International Journal of Innovative Technology and Exploring Engineering (IJITEE)
ISSN: 2278-3075 (Online), Volume-10 Issue-4, February 2021

Learning Rate Optimization in CNN for Accurate Ophthalmic Classification

Mahmoud Smaida, Serhii Yaroshchak, Ahmed Y. Ben Sasi

Abstract: ***One of the most important hyper-parameters for model training and generalization is the learning rate. Recently, many research studies have shown that optimizing the learning rate schedule is very useful for training deep neural networks to get accurate and efficient results. In this paper, different learning rate schedules using some comprehensive optimization techniques have been compared in order to measure the accuracy of a convolutional neural network CNN model to classify four ophthalmic conditions. In this work, a deep learning CNN based on Keras and TensorFlow has been deployed using Python on a database that contains 1692 images, which consists of four types of ophthalmic cases: Glaucoma, Myopia, Diabetic retinopathy, and Normal eyes. The CNN model has been trained on Google Colab. GPU with different learning rate schedules and adaptive learning algorithms. Constant learning rate, time-based decay, step-based decay, exponential decay, and adaptive learning rate optimization techniques for deep learning have been addressed. Adam adaptive learning rate method. has outperformed the other optimization techniques and achieved the best model accuracy of 92.58% for training set and 80.49% for validation datasets, respectively.***

Keywords: ***CNN model, Deep learning, Ophthalmic classification, Time-based decay, Step-based decay, Exponential decay, Adaptive learning rate.***

I. INTRODUCTION

Stochastic gradient descent SGD method is an iterative algorithm used for optimizing an objective function with suitable smoothness properties [1]. This method used the learning rate to determine the step size at each iteration in order to minimize the loss function. To train a neural network model, a loss function is defined to measure the difference between the model's predictions and the label to be predicted. A specific set of weights in the neural network are searched that can make an accurate prediction, which automatically leads to a lower value of the loss function. The learning rate in machine learning is a tuning parameter used to determine the step size at each iteration in order to minimize the loss function using gradient decent algorithms. Too high learning rate will make the learning jump over minimum and too low learning rate will lead learning to take long time and may get stuck at local minimum [2].

Often the learning rate varies during training according to the learning rate schedule or by using the adaptive learning rate to achieve faster convergence, prevent fluctuations, and avoiding getting /stuck into an unwanted local minimum.

The purpose of learning rate schedule is to change the learning rate during learning often each epoch/iteration. This is done with the use of both decay and momentum. Decay leads to fix the learning in a pretty place and avoid oscillations. Momentum looks like a ball rolling down a hill and the aim is to settle it at the lowest point of the hill. When increasing momentum too high the ball will roll over minima to be found, and too low will not fulfill its purpose. Time-based decay, step-based decay and exponential decay are the most common learning rate schedules [3]. All of these algorithms are based on the systematic gradient descent optimization. However, this systematic approach can be extended to weight optimization through some mathematical equations. Thus, leading to build more effective optimization algorithms that allow the neural network to adequately learn faster and achieve better performance. In this paper, constant learning rate and learning rate decay schedules such as time-based decay, step-based decay and exponential decay were used. Furthermore, adaptive learning rate algorithms were addressed using Adagrad, Adadelta, RMSProp and Adam. Model performances using different learning rate schedules and adaptive learning rate algorithms were compared.

II. OPTIMIZATION TECHNIQUES IN DEEP LEARNING

A lot of gradient descent algorithms have been used in deep neural networks in order to increase the accuracy of the models. Learning rate is the most important factor in stochastic gradient descent algorithm to increase the model accuracy. The following are some of the most common learning rate schedules:

A. Time-based decay

In the time-based decay, the learning rate changes depend on the learning rate from the previous time epoch. The mathematical formula of the learning rate is [3]:

$$\eta_{n+1} = \frac{\eta_n}{1+dn} \quad (1)$$

Where η is the learning rate/epoch, d is the decay, and n is the epoch number.

Manuscript received on January 08, 2021.
Revised Manuscript received on January 15, 2021.
Manuscript published on February 28, 2021.
* Correspondence Author
Mahmoud Smaida*, Computer department, The National University of Water and Environmental Engineering, Revine, Ukraine. Email: m.e.smaida@nuwm.edu.ua
Serhii Yaroshchak, Applied Mathematics, The National University of Water and Environmental Engineering, Revine, Ukraine. Email: s.v.yaroshchak@nuwm.edu.ua
Ahmed Y. Ben Sasi, Computer department, The College of Industrial Technology, Misurata, Libya. Email: prof_ahmed@cit.edu.ly

B. Step-based decay

In the step-based decay, the learning rate changes according to some pre-defined steps. The mathematical form of the step-based decay is defined as [3]:

$$\eta_n = \eta_0 d^{floor\,(\frac{1+n}{r})} \quad (2)$$

Where η_n is the learning rate at epoch number n, η_0 is the initial learning rate, d is how much the learning rate should change at each drop, and r is how often the rate should be dropped. The *floor* function here drops the value of its input to 0 for all values smaller than 1.

C. Exponential decay

It is similar to the step-based decay but it is used a decreasing exponential function instead of using steps. The mathematical form defined as [3]:

$$\eta_n = \eta_0 e^{-dn} \quad (3)$$

Where d is a decay parameter, and n is the epoch number.

D. Adaptive learning rate

Adaptive learning rate is referred when the performance of the model on the training dataset can be monitored by the learning algorithm, and in response the learning rate can be adjusted [4]. In traditional gradient descent, each forward pass produces a loss value that can be used for optimization. and backpropagation generates the actual gradients in order to perform the optimization, Therefore, the optimizer algorithm used determines how optimization is performed to apply what change in the weights of the neural network in order to improve loss during the next iteration. Many new optimizers have been created over the years and mostly belong to the class of adaptive optimizers. Unlike gradient descent optimizers, which statically adapt weights at a fixed learning rate across all parameters, adaptive optimizers are more flexible. Some of these optimizers are as follows:

1. **Adagrad**

Adagrad is an optimization strategy that keeps running the sum of squared gradients during optimization. There is no momentum term in this strategy. Adagrad method is to set parameters appropriate to the learning rate, to perform large updates to sporadic parameters and small updates to frequent parameters. Therefore, Adagrad method is very suitable for processing dispersed data. Adagrad method determines different learning rates for different parameters θ based on the previous gradients calculated for each parameter. At each time step t, Adagrad method determines a different learning rate for each parameter θ, updates the corresponding parameters, and then performs conducting routing. For simplicity, the gradient of the loss function for the parameter θ_t at time t was set as g_t.The parameters update form are as follows [5]:

$$G_t = \sum_{n=0}^{t} g^2{}_n \quad (4)$$

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{G_t + \epsilon}} \cdot g_t \quad (5)$$

In the above formula, ϵ is the smoothing factor to avoid the divisor to zero. If G_t in a given direction is smaller, the corresponding learning rate is greater, that is, larger updates of the parameters which occur infrequently are made. Over time, the G_t gets bigger and bigger, making the learning rate smaller and smaller. Therefore, there is no need to set the learning rate manually. In most Adagrad applications, the default value of the learning rate is 0.01. Since G_t is the sum of squares, each term is positive. Therefore, as the training process progresses, G_t will continue to grow. This means that the learning rate will continue to decrease, and the model convergence will become slower and slower. Furthermore, the training will take a long time, and even the learning rate will eventually be so small that the model stops learning completely.

2 Adadelta

It is a slight improvement on Adagrad., and works to reduce its aggressive, monotonically decreasing learning rate. Instead of accumulating all past squared gradients, Adadelta restricts the window of accumulated past gradients to some fixed size window. The parameters update form are as [5], [6]:

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{E[g^2]_t + \epsilon}} \cdot g_t \quad (6)$$

$$E[g^2]_t = \gamma E[g^2]_{t-1} + (1-\gamma) g^2{}_t \quad (7)$$

where, $E[g^2]_t$ is a weighted average at time t and γ is set to 0.95 as default value.

3 Root Mean Square Propagation RMSProp

RMSprop can be considered as a special case of Adadelta. It has been developed independently from the need to resolve Adagrad's radically diminishing learning rates. RMSprop divides the learning rate by an exponentially decaying average of squared gradients. Where γ is set to 0.9, while the default value for the learning rate η is 0.001 [5], [6].

$$E[g^2]_t = 0.9E[g^2]_{t-1} + 0.1 g^2{}_t \quad (8)$$

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{E[g^2]_t + \epsilon}} \cdot g_t \quad (9)$$

4 Adam

Adam is another type of adaptive optimizers that computes adaptive learning rates for each parameter. Adam also stores an exponentially decaying average of past squared gradients v_t like Adadelta and RMSprop and keeps an exponentially decaying average of past gradients m_t, similar to momentum. Momentum looks like a ball running down a slope, Adam running down a slope like a heavy ball with friction. Adam computes the decaying averages of past gradients and past squared gradients m_t and v_t as follows [5], [6]:

$$m_t = \beta_1 m_{t-1} + (1-\beta_1)\, g_t \quad (10)$$

$$v_t = \beta_2 v_{t-1} + (1-\beta_2)\, g^2{}_t \quad (11)$$

The first formula uses an attenuation coefficient β_1 to calculate the mean gradient; The second formula uses an attenuation coefficient β_2 to calculate the mean squared gradient. Usually both m_t and v_t are initialized to zero vectors. Since $\beta_1$and β_2 are attenuation coefficients close to 1, m_t and v_t will always be near to 0 at the start. To solve this problem, the following methods are used to optimize m_t and v_t [5], [6]:

$$\hat{m}_t = \frac{m_t}{1-\beta^t{}_1} \quad (12)$$

$$\hat{v}_t = \frac{v_t}{1-\beta^t{}_2} \quad (13)$$

The authors of Adam proposed default values of 0.9 for β_1, 0.999 for β_2, and 10^{-8} for ϵ. They then use these to update the parameters as shown in Adadelta and RMSprop, which yields to the Adam update rule:

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{\hat{v}_t + \epsilon}} . \hat{m_t} \quad (14)$$

III. RESEARCH METHODOLOGY

The block diagram of the proposed methodology CNN model is shown in Fig.2. In this research, a deep learning convolutional neural network CNN based on Keras and TensorFlow was deployed using Python for medical image classification. A number of different images, which contains four types of eye cases, namely Glaucoma, Myopia, Diabetic retinopathy, and Normal were used as a dataset. Firstly, the dataset was analyzed and preprocessed of four folders which contain 1692 of Glaucoma, Myopia, Diabetic retinopathy and Normal images, where 1200 images used for training, 246 images for testing and 246 images used for validation purpose. Which mean 15% of the total images were used for testing and 15% for validation purpose.

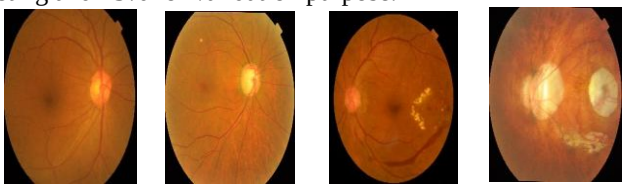

Fig. 1: Normal Fundus, Glaucoma, Diabetic retinopathy and Myopia

Next steps include the CNN model architectures, training the model, and applying all the optimizers mentioned above and obtain the accuracy metric. Finally. the obtained results of these optimization algorithms that allow CNN to learn faster and achieve better performance were compared.

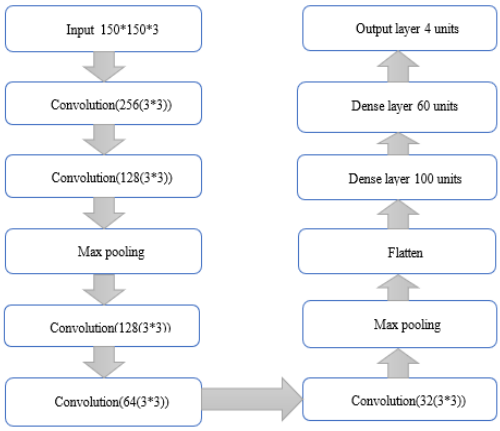

Fig.2 Block diagram of CNN model

As shown in Fig.2, the size of input image is set to 150×150 pixels with 3 RGB channels. To extract the features from the image, two convolution layers were used: the first is 256 filters of size 3×3 pixels and the second is 128 filters of size 3×3 pixels. For the pooling layer, a window of size 2×2 pixels were used, which compresses the original image size for further processing. After that, another three convolution layers were used of 128, 64, and 32 filters with size 3×3 pixels with a maximum pooling size 2×2 pixels. Then, fully connection is used (Dense 100 units and 60 units) and output layer (4 units) to predict the eye cases. CNNs adjust their filter weights through backpropagation, which means that after the forward pass, the network is able to look at the loss function and make a backward pass to update the weights.

The CNN model performance was evaluated against training, testing, and validation datasets using the accuracy measure, which reflects the ability of the model to classify the whole database into four categories with almost no error. The following represents the accuracy measure equation [7]:

$$\text{Accuracy} = \text{TP} / \text{TP} + \text{FP} + \text{FN} + \text{TN} \quad (15)$$

Where, TP is True Positive: Your predicted positive and it is true, TN is True Negative: Your predicted negative and it is true, FP is False Positive: your predicted positive but it is false and False Negative FN: Your predicted negative but it is false.

IV. EXPERIMENTAL RESULTS

According to the optimization algorithms explained above, all these algorithms were implemented using Python language on Google Colab. GPU, and applied to the CNN model to classify the ophthalmic cases in the dataset into Glaucoma, Myopia, Diabetic Retinopathy, and Normal categories to be compared for accuracy performances.

A. Constant learning rate results

Constant learning rate is the default learning rate schedule in SGD. Both momentum and decay rate are set to zero. It is difficult to choose an appropriate learning rate. By trying a set of learning rates, lr was set to 0.001 which showed a good performance to start. This can serve as the basis for further experiments with different learning rate strategies. In constant learning rate schedule, the model was defined and the learning rate value was set to constant as follows.

```
# define CNN_ model
model1 = CNN_model()
epochs = 100
# define SGD optimizer and set to default except lr
learning_rate = 0.001
sgd = SGD(lr=learning_rate, momentum=0.0, decay=0.0, nesterov=False).
```

Then, compiling, fitting the CNN model and plotting the model accuracy as shown in Fig.3.

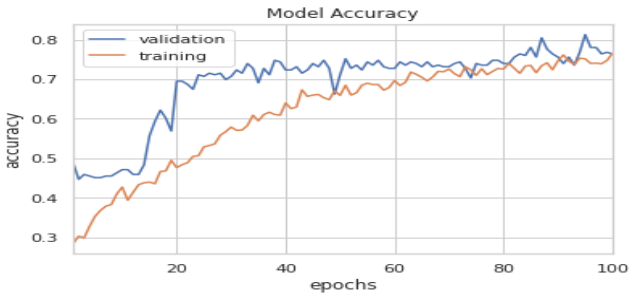

Fig.3 Constant learning rate accuracy graph

B. Time-based decay results

In the time-based decay, the SGD optimizer takes decay and learning rate and updates the learning rate by a decreasing factor in each epoch.

Momentum is another argument in SGD Optimizer that can be introduced for faster convergence. Unlike the classic SGD,

the parameter vector momentum method helps to build velocity in any direction with a constant gradient descent to prevent oscillations. Momentum was set between 0.5 to 0.9. The time-based decay can be implemented as follows, and the plot the model accuracy is shown in Fig.4.:

```
learning_rate = 0.001
decay_rate = learning_rate / epochs
learning_rate = learning_rate * (1. / (1. + decay_rate * epoch_number))
momentum = 0.5
sgd = SGD(lr=learning_rate, momentum=momentum, decay=decay_rate, nesterov=False)
```

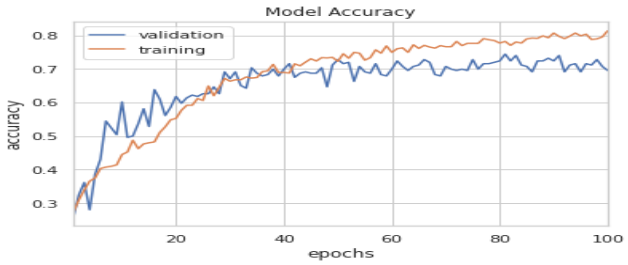

Fig.4 Time-based decay accuracy graph

C. Step-based decay results

The idea of step-based decay optimizer is to drop the learning rate by a factor every few epochs. The mathematical form was implemented to drop the learning rate by half every 10 epoch.. The learning rate schedule plot of this optimizer is illustrated in Fig.5, and the plot the model accuracy is shown in Fig.6. The step-based decay function can be defined and return the updated learning rates for use in SGD optimizer as follows:

```
def step_decay(epoch_number):
    initial_lrate = 0.001
    drop = 0.5
    epochs_drop = 10.0
    lrate = initial_lrate * math.pow(drop, math.floor((epoch_number)/epochs_drop))
 return lrate
```

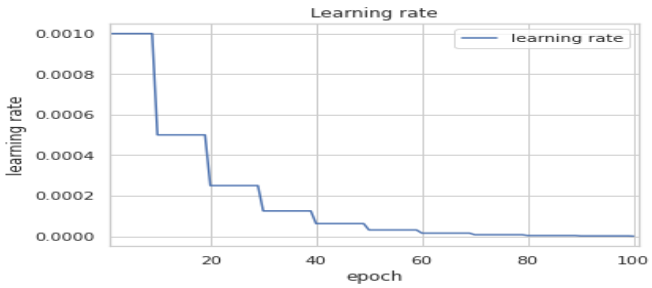

Fig.5 Learning rate of step-based decay

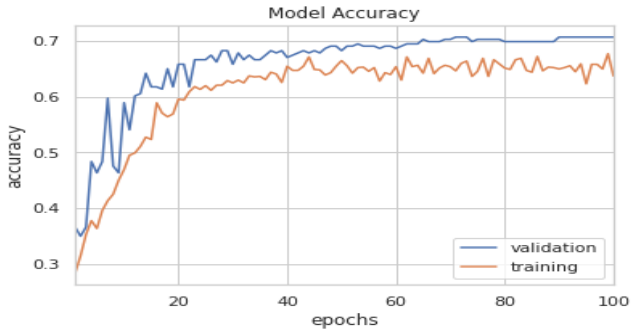

Fig.6. Step-based decay accuracy graph

D. Exponential decay results

The mathematical form of this optimizer is an exponential decay, and can be implemented in Python using the following function:

```
def exp_decay(epoch_number):
    initial_lrate = 0.001
    drop = 0.1
    lrate = initial_lrate * np.exp(-drop*epoch_number)
 return lrate
```

The learning rate schedule plot of this optimizer is illustrated in Fig.7, and the plot the model accuracy is shown in Fig.8.

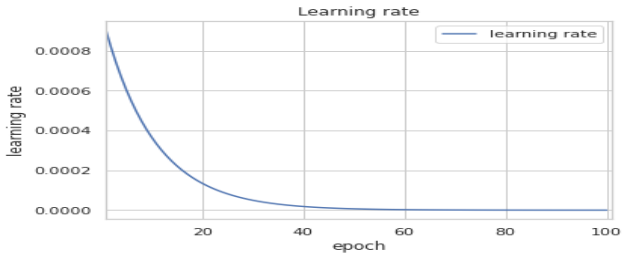

Fig.7 Learning rate of exponential Decay

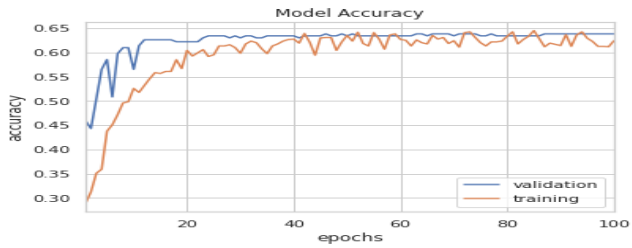

Fig.8 Exponential decay accuracy graph

E. Comparison of constant and learning rate decay schedules

The accuracy performances of CNN model using constant learning rate, time-based decay, step decay and exponential decay were evaluated. Table.1. and Fig.9 show the obtained experimental results of model accuracy comparison using learning rate decay schedules on training and validation sets, which illustrate no significant differences.

Table.1 Model accuracy comparison using learning rate decay schedules

Optimization techniques	Accuracy on training set	Accuracy on validation set
Constant learning rate	0.7658	0.7642
Time-based decay	0.8133	0.6951
Step-based decay	0.6375	0.7073
Exponential decay	0.6242	0.6382

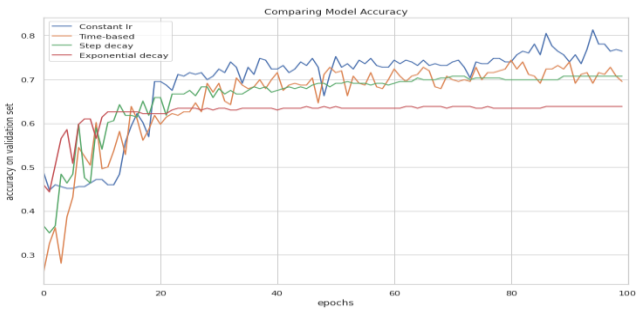

Fig.9 Learning rate decay schedules accuracy comparison graph

F. Adaptive learning rate results

The four adaptive learning algorithms can be implemented using Python. These optimizers recommended to leave their hyper-parameters at default values expect *lr* sometimes as follows:

optimizer=Adagrad(lr=0.01, epsilon=1e-08, decay=0.0)
optimizer=Adadelta(lr=1.0, rho=0.95, epsilon=1e-08, decay =0.0)
optimizer=RMSprop(lr=0.001, rho=0.9, epsilon=1e-08, deca y=0.0)
optimizer=Adam(lr=0.001, beta_1=0.9, beta_2=0.999, epsil on=1e-08, decay=0.0)

The accuracy performances of CNN model using different adaptive learning rate methods were evaluated on training and validation sets. Table.2. and Fig.10 show that the best performance was achieved by Adam which gives the top model accuracy among other adaptive learning rate algorithms.

Table.2. Model accuracy comparison using adaptive learning rate algorithms

Optimization techniques	Accuracy on training set	Accuracy on validation set
Adagrad	0.8167	0.7114
Adadelta	0.9042	0.7724
RMSprop	0.8400	0.7439
Adam	0.9258	0.8049

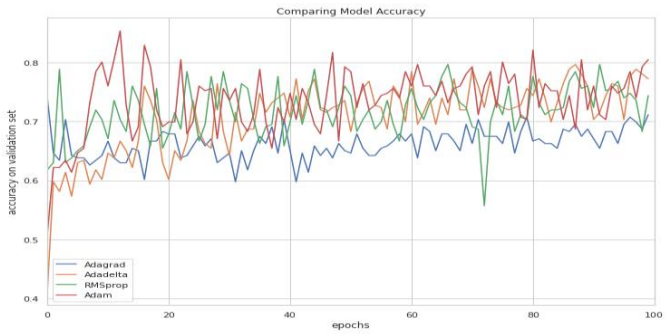

Fig.10 Adaptive learning rate algorithms accuracy comparison graph

Finally, the accuracy performances of all the learning rate schedules and adaptive learning rate methods were plotted for comparison as shown in Fig.11.

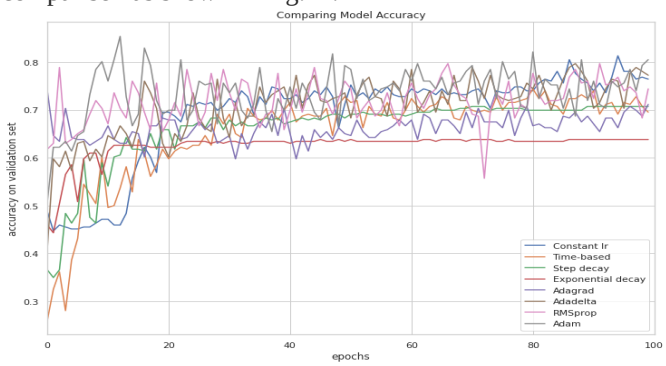

Fig.11 Accuracy performances of different learning rate schedules and adaptive learning algorithms

V. CONCLUSIONS

In this research, a deep learning CNN based on Keras and TensorFlow has been deployed using Python on a database of four types of ophthalmic cases: Glaucoma, Myopia, Diabetic retinopathy, and Normal eyes.The implemented CNN model has been trained, tested and validated on Google Colab. GPU with different learning rate schedules and adaptive learning algorithms. Constant learning rate, time-based decay, step-based decay, exponential decay, and adaptive learning rate optimization techniques for deep learning have been addressed. In conclusion, Adam adaptive learning rate method. has outperformed the other optimization techniques and achieved the best model accuracy of 92.58% for training set and 80.49% for validation datasets, respectively. Further work can be proposed using different networks in order to enhance the accuracy.

REFERENCES

1. Bottou, Léon. "Large-scale machine learning with stochastic gradient descent." *Proceedings of COMPSTAT'2010. Physica-Verlag HD*, 2010. 177-186.
2. Buduma, Nikhil, and Nicholas Locascio. "Fundamentals of deep learning: designing next-generation machine intelligence algorithms. " *O'Reilly Media, Inc.*", 2017.
3. Lau, Suki. "Learning rate schedules and adaptive learning rate methods for deep learning." *Towards Data Scienc*e 2017.
4. Goodfellow, I., Y. Bengio, and A. Courville. "Deep learning, series, the adaptive computation and machine learning series." 2016.
5. Some state of the art optimizers in neural networks, https://hackernoon.com/some-state-of-the-art-optimizers-in-neural-networks-a3c2ba5a5643, on 9/2020.
6. "An overview of gradient descent optimization algorithms", https://ruder.io/optimizing-gradient-descent/, on 9/2020.
7. Visa, Sofia, et al. "Confusion Matrix-based Feature Selection." MAICS 710 (2011): 120-127.
8. "Learning rate schedules and adaptive learning rate methods for deep learning," https://towardsdatascience.com/learning-rate-schedules-and-adaptive-learning-rate-methods-for-deep-learning-2c8f433990d1, on 9/2020.
9. Christian Daniel, Jonathan Taylor, and Sebastian Nowozin. Learning step size controllers for robust neural network training. *In Thirtieth AAAI Conference on Artificial Intelligence*, 2016.
10. Smith, Samuel L., et al. "Don't decay the learning rate, increase the batch size." *arXiv preprint arXiv:1711.00489* 2017.
11. Clevert, Djork-Arné, Thomas Unterthiner, and Sepp Hochreiter. "Fast and accurate deep network learning by exponential linear units (elus)." *arXiv preprint arXiv:*1511.07289 2015.
12. Zeiler, Matthew D. "Adadelta: an adaptive learning rate method." *arXiv preprint arXiv:*1212.5701 2012.
13. Sutskever, Ilya, et al. "On the importance of initialization and momentum in deep learning." , *International conference on machine learning*. 2013.
14. Park, Jieun, Dokkyun Yi, and Sangmin Ji. "A novel learning rate schedule in optimization for neural networks and its convergence." *Symmetry* 12.4 2020: 660.
15. Ruder, Sebastian. "An overview of gradient descent optimization algorithms.", *arXiv preprint arXiv:*1609.04747 2016.
16. Chin, Wei-Sheng, et al. "A learning-rate schedule for stochastic gradient methods to matrix factorization." *Pacific-Asia Conference on Knowledge Discovery and Data Mining. Springer, Cham*, 2015.
17. Zulkifli, Hafidz. "Understanding learning rates and how it improves performance in deep learning." *Software testing fundalmentals* 2018.
18. Dauphin, Yann N., et al. "Identifying and attacking the saddle point problem in high-dimensional non-convex optimization." , *Advances in neural information processing systems*. 2014.
19. Zhang, Sixin, Anna E. Choromanska, and Yann LeCun. "Deep learning with elastic averaging SGD." , *Advances in neural information processing systems.* 2015.
20. Dozat, Timothy. "Incorporating nesterov momentum into Adam." , 2016.
21. Brownlee, J. "Using learning rate schedules for deep learning models in python with keras." (2016).

AUTHORS PROFILE

Mahmoud Smaida, PhD student in National University of Water and Environmental Engineering, Revine, Ukraine. I participated in many conferences and published in journals as follows:

- Industry 4.0 International conference held on 14 - 16 October 2019 at SEGi University, Kota Damansara, Kuala Lumpur, Malaysia. Title of the paper: Development of neural network algorithms for automation of early diagnostics of eye.
- International scientific and technical conference, held on November 27-29, 2019 at Ivano-Frankivsk university, Ukraine. Title of the paper: Features of the use of neural network to the classification of eye diseases.
- Published in International Journal of Innovative Science and Research Technology, Volume 4, Issue 12, December-2019. Title of the paper: Comparative Study of Image Classification Algorithms for Eyes Diseases Diagnostic.
- International Conference on COMPUTATIONAL LINGUISTICS AND INTELLIGENT SYSTEMS, Lviv, Ukraine. Held on April, 2020, 23-24. Title of the paper: Bagging of Convolutional Neural Networks for Diagnostic of Eye Diseases. (Scopus).
- Published in International Journal of Scientific and Research Publications. Title of the paper: Using Ensemble Learning for Diagnostics of Eye Diseases. published for October 2020, Volume 10, Issue 10 publication under ISSN 2250-3153.

Serhii Yaroshchak Machine Learning Engineer – PRO IT and PhD of Philosophy in Mathematical Modeling and Numerical Methods. He received his PhD degree from Lviv Polytechnic National University, Lviv, Ukraine in 2012. Dr, Yaroshchak is currently One of the faculty members of the National University of Water and Environmental Engineering, Revine, Ukraine. His research interests include Machine learning, Deep learning and mathematical Sciences. Dr Yaroshchak has published numerous papers in conferences and jurnal papers (https://scholar.google.com/citations?user=wlpfL_8AAAAJ&hl=en_US).
He is Supervised many research papers, among them the research papers of the student Mahmoud Smaida, mentioned above.

Ahmed Y. Ben Sasi, Professor, was born in Misurata, Libya, in 1969. He received his B.Sc. degree in Computer Engineering from the Engineering Faculty, Tripoli, Libya, in 1992, M.Sc. degree in Advanced Control Engineering from the University of Manchester Institute of Science and Technology (UMIST), U.K, in 1999, and Ph.D. degree in Maintenance and Control Engineering from the University of Manchester, U.K, in 2005. Prof, Ben Sasi is currently the Dean of the College of Industrial Technology (CIT), Misurata, Libya. His research interests include image processing, electro-mechanical condition monitoring, genetic and neuro-fuzzy logic algorithms, and biomedical control engineering applications. Prof, Ben Sasi is a reviewer for several conference and journal papers.