International Journal of Innovative Technology and Exploring Engineering (IJITEE)
ISSN: 2278-3075 (Online), Volume-9 Issue-1, November 2019

Learning Rate Scheduling Policies

A. Agnes Lydia, F. Sagayaraj Francis

Abstract*: With the availability of high processing capability hardwares at less expensive prices, it is possible to successfully train multi-layered neural networks. Since then, several training algorithms have been developed, from algorithms which are statically initialized to algorithms which adaptively change. It is observed that to improve the training process of neural networks, the hyper-parameters are to be fine tuned. Learning Rate, Decay rate, number of epochs, number of hidden layers and number of neurons in the network are some of the hyper-parameters in concern. Of these, the Learning rate plays a crucial role in enhancing the learning capability of the network. Learning rate is the value by which the weights are adjusted in a neural network with respect to the gradient descending towards the expected optimum value. This paper discusses four types of learning rate scheduling which helps to find the best learning rates in less number of epochs. Following these scheduling methods, facilitates to find better initial learning rate value and step-wise updation during the later phase of the training process. In addition the discussed learning rate schedules are demonstrated using COIL-100, Caltech-101 and CIFAR-10 datasets trained on ResNet. The performance is evaluated using the metrics, Precision, Recall and F1-Score. The results analysis show that, depending on the nature of the dataset, the performance of the Learning Rate Scheduling policy varies. Hence the choice of the scheduling policy to train a neural network is made, based on the data.

***Keywords*: Decay Rate, Hyper-parameter tuning, Learning Rates, Neural Network, Scheduling Policy.**

I. INTRODUCTION

Neural Networks are models with successive layers of neurons that have been in existence for decades. These Neural Networks can be trained in both Supervised and Unsupervised [1] manner. To train the network in Supervised manner, an algorithm was developed in late 1970s called *Backpropagation* [2]. Backpropagation uses gradient descent method to compute the increase or decrease in the learning pattern of the network. A typical neural network updates the primary parameters, such as *Weights* and *Biases* using the Gradient Descent method as,

$$\theta^t = \theta^{t-1} - \lambda \frac{\partial L}{\partial \theta} \quad (1)$$

where, θ is the value of the gradient at time t, λ is the learning rate, and $\frac{\partial L}{\partial \theta}$ is the rate of change in loss function with respect to the gradient. The goal of the training phase is to converge towards an optimum point, at which the predicted value and the expected value are almost same and with minimum loss.

Hyper-parameters are components that can impact the learning behavior of the neural network. To enhance the performance of the neural network, a few of these hyper-parameters can be modified. Some of the tunable hyper-parameters of a neural network are, *Learning Rate*, *Decay Rate*, *Epochs*, *Hidden Layers* and number of *Neurons* in each layer [3]. Of these, Learning Rate is the predominantly used hyper-parameter that can be altered in several ways. It is observed that networks with smaller learning rates learn slower and neural networks with larger learning rates learn faster and might skip the optimum value. To overcome this, several learning rate scheduling policies were developed.

II. LEARNING-RATE SCHEDULING POLICIES

Training a large neural network is a difficult optimization task. The *Stochastic Gradient Descent* (SGD) algorithm[4] has established to achieve increased performance and faster learning, using learning rates that varies during the training phase. This process is referred to as *Learning Rate Annealing* or *Adaptive Learning Rates* or *Learning Rate Scheduling* [5]. Learning Rate Scheduling enables to find the best suitable weights for the neurons in the early stages with larger learning rates and, later finds the optimum weights by using smaller learning rates. This reduces the number of epochs to train the network to find the optimum weights, and learning the good weights earlier followed by fine tuning them later. Most of the networks perform better with initial learning rates, $\lambda = \{1e^{-1}, 1e^{-2}, 1e^{-3}\}$. Some of the scheduling policies are discussed further in detail.

A. Time-Based Learning Rate Schedule

While training a neural network using Stochastic Gradient Descent algorithm, it requires several arguments to be passed manually, one of which is the *Decay rate*. At the end of every epoch, the learning rate changes as,

$$\lambda_t = \lambda_0 * \frac{1.0}{1.0 + d + i_t} \quad (2)$$

Revised Manuscript Received on November 30, 2019.
* Correspondence Author
A. Agnes Lydia*, Ph.D., in Computer Science and Engineering at Pondicherry Engineering College, Pondicherry, India.
F. Sagayaraj Francis, Professor in Department of Computer Science and Engineering at Pondicherry Engineering College, Pondicherry, India.

where, λ_t is the current learning rate, λ_0 is the initial learning rate, d is the decay rate, and i_t denotes the epochs. By default the decay arguments is assigned the value 0 which makes in inactive. When the decay rate is specified explicitly, it decreases the present decay rate from the previous epoch by the specified value.

B. Drop-based Learning Rate Schedule

This scheduling policy systematically drops the learning rate during training the network. The learning rate is reduced to half of its initial value for every batch of epochs. The learning rate is updated as,

$$\lambda_t = \lambda_0 * D^{(\frac{i_t}{i_D})} \quad (3)$$

where, λ_t is the current learning rate, λ_0 is the initial learning rate, D is the Drop rate, i_t epochs and i_D epochs batch size. The experiments conducted in this paper, has a learning rate drop by a factor of 0.25 every 10 epochs.

Epochs	Learning Rate (λ)
0	0.01000
1	0.00836
2	0.07190
...	...
39	0.00119
40	0.00116

Table – I: Drop in Learning Rate

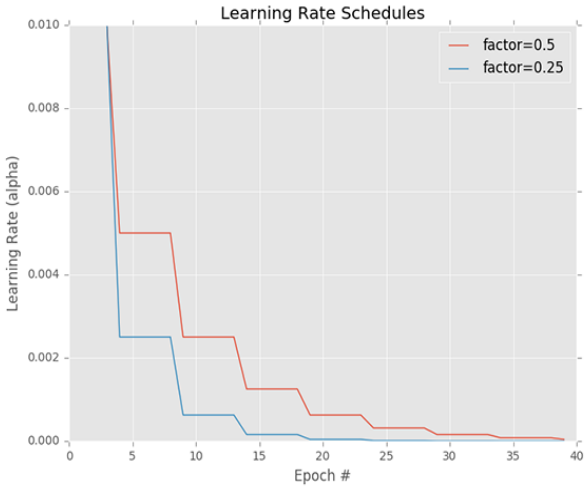

Fig.1: Drop-based Learning Rate Schedule

C. Linear Learning Rate Schedule

In Linear Learning Rate Schedule the learning rate is decayed to zero over a fixed number of epochs. The pace at which the learning rate decays depends on the exponent value assigned to the argument *decay*. In Linear Learning Rate Scheduling, the exponent is assigned the value 1.

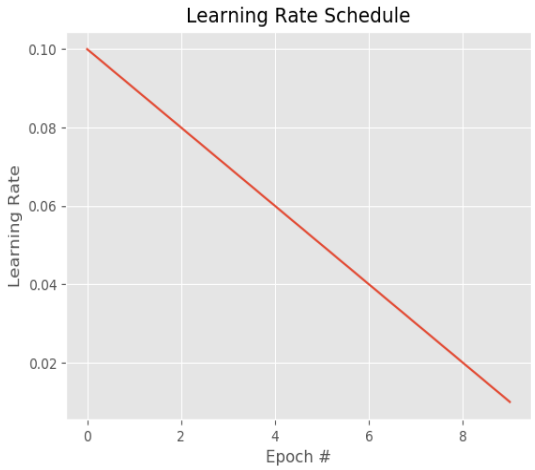

Fig.2: Linear Learning Rate Schedule

D. Polynomial Learning Rate Schedule

The Polynomial Learning Rate Schedule is similar to Linear Learning Rate Scheduling, except that the argument is assigned values greater than 1.

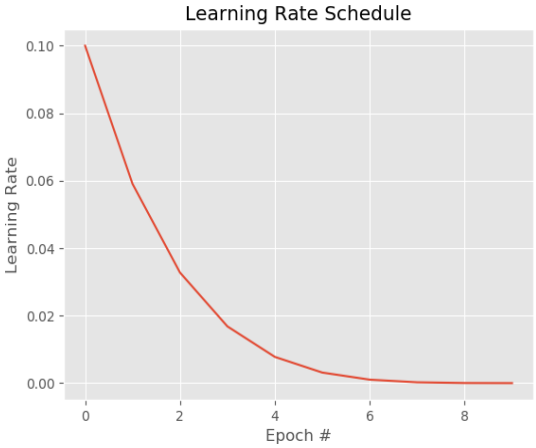

Fig.3: Polynomial Learning Rate Schedule

III. MODEL ARCHITECTURE

A. ResNet

Universal Approximation Theorem [6] states that, "*Given enough capacity, a feed-forward neural network with a single hidden layer is sufficient to represent any function*". But this leads to the layer handling massive amount of data and is susceptible to over-fitting of data. This gave way to the development of deeper neural networks with many hidden layers. Some of the state-of-art convolutional neural networks like AlexNet [7], VGGNet [8] and GoogleNet [9], exhibit the fact that, increasing the hidden layers does not improve the performance of the network. Deeper networks suffer from *Vanishing Gradient Descent problem* [10], i.e., as the gradient is backpropagated to preceding layers, repeated multiplication might make the gradient value to become infinitively small. This makes the network performance to degrade rapidly.

Several techniques were proposed to handle to Vanishing Gradient Descent problem, like the normalized initialization of weights [11] and normalization of intermediate layers of the network [12]. This enables the network to converge faster using Stochastic Gradient Descent trained with Backpropagation. This eventually gave rise to the problem of *Network degradation*, i.e., as the network grows deeper the accuracy of the network gets saturated. To overcome this scenario, *Kaiming He et al.*, proposed a network with the ideology of introducing *identity shortcut connections* in deep neural networks that skips one or more intermediate layers. This network is termed as the Residual Network (ResNet)[13].

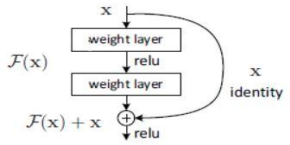

Fig.4: A Single Residual Block

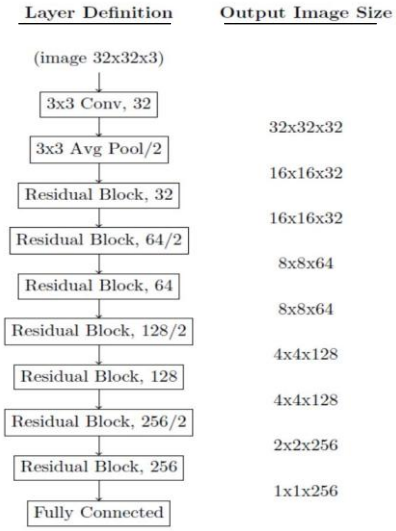

Fig.5: ResNet Architecture

Instead of passing through every stacked layer that is fit with a desired underlying mapping, ResNet explicitly fits a residual mapping. Assuming the desired underlying mapping as $H(x)$, we explicitly let the stacked layers to fit another mapping $F(x) = H(x) - x$. The original mapping is recast into $F(x) + x$, which represents the feed-forward neural network with identity shortcut connections. These shortcut connections perform identity mapping and their outputs are added to the output of the stacked layers. This modified architecture does not increase the trainable parameters or increase the computational complexity.

IV. EXPERIMENTAL RESULTS

A. Analysis on COIL-100

Columbia Object Image Library (COIL) is a dataset collected by the Centre for Research on Intelligent Systems at the Department of Computer Science, Columbia University[14]. It contains images of 100 different objects. The images are placed in a black background, and 72 instances of each object are captured through 360°, with each instance pose varying at 5°. These objects have a variety of geometric complexities and reflectance characteristics.

Table – I: Results of Scheduling Policies (COIL-100)

Schedule	Precision	Recall	F1-Score
Time-Based	0.85	0.85	0.85
Drop-Based	0.84	0.84	0.84
Linear	0.85	0.84	0.84
Polynomial	0.84	0.85	0.85

On training the ResNet using COIL-100 dataset on the discussed scheduling policies, the results show that the performance is better on using Time-Based Learning Rate Scheduling policy.

B. Analysis on Caltech-101

Caltech-101 is a dataset compiled by Fei-Fei Li, et al. at the California Institute of Technology [15]. It consists of image samples of 101 different categories with different backgrounds. Unlike COIL-100, Caltech-101 has imbalanced number of instances under each category. Each image is about 300x200 pixels.

Table – II: Results of Scheduling Policies (Caltech-101)

Schedule	Precision	Recall	F1-Score
Time-Based	0.63	0.62	0.62
Drop-Based	0.67	0.67	0.67
Linear	0.72	0.72	0.72
Polynomial	0.71	0.71	0.71

On training the ResNet using Caltech-101 dataset on the different scheduling policies, the results show that the performance is better on using Linear Learning Rate Scheduling policy.

C. Analysis on CIFAR-10

Canadian Institute for Advanced Research (CIFAR) is a collection of coloured images belonging to 10 different classes [16]. It contains 60,000 images, each of size 32x32. This is one of the commonly used benchmarking dataset, because of its low resolution images it facilitates faster training of neural networks.

Table –VI: Results of Scheduling Policies (CIFAR-10)

Schedule	Precision	Recall	F1-Score
Time-Based	0.70	0.70	0.70
Drop-Based	0.80	0.77	0.77
Linear	0.83	0.83	0.83
Polynomial	0.79	0.79	0.79

On training the ResNet using CIFAR-10 dataset on the different scheduling policies, the results show that the performance is better on using Linear Learning Rate Scheduling policy.

V. CONCLUSION

This article concentrates on fine-tuning only a single hyper-parameter i.e., learning rate, to improve the training of a neural network along with several scheduling policies, by which these learning rates can be initialized and upgraded during the training process. From the experiments conducted, it is analysed that, using these scheduling policies will reduce the number of epochs to attain the optimum value.

Retrieval Number: A4648119119/2019©BEIESP
DOI: 10.35940/ijitee.A4648.119119
Journal Website: www.ijitee.org

Published By:
Blue Eyes Intelligence Engineering
& Sciences Publication

And training on a ResNet architecture, converges even faster and overcomes the problem of Network degradation. With this architecture, the network was trained on COIL-100, Caltech-101, and CIFAR-10. The performance of the scheduling policies varies on different datasets, showing that the choice of the learning rate scheduling policy depends on the nature of the data to be trained on.

REFERENCES

1. Schmidhuber, Jürgen. "Deep learning in neural networks: An overview." *Neural networks* 61 (2015): 85-117.
2. Leung, Henry, and Simon Haykin. "The complex backpropagation algorithm." *IEEE Transactions on signal processing* 39.9 (1991): 2101-2104.
3. Domhan, Tobias, Jost Tobias Springenberg, and Frank Hutter. "Speeding up automatic hyperparameter optimization of deep neural networks by extrapolation of learning curves." *Twenty-Fourth International Joint Conference on Artificial Intelligence*. 2015.
4. Ruder, Sebastian. "An overview of gradient descent optimization algorithms." *arXiv preprint arXiv:1609.04747* (2016).
5. Darken, Christian, Joseph Chang, and John Moody. "Learning rate schedules for faster stochastic gradient search." *Neural Networks for Signal Processing II Proceedings of the 1992 IEEE Workshop*. IEEE, 1992.
6. Huang, Guang-Bin, Lei Chen, and Chee Kheong Siew. "Universal approximation using incremental constructive feedforward networks with random hidden nodes." *IEEE Trans. Neural Networks* 17.4 (2006): 879-892.
7. Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E. Hinton. "Imagenet classification with deep convolutional neural networks." *Advances in neural information processing systems*. 2012.
8. Simonyan, Karen, and Andrew Zisserman. "Very deep convolutional networks for large-scale image recognition." *arXiv preprint arXiv:1409.1556* (2014).
9. Szegedy, Christian, et al. "Going deeper with convolutions." *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015.
10. Hochreiter, Sepp, et al. "Gradient flow in recurrent nets: the difficulty of learning long-term dependencies." (2001).
11. Glorot, Xavier, and Yoshua Bengio. "Understanding the difficulty of training deep feedforward neural networks." *Proceedings of the thirteenth international conference on artificial intelligence and statistics*. 2010.
12. Bjorck, Nils, et al. "Understanding batch normalization." *Advances in Neural Information Processing Systems*. 2018.
13. He, Kaiming, et al. "Deep residual learning for image recognition." *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016.
14. Khalid, El Asnaoui, et al. "An efficient Descriptor for Image Retrieval: Application to COIL-100 Database."
15. Fei-Fei, Li, Rob Fergus, and Pietro Perona. "One-shot learning of object categories." *IEEE transactions on pattern analysis and machine intelligence* 28.4 (2006): 594-611.
16. Krizhevsky, Alex, and Geoff Hinton. "Convolutional deep belief networks on cifar-10." *Unpublished manuscript* 40.7 (2010): 1-9.

AUTHORS PROFILE

A. Agnes Lydia, received her Master Degree in Computer Science and Applications from Pondicherry University, Pondicherry, India. She is currently pursuing Ph.D., in Computer Science and Engineering at Pondicherry Engineering College, Pondicherry, India. Her research interests include, Image Recognition, Feature Extraction and Deep Neural Networks. She has published in two International Journals and one International Conference.

F. Sagayaraj Francis, received his Ph.D., in Computer Science and Engineering from Pondicherry University, Pondicherry, India. He is currently working as Professor in Department of Computer Science and Engineering at Pondicherry Engineering College, Pondicherry, India. His research interests include Database Management Systems, Data Analysis, Geographical Information Systems, Big Data, and Information Systems. He has published in 30 journals and more than 20 International Conferences.