning rate with energy index in deep learning,” *Neural Networks*, vol. 110, pp. 225–231, 2019, doi: 10.1016/j.neunet.2018.12.009.

[24] C. E. Nwankpa, “Advances in optimisation algorithms and techniques for deep learning,” *Advances in Science, Technology and Engineering Systems Journal*, vol. 5, no. 5, pp. 563–577, 2020, doi: 10.25046/aj050570.

[25] D. Soydaner, “A comparison of optimization algorithms for deep learning,” *International Journal of Pattern Recognition and Artificial Intelligence*, vol. 34, no. 13, p. 2052013, 2020, doi: 10.1142/S0218001420520138.

[26] T. Dozat, “Incorporating nesterov momentum into adam,” in *International Conference on Learning Representations (ICLR)*, 2016, pp. 1–4, [Online]. Available: https://cs229.stanford.edu/proj2015/054_report.pdf.

[27] S. H. Haji and A. M. Abdulazeez, “Comparison of optimization techniques based on gradient descent algorithm: A review,” *PalArch’s Journal of Archaeology of Egypt/Egyptology*, vol. 18, no. 4, pp. 2715–2743, 2021, [Online]. Available: https://archives.palarch.nl/index.php/jae/article/view/6705.

[28] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning*. Cambridge, MA, USA: MIT press, 2016.

[29] C. Garbin, X. Zhu, and O. Marques, “Dropout vs. batch normalization: an empirical study of their impact to deep learning,” *Multimedia Tools and Applications*, vol. 79, no. 19, pp. 12777–12815, 2020, doi: 10.1007/s11042-019-08453-9.

[30] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, “Gradient-based learning applied to document recognition,” *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998, doi: 10.1109/5.726791.

[31] G. Cohen, S. Afshar, J. Tapson, and A. Van Schaik, “Emnist: Extending mnist to handwritten letters,” in *2017 international joint conference on neural networks (IJCNN)*, 2017, pp. 2921–2926, doi: 10.1109/IJCNN.2017.7966217.

[32] A. Sharma, H. Bhardwaj, A. Bhardwaj, A. Sakalle, D. Acharya, and W. Ibrahim, “A machine learning and deep learning approach for recognizing handwritten digits,” *Computational Intelligence and Neuroscience*, vol. 2022, 2022, doi: 10.1155/2022/9869948.

[33] A. Baldominos, Y. Saez, and P. Isasi, “A survey of handwritten character recognition with mnist and emnist,” *Applied Sciences*, vol. 9, no. 15, p. 3169, 2019, doi: 10.3390/app9153169.

[34] A. Baldominos, Y. Saez, and P. Isasi, “Hybridizing evolutionary computation and deep neural networks: an approach to handwriting recognition using committees and transfer learning,” *Complexity*, vol. 2019, 2019, doi: 10.1155/2019/2952304.

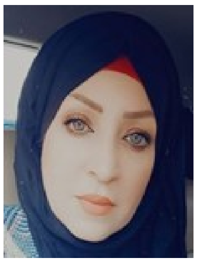
[35] R. Anuradha, N. Saranya, M. Priyadharsini, and G. D. Kumar, “Assessment of extended MNIST (EMNIST) dataset using capsule networks,” in *2019 International Conference on Intelligent Sustainable Systems (ICISS)*, 2019, pp. 263–266, doi: 10.1109/ISS1.2019.8908006.

BIOGRAPHIES OF AUTHORS

Abdulhakeem Qusay Albayati received the B.Sc. degree in computer engineering from Al-Mustansiriya University, Iraq in 2004. He received the M.Sc. degree in computer engineering from the University of Technology, Iraq in 2020. Since 2020, he has been an assistant lecturer at the Computer Engineering Department in the University of Technology-Iraq. His research interests include artificial intelligence, deep learning, machine learning, natural language processing in DL, and computer vision in DL. He has served as a reviewer for IEEE Access journal. He can be contacted at email: Abdulhakeem.Q.Ali@uotechnology.edu.iq.

Sarmad A. Jameel Altaie received the B.Sc. degree in mathematics and computer applications from Al-Nahrain University, Iraq in 2001. He received the M.Sc. degree in mathematics and computer applications from Al-Nahrain University, Iraq in 2005. Currently, he is a senior lecturer at the Computer Engineering Department, University of Technology-Iraq. His research interests include fuzzy set theory, fuzzy control, fuzzy dynamical systems, fractional derivatives, numerical methods, and approximate – analytical methods. He has served as a reviewer for IEEE Access journal. He can be contacted at email: sarmad.a.altaie@uotechnology.edu.iq.

Wasseem N. Ibrahem Al-Obaydy received the B.Sc. degree in computer and software engineering from University of Technology, Iraq in 2003. He received the M.Sc. degree in computing from University of Buckingham, United Kingdom in 2011. Currently, he is working as a lecturer at the computer engineering department, college of engineering, Al-Iraqia University, Iraq. His research interests include face recognition, feature extraction, image processing, computer vision, and pattern recognition. He has served as a reviewer for IEEE Access journal. He can be contacted at email: wasseem.nahi@aliraqia.edu.iq.



Farah Flayyeh Alkhalid received the B.Sc. degree in computer and software engineering from Al-Mustansiriya University, Iraq in 2004. She received the M.Sc. degree in computer engineering from Al-Nahrain University, Iraq in 2013. She work as an assistant professor at the control and systems engineering department, University of Technology-Iraq. Her research interests include network and communication, database systems, computer maintenance, microprocessor and tiny microprocessor, and deep learning. She can be contacted at email: farah.f.alkhalid@uotechnology.edu.iq.