Appendix: For CALR Paper

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.

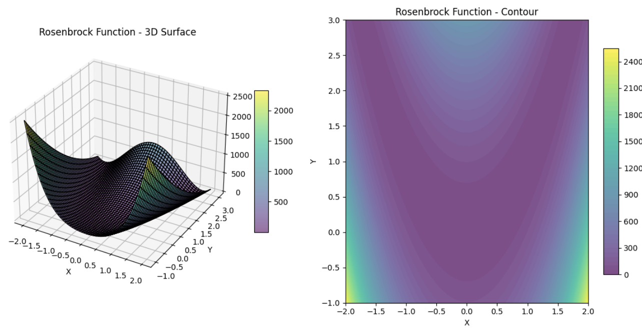

Figure 8: 3D visualizations of the Rosenbrock Function. The narrow, curved valley demonstrates the challenges in navigating the landscape.

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.

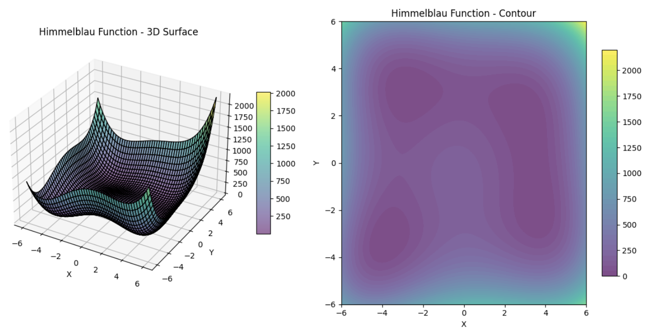

Figure 9: 3D visualizations of the Himmelblau Function. The multiple valleys and peaks highlight the complexity of the landscape.

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.

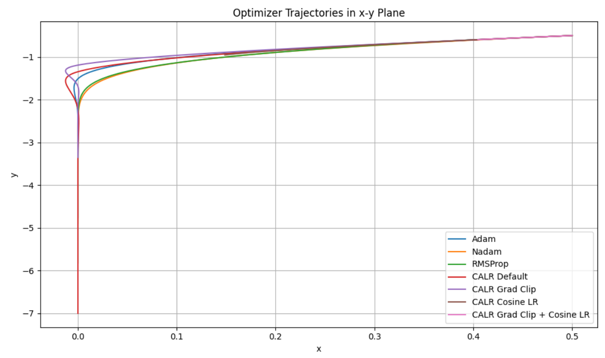

Figure 10: Optimizer Trajectories in the $x - y$ Plane on the Saddle Point Function. CALR variants demonstrate more stable navigation.

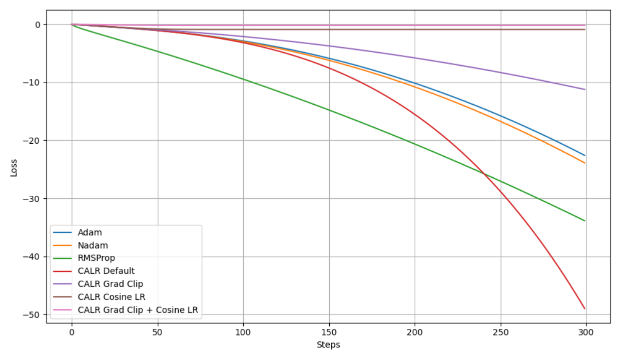

Figure 11: Loss vs. Steps for Optimizers on the Saddle Point Function. CALR achieves faster and stable convergence.

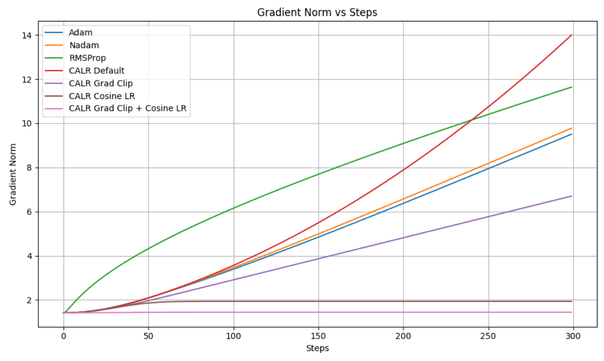

Figure 12: Gradient Norm vs. Steps for Optimizers on the Saddle Point Function. CALR maintains stable gradients across iterations.

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.

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.

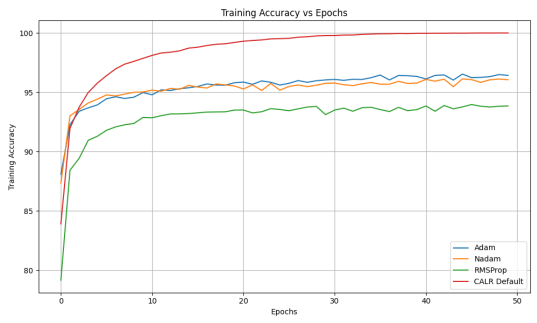

Figure 15: Training accuracy comparison for MNIST. This figure shows CALR's ability to rapidly improve accuracy across epochs, achieving stable and consistent performance compared to Adam, Nadam, RMSProp, and SGD.

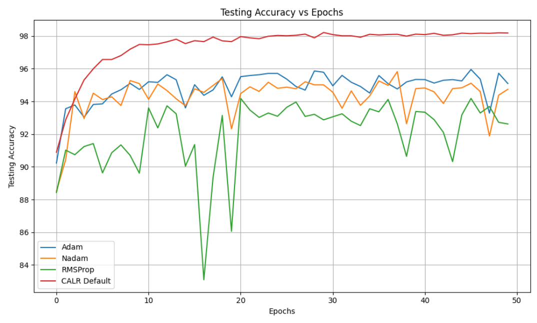

Figure 16: Testing accuracy comparison for MNIST. CALR achieves the highest testing accuracy among all tested optimizers, demonstrating its superior generalization capabilities.

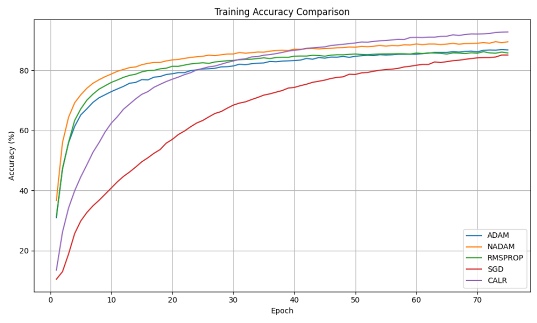

Figure 19: Training accuracy comparison for CIFAR-10. CALR achieves competitive training accuracy, consistently outperforming RMSProp and SGD, and closely matches the performance of Adam and Nadam. This demonstrates CALR's effectiveness in improving training performance over time.

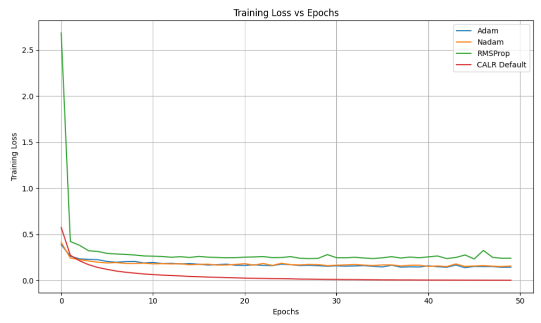

Figure 17: Training loss comparison for MNIST. The figure highlights CALR's optimization efficiency, achieving the fastest reduction in training loss compared to other methods.

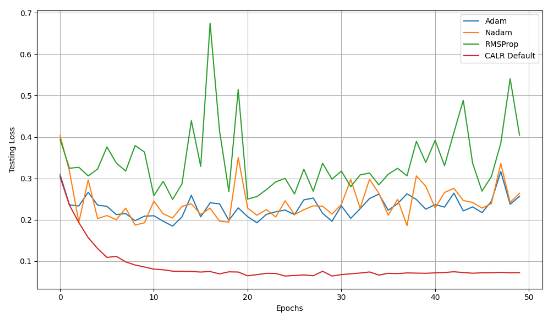

Figure 18: Testing loss comparison for MNIST. CALR demonstrates its stability and robustness by maintaining a significantly lower testing loss throughout the training process compared to Adam, Nadam, RMSProp, and SGD.

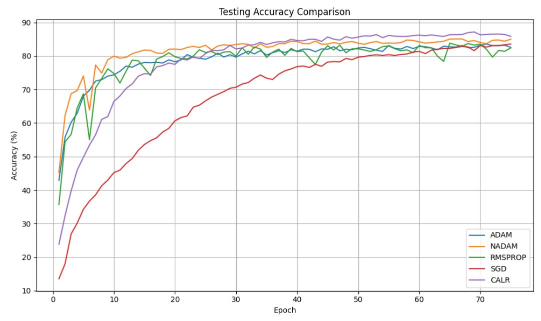

Figure 20: Testing accuracy comparison for CIFAR-10. CALR consistently achieves high testing accuracy, surpassing other optimizers and showcasing its robust generalization ability on unseen data.

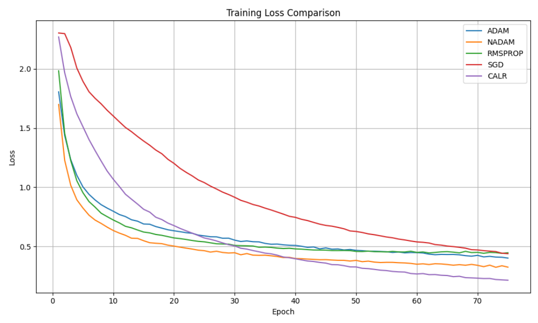

Figure 21: Training loss comparison for CIFAR-10. CALR demonstrates a smooth and steady decline in training loss compared to other optimizers, highlighting its efficiency in reducing loss during training.

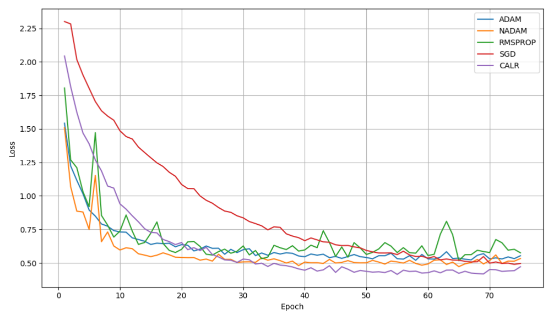

Figure 22: Testing loss comparison for CIFAR-10. CALR demonstrates superior performance, achieving consistently lower testing loss compared to Adam, Nadam, RMSProp, and SGD. This underlines CALR's ability to generalize effectively across complex optimization landscapes.

CALR Variant with Diagonal Hessian Curvature Estimation

To further evaluate the potential of using more refined curvature estimates, we implemented a variant of CALR that incorporates a diagonal approximation of the Hessian matrix. This approach leverages PyTorch's automatic differentiation tools to produce curvature estimates that more directly reflect second-order information than the gradient-magnitude heuristic used in standard CALR. Due to computational and memory overhead, we limited these experiments to a synthetic toy problem and a small subset of MNIST.

Figure 23 shows that CALR with diagonal Hessian converges significantly faster and reaches a lower loss compared to the gradient-magnitude-based CALR. A similar trend is observed in the MNIST subset (Figure 24), although the differences are less pronounced due to the network size and input complexity.

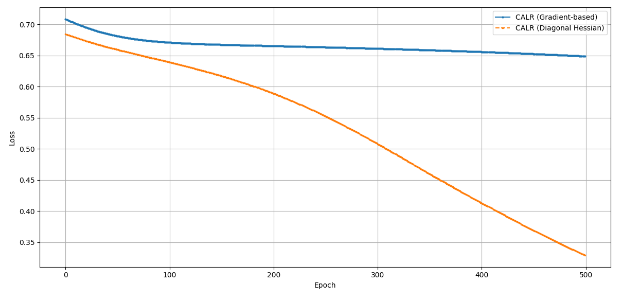

Figure 23: Loss curve comparison between CALR (Gradient-based) and CALR (Diagonal Hessian) on a toy problem. The diagonal Hessian variant shows significantly faster convergence.

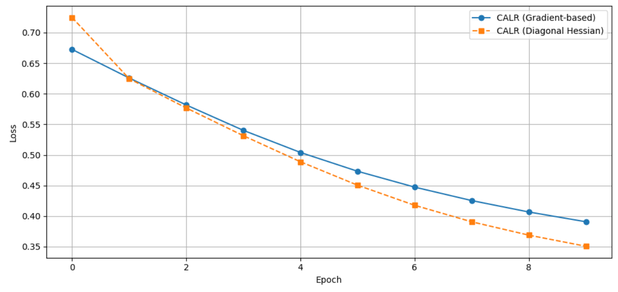

Figure 24: Loss comparison on a small MNIST subset using CALR variants. The diagonal Hessian variant achieves slightly better convergence.

These results suggest that incorporating more accurate curvature approximations can further enhance CALR's performance. However, the increased computational cost makes it challenging to scale to full datasets. Exploring efficient Hessian approximations in CALR for large-scale deep learning remains an important direction for future research.

Extended results and reproducible code are available at (Maduranga 2025).