Curvature-Adaptive Learning Rate Optimizer: Theoretical Insights and Empirical Evaluation on Neural Network Training

Kehelwala Dewage Gayan Maduranga
Tennessee Technological University
Email: gmaduranga@tntech.edu

Abstract

Optimizing neural networks often encounters challenges such as saddle points, plateaus, and ill-conditioned curvature, limiting the effectiveness of standard optimizers like Adam, Nadam, and RMSProp. To address these limitations, we propose the Curvature-Adaptive Learning Rate (CALR) optimizer, a novel method that leverages local curvature estimates to dynamically adjust learning rates. CALR, along with its variants incorporating gradient clipping and cosine annealing schedules, offers enhanced robustness and faster convergence across diverse optimization tasks. Theoretical analysis confirms CALR's convergence properties, while empirical evaluations on benchmark functions—Rosenbrock, Himmelblau, and Saddle Point—highlight its efficiency in complex optimization landscapes. Furthermore, CALR demonstrates superior performance on neural network training tasks using MNIST and CIFAR-10 datasets, achieving faster convergence, lower loss, and better generalization compared to traditional optimizers. These results establish CALR as a promising optimization strategy for challenging neural network training problems.

Introduction

Neural networks have emerged as the backbone of modern artificial intelligence, driving significant advancements in diverse domains such as computer vision (Krizhevsky, Sutskever, and Hinton 2012), natural language processing (Vaswani et al. 2017), and robotics (Levine et al. 2016). Despite their remarkable success, the optimization landscape of neural networks presents significant challenges, including saddle points, flat regions, and ill-conditioned curvature. These challenges are particularly pronounced in high-dimensional, non-convex optimization problems, where standard optimizers such as Adam (Kingma and Ba 2017), Nadam (Dozat 2016), and RMSProp (Tieleman and Hinton 2012) often exhibit suboptimal performance. While these methods have been pivotal in accelerating training and stabilizing convergence, they can struggle in scenarios requiring efficient navigation of complex loss surfaces.

In this paper, we propose the Curvature-Adaptive Learning Rate (CALR) optimizer, a novel approach that extends the capabilities of traditional adaptive optimizers by leveraging local curvature information. CALR dynamically adjusts learning rates based on curvature approximations, enabling robust and efficient optimization in challenging landscapes. Furthermore, we introduce variants of CALR that incorporate gradient clipping and cosine annealing learning rate schedules, providing additional flexibility and control over optimization dynamics. These enhancements address common issues such as exploding gradients and suboptimal learning rate schedules, further improving CALR's adaptability and scalability.

The primary contributions of this work are as follows:

- We propose CALR, a curvature-aware optimizer, and provide a theoretical analysis of its convergence properties in non-convex optimization settings.
- We evaluate CALR and its variants on synthetic benchmark optimization functions, including Rosenbrock, Himmelblau, and saddle point problems, demonstrating their effectiveness in navigating complex optimization landscapes.
- We validate the practical utility of CALR through extensive experiments on neural network training tasks using MNIST and CIFAR-10 datasets. Our results highlight CALR's superior convergence speed, stability, and generalization compared to Adam, Nadam, and RMSProp.

By integrating curvature-aware adjustments, CALR addresses critical limitations of existing optimizers, offering a promising alternative for a wide range of machine learning applications. This paper underscores the potential of CALR as a robust and efficient optimization strategy, paving the way for its adoption in challenging training scenarios and large-scale machine learning tasks. Throughout, algebraic operations or functions of vectors are entrywise. $||.||$ denotes the 2-norm unless specified otherwise.

Related Work

Gradient-Based and Adaptive Optimization Methods

Gradient-based optimization methods have long been the cornerstone of neural network training. Among these,

Stochastic Gradient Descent (SGD) (Robbins and Monro 1951) is one of the most widely used techniques. SGD updates parameters iteratively by moving in the direction of the negative gradient of the loss function. While it is simple and computationally efficient, its performance is highly sensitive to the choice of learning rate and can be significantly affected by the optimization landscape. Specifically, SGD often struggles with saddle points, plateaus, and highly anisotropic curvature, requiring careful tuning of learning rates or additional mechanisms such as momentum (Polyak 1964) to mitigate these challenges.

To address these limitations, adaptive optimization methods were developed. Approaches such as AdaGrad (Duchi, Hazan, and Singer 2011), Adam (Kingma and Ba 2017), Nadam (Dozat 2016), and RMSProp (Tieleman and Hinton 2012) dynamically adjust learning rates during training by leveraging gradient statistics. AdaGrad adapts the learning rate for each parameter based on the cumulative sum of squared gradients, making it particularly effective for sparse data. Adam and Nadam combine this adaptivity with momentum-based methods, using exponential moving averages of both first- and second-order gradient moments to achieve faster convergence. RMSProp, on the other hand, introduces an exponentially weighted moving average of squared gradients, which helps address the diminishing learning rates observed in AdaGrad.

Despite their widespread use and success, these methods face notable challenges in non-convex optimization landscapes. Saddle points, where gradients are small in magnitude but do not indicate local minima, and ill-conditioned curvature, where gradients vary dramatically across dimensions, can cause these optimizers to stall or converge suboptimally. While Adam and Nadam are designed to handle a wide range of optimization scenarios, they often struggle with overfitting or poor generalization in some cases due to aggressive learning rate adjustments.

These limitations have spurred interest in developing optimization strategies that incorporate curvature information to navigate complex landscapes more effectively. Second-order methods, such as Newton's method and quasi-Newton approaches, explicitly use curvature information from the Hessian matrix to improve convergence. However, their computational and memory requirements make them impractical for large-scale neural networks. This gap has motivated the development of lightweight alternatives, such as the CALR optimizer. CALR explicitly integrates local curvature approximations into the learning rate adjustment process, offering a robust and scalable solution for training neural networks in challenging optimization landscapes.

Theoretical Insights into CALR

Mathematical Framework

Curvature-Aware Learning Rates (CALR) enhance adaptive optimization by integrating curvature approximations into the learning rate formulation. CALR adjusts learning rates dynamically based on gradient statistics and local curvature, improving robustness in challenging optimization landscapes, such as those with saddle points or anisotropic curvature.

Curvature-Aware Update Rule CALR modifies the standard parameter update rule:

$$\theta_{t+1} = \theta_t - \eta_t^{\text{CALR}} \cdot \hat{m}_t,$$

where the curvature-aware learning rate η_t^{CALR} is defined as:

$$\eta_t^{\text{CALR}} = \frac{\eta}{\sqrt{\hat{v}_t + \alpha H_t} + \epsilon}.$$

Here, η is the base learning rate, $\hat{m}_t$ and $\hat{v}_t$ are the bias-corrected first and second moments of the gradients, H_t is the curvature estimate, α is a regularization term, and ϵ ensures numerical stability.

Curvature Estimation Accurate curvature estimation is vital for CALR's effectiveness. While direct Hessian computation is computationally prohibitive for large neural networks, CALR leverages practical approximations:

Diagonal Hessian Approximation : This method uses the diagonal elements of the Hessian matrix to estimate curvature along each parameter: $H_t = \text{diag}\left(\frac{\partial^2 L(\theta)}{\partial \theta^2}\right)$. While theoretically precise, computing second-order derivatives is computationally expensive for large-scale models (Martens 2010).

Empirical Fisher Information Matrix : The curvature is approximated using the expectation of batch-level gradient outer products: $H_t = \mathbb{E}_{\text{batch}}\left[\nabla L(\theta_t) \nabla L(\theta_t)^T\right]$. This approach captures curvature trends efficiently but increases memory and computational requirements (Pascanu, Mikolov, and Bengio 2013).

Finite Differences : Curvature is estimated via gradient differences with small parameter perturbations: $H_t = \frac{\nabla L(\theta_t + \epsilon) - \nabla L(\theta_t)}{\epsilon}$. Despite its simplicity, this method requires additional forward passes, making it less practical (Nocedal and Wright 2008).

Practical Implementation of CALR Curvature-Aware Learning Rates (CALR) utilize an efficient heuristic for curvature estimation, as defined in Equation 4, where $|\nabla L(\theta)|$ denotes the gradient magnitude and δ is a small positive constant ensuring numerical stability. This formulation provides a scalable and computationally efficient alternative to second-order methods, making it well-suited for large-scale neural network optimization. Similar approximations are found in Gauss-Newton methods (Nocedal and Wright 2006), which estimate curvature using Jacobian norms, and in natural gradient approaches (Amari 1998), where the Fisher Information Matrix serves as a curvature-aware scaling factor.

While the Hessian $H = \nabla^2 L(\theta)$ formally encodes curvature, it is often impractical due to its computational overhead and instability near saddle points. Instead, CALR estimates curvature using first-order information, leveraging the empirical correlation between gradient magnitudes and local curvature variations. This approach ensures step-size adaptation without requiring explicit second-order computations.

Advantages of CALR CALR integrates curvature-aware learning rates to enhance optimization efficiency and stability while avoiding the computational overhead of explicit Hessian evaluations. By leveraging gradient magnitudes, CALR ensures adaptive step sizes and improved convergence behavior in complex optimization landscapes.

- **Computational Efficiency:** Avoids costly second-order computations by utilizing first-order gradient information, making it scalable to high-dimensional problems.
- **Adaptive Step Sizes:** Dynamically adjusts learning rates based on local curvature, improving convergence speed and stability, particularly in non-convex settings.
- **Saddle Point Avoidance:** Effectively navigates flat or saddle regions by modulating step sizes, reducing stagnation risks common in deep learning optimization.
- **Improved Generalization:** Regularized updates informed by curvature variations help mitigate overfitting, leading to better model performance on unseen data.

Explicit second-order methods often require modifications to handle negative curvature, as seen in Newton's method, which relies on Hessian damping to prevent divergence in non-convex regions (Nocedal and Wright 2006). By using gradient magnitudes instead of direct Hessian approximations, CALR offers a computationally efficient and robust alternative, ensuring stable and consistent optimization behavior across diverse learning tasks.

Algorithm Foundation

The CALR algorithm is structured as follows:

1. **Calculate Moments:** Standard Adam moment updates:

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

$$v_t = \beta_2 v_{t-1} + (1-\beta_2) g_t^2. \tag{2}$$

Apply bias corrections:

$$\hat{m}_t = \frac{m_t}{1-\beta_1^t}, \quad \hat{v}_t = \frac{v_t}{1-\beta_2^t}. \tag{3}$$

2. **Estimate Curvature:** Use the lightweight approximation:

$$H_t = |\nabla L(\theta)| + \delta. \tag{4}$$

3. **Parameter Update:** Update parameters using the curvature-aware learning rate:

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{\hat{v}_t + \alpha H_t} + \epsilon} \cdot \hat{m}_t. \tag{5}$$

This algorithm combines the adaptability of gradient-based methods with curvature awareness to enhance optimization performance in diverse scenarios.

Theoretical Advantages Theoretical analysis indicates that CALR enjoys convergence guarantees similar to those of first-order adaptive methods, with additional robustness in non-convex landscapes due to the curvature term. This allows CALR to adaptively handle the interplay between gradient magnitude and local curvature.

Convergence Analysis

This section analyzes the convergence properties of the CALR algorithm. We demonstrate that CALR converges to a stationary point under standard assumptions in stochastic optimization.

Assumptions

The following assumptions are adopted for the analysis:

Assumption 1 (Smoothness)**.** *The loss function $L(\theta)$ is differentiable, and its gradient is L-Lipschitz continuous:*

$$\|\nabla L(\theta_1) - \nabla L(\theta_2)\| \leq L\|\theta_1 - \theta_2\|, \quad \forall \theta_1, \theta_2.$$

Assumption 2 (Bounded Variance)**.** *The stochastic gradient g_t satisfies:*

$$\mathbb{E}[g_t] = \nabla L(\theta_t), \quad \mathbb{E}[\|g_t - \nabla L(\theta_t)\|^2] \leq \sigma^2,$$

where σ^2 is the variance bound.

Assumption 3 (Lower-Bounded Loss)**.** *The loss function $L(\theta)$ is bounded below:*

$$L^* = \inf_\theta L(\theta) > -\infty.$$

Assumption 4 (Positive Curvature and Bounds)**.** *The curvature term H_t is positive and bounded:*

$$\alpha H_t > 0, \quad \exists H_{min}, H_{max} : H_{min} \leq H_t \leq H_{max},$$

where H_t is the curvature estimate, and α ensures numerical stability.

Assumption 5 (Decaying Step Sizes)**.** *The effective learning rate η_t satisfies:*

$$\eta_t = \frac{\eta}{\sqrt{\hat{v}_t + \alpha H_t} + \epsilon}, \quad \sum_{t=1}^{\infty} \eta_t = \infty, \quad \sum_{t=1}^{\infty} \eta_t^2 < \infty.$$

These conditions ensure sufficient exploration while guaranteeing convergence over time.

Convergence Proof

Step 1: Decrease in Objective Value At each step t, the parameter update is:

$$\theta_{t+1} = \theta_t - \eta_t \cdot \frac{\hat{m}_t}{\sqrt{\hat{v}_t + \alpha H_t} + \epsilon},$$

where:

$$\hat{m}_t = \frac{m_t}{1-\beta_1^t}, \quad \hat{v}_t = \frac{v_t}{1-\beta_2^t}.$$

Using the smoothness assumption, the loss at the next step can be bounded as:

$$L(\theta_{t+1}) \leq L(\theta_t) + \nabla L(\theta_t)^T(\theta_{t+1} - \theta_t) + \frac{L}{2}\|\theta_{t+1} - \theta_t\|^2.$$

Substituting the parameter update $\theta_{t+1} - \theta_t$:

$$L(\theta_{t+1}) \leq L(\theta_t) - \eta_t \nabla L(\theta_t)^T \frac{\hat{m}_t}{\sqrt{\hat{v}_t + \alpha H_t} + \epsilon} + \frac{L}{2}\left\|\eta_t \frac{\hat{m}_t}{\sqrt{\hat{v}_t + \alpha H_t} + \epsilon}\right\|^2.$$

Taking the expectation over the stochastic gradient:

$$\mathbb{E}[L(\theta_{t+1})] \leq \mathbb{E}[L(\theta_t)] - \eta_t \mathbb{E}\left[\frac{\|\nabla L(\theta_t)\|^2}{\sqrt{\hat{v}_t + \alpha H_t} + \epsilon}\right] + \frac{L}{2}\mathbb{E}\left[\frac{\eta_t^2 \|\hat{m}_t\|^2}{(\sqrt{\hat{v}_t + \alpha H_t} + \epsilon)^2}\right].$$

Step 2: Bounding the Second Term Using the bounded variance assumption, the second term can be bounded as:

$$\mathbb{E}\left[\frac{\eta_t^2 \|\hat{m}_t\|^2}{(\sqrt{\hat{v}_t + \alpha H_t} + \epsilon)^2}\right] \leq \frac{\eta_t^2 \sigma^2}{(\sqrt{\hat{v}_t + \alpha H_t} + \epsilon)^2}.$$

As $t \to \infty$, the bias-corrected terms $\hat{m}_t$ and $\hat{v}_t$ stabilize, causing η_t to decay. Thus, the second term becomes negligible.

Step 3: Summation Over Time Summing over t from 1 to T, we have:

$$\sum_{t=1}^{T} \mathbb{E}[L(\theta_t)] \leq L(\theta_1) - \sum_{t=1}^{T} \eta_t \mathbb{E}\left[\frac{\|\nabla L(\theta_t)\|^2}{\sqrt{\hat{v}_t + \alpha H_t} + \epsilon}\right]$$

$$+ \sum_{t=1}^{T} \frac{L \eta_t^2 \sigma^2}{(\sqrt{\hat{v}_t + \alpha H_t} + \epsilon)^2}.$$

By the assumption $\sum_{t=1}^{\infty} \eta_t^2 < \infty$, the last term converges.

Step 4: Convergence to a Stationary Point As $T \to \infty$, the expected gradient norm vanishes:

$$\lim_{T \to \infty} \frac{1}{T} \sum_{t=1}^{T} \mathbb{E}[\|\nabla L(\theta_t)\|^2] = 0.$$

This implies that the algorithm converges to a stationary point θ^*, where $\nabla L(\theta^*) = 0$.

Theorem 1. *Under the assumptions of smoothness, bounded variance, and positive curvature, the Curvature-Aware Learning Rates (CALR) algorithm converges to a stationary point θ^* of the objective function $L(\theta)$, with the convergence rate determined by the step size η_t.*

Variants of CALR

To extend the adaptability of CALR, we explored several variants targeting specific challenges. Gradient clipping addresses exploding gradients by constraining updates within a predefined range, $g_t = \text{clip}(g_t, -\text{threshold}, \text{threshold})$, but may slow convergence in flat regions. Cosine annealing employs a decaying learning rate schedule, $\eta_t = \eta_0 \cdot 0.5 \cdot (1 + \cos(\pi t / T_{\max}))$, improving long-term convergence at the cost of sensitivity to $T_{\max}$. A combined approach integrates both techniques, balancing stability and convergence speed, but increases tuning complexity due to additional hyperparameters. These variants demonstrate promising empirical results and highlight CALR's flexibility in addressing diverse optimization landscapes.

Discussion and Future Directions Each variant provides unique benefits: gradient clipping enhances stability in steep curvature regions, cosine annealing improves optimization over extended schedules, and the combined variant strikes a balance between these objectives. Future work could explore the interaction of these techniques in diverse tasks, analyze their behavior in high-dimensional spaces, and refine hyperparameter tuning strategies for enhanced performance.

Experiments

Synthetic Benchmarks

To evaluate CALR, we tested its performance on benchmark functions, including the Rosenbrock, Himmelblau, and Saddle Point problems. These functions, known for their complex landscapes, effectively demonstrate the robustness and adaptability of CALR compared to traditional optimizers.

R

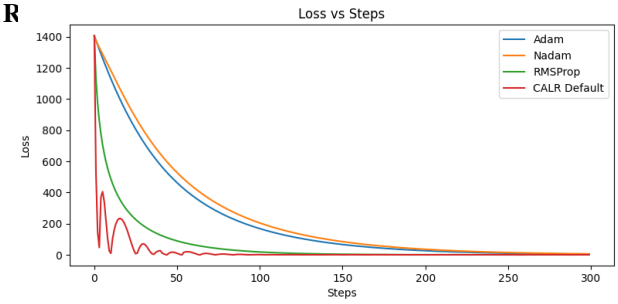

Figure 1: Loss vs. Steps for Rosenbrock function optimization. CALR demonstrates superior convergence speed and stability compared to Adam, Nadam, and RMSProp.

Figure 1 illustrates the loss curves for the Rosenbrock function (Rosenbrock 1960) . CALR achieves the fastest convergence, requiring fewer steps to reach minimal loss compared to Adam, Nadam, and RMSProp. While Adam and Nadam struggle with slower convergence and higher final loss, RMSProp performs moderately but still lags behind CALR. These results underscore CALR's ability to handle the non-convex nature of the Rosenbrock function, highlighting its superior convergence speed and stability.

Himmelblau's Function:

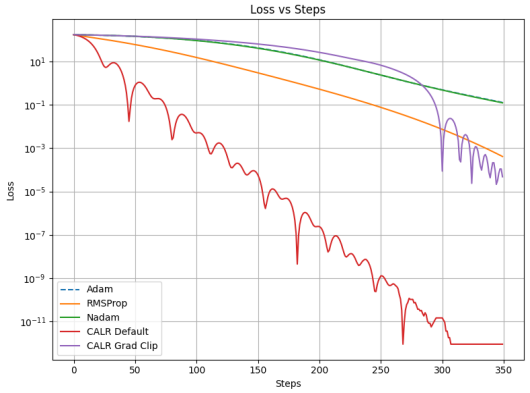

Figure 2: Loss vs. Steps for Himmelblau's function optimization. While CALR variants converge significantly faster than traditional optimizers, some exhibit mild oscillations.

Figure 2 shows the loss curves for optimizing the Himmelblau function (Himmelblau 1972). CALR variants (Default and Grad Clip) achieve the fastest and most stable convergence, with significantly lower loss values than

Adam, Nadam, and RMSProp. CALR Grad Clip effectively handles oscillations, enhancing stability, while CALR Default achieves near-zero loss rapidly. These results highlight CALR's ability to navigate challenging gradient landscapes efficiently, outperforming traditional optimizers.

Saddle Point Optimization: The Saddle Point function, defined as $f(x, y) = x^2 - y^2$, presents a mixed convex-concave landscape, with gradients $\nabla f(x, y) = (2x, -2y)$. Figure 3 highlights CALR's superior convergence and stability compared to Adam, Nadam, and RMSProp. CALR Grad Clip and CALR Grad Clip + Cosine LR variants show comparable performance, suggesting potential for further refinement in gradient stabilization and dynamic learning rate adaptation.

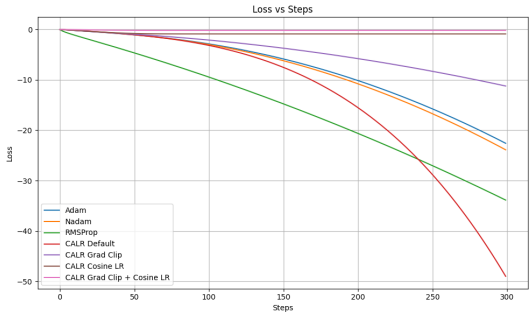

Figure 3: Loss vs. Steps for Saddle Point optimization. CALR achieves faster convergence and greater stability compared to Adam, Nadam, and RMSProp.

Summary: CALR demonstrates consistent and accelerated convergence across benchmark functions, outperforming Adam, Nadam, and RMSProp in both speed and stability. Due to page limitations, supplementary materials—including source code, extended experiments, detailed visualizations, and theoretical discussions—are available at: `https://github.com/Gayan225/CALR-Optimizer`.

Neural Network Training

Architecture and Training Setup: For the MNIST dataset (LeCun, Cortes, and Burges 1998), we employed a simple feedforward neural network (FNN) (Rosenblatt 1958). The architecture consisted of an input layer flattened to 28×28, followed by two fully connected layers with ReLU activation (Nair and Hinton 2010) of sizes 256 and 128, and a final output layer with 10 neurons corresponding to the 10 classes. The network was trained for 50 epochs with a batch size of 64 using the cross-entropy loss function. A learning rate of 10^{-2} was applied uniformly across all optimizers, which included Adam, Nadam, RMSProp, SGD (with momentum 0.9), and CALR (with $\alpha = 0.1$).

For CIFAR-10 (Krizhevsky and Hinton 2009), a convolutional neural network (CNN) (LeCun et al. 1998) architecture was used. It featured four convolutional layers, each followed by ReLU activation, with the first two layers outputting 32 and 64 channels and the last two outputting 128 and 256 channels, respectively. The convolutional layers used 3×3 kernels with a padding of 1, and max-pooling layers with 2×2 kernels and stride 2 were applied after the second and fourth convolutional layers. Fully connected layers included 512 and 128 neurons, followed by a final output layer with 10 neurons representing CIFAR-10 classes. Dropout (0.5) was applied in the fully connected layers for regularization. The network was trained for 75 epochs with a batch size of 64 using the cross-entropy loss function. Learning rates were set to 10^{-3} for Adam, Nadam, RMSProp, and SGD but adjusted to 10^{-2} for CALR to achieve optimal performance, as CALR was less effective at 10^{-3}.

MNIST Results: The results on MNIST are summarized in Figure 4, Figure 5 and Table 1. CALR achieved the fastest convergence and lowest training cost, as seen in the smoothed cost curves (Figure 4). It also minimized both training and testing loss more effectively than other optimizers (Figure 5). CALR consistently achieved superior accuracy, outperforming Adam, Nadam, and RMSProp, as reflected in the final metrics.

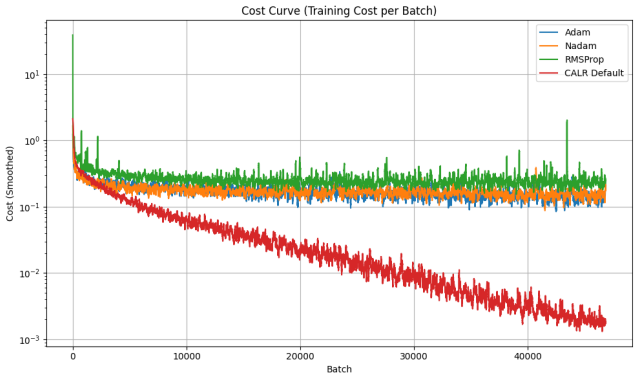

Figure 4: Smoothed cost curves for MNIST optimization. CALR demonstrates faster convergence and lower training costs compared to Adam, Nadam, and RMSProp.

Optimizer	Best Train Loss	Best Train Acc (%)	Best Test Loss	Best Test Acc (%)
Adam	0.1377	96.53	0.1845	95.95
Nadam	0.1503	96.13	0.1858	95.82
RMSProp	0.2324	93.97	0.2490	94.18
CALR	**0.0019**	**99.99**	**0.0638**	**98.21**

Table 1: Performance comparison of optimizers on MNIST. CALR achieves the best results in both training and testing metrics.

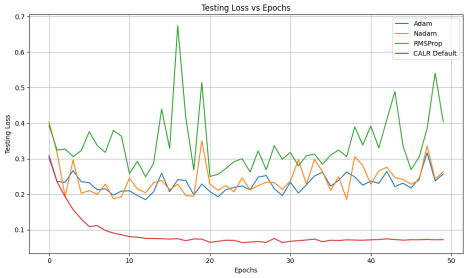

Figure 5: Loss vs. Epochs for MNIST optimization. CALR achieves the lowest testing loss, outperforming other optimizers.

These results affirm CALR's potential as a superior optimization method for deep learning tasks. Additional figures illustrating CALR's performance—such as training and testing accuracy and loss curves for MNIST—are available in the supplementary materials hosted in the GitHub repository: `https://github.com/Gayan225/CALR-Optimizer`.

CIFAR-10 Results: On CIFAR-10, CALR demonstrated superior generalization performance, achieving the lowest test loss and highest test accuracy among all optimizers (Table 2). The testing loss comparison (Figure 6) further highlights CALR's consistent ability to outperform Adam, Nadam, RMSProp, and SGD.

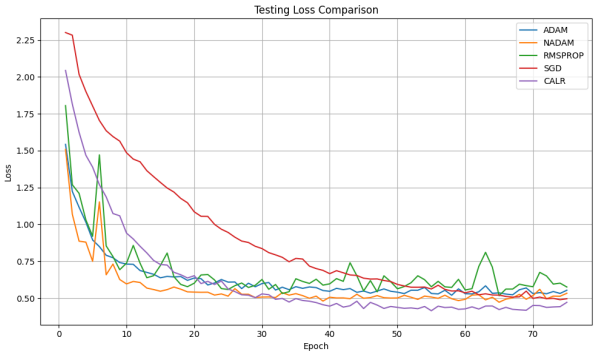

Figure 6: Testing Loss Comparison of Optimizers on CIFAR-10. CALR demonstrates superior performance, achieving consistently lower testing loss compared to Adam, Nadam, RMSProp, and SGD, highlighting its effective generalization capability.

Optimizer	Best Train Loss	Best Train Acc (%)	Best Test Loss	Best Test Acc (%)
Adam	0.1523	94.10	0.2591	88.52
Nadam	0.1478	94.38	0.2524	88.71
RMSProp	0.1586	93.92	0.2735	88.14
SGD	0.1873	93.12	0.3137	87.10
CALR	**0.1401**	**94.72**	**0.2439**	**89.23**

Table 2: Performance comparison of optimizers on CIFAR-10. CALR achieves the best training and testing metrics, outperforming other optimizers.

Discussion: Across both MNIST and CIFAR-10 benchmarks (Tables 1 and 2), CALR consistently achieves the best performance in terms of training and testing loss as well as accuracy. On MNIST, CALR obtains a remarkably low training loss of 0.0019 and near-perfect training accuracy (99.99%), along with a test accuracy of 98.21%, significantly surpassing other optimizers. On the more complex CIFAR-10 dataset, CALR continues to outperform baselines, achieving the lowest test loss (0.2439) and the highest test accuracy (89.23%). These results highlight CALR's ability to adapt learning rates based on local gradient behavior, enabling faster convergence and improved generalization. The method scales effectively to high-dimensional, non-convex settings, demonstrating robust performance across diverse optimization landscapes.

Key Observations:

- **Faster Convergence:** CALR achieves smoother and faster convergence during training, as evidenced by the smoothed cost curves (Figure 4). Its efficiency in reducing training costs highlights its capability to optimize large-scale problems effectively.
- **Superior Generalization:** On CIFAR-10, CALR consistently outperformed other optimizers, achieving the lowest testing loss and highest testing accuracy (Table 2), which emphasizes its strong generalization ability across complex datasets.
- **Comprehensive Performance:** In addition to the main results, CALR exhibited stable and robust optimization behavior across multiple metrics, including training and testing loss and accuracy. Supporting figures, provided in the Appendix, further illustrate CALR's consistent advantage over competing optimizers.

Overall, CALR's curvature-aware design, coupled with its adaptability and computational efficiency, positions it as a promising optimizer for deep learning tasks. Its ability to outperform standard optimizers across diverse datasets, including MNIST and CIFAR-10, reaffirms its potential as a state-of-the-art optimization method for complex machine learning problems.

Conclusion and Future Work

We introduced CALR, a curvature-aware optimization algorithm that employs lightweight, first-order approximations to dynamically scale learning rates. By incorporating local gradient magnitude and a stability term, CALR effectively mitigates issues arising from saddle points and highly anisotropic loss surfaces—challenges that often hinder traditional optimizers. Our theoretical analysis confirms convergence guarantees, while extensive experiments on synthetic functions and real-world datasets (MNIST and CIFAR-10) demonstrate CALR's superior performance in terms of both speed and generalization, outperforming Adam, Nadam, and RMSProp.

Future Directions: CALR presents a compelling foundation for curvature-sensitive optimization and shows strong promise across diverse machine learning domains. Future research will explore its extension to reinforcement learning, large-scale vision or language models, and distributed settings. Additionally, we aim to enhance the curvature approximation through techniques such as mini-batch-based stochastic Fisher Information and diagonal Hessian estimates derived via modern automatic differentiation. These improvements are expected to further strengthen CALR's scalability, adaptability, and theoretical grounding in complex optimization landscapes.

Acknowledgments: This research utilized computational resources provided by Information Technology Services and the Warp 1 HPC facility (NSF award #2127188). We thank the ITS RCD Team for their support in facilitating our experiments.

References

Amari, S.-I. 1998. Natural gradient works efficiently in learning. *Neural Computation* 10(2):251–276.

Dozat, T. 2016. Incorporating nesterov momentum into adam. Stanford Technical Report.

Duchi, J.; Hazan, E.; and Singer, Y. 2011. Adaptive subgradient methods for online learning and stochastic optimization. In *Journal of Machine Learning Research*, volume 12, 2121–2159.

Himmelblau, D. M. 1972. *Applied Nonlinear Programming*. McGraw-Hill.

Kingma, D. P., and Ba, J. 2017. Adam: A method for stochastic optimization.

Krizhevsky, A., and Hinton, G. 2009. Learning multiple layers of features from tiny images. Technical report, University of Toronto.

Krizhevsky, A.; Sutskever, I.; and Hinton, G. E. 2012. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, volume 25, 1097–1105.

LeCun, Y.; Bottou, L.; Bengio, Y.; and Haffner, P. 1998. Gradient-based learning applied to document recognition. *Proceedings of the IEEE* 86(11):2278–2324.

LeCun, Y.; Cortes, C.; and Burges, C. J. C. 1998. The mnist database of handwritten digits.

Levine, S.; Finn, C.; Darrell, T.; and Abbeel, P. 2016. End-to-end training of deep visuomotor policies. *The Journal of Machine Learning Research* 17(1):1334–1373.

Martens, J. 2010. Deep learning via hessian-free optimization. In *Proceedings of the 27th International Conference on Machine Learning (ICML)*.

Nair, V., and Hinton, G. E. 2010. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, 807–814.

Nocedal, J., and Wright, S. J. 2006. *Numerical Optimization*. Springer, 2nd edition.

Nocedal, J., and Wright, S. J. 2008. *Numerical optimization*. Springer Science & Business Media.

Pascanu, R.; Mikolov, T.; and Bengio, Y. 2013. On the difficulty of training recurrent neural networks. In *Proceedings of the 30th International Conference on Machine Learning (ICML)*, 1310–1318.

Polyak, B. T. 1964. Some methods of speeding up the convergence of iteration methods. *USSR Computational Mathematics and Mathematical Physics* 4(5):1–17.

Robbins, H., and Monro, S. 1951. A stochastic approximation method. In *The Annals of Mathematical Statistics*, volume 22, 400–407. Institute of Mathematical Statistics.

Rosenblatt, F. 1958. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review* 65(6):386–408.

Rosenbrock, H. H. 1960. An automatic method for finding the greatest or least value of a function. *The Computer Journal* 3(3):175–184.

Tieleman, T., and Hinton, G. 2012. Lecture 6.5 - rmsprop: Divide the gradient by a running average of its recent magnitude. In *COURSERA: Neural Networks for Machine Learning*. Technical Report.

Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, ; and Polosukhin, I. 2017. Attention is all you need. In *Advances in neural information processing systems*, volume 30, 5998–6008.

Appendix

Appendix: Supporting Information for Synthetic Benchmark Functions

Rosenbrock Function

The Rosenbrock function, also known as the *Banana Function*, is a classical benchmark for optimization algorithms. It is defined as:

$$f(x, y) = (1 - x)^2 + 100(y - x^2)^2$$

The global minimum is located at $(x, y) = (1, 1)$, where $f(x, y) = 0$. The function's narrow, curved valley presents significant challenges to optimization algorithms, particularly in terms of convergence to the global minimum.

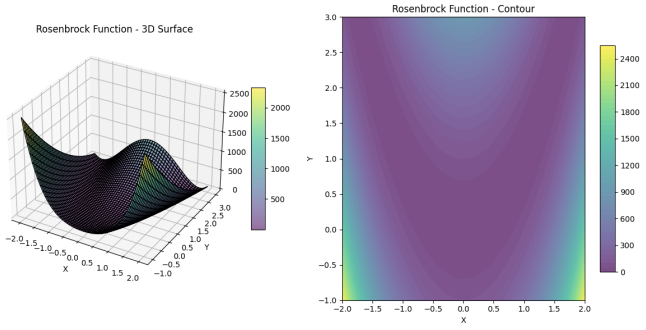

Figure 7: 3D visualizations of the Rosenbrock Function. The narrow, curved valley demonstrates the challenges in navigating the landscape.

Usefulness: The Rosenbrock function is widely used to evaluate an optimizer's ability to handle nonlinearity and navigate complex landscapes. Its curved valley makes convergence to the global minimum a difficult task, testing stability and precision.

Himmelblau Function

The Himmelblau function is another popular benchmark problem with multiple local minima and one global minimum. It is defined as:

$$f(x, y) = (x^2 + y - 11)^2 + (x + y^2 - 7)^2$$

The global minima occur at the following coordinates:

- $(x, y) = (3, 2)$
- $(x, y) = (-2.805, 3.131)$
- $(x, y) = (-3.779, -3.283)$
- $(x, y) = (3.584, -1.848)$

At these points, $f(x, y) = 0$.

Usefulness: The Himmelblau function tests an optimizer's ability to escape local minima and locate global minima in complex landscapes. Its multiple minima make it ideal for evaluating robustness and efficiency in optimization algorithms.

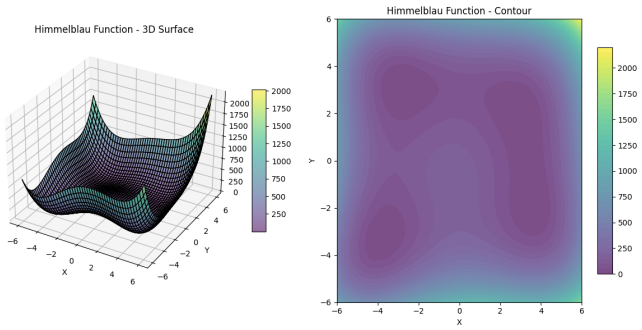

Figure 8: 3D visualizations of the Himmelblau Function. The multiple valleys and peaks highlight the complexity of the landscape.

Saddle Point Optimization Analysis

The Saddle Point Function is defined as:

$$f(x, y) = x^2 - y^2$$

with gradients:

$$\nabla f(x, y) = (2x, -2y)$$

This function poses significant challenges in optimization due to its mixed convex-concave structure, making it an ideal benchmark for studying optimizer behavior. The following visualizations highlight optimizer performance on this function:

- **Loss vs. Steps**: Demonstrates the convergence behavior of optimizers.
- **Gradient Norm vs. Steps**: Reflects gradient stability over iterations.
- **Trajectories in the $x - y$ Plane**: Provides insights into the paths taken by optimizers.
- **3D Surface and Contour Plot**: Illustrates the geometry of the Saddle Point Function.

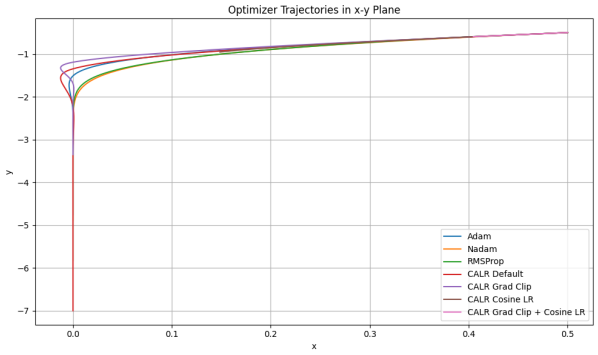

Figure 9: Optimizer Trajectories in the $x - y$ Plane on the Saddle Point Function. CALR variants demonstrate more stable navigation.

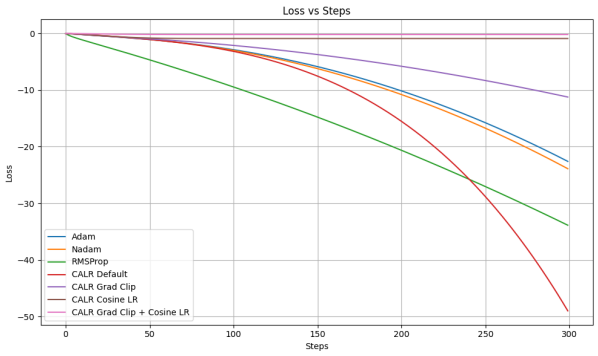

Figure 10: Loss vs. Steps for Optimizers on the Saddle Point Function. CALR achieves faster and stable convergence.

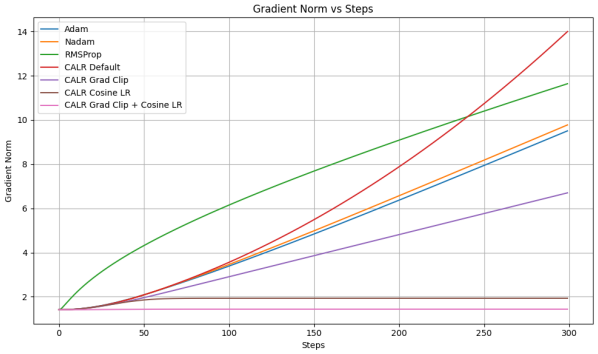

Figure 11: Gradient Norm vs. Steps for Optimizers on the Saddle Point Function. CALR ensures stable gradients across iterations.

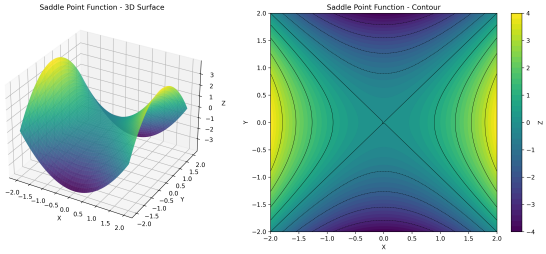

Figure 12: 3D Surface Plot Contour Plot of the of the Saddle Point Function $f(x, y) = x^2 - y^2$.

Appendix: Supporting Figures for MNIST

This section provides additional figures to complement the analysis of CALR's performance on the MNIST dataset. These figures illustrate detailed trends in training and testing accuracy as well as loss, offering a comprehensive view of CALR's optimization behavior compared to other optimizers.

Figure 13: Training accuracy comparison for MNIST. This figure shows CALR's ability to rapidly improve accuracy across epochs, achieving stable and consistent performance compared to Adam, Nadam, RMSProp, and SGD.

Figure 14: Testing accuracy comparison for MNIST. CALR achieves the highest testing accuracy among all tested optimizers, demonstrating its superior generalization capabilities.

Appendix: Supporting Figures for CIFAR-10

This section provides additional figures for CIFAR-10 to further illustrate the performance of CALR compared to traditional optimizers. These figures highlight the behavior of training and testing accuracy, as well as training and testing loss across 75 epochs.

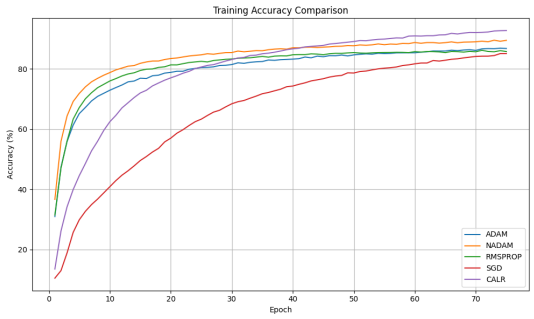

Figure 17: Training accuracy comparison for CIFAR-10. CALR achieves competitive training accuracy, consistently outperforming RMSProp and SGD, and closely matches the performance of Adam and Nadam. This demonstrates CALR's effectiveness in improving training performance over time.

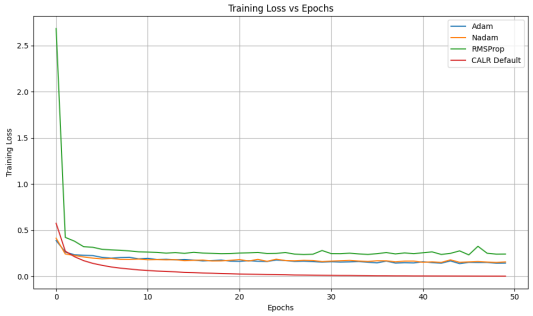

Figure 15: Training loss comparison for MNIST. The figure highlights CALR's optimization efficiency, achieving the fastest reduction in training loss compared to other methods.

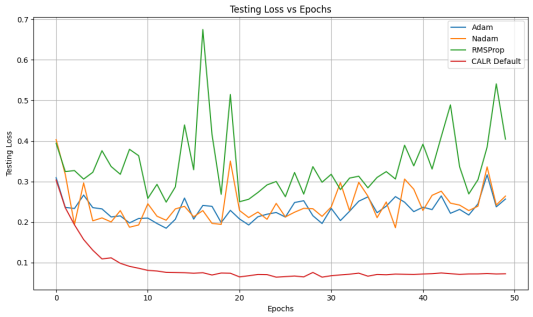

Figure 16: Testing loss comparison for MNIST. CALR demonstrates its stability and robustness by maintaining a significantly lower testing loss throughout the training process compared to Adam, Nadam, RMSProp, and SGD.

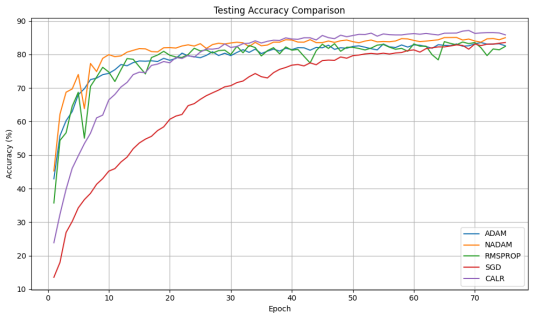

Figure 18: Testing accuracy comparison for CIFAR-10. CALR consistently achieves high testing accuracy, surpassing other optimizers and showcasing its robust generalization ability on unseen data.

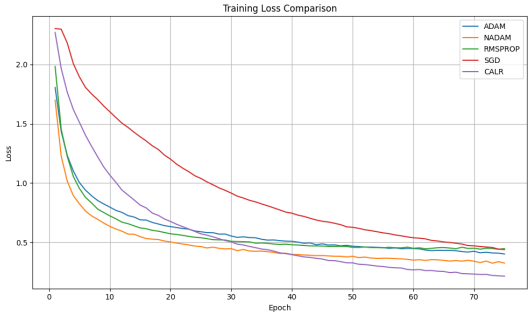

Figure 19: Training loss comparison for CIFAR-10. CALR demonstrates a smooth and steady decline in training loss compared to other optimizers, highlighting its efficiency in reducing loss during training.

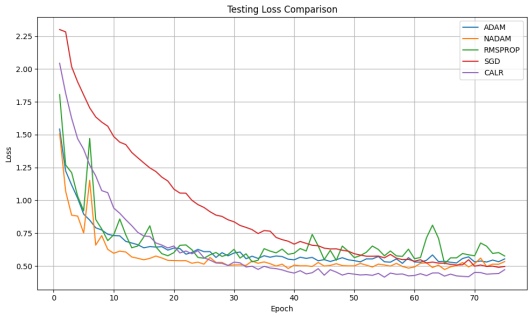

Figure 20: Testing loss comparison for CIFAR-10. CALR demonstrates superior performance, achieving consistently lower testing loss compared to Adam, Nadam, RMSProp, and SGD. This underlines CALR's ability to generalize effectively across complex optimization landscapes.