

Comparative Analysis on Deep Learning Optimization Techniques

¹P. Alekhya, ²S. Nitish, ³Y. Harshitha

^{1,2,3}Student – 8th Semester Department of Computer Science & Engineering

^{1,2,3}Lendi Institute of Engineering and Technology, Jonnada, Vizianagaram

Abstract: Deep learning & especially Convolutional Neural Networks (CNNs) are taking a different shape in the area of Image Recognition & Classification. Performance of any CNN model depends on various parameters such as size of the dataset, number of classes, weights of the model, hyperparameters and mainly on optimizers. Generally, optimizers are used to optimize the model parameters in any learning algorithm. The purpose of an optimizer is to adjust model weights to maximize a loss function. The loss function is used as a way to measure how well the model is performing. In order to reduce the loss and increase the accuracy, we are using the optimizers. In this project, we are doing a comparative analysis of optimizers like mini-batch GD, momentum GD, RMS prop, Adam, Adagrad and Adadelta on datasets like MNIST, CIFAR10, Kaggle Flowers. The comparison will be made between the loss and accuracy at every epoch.

Key Words: Deep Learning, Optimizers, SGD, Adam, RMS prop, Adagrad, Adadelta

I. INTRODUCTION:

Deep Learning: Deep learning is a subset of machine learning that involves training artificial neural networks to perform complex tasks, such as image and speech recognition, natural language processing, and decision-making. It is based on the idea of creating multiple layers of artificial neurons that can learn to recognize patterns in data and make predictions or decisions based on that information. Deep learning algorithms use large amounts of data to train these neural networks, allowing them to automatically identify and learn from patterns in the data. This makes them particularly useful for tasks such as image and speech recognition, where the input data is complex and difficult to analyze using traditional programming techniques.

Some of the most popular deep learning techniques include convolutional neural networks (CNNs) for image and video analysis, recurrent neural networks (RNNs) for natural language processing and speech recognition, and deep reinforcement learning for decision-making tasks. Deep learning has numerous applications in fields such as healthcare, finance, and self-driving cars, and it continues to be an active area of research and development.

Role of algorithms in deep learning: Algorithms play a crucial role in deep learning as they provide the framework for training and optimizing deep neural networks. In deep learning, an algorithm is used to adjust the weights and biases of the network during training so that it can accurately predict the output for a given input. Algorithms used in deep learning include convolutional neural network (CNN) algorithms, recurrent neural network (RNN) algorithms, and generative adversarial network (GAN) algorithms. These algorithms provide specialized techniques for processing different types of data and solving specific types of problems. Algorithms are the backbone of deep learning and provide the means for building complex models that can learn from vast amounts of data and make accurate predictions.

Role of Optimizers in deep learning: Optimizers are a key component of deep learning algorithms and play a critical role in training deep neural networks.

- They are responsible for adjusting the weights and biases of the network during training so that it can accurately predict the output for a given input.
- The primary role of optimizers is to minimize the loss function of the neural network. This is achieved by adjusting the weights and biases of the network based on the gradients of the loss function with respect to these parameters.
- The gradients represent the direction of steepest descent of the loss function, and optimizers adjust the weights and biases in this direction to minimize the loss.
- One of the key challenges in deep learning is to find the optimal set of weights and biases that minimize the loss function. Optimizers play a critical role in this process, allowing the network to adjust its parameters and find the optimal solution.
- They enable the training of deep neural networks that can learn complex representations of the input data and solve a wide range of problems.

The primary goal of this research is to contrast different optimizers with optimization techniques. Here, the loss, MSE and accuracy of optimizers are calculated for each epoch. The optimum optimizer for the dataset is one with lower loss & MSE and higher accuracy.

II. LITERATURE SURVEY:

The present paper[1] , First-Order Optimization Algorithms in Deep Learning provides a comparison of the training strategies for convolutional neural networks, which are employed in image recognition applications. In this project, the Oxford17 category flower dataset to train and assess the CNN model was used. It contains 17 categories of common flowers in the UK with 80 images for each class. At the decision of problems of recognition the Adamax algorithm should be used. At the same time, however, the problem of selecting the optimal values of the algorithms parameters that provide top speed of learning still remains open.

Gradient descent optimisation methods, despite becoming more and more common, are frequently utilised as "black-box" optimizers because it is difficult to find concrete justifications for their advantages and disadvantages. At every state-of-the-art Deep Learning library contains implementations of various algorithms to optimize gradient descent. This article[2] aims to provide the reader with intuitions with regard to the behaviour of different algorithms that will allow her to put them to use. In this article, they first examined the three different types of gradient descent, with minibatch gradient descent being the most widely used. The other ways to enhance SGD, including batch normalisation, early stopping, curriculum learning, and shuffling.

In this study [3], they examined the use of covariance noise in building deep neural network optimisation algorithms that can potentially display perfect learning behaviour. A stochastic optimisation algorithm's batch-size selection has a significant impact on both optimisation and generalisation. In most cases, increasing the batch size employed worsens generalisation while improving optimisation. The proposed method in this paper, studying the gradient variance is not sufficient in deducing any information on the optimization behavior. Rather, it is the structure of the covariance matrix of the noise which plays a critical role. For large-batch training with diagonal Fisher, and found that despite having a high gradient variance, it still attains an ideal optimization performance.

In-depth analysis of cutting-edge deep learning applications for speech and audio processing, natural language processing (NLP), and visual data processing are provided in this paper[4]. The most effective method for managing enormous volumes of data in the age of artificial intelligence is a very challenging problem that is also quite motivating. This research argues that stochastic gradient descent (SGD), among machine learning models, is not only straightforward but also exceedingly efficient. The first type of typical strategy involves using as many training samples as you can while carrying out a minimal amount of calculations on each sample. Finally, this paper began by outlining the various deep learning applications. Gradient descent methods for enhancing SGD were later introduced in a variety of forms.

In this paper[5], for locating the global optimizers of nonconvex optimisation problems, they devised the "AdaVar" algorithm, a new gradient descent method with extra stochastic terms. The adaptive tweaking of randomness based on the value of the objective function is a crucial part of the method. The temperature is state-dependent in the language of simulated annealing. With this, it was able to demonstrate how the algorithm converges globally at an algebraic rate in both parameter space and probability. By adopting a more straightforward control of the noise term, this rate is improved over the classical rate. The discrete configuration of the algorithm itself serves as the foundation for the convergence proof. In order to alter implementation details and develop best practices, final conclusion was that it is critical to use the algorithms in a variety of actual circumstances.

Traditional machine learning and control optimisation techniques mainly rely on first-order update rules[6]. The area of meta-learning is motivated by the fact that choosing the best method and hyperparameters for a given job frequently involves trial and error or practitioner intuition. By developing a meta-learning framework in which the inner loop optimisation step entails solving a differentiable convex optimisation problem, and generalised a large family of existing update rules (DCO). It has been demonstrated that the theoretical attraction of this strategy by demonstrating that, provided the meta-learner has adequate exposure to related tasks, it allows one-step optimisation of a family of linear least squares problems. On a variety of illustrative experimental setups, different instantiations of the DCO update algorithm are compared to conventional optimizers. In the context of meta-learning, this work provided a novel DCO-based approach for optimizer design. The existing first-order update rules' innate convexity is maintained via the DCO meta-learning framework

III. METHODOLOGY:

3.1 Comparative analysis on Optimizers:

In deep learning, optimizers are utilized to change a model's parameters. Optimizers are algorithms or techniques that modify the weights and learning rates of your neural network in order to minimize the risks. We performed a comparative analysis of optimizers for the MNIST dataset as part of this research, which is titled Comparative Analysis on Deep Learning Optimization Methods. Here, we examined a number of optimizers, calculating their accuracy, MSE and loss at each epoch. Every optimizer was taken into account in this project across many epochs. The graphs will be displayed with the results after the loss, MSE and accuracy have been determined. These graphs show the loss, MSE and accuracy from the optimizers we utilised for the analysis. The dataset will be optimised using the algorithms that have the best results.

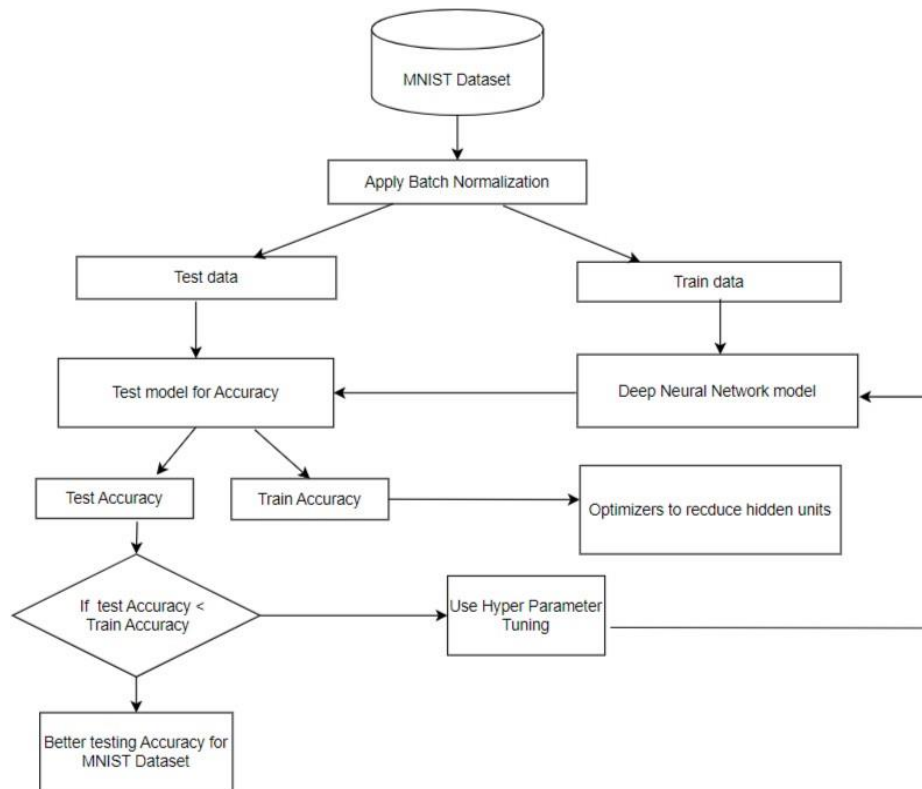


Fig. 3.1: Block diagram for Role of Optimizers in Deep Learning

We discovered that the main role of CNN is played by optimizers based on the current systems. So, the comparison of optimisation strategies is the main focus of our project. Since there are many limitations in the first order optimization algorithms, we have used the adaptive gradient optimization algorithms like Adam, AdaGrad, RMSProp, NAdam and Adamax along with first order optimization algorithms. So that we can reduce the limitations occurred in the existing systems. Since we are using adaptive gradient optimization algorithms, there are some advantages in our project.

Algorithm for comparative analysis of Optimizers:

Step-1: An image from the dataset should be preprocessed.

Step-2: Develop suitable CNN model for the image that is preprocessed.

Step-3: Ensure to optimize the CNN with Optimizers like Adam, NAdam, Adamax, SGD, Adagrad, Adadelata, RMSProp so on.

Step-4: Calculate the metrics such as MSD (Mean squared Error), Loss, Accuracy.

Step-5: Compare the calculated metrics.

Step-6: Ensure that suitable optimizer is selected.

IV. Experimental Results and Analysis:

4.1 Libraries used:

Google colab:

Colaboratory, commonly known as "Colab," is a Google Research product. Colab is particularly well suited to machine learning, data analysis, and education. It enables anyone to create and execute arbitrary Python code through the browser.

TensorFlow:

The TensorFlow platform assists you in putting best practises for data automation, model tracking, performance monitoring, and model retraining into practise. It is an open source machine learning platform that runs from beginning to end. Learn about the adaptable ecosystem of tools, libraries, and community resources available with TensorFlow.

Keras:

Keras is a Python-based deep learning API that runs on top of the TensorFlow machine learning platform. Keras is the TensorFlow platform's high-level API: an approachable, highly productive interface for tackling machine learning issues, with a focus on current deep learning. It provides fundamental abstractions and building blocks for rapidly designing and shipping machine learning applications. Keras enables engineers and researchers to fully exploit the TensorFlow platform's scalability and cross-platform capabilities: you can run Keras on TPU or massive clusters of GPUs, and you can export your Keras model to run in the web or on a mobile device.

Python:

Python is an interpreted, high-level, general-purpose programming language. Created by Guido van Rossum and first released in 1991, Python's design philosophy emphasizes code readability with its notable use of significant whitespace. Its language

constructs and object-oriented approach aim to help programmers write clear, logical code for small and large-scale projects. Python is dynamically typed and garbage-collected. It supports multiple programming paradigms, including procedural, object-oriented, and functional programming.

NumPy:

NumPy is the fundamental package for scientific computing in Python. It is a Python library that provides a multidimensional array object, various derived objects (such as masked arrays and matrices), and an assortment of routines for fast operations on arrays, including mathematical, logical, shape manipulation, sorting, selecting, I/O, discrete Fourier transforms, basic linear algebra, basic statistical operations, random simulation and much more. At the core of the NumPy package, is the ndarray object.

Matplotlib:

Matplotlib is an excellent Python visualisation package for 2D array charts. Matplotlib is a multi-platform data visualisation package based on NumPy arrays and designed to operate with the SciPy stack as a whole. One of the most significant advantages of visualisation is that it provides us with visual access to massive volumes of data in simply digestible representations. Matplotlib has a variety of plots such as line, bar, scatter, histogram, and so on.

Tensorboard:

TensorBoard is a strong open source toolbox for measuring and visualising data within individual models as well as comparing performance across models. There are also some robust debugging capabilities offered to help you visually examine the model. TensorBoard was designed for TensorFlow, however it is now supported by other frameworks such as PyTorch. TensorBoardX is a project that adds support for TensorBoard to additional frameworks like as Chainer, MXnet, and others.

Metrics:

Metrics are used to monitor and measure a model's performance (during training and testing), and they do not have to be differentiable. However, if the performance metric is differentiable for some tasks, it can also be utilised as a loss function (possibly with some regularisations added), such as MSE. Metric functions are similar to loss functions in that the outcomes of metric evaluation are not considered for training the model.

4.2 Comparison of optimizers

Comparison of Optimizers w.r.t loss and epochs:

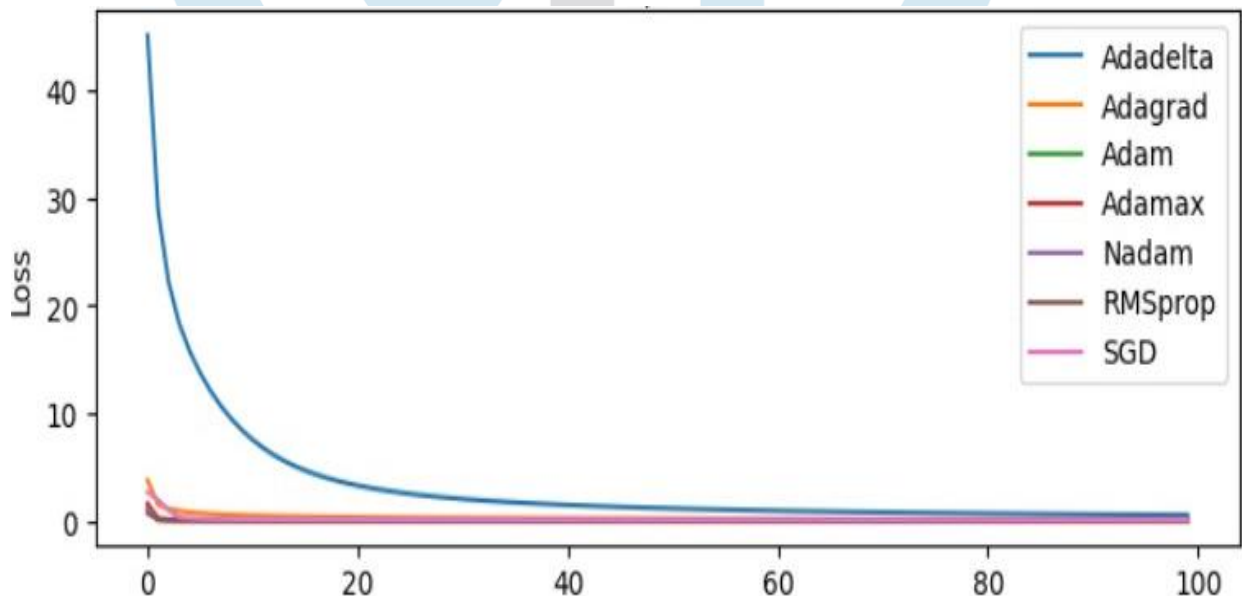


Fig.4.2.1: Comparison between loss and epochs

Comparison of Optimizers w.r.t MSE and epochs:

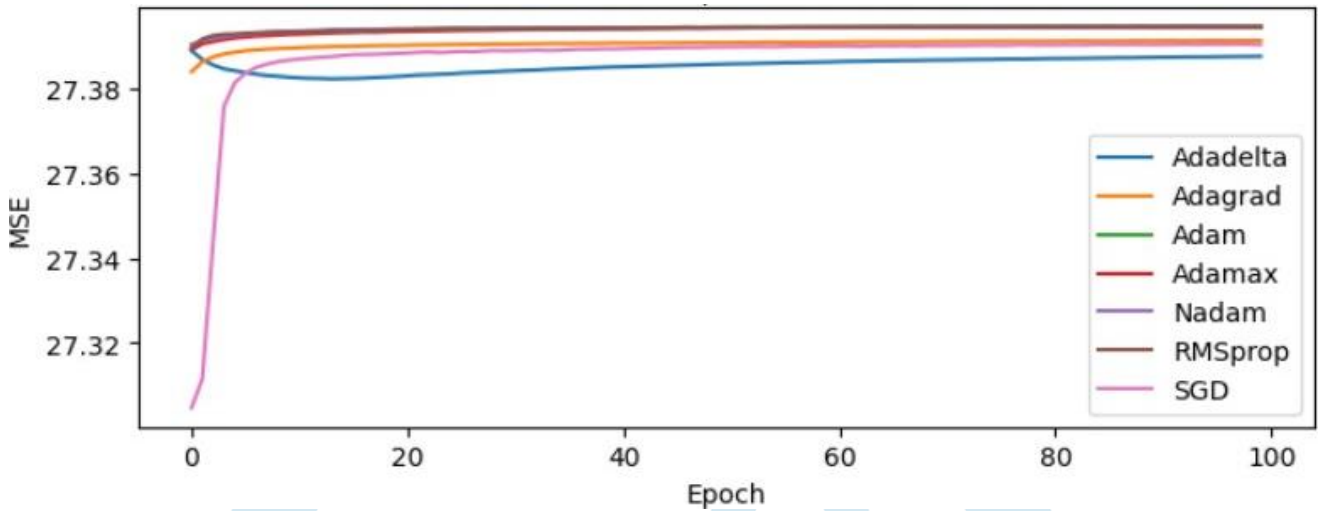


Fig. 4.2.2: Comparison between MSE and epochs

Comparison of Optimizers w.r.t accuracy and epochs:

Accuracy Comparison for Optimizers

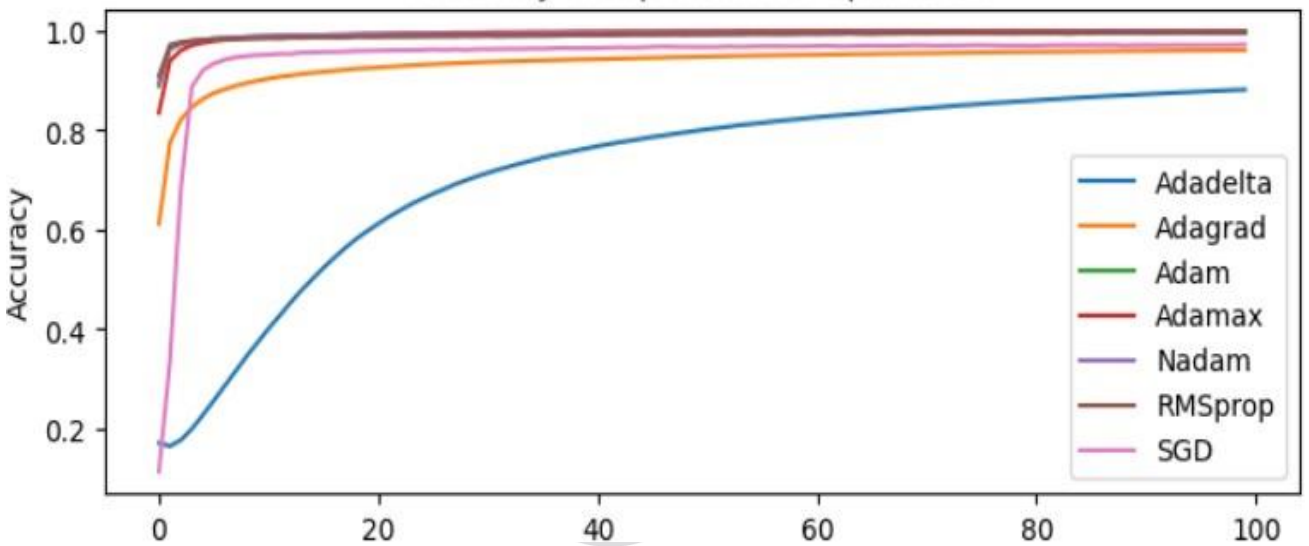


Fig. 4.2.3: Comparison between accuracy and epochs

V. CONCLUSION:

The goal of deep learning is to lower generalisation error. We must be mindful of overfitting and use the optimisation technique to reduce the training error in order to achieve the latter. The goal of optimisation isto increase the model's accuracy while reducing the likelihood of errors or losses resulting from these predictions. The accuracy of predictions and classifications is increased, and error is reduced, through optimisation. In this project, we compared optimizers by analyzing their accuracy, mean squared error, and loss. Then, the graphs with regard to epochs were used to represent the metrics of the optimizers (loss, MSE, and accuracy). The

optimizer with the lowest loss and MSE and the highest accuracy was chosen in the end. We found that RMS prop and Adam were the most appropriate for our project.

REFERENCES:

1. K Oliinyk, 2020, First-Order Optimization (Training) Algorithms in Deep Learning, Kharkiv National University of Radio Electronics
2. S Ruder, 2016, An overview of gradient descent optimization algorithms, Insight Centre for DataAnalytics.
3. Yeming Wen, 2020, An Empirical Study of Large-Batch Stochastic Gradient Descent with Structured Covariance Noise, University of Toronto.
4. Y Tian, Jan 2023, Recent Advances in Stochastic Gradient Descent in Deep Learning, Advances in Network Modeling, Analysis and Optimization.
5. Yunan Yang, Nov 2022, An Algebraically Converging Stochastic Gradient Descent Algorithm for Global Optimization, Optimization and Control, Machine Learning.
6. Tanmay Gautam, March 2023, Meta-Learning Parameterized First-Order Optimizers using Differentiable Convex Optimization, Machine Learning (cs.LG); Artificial Intelligence (cs.AI); Optimization and Control

