Optimization of Deep Learning Model Using an Improved Three-term Conjugate Gradient Algorithm

Ibidapo Dada (dman4computer@gmail.com)
Federal University of Agriculture Abeokuta https://orcid.org/0000-0003-3336-5332
Adio Akinwale
Federal University of Agriculture Abeokuta
Idowu Osinuga
Federal University of Agriculture Abeokuta
Moses Olubiyi
Federal University of Agriculture Abeokuta

Research Article

Keywords: deep learning, optimization, conjugate gradient method

Posted Date: March 1st, 2022

DOI: https://doi.org/10.21203/rs.3.rs-1308110/v1

OPTIMIZATION OF DEEP LEARNING MODEL USING AN IMPROVED THREE-TERM CONJUGATE GRADIENT ALGORITHM

Dada Ibidapo D.[1], Akinwale Adio T.[1], Osinuga Idowu A.[2], Olubiyi Moses O[2].

[1]Department of Computer Science, Federal University of Agriculture, Abeokuta, Nigeria

[2]Department of Mathematics, Federal University of Agriculture, Abeokuta, Nigeria

Corresponding email: dman4computer@gmail.com

Abstract- The aim of this study is to present a new type of optimization algorithm to train deep learning model. In other to achieve this, we used a convex combination method to combine the coefficients of Fletcher-Reeves (FR) and Polak-Ribiere-Polyak (PRP) on a three-term conjugate gradient method. This algorithm called Three-term PRP-FR Algorithm was implemented from scratch using python programming language alongside some existing optimizers. These optimizers were evaluated and compared based on the convergence of these optimizers using the accuracy results on the popularly known digit recognition dataset (MNIST dataset).

Keywords- deep learning, optimization, conjugate gradient method.

1.0 INTRODUCTION

Deep learning with neural networks has been the dominant methodology of training new machine learning models for the past decade (Gaurav 2021). The field of deep learning is concerned with choosing the right methods for training deep neural networks (San *et al.* 2014; Xu *et al.* 2015; Lou *et al.* 2021). One of the major aspects of deep learning is to formulate useful optimization algorithms for finding model parameter values that reduce the cost function (Bottou *et al.* 2018). Research in the field of optimization has a long history and it's evolved constantly. Applying a good optimization method can reduce the number of iterations and also can obtain better characteristics when compared to each other.

Ruder (2017) Presents the gradient descent optimization algorithm in survey research, has the earliest and commonly used first-order optimization method. The gradient descent method works by updating variables iteratively in the (opposite) direction of the objective function's gradients. The update is carried out for the objective function to eventually converge to its optimum value. The learning rate influences the number of iterations required to achieve the optimal value by determining the step size in each iteration (Ruder 2017). It is used to minimize the cost function of an objective function.

The gradient descent algorithm in general form:

$$\text{Let } x \coloneqq x_0 \in R^n \; be \; an \; inital \; point$$
$$\text{While } \|\nabla f(x)\|_2 > \varepsilon$$
$$x \coloneqq x - \propto \nabla f(x)$$

Gradient descent algorithm take time to converge to global optima based on the value of α (learning rate) chosen also the learning rate does not update automatically.

According to a survey research conducted by Shiliang *et al.* (2019), presents the stochastic gradient method in which was proposed by Robbins and Monro in 1951, because of the high computational complexity of the batch gradient descent on each iteration especially for large-scale data. The idea behind stochastic gradient descent (SGD) is to use one random sample to update the gradient per iteration. In large numbers of samples, SGD is independent and can achieve convergence speed (Roux *et al.* 2012; Johnson *et al.* 2013; Agarwal *et al.* 2009). The algorithm repeatedly updates each parameter β_j using a learning rate α.

$$Repeat\{$$
$$\beta_j \coloneqq \beta_j - \alpha \frac{\partial}{\partial \beta_j} J(\beta) \; \}$$

The limitations of Stochastic Gradient Descent are appropriate learning rate is difficult to choose and the solution to some problems can be trapped at a saddle point. To solve the problem of trapping at a saddle point, a method was proposed by Robbin and Monro (1951) called the mini-batch gradient descent (MSGD). Ruder (2017) Presented that MSGD uses 50 to 256 independent identically samples to update parameters in each iteration. This method helps to reduce the variance of the gradients and also makes convergence more stable. With these characteristics, MSGD has a great chance of reaching the global optimal solution for complex problems. Drawback for MSGD is that choosing a too small learning rate might lead to very slow

convergence and a too large learning rate make it difficult to convergence. (Robbins *et al.* 1951; Darken *et al.* 1992) proposed a way to solve this problem which is setting a predefined list of learning rates or giving a condition to adjust the learning rate during the learning process.
Many researchers have worked to improve SGD method. An example was the idea of momentum which was proposed by Polyak (1964). The method is to be applied in SGD.
The algorithm repeatedly updates each parameter β_j.

$$Repeat\{$$
$$\beta_j \coloneqq \beta_j - \alpha \frac{\partial}{\partial \beta_j} J(\beta) + V^{old}.\ mtm$$
$$\}$$

where V^{old} was introduced to speed up the convergence rate and it represents the direction and the rate at which the parameters moves in their own space. $V^{old}$is known as the previous update which is now multiplied by a momentum factor in the range of [0,1] written as mtm. Choosing proper momentum plays a great role from escaping from the local minimum in training process and can also help the searching process to reach convergence more quickly. Research from Ruder (2017) shows through many experiments that the most appropriate setting for the momentum factor is 0.9.
(Nesterov 1983; Sutskever *et al.* 1983) made improvement on the traditional momentum. β was added to $V^{old}.\ mtm$ denoted as $\tilde{\beta}$ which is used for updating the parameters. Parameters update formulae are given as follows:

$$\begin{cases} \tilde{\beta} = \beta + V^{old}.\ mtm, \\ V = \alpha \frac{\partial}{\partial \beta_j} J(\beta) + V^{old}.\ mtm \\ \beta' = \beta + V \end{cases}$$

Updating the future position of the gradient instead of the current is a great improvement over the traditional momentum which improves the convergence from O(1/k) to O($1/k^2$), when the batch optimization is used (Nesterov 1983). However, the issue of how to determine the learning rate is still a big challenge.
In the work of Baird *et al.* (1999), learning rate decay factor d is used in SGD's momentum method, which decreases the learning rate during the iteration period. The learning rate decay formula is given as:

$$\alpha_t = \frac{\alpha_0}{1 + d.t} \tag{1}$$

where $\alpha_t$is the learning rate at t^{th} iteration, $\alpha_0$is the original rate, d is an element in [0, 1]. d determines how fast the learning rate decay.
(Duchi *et al.* 2011) proposed a direct improved algorithm to SGD called AdaGrad algorithm. The AdaGrad algorithm worked on adjusting the learning rate based on the gradient of previous iterations. The update formulae for the SGD is given as:

$$\beta_j \coloneqq \beta_j - \alpha \frac{\partial}{\partial \beta_j} J(\beta) \tag{2}$$

while in the AdaGrad modifies the learning rate α at point j for every parameter β_j based on the gradient of all previous iterations.

$$\beta_j \coloneqq \beta_j - \frac{\alpha}{\sqrt{\sum_{i=1}^{j} \left(\frac{\partial}{\partial \beta_j} J(\beta) \right)^2 + \epsilon}} \frac{\partial}{\partial \beta_j} J(\beta) \tag{3}$$

One of Adagrad's main benefits is that it eliminates the need to manually tune the learning rate. Most implementations use a default value of 0.01 and leave it at that. Drawback of Adagrad are the algorithm still needs to set the global learning rate manually and as the training time increases, the accumulated gradient will become larger and larger, making the learning rate tend to zero, resulting in ineffective parameter update. (Matthew 2012) made an improvement on AdaGrad algorithm to AdaDelta algorithm which provide solution to the problem that makes the learning rate tend to zero. Instead of summing all past gradients, Adadelta method accumulates past gradients in a window over a fixed period. The drawback of AdaDelta update process may not converge.

(Kingma *et al.* 2014) proposed another improved SGD method called the Adaptive moment estimation (Adam), which is a combination of adaptive methods and the momentum methods. It uses the first-order moment estimation and the second-order moment estimation of the gradient to dynamically adjust the learning rate of each parameter. The method is best used for non-convex optimization problems with large data and for higher dimensional space but it may not converge in some cases. Adam performs very well in practice when compared to other adaptive methods.

2.0 OVERVIEW OF RELATED METHODS

The following unconstrained optimization problem can be solved by modelling the nonlinear conjugate gradient method

$$min\{f(x)|x \in R^n\} \quad (4)$$

where $f: R^n \to R$ is continuously differentiable, $f(x)$ is an objective function and $x \in R^n$ is a vector with independent variables. The CGM is commonly solved by iterative method which is defined as follows:

$$x_{k+1} = x_k + \propto_k d_k, \quad k = 0,1,2, \ldots \quad (5)$$

where x_k is the current iterative point, $\propto_k$ is the learning rate (also known as step size) and d_k is the search direction of conjugate gradient method. The search direction of conjugate gradient method d_k can be defined as follows:

$$d_k = \begin{cases} -g_k & k=0 \\ -g_k + \beta_k d_{k-1} & k>1 \end{cases} \quad (6)$$

where g_k is the gradient at point x_k. β_k is a parameter known as conjugate gradient (CG) coefficient. Different change in the parameter β_k correspond conjugate gradient methods. Some common conjugate gradient method are Fletcher-Reeves (FR) (Fletcher *et al.* 1964), Polak-Ribiere-Polyak (PRP) (Polyak 1964), Hestenes-Stiefel (HS) (Hestenes *et al.* 1952), Liu-Storey (LS) (Liu 1991), Dai-Yuan (DY) (Dai 2001) and conjugate descent (CD) (Fletcher *et al.* 1964) and their parameters β_k are defined respectively as:

$$\beta_k^{FR} = \frac{g_k^T g_k}{\|g_{k-1}\|^2} \quad (7)$$

$$\beta_k^{PRP} = \frac{g_k^T(g_k - g_{k-1})}{\|g_{k-1}\|^2} \tag{8}$$

$$\beta_k^{HS} = \frac{g_k^T(g_k - g_{k-1})}{d_{k-1}^T(g_k - g_{k-1})} \tag{9}$$

$$\beta_k^{LS} = \frac{-g_k^T(g_k - g_{k-1})}{d_{k-1}^T g_{k-1}} \tag{10}$$

$$\beta_k^{DY} = \frac{g_k^T g_k}{d_{k-1}^T(g_k - g_{k-1})} \tag{11}$$

$$\beta_k^{CD} = \frac{-g_k^T g_k}{d_{k-1}^T g_{k-1}} \tag{12}$$

3.0 PROPOSED THREE-TERM CONJUGATE GRADIENT METHOD

We develop a modified three-term conjugate gradient method guided by the work of Jiankun *et al.*. (2019), to optimize a deep learning model. Jiankun *et al.*. *(*2019), proposed a three-term conjugate gradient method, where the search direction is given as:

$$d_k = \begin{cases} -g_k & k=0 \\ -g_k + \beta_k d_{k-1} + \theta_k y_{k-1} & k>1 \end{cases} \tag{13}$$

Where $g_k = \nabla f(x)$ is the gradient of f at $x, y_{k-1} = g_k - g_{k-1}$ and

$$\beta_k = \frac{g_k^T y_{k-1}}{\|g_{k-1}\|^2} \tag{14}$$

$$\theta_k = \frac{g_k^T d_{k-1}}{\|d_{k-1}\|^2} \tag{15}$$

Motivated by the above methods, we propose a hybridized method for β_k as a convex combination of Polak-Ribiere-Polyak (PRP) and Fletcher-Reeves (FR) methods named as β_k^{CHM} given as:

$$\beta_k^{CHM} = (1 - \emptyset)\beta_k^{PRP} + \emptyset\beta_k^{FR} \tag{16}$$

Where $0 < \emptyset < 1$,

$$\beta_k^{PRP} = \frac{g_k^T(g_k - g_{k-1})}{\|g_{k-1}\|^2} \tag{17}$$

$$\beta_k^{FR} = \frac{g_k^T g_k}{\|g_{k-1}\|^2} \tag{18}$$

Therefore, the new search direction is given as:

$$d_k = \begin{cases} -g_k & k=0 \\ -g_k + {\beta_k}^{CHM} d_{k-1} + \theta_k y_{k-1} & k>1 \end{cases} \quad (19)$$

The develop algorithm is describe below.

Three-term PRP-FR Algorithm: A convex combination of PRP and FR on Three-term Conjugate Gradient method

---

1: Choose $0 < \emptyset < 1$. Set initial point $x_0 \in \mathbb{R}^n$, $\in > 0$ and $k = 0$.
2: Set $g_0 = \nabla f(x_0)$, if $\|g_{k-1}\| \leq \in$, then stop.
3: **For** $k = 0,1,$ ... **do**
4: Compute d_k as in (19).
5: Compute step size $\propto_k$.
6: Update a new point x_{k+1} using (5).
7: Set $k = k + 1$.
8: If $\|g_{k-1}\| \leq \in$, then stop.
9: Compute ${\beta_k}^{PRP}$, ${\beta_k}^{FR}$ as in (17) and (18) respectively.
10: Compute ${\beta_k}^{CHM}$ as in (16)
11: **end for**

4.0 Convergence analysis

This section provides theoretical proofs for the three-term PRP-FR algorithm. The sufficient descent condition for the method is provided here and its global convergence is established. Below, we show the descent property of the proposed CGM-Algorithm denoted by

The sufficient descent condition

$$d_k = \begin{cases} -g_k & k=0 \\ -g_k + {\beta_k}^{CHM} d_{k-1} + \theta_k y_{k-1} & k>1 \end{cases} \quad (20)$$

where

$${\beta_k}^{CHM} = (1 - \emptyset){\beta_k}^{PRP} + \emptyset{\beta_k}^{FR} \quad (21)$$

$${\beta_k}^{PRP} = \frac{{g_k}^T (g_k - g_{k-1})}{\|g_{k-1}\|^2} \quad (22)$$

$${\beta_k}^{FR} = \frac{{g_k}^T g_k}{\|g_{k-1}\|^2} \quad (23)$$

$$\theta_k = -\frac{{g_k}^T d_{k-1}}{\|d_{k-1}\|^2} \quad (24)$$

Lemma 1

The search direction d_k in equation (20) satisfies the descent condition

$$g_k^T d_k \leq 0 \tag{25}$$

Proof:
From (20)

$$d_k = -g_k + (1-\emptyset)\beta_k{}^{PRP} d_{k-1} + \emptyset\beta_k{}^{FR} + \theta_k y_{k-1}$$

$$= -g_k + \beta_k{}^{PRP} d_{k-1} - \emptyset\beta_k{}^{PRP} d_{k-1} + \emptyset\beta_k{}^{FR} + \theta_k y_{k-1}$$

$$= -g_k + \frac{g_k{}^T(g_k - g_{k-1})}{\|g_{k-1}\|^2} d_{k-1} - \emptyset \frac{g_k{}^T(g_k - g_{k-1})}{\|g_{k-1}\|^2} d_{k-1} + \emptyset \frac{g_k{}^T g_k}{\|g_{k-1}\|^2} - \frac{g_k{}^T d_{k-1}}{\|d_{k-1}\|^2} y_{k-1}$$

$$\leq \frac{g_k{}^T(g_k - g_{k-1})}{\|g_{k-1}\|^2} d_{k-1} + \emptyset \frac{g_k{}^T g_k}{\|g_{k-1}\|^2}$$

$$= \frac{g_k{}^T((g_k - g_{k-1})d_{k-1} + \emptyset(g_k - g_{k-1})}{\|g_{k-1}\|^2}$$

Multiplying through by g_k

$$g_k^T d_k \leq -\|g_{k-1}\|^2 \left(\frac{((g_k - g_{k-1})d_{k-1} + \emptyset(g_k - g_{k-1})}{\|g_{k-1}\|^2} \right)$$

putting

$$c = \left(\frac{((g_k - g_{k-1})d_{k-1} + \emptyset(g_k - g_{k-1})}{\|g_{k-1}\|^2} \right)$$

then

$$g_k^T d_k \leq -c\|g_{k-1}\|^2 < 0 \tag{26}$$

Global Convergence Analysis

Next, we show that the proposed CGM-Algorithm with $\beta_k{}^{CHM}$ converges globally.

Assumption 1

1a. If f is bounded in the level set $Z = \{x \epsilon R^n : f(x) \leq f(x_0)\}$ in some initial point.

b. If f is continuously differentiable and its gradient is Lipschtiz continuous, there exist $P > 0$ such that:

$$\|g(x) - g(y)\| \leq P\|x - y\| \forall\, x, y\ \in N$$

c. If f is uniformly convex function, then there exist a constant $\mu > 0$ such that:

$$(\nabla f(x) - \nabla f(y))^T (x - y) > \mu\|x - y\|^2, for\ any\ x, y \in S$$

From assumption 1a, there exist a constant Q such that:

$$\|x\| \leq Q\ \forall\ x \in Z\ and\ \|\nabla f(x)\| \leq D, \forall\ x \in S \tag{27}$$

Theorem

Suppose that assumption (1), equation (26) and equation (27) holds, then the conjugate gradient method of the form:

$$d_k = -g_k + {\beta_k}^{CHM} d_{k-1} + \theta_k y_{k-1}$$

Where $\propto_k$ is fixed at 0.1, then of the cost function is uniformly convex, set Z, then

$$\lim_{k \to \infty} \|g_k\| = 0$$

Proof

$$d_k = -g_k + {\beta_k}^{CHM} d_{k-1} + \theta_k y_{k-1}$$

$$d_k = \left\| -g_k + {\beta_k}^{CHM} d_{k-1} + \theta_k y_{k-1} \right\|^2$$

$$\leq \|g_k\|^2 + {\beta_k}^{CHM} \|d_{k-1}\|^2 + \theta_k \| y_{k-1}\|^2$$

putting $\theta_k = A$ and ${\beta_k}^{CHM} = B$

$$\|d_k\|^2 \leq \|g_k\|^2 + B\|d_{k-1}\|^2 + \mathrm{A}\| y_{k-1}\|^2$$

$$\|d_k\|^2 \leq \frac{1}{D^{-2}} (\|g_k\|^2 D^{-2} + BD^{-2}\|d_{k-1}\|^2 + \mathrm{A}D^{-2}\| y_{k-1}\|^2)$$

putting E = $\|g_k\|^2 D^{-2} + BD^{-2}\|d_{k-1}\|^2 + \mathrm{A}D^{-2}\| y_{k-1}\|^2$

then

$$\|d_k\|^2 \leq E\left(\frac{1}{D^{-2}}\right)$$

$$\sum_{k=1}^{\infty} \frac{1}{\|d_k\|^2} \leq \frac{1}{E} D^{-2} \sum_{k \geq 1} 1 = \infty$$

$$\Rightarrow \lim_{k\to\infty} \|g_k\| = 0 \; as \; required.$$

5.0 Numerical Experiments

All experiments were implemented on a PC with the Hardware configuration of PC workstation with Intel® Core™ i5-5020U CPU @ 2.20GHz, 8GB of RAM; and Python is the main programming language that is used to implement this study. This is due to the availability of vast amount of open source python-based libraries and packages.

5.1 Dataset

We train and evaluate the proposed optimizer and compare the result with some existing optimizer such as FR-CGM, PRP-CGM, HS-CGM, stochastic gradient descent (SGD), momentum stochastic gradient descent (MSGD). The Experiments was based on digit recognition using the popular MNIST dataset for image classification. The dataset comprises 42000 dataset. The MNIST dataset was downloaded manually from https://www.kaggle.com.

5.2 Parameters setting

The stop criteria used in the experiment for all the optimization algorithms convergence is set to $\|g_k\| \leq 10^{-4}$. Fixed learning rate α is set to 0.1. For $\emptyset \; in \; {\beta_k}^{CHM}$ is set to 0.7. Decay factor is set to 0.5 for momentum stochastic gradient descent (MSGD).

5.3 Discussion of Results

For each optimization algorithms used in these experiments, the program were executed ten times since the initial weight and biases were based on randomization number. Therefore, the best results were depicted in Table 1.

Table 1: Accuracy metric Scores based on Number of iterations for different optimizers on MNIST dataset

Number of iterations / optimization algorithms	50	100	150	200	300	350	500
Three-term PRP-FR	0.67%	0.82%	0.86%	0.90%	-	-	-
PRP	0.53	0.68%	0.74%	-	-	-	-
FR	-	-	-	-	-	-	-
HS	0.68%	0.77%	0.81%	0.83%	0.86%	-	-
SGD	0.43%	0.64%	0.73%	0.78%	0.82%	0.83%	0.85%
MSGD	0.65%	0.75%	0.79%	0.80%	0.84%	0.85%	

The numerical results of the proposed algorithm and some other algorithms were shown in table 1. The numerical results indicated that the proposed algorithm (Three-term PRP-FR) have made significant performance among all algorithms. The proposed algorithm (Three-term PRP-FR)

reach optimal value after 200 iterations with 0.9% accuracy which outperformed other algorithms with Three-term PRP gives 0.74% accuracy after 150 iterations, Three-term HS gives 0.86% accuracy after 300 iterations, SGD gives 0.85% accuracy after 500 iterations and MSGD gives 0.85% accuracy after 350 iterations. Three-term FR fails to solve the problem.

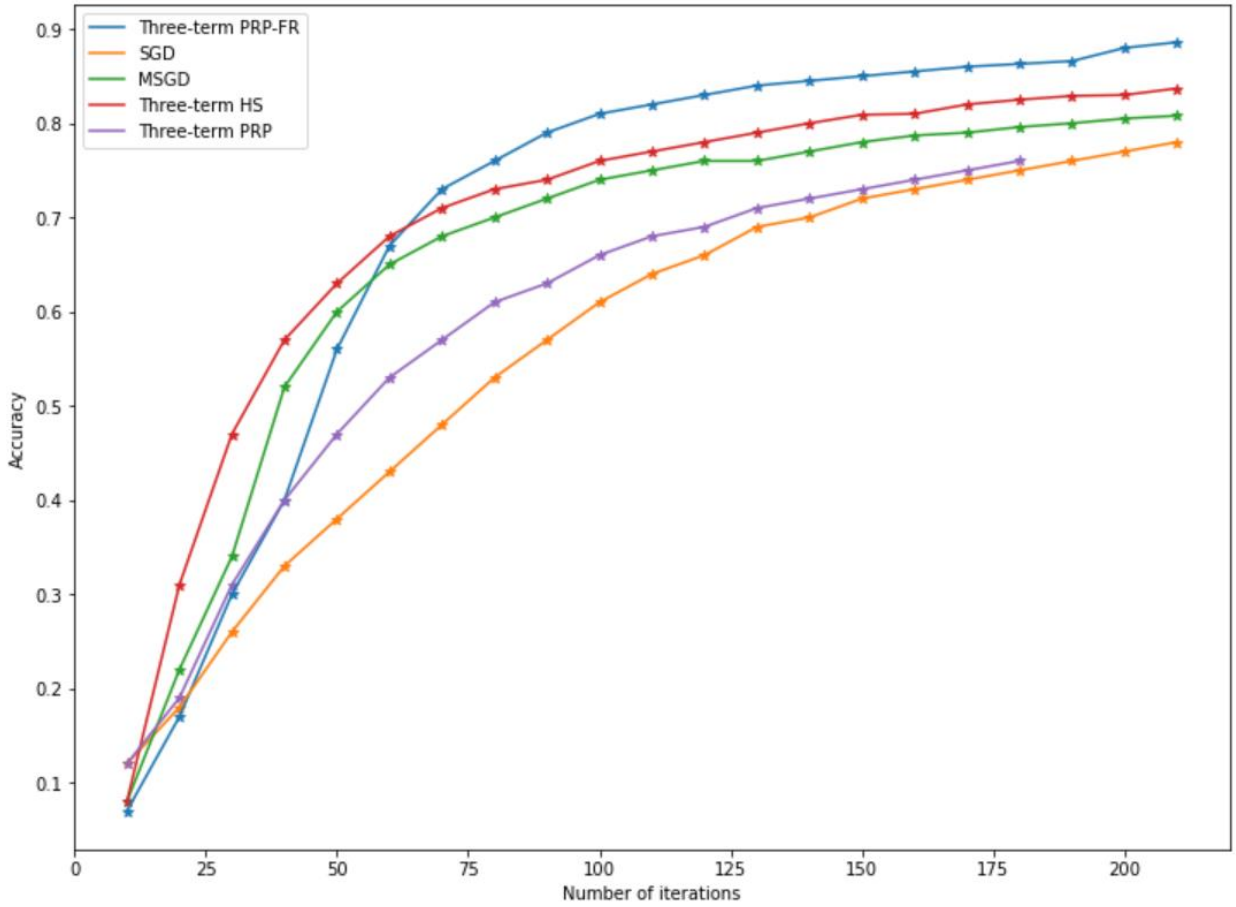


Figure1: Accuracy metric Scores based on Number of iterations for different optimizers on MNIST dataset

Considering figure 1, the comparison of different optimization algorithm on MNIST dataset using 200 iteration. It was observed that the proposed algorithm Three-term PRP-FR converges to local minimum of the loss function but HS, SGD, MSGD needs more number of iterations in order to converges to local minimum of the loss function. PRP converges to local minimum but with the lowest value of accuracy.

Table 2: Predicted value for each optimizer

Predicted output / Input data	**Label**	Three-term PRP-FR	PRP	HS	SGD	MSGD
	1	1	1	1	1	1

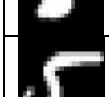
		0	0	1	0	0	0
		4	9	8	8	8	8
		7	9	0	9	9	9
		8	8	8	5	5	5
		6	6	1	6	6	6
		5	5	8	5	5	5
		2	2	2	2	2	2

Table 2 shows some selected original input data from about 20% of the handwriting recognition dataset (MNIST dataset) and the original label for testing. The predicted value for the optimizer were also shown.

6.0 Conclusion

In this study, a new optimization algorithms was proposed to solve deep learning model problem. The algorithm performs was tested using the popularly known handwriting recognition dataset (MNIST dataset) and was compared with some existing optimization algorithm. Therefore, from the experiment on the popularly known handwriting recognition dataset (MNIST dataset). The proposed algorithm Three-term PRP-FR proves to be a good optimization algorithm in terms of number of iterations and accuracy.

Author contributions

All authors contributed to the study.
Methodology: Dada Ibidapo
Numerical Experiments: Dada ibidapo;
Writing: Dada Ibidapo and Olubiyi Moses;
Review and editing: Akinwale Adio and Osinuga Idowu
Supervision: Akinwale Adio and Osinuga Idowu.

References

Agarwal A., M. J. Wainwright, P. L. Bartlett, and P. K. Ravikumar, "Information-theoretic lower bounds on the oracle complexity of convex optimization," in Advances in Neural Information Processing Systems, 2009, pp. 1–9.

Baird III L. C. and A. W. Moore, "Gradient descent for general reinforcement learning," in Advances in Neural Information Processing Systems, 1999, pp. 968–974.

Bottou L, F. E. Curtis, and J. Nocedal, “Optimization methods for large scale machine learning,” SIAM
review, vol. 60, no. 2, pp. 223-311, 2018.

Dai Y. H. and Yuan Y., (2001). “An efficient hybrid conjugate gradient method for unconstrained optimization”. Annals of Operations Research 103, 3347.

Darken C., J. Chang, and J. Moody, “Learning rate schedules for faster stochastic gradient search,” in Neural Networks for Signal Processing, 1992, pp. 3–12.

Duchi J., E. Hazan, and Y. Singer, “Adaptive subgradient methods for online learning and stochastic optimization,” Journal of Machine Learning Research, vol. 12, pp. 2121–2159, 2011.

Fletcher R. and Reeves C. M., (1964). Function minimization by conjugate gradients, Computer Journal 7, 149 - 154.

Gaurav Menghani (2021). “Efficient Deep learning: A Survey on making Deep learning Models Smaller,
Faster, and Better”, Google Research, Mountain view, California, USA, 95054.

Hestenes M. R. and Stiefel E., (1952). Methods of Conjugate Gradients for Solving Linear Systems. Journal of Research of the National Bureau of Standards, 49(6), Research Paper 2379.

Jiankun Liu & Shouqiang Du (2019). Modified Three-Term Conjugate Gradient Method and its applications. Mathematical Problems in Engineering, 2019.

Johnson R. and T. Zhang, “Accelerating stochastic gradient descent using predictive variance reduction,” in Advances in Neural Information Processing Systems, 2013, pp. 315–323.

Kingma, D. and Ba, J. 2014. Adam: A Method for Stochastic Optimization. arXiv preprint arXiv:1412.6980.

Liu, Y., & Storey, C. (1991). Efficient generalized conjugate gradient algorithms, part 1: theory. Journal of Optimization Theory and Applications, 69(1), 129–137.

Lou Y., Y. He, L. Wang and G. chen, “Predicting network controllability robustness: A convolutional
neural network approach,” IEEE Transactions on Cybermetics, 2021.

Matthew D. Zeiler. ADADELTA: An Adaptive Learning Rate Method. arXiv preprint arXiv:1212.5701,
2012.

Nesterov Y., “A method for unconstrained convex minimization problem with the rate of convergence O(1/k2),” Doklady Akademii Nauk SSSR, vol. 269, pp. 543–547, 1983.

Polyak B. T., “Some methods of speeding up the convergence of iteration methods,” USSR Computational Mathematics and Mathematical Physics, vol. 4, pp. 1–17, 1964.

Polyak B. T., “Some methods of speeding up the convergence of iteration methods,” USSR Computational Mathematics and Mathematical Physics, vol. 4, pp. 1–17, 1964.

Robbins H. and S. Monro, “A stochastic approximation method,” The Annals of Mathematical Statistics, pp. 400–407, 1951.

Roux N. L., M. Schmidt, and F. R. Bach, “A stochastic gradient method with an exponential convergence rate for finite training sets,” in Advances in Neural Information Processing Systems, 2012, pp. 2663– 2671.

Ruder S., “An overview of gradient descent optimization algorithms,”arXiv preprint arXiv:1609.04747,

2017
San P.P., S. H. Ling, Nuryani, and H. Nguyan, “Evolvable rough-block-based neural network and its biomedical application to hypoglycemia detection system,” IEEE Transactions on Cybermetics, vol. 44, no 8, pp. 1338-1349, 2014.
Shiliang Sun, Zehui Cao, Han Zhu, and Jing Zhao (2019). “A Survey of Optimization Methods from a Machine Learning Perspective”. arXiv:1906.06821v2 [cs.LG] 23 Oct 2019.
Sutskever I., 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.
Xu W., J. Cao, M. Xiao, D. W. C. Ho and G. Wen, “A new framework for analysis on stability and bifurcation in a class of neural networks with discrete and distributed delays” IEEE Transactions on Cybermetics, vol. 45, no 10, pp. 2224-2236, 2015.

Statements & Declarations

Funding
No funds, grants, or other support were received during the preparation of this manuscript.

Competing Interests
Authors have no relevant financial or non-financial interests to disclose.