A Study of Momentum-Based and Adaptive Optimization Methods in Deep Learning

Om Sai Krishna Madhav Lella

Michigan State University

lellaom@msu.edu

December 7, 2024

Overview

1. Introduction
2. Methodology
3. Results
4. Discussion and Conclusion

Introduction: Background and Motivation

Background:

- Optimization is fundamental to deep learning, significantly influencing the performance of the model.
- Optimizers in deep learning are designed to minimize loss functions such as Mean Squared Error, Cross-Entropy Loss, and others.

Motivation:

- Choosing the right optimizer is important as optimizer choice greatly affects training efficiency and model performance.
- Momentum-based and adaptive optimizers help us mitigate common issues faced in traditional gradient descent algorithms.

Introduction: Objective and Research Questions

Objective: To implement the following optimization methods as reusable classes, understand their key differences, and evaluate their performance across various deep learning tasks: SGD, Rprop, Adagrad, RMSprop, Adadelta, and Adam.

Research Questions

- **Q1:** What are the fundamental differences among the optimizers?
- **Q2:** How does the convergence of Stochastic Gradient Descent vary with momentum?
- **Q3:** What is the impact of varying learning rate and batch size on the convergence of different optimizers?
- **Q4:** Which optimizers perform best and demonstrate the highest robustness across various deep learning problems?

Methodology: Overview of Algorithms (1/3)

1. **Stochastic Gradient Descent With Momentum (SGD):**

$$v_i^{(k+1)} = \beta v_i^{(k)} - \alpha g_i^{(k)}$$
$$x_i^{(k+1)} = x_i^{(k)} + v_i^{(k+1)}$$

2. **Resilient Propagation (Rprop):**

$$\Delta_i^{(k+1)} = \begin{cases} \min(\eta^+ \Delta_i^{(k)}, \Delta_{\max}), & \text{if } g_i^{(k)} g_i^{(k-1)} > 0 \\ \max(\eta^- \Delta_i^{(k)}, \Delta_{\min}), & \text{if } g_i^{(k)} g_i^{(k-1)} < 0 \\ \Delta_i^{(k)}, & \text{if } g_i^{(k)} g_i^{(k-1)} = 0 \end{cases}$$

$$x_i^{(k+1)} = \begin{cases} x_i^{(k)} - \Delta_i^{(k+1)}, & \text{if } g_i^{(k)} > 0 \\ x_i^{(k)} + \Delta_i^{(k+1)}, & \text{if } g_i^{(k)} < 0 \\ x_i^{(k)}, & \text{if } g_i^{(k)} = 0 \end{cases}$$

Methodology: Overview of Algorithms (2/3)

3. **Adaptive Gradient (Adagrad):**

$$s_i^{(k)} = \sum_{j=1}^{k} \left(g_i^{(j)} \right)^2$$

$$x_i^{(k+1)} = x_i^{(k)} - \frac{\alpha}{\epsilon + \sqrt{s_i^{(k)}}} g_i^{(k)}$$

4. **Root Mean Square Propagation (RMSprop):**

$$s_i^{(k+1)} = \gamma s_i^{(k)} + (1 - \gamma) \left(g_i^{(k)} \right)^2$$

$$x_i^{(k+1)} = x_i^{(k)} - \frac{\alpha}{\epsilon + \sqrt{s_i^{(k+1)}}} g_i^{(k)}$$

Methodology: Overview of Algorithms (3/3)

5. **Adaptive Delta (Adadelta):**

$$
\begin{aligned}
s_i^{(k+1)} &= \gamma s_i^{(k)} + (1-\gamma)\left(g_i^{(k)}\right)^2 \\
r_i^{(k+1)} &= \gamma r_i^{(k)} + (1-\gamma)\left(\Delta x_i^{(k)}\right)^2 \\
\Delta x_i^{(k+1)} &= -\frac{\sqrt{r_i^{(k)}+\epsilon}}{\sqrt{s_i^{(k)}+\epsilon}} g_i^{(k)} \\
x_i^{(k+1)} &= x_i^{(k)} + \Delta x_i^{(k+1)}
\end{aligned}
$$

6. **Adaptive Moment Estimation (Adam):**

$$
\begin{aligned}
v_i^{(k+1)} &= \gamma_v v^{(k)} + (1-\gamma_v) g_i^{(k)} \\
s_i^{(k+1)} &= \gamma_s s^{(k)} + (1-\gamma_s)\left(g^{(k)}\right)^2 \\
\hat{v}_i^{(k+1)} &= \frac{v_i^{(k+1)}}{1-\gamma_v^k} \\
\hat{s}_i^{(k+1)} &= \frac{s_i^{(k+1)}}{1-\gamma_s^k} \\
x_i^{(k+1)} &= x_i^{(k)} - \frac{\alpha \hat{v}_i^{(k+1)}}{\epsilon + \sqrt{\hat{s}_i^{(k+1)}}}
\end{aligned}
$$

Methodology: Deep Learning Tasks

Auto MPG Prediction

- **Objective:** Predict MPG (fuel efficiency)
- **Loss Function:** MAE with L2 Regularization

California Housing Price Prediction

- **Objective:** Predict house price
- **Loss Function:** MSE with L2 Regularization

Breast Cancer Prediction

- **Objective:** Classify tumor as malignant or benign
- **Loss Function:** BCE Loss with L2 Regularization

Fashion MNIST Classification

- **Objective:** Classify images into one of 10 fashion item classes
- **Loss Function:** Cross-Entropy Loss with L2 Regularization

Methodology: Loss Functions

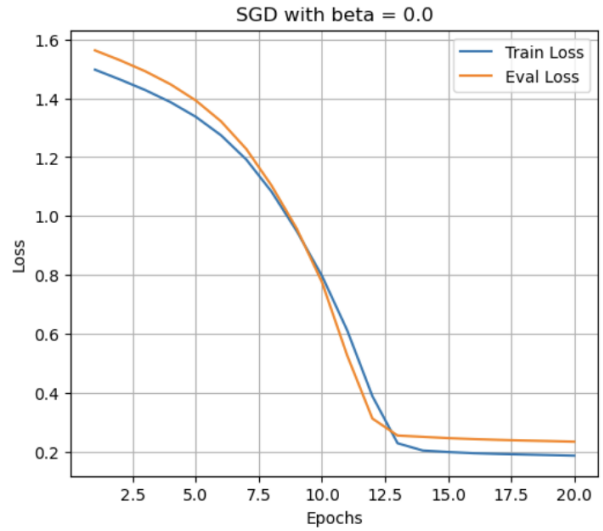
$$\text{MAE with L2} = \frac{1}{N}\sum_{i=1}^{N}|y_i - \hat{y}_i| + \lambda\sum_{j=1}^{d} w_j^2$$

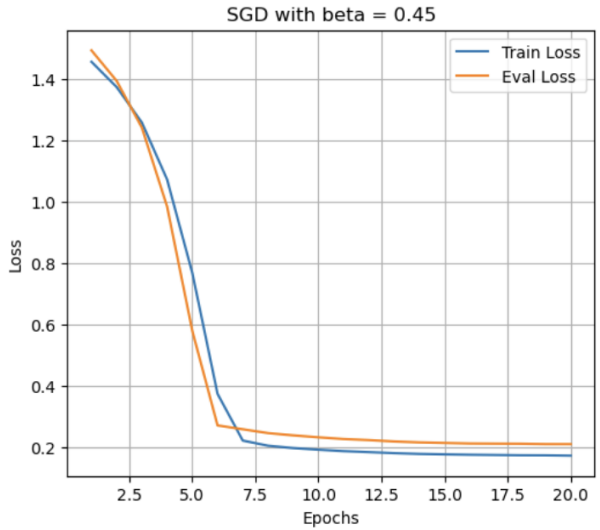
$$\text{MSE with L2} = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2 + \lambda\sum_{j=1}^{d} w_j^2$$

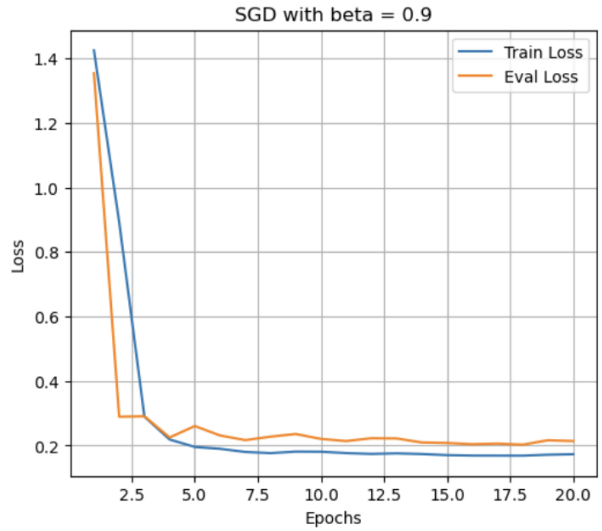
$$\text{BCE Loss with L2} = -\frac{1}{N}\sum_{i=1}^{N}[y_i \log(\hat{y}_i) + (1 - y_i)\log(1 - \hat{y}_i)] + \lambda\sum_{j=1}^{d} w_j^2$$

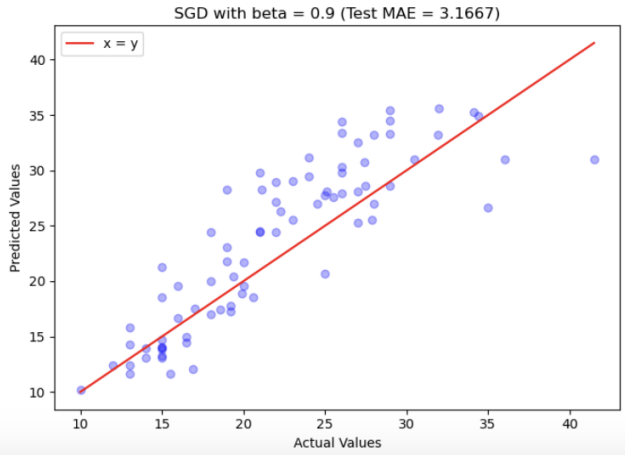
$$\text{Cross-Entropy Loss with L2} = -\frac{1}{N}\sum_{i=1}^{N}\sum_{k=1}^{K} y_{i,k}\log(\hat{y}_{i,k}) + \lambda\sum_{j=1}^{d} w_j^2$$

Results: Convergence of SGD vs Momentum Coefficient

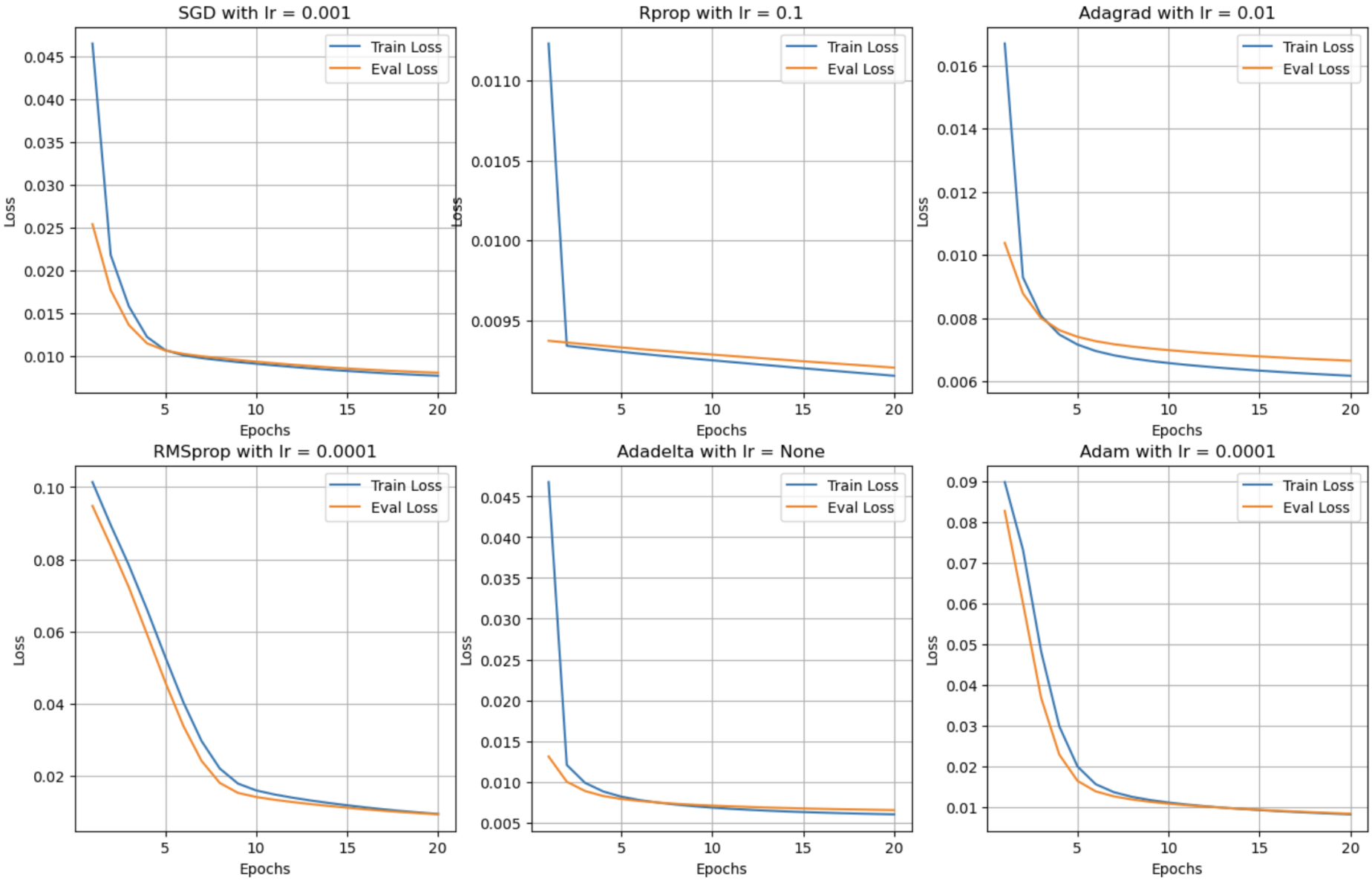




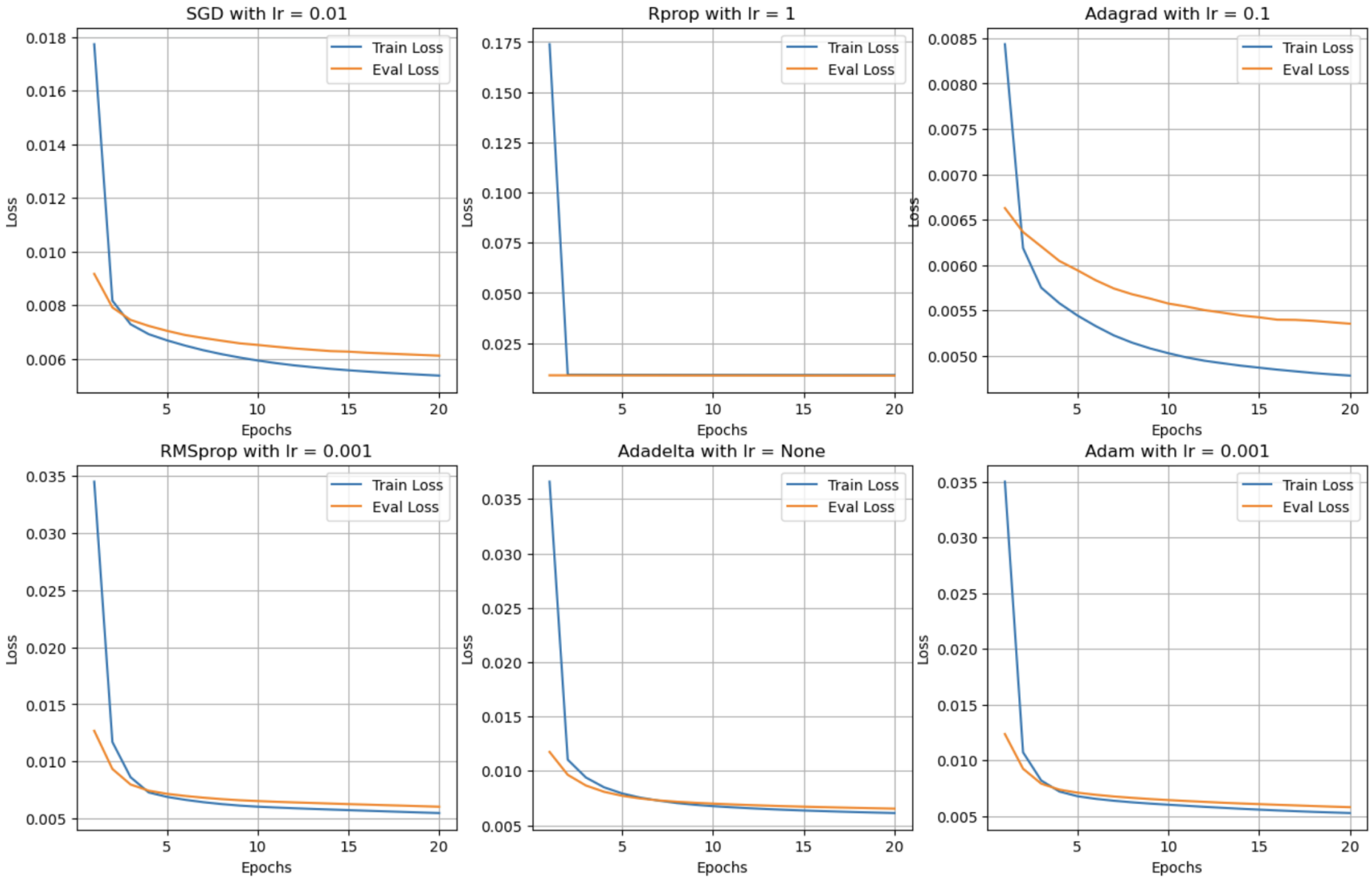




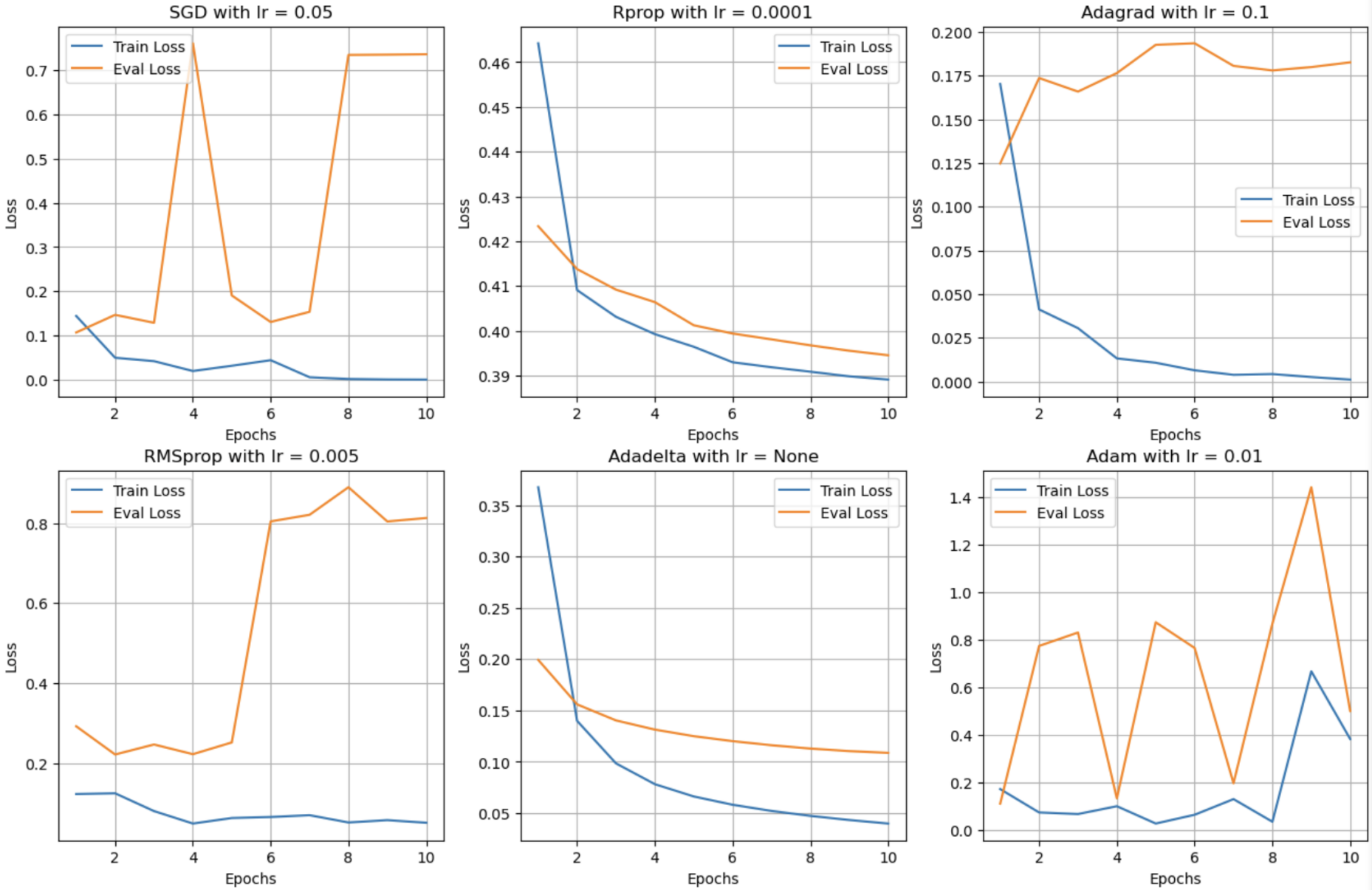
Results: Optimization vs Learning Rate (1/2)



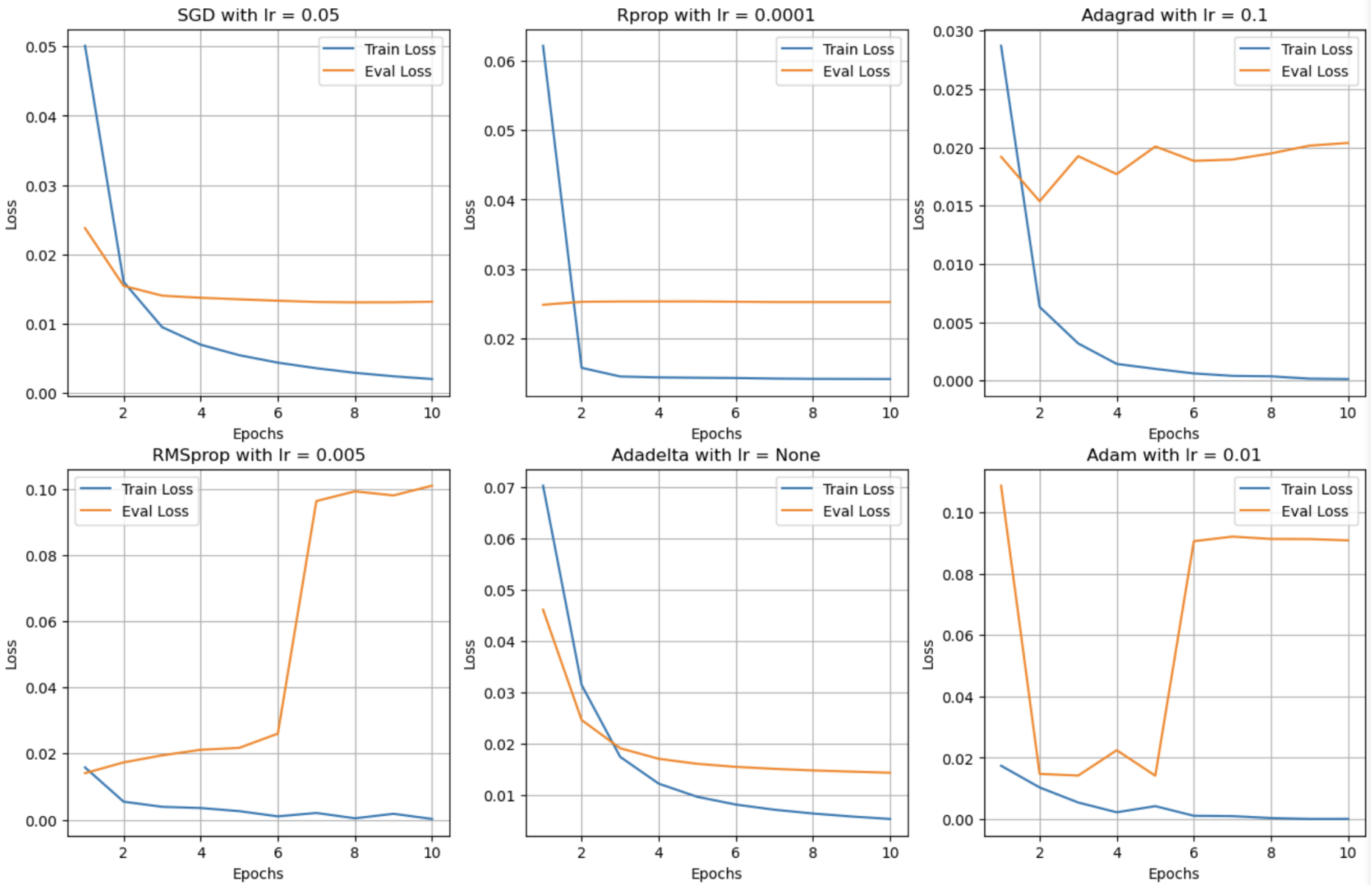
Results: Optimization vs Learning Rate (2/2)



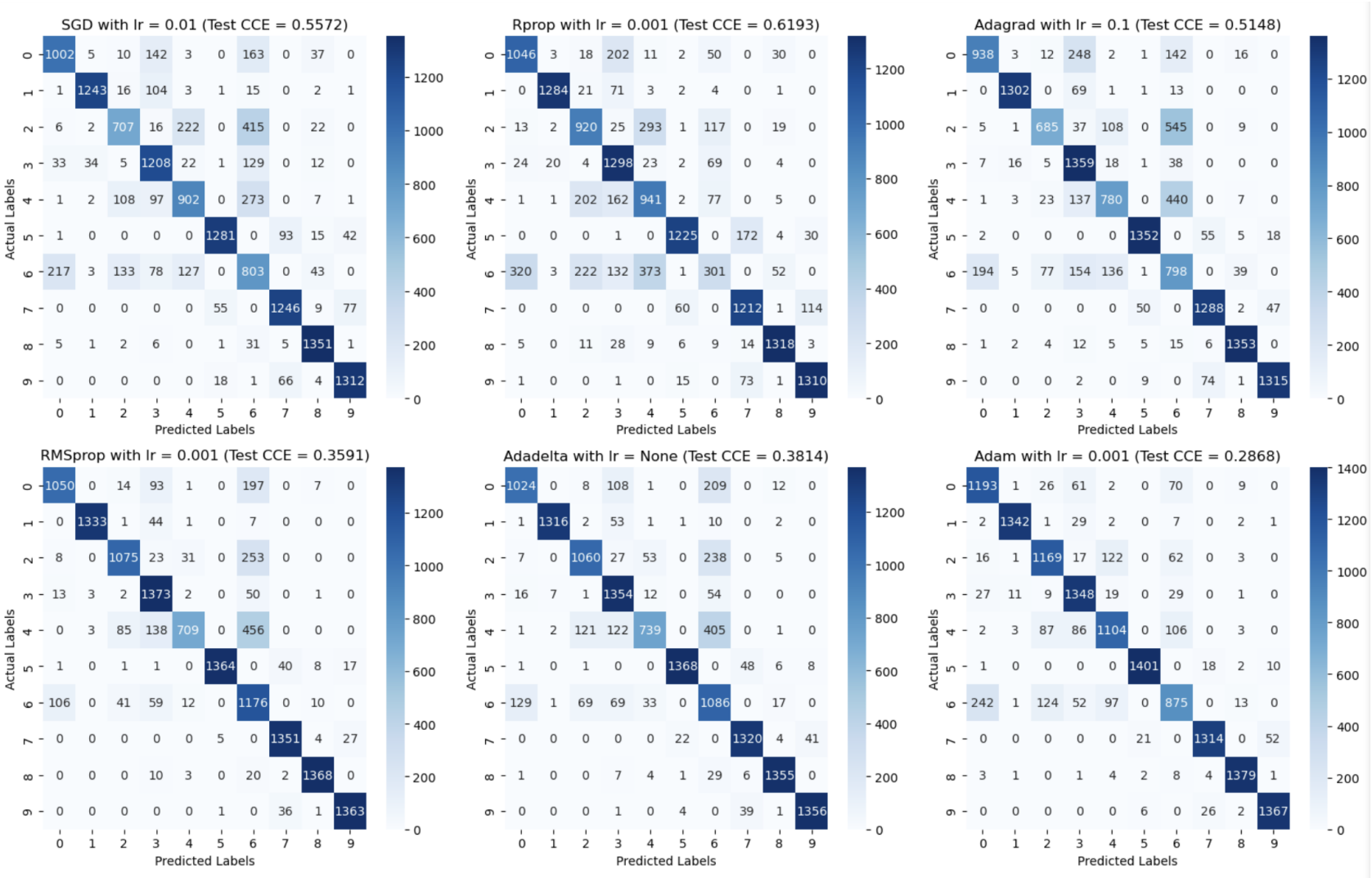
Results: Optimization With Batch Size = 1



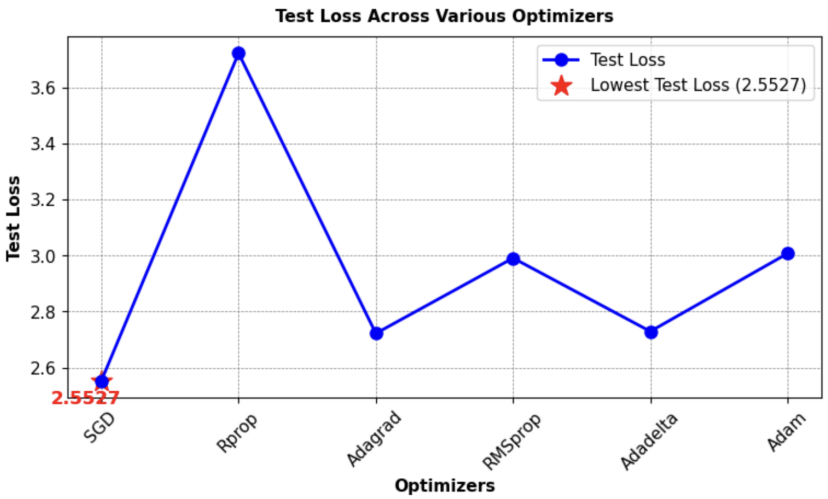
Results: Optimization With Batch Size = 8

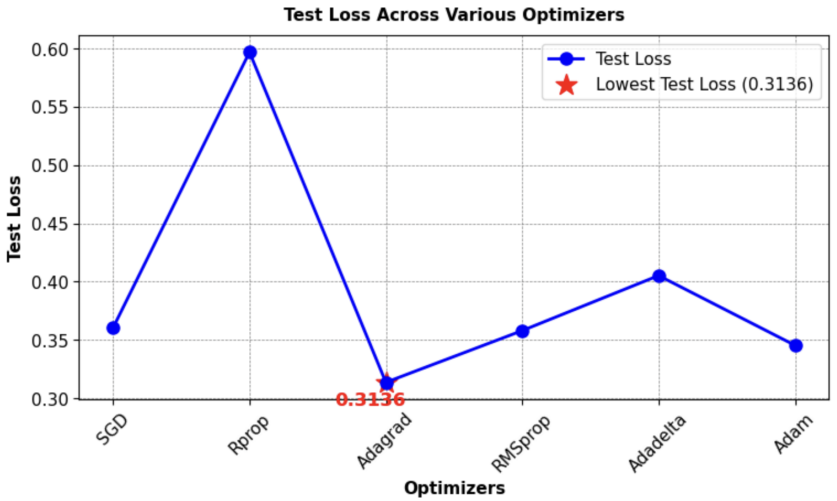


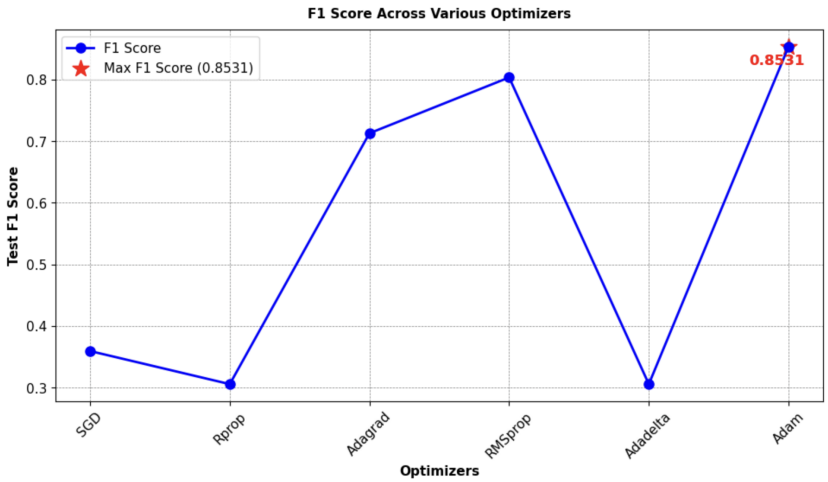
Results: Optimizer Performance

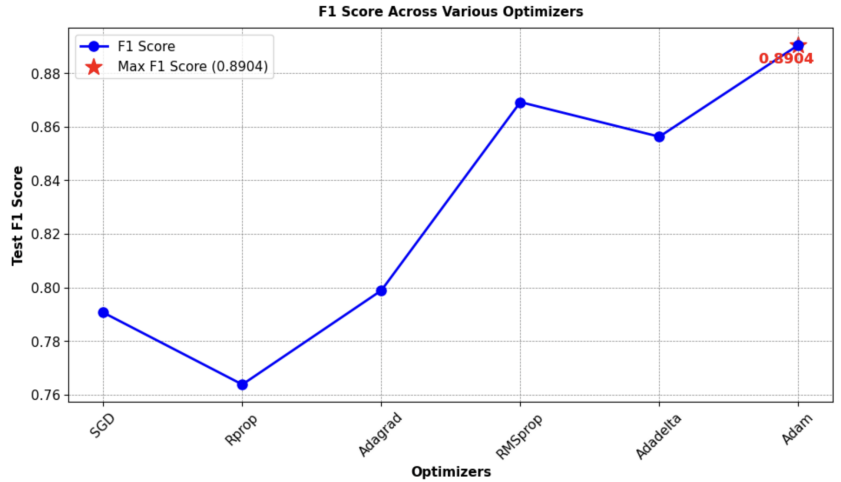


Results: Comparison of Optimizer Performance









Discussion and Conclusion

- **Momentum vs Adaptive Methods:** Momentum increases traversal speed and avoids local minima, while adaptive methods adjust learning rates for each variable, providing better updates for consistently high gradients.
- **Learning Rate Sensitivity:** Adaptive methods are less sensitive to learning rate compared to momentum-based methods, with Adagrad's learning rate decreasing monotonically over time.
- **Optimizer Variations:** Rprop considers only the sign of gradients, Adadelta eliminates the learning rate parameter, and Adam combines momentum with RMSprop and Adadelta benefits.

Discussion and Conclusion

- **Convergence of SGD vs Momentum Coefficient:** As the momentum increases, SGD converges faster, but it may not be optimal if the momentum is too high.
- **Convergence vs Learning Rate:** As the learning rate increases, the algorithms converge to the minima faster, but the updates might be noisy.
- **Convergence vs Batch Size:** Lower batch sizes leads to noisy updates but higher batch sizes might lead to overfitting. Proper batch size need to be chosen for efficient training of the neural network.
- **Robust Optimizer:** The Adam optimizer is the most robust optimizer, as it performs consistently well across all the test problems. It uses both the moving average of the first and second moments of the gradients to update the parameters.

References

Mykel J. Kochenderfer, Tim Allan Wheeler, Algorithms for optimization (2019, MIT Press).

Thank You!