*i*Journals: International Journal of Software & Hardware Research in Engineering (IJSHRE)
ISSN-2347-4890
Volume 10 Issue 7 July 2022

Comparative Study on Image Classification Using Different Optimizer

Arshiya Mobeen M[1]
[2]Assistant Professor
Master of Computer Applications
MEASI Institute of Information Technology
arshiyamobeen@gmail.com

Saistha N[2]
[2]Assistant Professor
Master of Computer Applications
MEASI Institute of Information Technology
saisthashaj@gmail.com

Abstract

The deep learning method is a promising computational technique, especially for image classification problems. One of them is the Convolutional Neural Network (CNN), which is the most popular neural network model used. Although CNN is highly accurate, overfitting is a problem that frequently occurs. The process of training a neural network to perform a task involves repeatedly testing it with example data and using the results to modify the parameters or weights within the model in such a way as to minimize errors occurring due to overfitting. The component that makes these changes to reduce overfitting is called the optimizer. Various optimizers such as Adagrad, RMSProp, AdaDelta and Adam can be used for model molding into its appropriate form by futzing with the weights. Aiming to overcome the problem in getting the optimized result, the research used various algorithms of weight optimization. This paper elaborates on convolutional network concept as well as the idea behind the use of optimizers. The goal of this paper is to conduct a performance evaluation of different optimizers and quantifying the speed and accuracy with which they perform an image classification task.

Keywords: AdaDelta, AdaGrad, Adam, CNN, CIFAR- 10, MNIST, RMSProp

1. INTRODUCTION

The rise of big data and the rapid popularization of highperformance computing devices in recent years have contributed to the unprecedented development of machine learning. In particular, the breakthroughs of deep learning in the computer vision industry have made people discussing about artificial intelligence again. Regarding image identification, this study intends to artificially extract features, and convert the features of an original image into useful features characterized by lower dimension and less noise.

Deep learning [1] inspired from brain simulators, the most acknowledging, promising and advance machine learning methodology that brings the breakthrough achievement for interpreting and classifying the image data. As compared with diverse state-of-art models of machine learning, deep learning is achieving uppermost and leading success in heterogeneous fields. Deep learning consists of well-built and strong models claiming outstanding achievement in resolving issues related to pattern recognition.

The successful extraction as well as selection of a feature is the specialty of this network that makes deep learning more powerful. In order to learn accurately, the different architectures available in the literature of deep learning model consist of numerous processing layers. Among other architectures, one powerful and successful one is the convolutional neural network [2].

2. CONVOLUTIONAL NEURAL NETWORK (CNNs)

The convolutional neural network has mainly been used as a classifier for processing images for the last decade. CNNs are a class of neural networks that allow greater extraction of features from

captured images [2]. Unlike classical models, CNNs take image data, train the model, and then classify the features automatically for healthier classification.

CNN is an algorithm with excellent performance in image processing. It belongs to deep learning category.CNN has multiple layers. CNN architecture consists of not only input-processing-output layers but also has convolution-pooling layers. Every layer is having neurons which are connected with other layer neurons. The responsibility of convolutional layer is to convolve the input and passing the generated output to other upcoming layers. The operations of convolution include the function of rectified Relu, slide, and shift.

High-level features extraction is the objective lies with this layer from image input. Pooling layer in this architecture is responsible for dimension reduction and this layer outcome goes to the next layer called fully connected layer. The combination of convolution-pooling-fully connected layers can be repeated 'n' number of times, where 'n' is any positive integer. Because of all the layers that CNN has, it becomes the powerful one which can resolve hard problems. This complete working of all the layers makes the convolutional network more efficient and powerful [3].

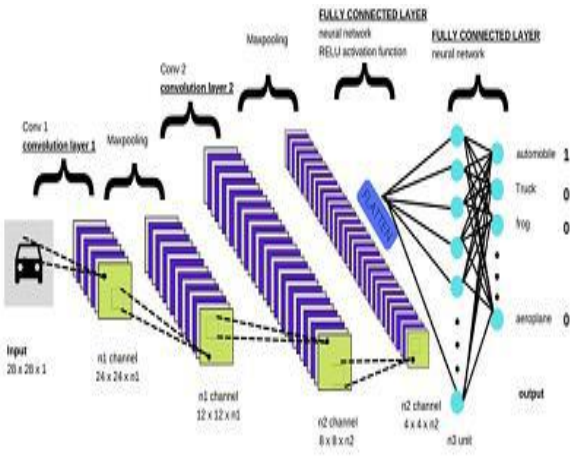

Fig 1. Architecture Diagram

To optimize the result, the adjustment of weights iteratively is performed during back propagation process [4]. There are many optimization algorithms present that adjusts the weights efficiently [5]. Because of these optimization algorithms, the different neural network models are able to perform well with good accuracy [5].

3. DATASET

CNN model is implemented using two datasets named MNIST, and CIFAR-10. Handwritten dataset having digits from 0 to 9 is MNIST [12]. This dataset has total 70 thousand images, in which 60,000 are present in training part and rest 10,000 present in testing part. In this, digits are represented in various ways. Diversity in digits is present due to diversity in handwriting style. The datasets – CIFAR-10 can be downloaded from the official website [11].

Understanding different formats of digits is a typical task for a machine. Convolution model is used with this dataset in this article. This document also uses CIFAR 10 dataset. This dataset has ten varieties of classes where each class is having 1000 images. All these 10 classes are completely mutually exclusive. In this, 50,000 images belong to the training part and 10,000 images belong to the testing class. Image recognition is the toughest job for a machine. Table 1 shows the brief description of used dataset with Convolutional Network

Table 1. Datasets used with CNN Network

Dataset Used	Epochs	Dataset Size
CIFAR 10	1,5	60,000

4. OPTIMIZATION ALGORITHMS

The way to optimize the neural network model's result is termed as optimization. To accomplish this task various optimizers or optimization algorithms are available. The methods whose purpose is to get more accurate result by adjusting again and again neural network parameters like learning rate, weights, etc. Aim of optimization is to achieve better result. This research is using CNN model with different optimizers to get better result in terms of accuracy[5].

The well known four optimizers called SGD, Adam, RMSProp, and Adadelta are judged and properly investigated using the unstructured as well as structured datasets. In case of structured datasets, Adam optimizer provides comparatively better quality of adversarial examples. Along with it, Adadelta optimizer with unstructured dataset provides comparatively better quality of adversarial examples. Additionally, adversarial examples transferability is not affected by optimizer choice. Jiandong et. al. [6] proposed the GRU model and algorithm for GPS map-matching that meet the simultaneously required efficiency and accuracy. With the purpose of weight optimization, the paper used diverse optimization algorithm such like SGD, Adam, Adadelta, and RMSprop with the GRU model.

Adagrad stands for Adaptive Gradient Algorithm. It is adaptive because each weight has a different learning rate, which is updated separately from the others. The rate of the updates are based on how often that parameter is used. This causes uncommon parameters to have a greater update rate than common parameters, which is beneficial when the dataset is sparse. However, because the calculation of learning rates is based on the historical squared gradients sum, the rates are continually decaying. This causes the algorithm to stop updating weights by any meaningful amount as training progresses.

RMSprop was developed to address Adagrad's weaknesses. It attempts to fix the continual decay problem by calculating weight updates based on the running average of the squared gradients, not the sum of all squared gradients. This keeps the weight update responsive to the more recent past of the model.

Adadelta is another optimizer that is very similar to Adagrad. Its goal is also to address the problem of continual decay. Adadelta only considers gradients that happened within some fixed window size. This allows it to keep learning even when many updates have been performed on the weights within the neural network. [6].

Adam, the most recently developed of the optimizers considered, stands for Adaptive Moment Estimation. Adam, like Adagrad, creates an individual learning rate for each parameter. It uses both the decaying average of past gradients and the decaying average of past squared gradients to update parameters [7, 8].

While training the deep learning model, we need to modify each epoch's weights and minimize the loss function.

An optimizer is a function or an algorithm that modifies the attributes of the neural network, such as weights and learning rate. Thus, it helps in reducing the overall loss and improves the accuracy. The problem of choosing the right weights for the model is a daunting task, as a deep learning model generally consists of millions of parameters. It raises the need to choose a suitable optimization algorithm for your application.

You can use different optimizers to make changes in your weights and learning rate. However, choosing the best optimizer depends upon the application. As a beginner, one evil thought that comes to mind is that we try all the possibilities and choose the one that shows the best results. This might not be a problem initially, but when dealing with hundreds of gigabytes of data, even a single epoch can take a considerable amount of time. In this paper, we describe and compare the accuracy and loss of each algorithm.

The terms that are used in comparison are

Epoch – The number of times the algorithm runs on the whole training dataset.
Sample – A single row of a dataset.

Batch – It denotes the number of samples to be taken to for updating the model parameters.

Learning rate – It is a parameter that provides the model a scale of how much model weights should be updated.

Cost Function/Loss Function – A cost function is used to calculate the cost that is the difference between the predicted value and the actual value.

Weights/ Bias – The learnable parameters in a model that controls the signal between two neurons.

5. COMPARATIVE STUDY

5.1 Adagrad (Adaptive Gradient

Descent) Deep Learning Optimizer

The adaptive gradient descent algorithm is slightly different from other gradient descent algorithms. This is because it uses different learning rates for each iteration. The change in learning rate depends upon the difference in the parameters during training. The more the parameters get changed, the more minor the learning rate changes. This modification is highly beneficial because real-world datasets contain sparse as well as dense features. So it is unfair to have the same value of learning rate for all the features[13]. The Adagrad algorithm uses the below formula to update the weights. Here the alpha(t) denotes the different learning rates at each iteration, n is a constant, and E is a small positive to avoid division by 0.

$$W_t = W_t - 1 - \eta'_t \frac{\partial L}{\partial w(t-1)}$$

$$\eta'_t = \frac{\eta}{sqrt(\alpha_t + \epsilon)}$$

The benefit of using Adagrad is that it abolishes the need to modify the learning rate manually. It is more reliable than gradient descent algorithms and their variants, and it reaches convergence at a higher speed[6].

One downside of AdaGrad optimizer is that it decreases the learning rate aggressively and monotonically. There might be a point when the learning rate becomes extremely small. This is because the squared gradients in the denominator keep accumulating, and thus the denominator part keeps on increasing[14]. Due to small learning rates, the model eventually becomes unable to acquire more knowledge, and hence the accuracy of the model is compromised.

5.2 RMSProp(Root Mean Square) Propagation Deep Learning Optimizer

RMSProp is one of the popular optimizers among deep learning enthusiasts. RMSProp is ideally an extension of the work Resilient back propagation (RPPROP). RPPROP resolves the problem of varying gradients. The problem with the gradients is that some of them were small while others may be huge. So, defining a single learning rate might not be the best idea. RPPROP uses the sign of the gradient adapting the step size individually for each weight. In this algorithm, the two gradients are first compared for signs. If they have the same sign, we're going in the right direction and hence increase the step size by a small fraction. Whereas, if they have opposite signs, we have to decrease the step size. Then we limit the step size, and now we can go for the weight update.

The problem with RPPROP is that it doesn't work well with large datasets and when we want to perform mini-batch updates. So, achieving the robustness of RPPROP and efficiency of mini-batches at the same time was the main motivation behind the rise of RMSProp[8]. RMSProp can also be considered an advancement in AdaGrad optimizer as it reduces the monotonically decreasing learning rate.

The algorithm mainly focuses on accelerating the optimization process by decreasing the number of function evaluations to reach the local minima. The algorithm keeps the moving average of squared gradients for every weight and divides the gradient by the square root of the mean square.

$$\nu(w,t) := \gamma\nu(w,t-1) + (1-\gamma)(\nabla Q_{i(w)})^2$$

where gamma is the forgetting factor. Weights are updated by the below formula

$$w := w - \frac{\eta}{\sqrt{\nu(w,t)}} \nabla Q_i(w)$$

In simpler terms, if there exists a parameter due to which the cost function oscillates a lot, we want to penalize the update of this parameter. Suppose you built a model to classify a variety of fishes. The model relies on the factor 'color' mainly to differentiate between the fishes. Due to which it makes a lot of errors. What RMSProp does is, penalize the parameter 'color' so that it can rely on other features too. This prevents the algorithm from adapting too quickly to changes in the parameter 'color' compared to other parameters. This algorithm has several benefits as compared to earlier versions of gradient descent algorithms. The algorithm converges quickly and requires lesser tuning than gradient descent algorithms and their variants.

The problem with RMSProp is that the learning rate has to be defined manually and the suggested value doesn't work for every application.

5.3 AdaDelta Deep Learning Optimizer

AdaDelta can be seen as a more robust version of AdaGrad optimizer. It is based upon adaptive learning and is designed to deal with significant drawbacks of AdaGrad and RMS prop optimizer. The main problem with the above two optimizers is that the initial learning rate must be defined manually. One other problem is the decaying learning rate which becomes infinitesimally small at some point[15]. Due to which a certain number of iterations later, the model can no longer learn new knowledge.

To deal with these problems, AdaDelta uses two state variables to store the leaky average of the second moment gradient and a leaky average of the second moment of change of parameters in the model.

$$S_t = \rho S_{t-1} + (1-\rho) g_t^2$$

$$X_t = X_t - 1 - g'_t$$

$$g'_t = \frac{a\sqrt{\Delta x_{t-1} + \epsilon}}{\sqrt{s_t + \epsilon}} \odot g_t$$

$$\Delta X_t = \rho \Delta X_{t-1} + (1-\rho) g'^2_t$$

Here S_t and delta X_t denotes the state variables, g'_t denotes rescaled gradient, delta X_{t-1} denotes squares rescaled gradients, and epsilon represents a small positive integer to handle division by 0.

5.4 Adam Deep Learning Optimizer

The name adam is derived from adaptive moment estimation. This optimization algorithm is a further extension of stochastic gradient descent to update network weights during training. Unlike maintaining a single learning rate through training in SGD, Adam optimizer updates the learning rate for each network weight individually. The creators of the Adam optimization algorithm know the benefits of AdaGrad and RMSProp algorithms, which are also extensions of the stochastic gradient descent algorithms[10]. Hence the Adam optimizers inherit the features of both Adagrad and RMS prop algorithms. In adam, instead of adapting learning rates based upon the first moment mean) as in RMS Prop, it also uses the second moment of the gradients. We mean the uncentred variance by the second moment of the gradients.

The adam optimizer has several benefits, due to which it is used widely. It is adapted as a benchmark for deep learning papers and recommended as a default optimization algorithm[9]. Moreover, the algorithm is straightforward to implement, has faster running time, low memory requirements, and requires less tuning than any other optimization algorithm.

$$m_t = \beta_1 m_t - 1 + (1-\beta_1)\left[\frac{\delta L}{\delta w_t}\right] v_t = \beta_2 v_{t-1} + (1-\beta_2)$$

$$\left[\frac{\delta L}{\delta w_t}\right]^2$$

The above formula represents the working of adam optimizer. Here B1 and B2 represent the decay rate of the average of the gradients.

6. RESULT COMPARISON

The table 2 shows the comparison of different optimizing algorithm based on accuracy and loss with Epoch 1 and Epoch 5

Table 2: Comparison of accuracy and loss with Epoch 1 & 5

Optimizer	Epoch 1		Epoch 5		Time
	Accuracy	Loss	Accuracy	Loss	
Adadelta	0.4512	2.3464	0.7676	1.7243	8.59
Adagrad	0.8211	0.7974	0.9121	0.3897	8.01
RMSprop	0.9574	0.0721	0.9765	0.0562	10.45
Adam	0.9662	0.0701	0.9799	0.0401	7.34

Based on the comparison, we analyze that the Adam optimizer shows the best accuracy and computation time when compared with other optimizing algorithm. RMSprop shows similar accuracy to that of Adam but with comparatively much larger computation time. Adagrad shows

moderate accuracy and computation time. Adadelta shows poor results with accuracy and time.

The Graph shows the accuracy of different optimizers with different Epoch.

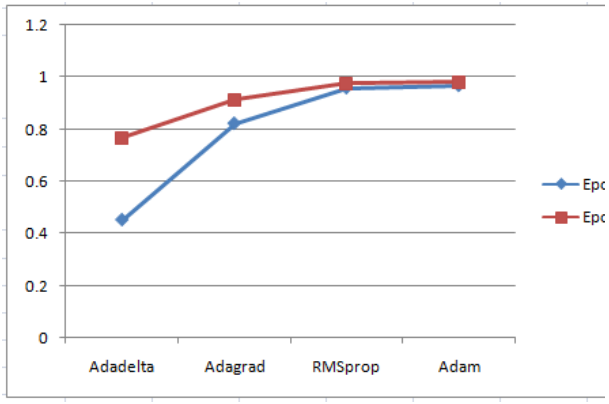

Fig 2. Accuracy of different optimizers

7. CONCLUSION

Adagrad works better than stochastic gradient descent generally due to frequent updates in the learning rate. It is best when used for dealing with sparse data. RMSProp shows similar results to that of gradient descent algorithm with momentum, just differs the way by which the gradients are calculated. Lastly comes the Adam optimizer that inherits the good features of RMSProp and other algorithms. The results of the Adam optimizer are generally better than every other optimization algorithms, have faster computation time, and require fewer parameters for tuning. Because of all that, Adam is recommended as the default optimizer for most of the applications. Choosing the Adam optimizer for your application might give you the best probability of getting the best results.

7. REFERENCES

[1]. Rathore H, Al-Ali AK, Mohamed A, Du X, Guizani M. A novel deep learning strategy for classifying different attack patterns for deep brain implants. IEEE Access. 2019 Feb 20;7:24154-64.

[2]. Xin R, Zhang J, Shao Y. Complex network classification with convolutional neural network. Tsinghua Science and technology. 2020 Jan 13;25(4):447-57.

[3].Weng Y, Zhou T, Liu L, Xia C. Automatic convolutional neural architecture search for image classification under different scenes. IEEE Access. 2019 Mar 28;7:38495-506.

[4]. Liang F, Shen C, Wu F. An iterative BP-CNN architecture for channel decoding. IEEE Journal of Selected Topics in Signal Processing. 2018 Jan 15;12(1):144-59.

[5]. Wang Y, Liu J, Mišić J, Mišić VB, Lv S, Chang X. Assessing optimizer impact on dnn model sensitivity to adversarial examples. IEEE Access. 2019 Oct 21;7:152766-76.

[6]. Zhao J, Gao Y, Yang Z, Li J, Feng Y, Qin Z, Bai Z. Truck traffic speed prediction under non-recurrent congestion: Based on optimized deep learning algorithms and GPS data. IEEE Access. 2019 Jan 1;7:9116-27.

[7]. Mainprice J, Hayne R, Berenson D. Goal set inverse optimal control and iterative replanning for predicting human reaching motions in shared workspaces. IEEE Transactions on Robotics. 2016 Jul 27;32(4):897-908.

[8]. Park SH, Bae YB, Fidan B, Ahn HS. Distance-based Mobile Node Localization of Fixed Beacons Using RMS Prop. In2019 58th Annual Conference of the Society of Instrument and Control Engineers of Japan (SICE) 2019 Sep 10 (pp. 376-381). IEEE.

[9]. D. P. Kingma and J. Ba, "Adam: A method for stochastics optimization," arXiv preprint arXiv:1706.08500, 2017.

[10] S. J. Reddi, S. Kale, and S. Kumar. 2018. "On the convergence of Adam and beyond". Proceedings of the International Conference on Learning Representations

[11].The apache software foundation, what is apachemahout. 2014.CIFAR Dataset

[12]. Mohapatra RK, Majhi B, Jena SK. Classification performance analysis of mnist dataset utilizing a multi-resolution technique. In2015 International Conference on Computing, Communication and and Security (ICCCS) 2015 Dec 4 (pp. 1-5). IEEE

[13]Hadgu AT, Nigam A, Diaz-Aviles E. Large-scale learning with AdaGrad on Spark. In2015 IEEE International Conference on Big Data (Big Data) 2015 Oct 1 (pp. 2828-2830). IEEE.

[14] Liu Y, Huangfu W, Zhang H, Long K. An efficient stochastic gradient descent algorithm to maximize the coverage of cellular networks. IEEE Transactions on Wireless Communications. 2019 May7;18(7):3424-36.

[15] Fleishman, G. M., & Thompson, P. M. (2017, April). Adaptive gradient descent optimization of

initial momenta for geodesic shooting in diffeomorphisms. In 2017 IEEE 14th International Symposium on Biomedical Imaging (ISBI 2017) (pp. 868-872). IEEE.