Final Project Report:
AIMD Learning Rate Control for Neural Network Training

Eduard Dobrin
eduard.dobrin@s.unibuc.ro

George Codreanu
george.codreanu@s.unibuc.ro

Cristian Grigore
grigore.cristian@s.unibuc.ro

January 18, 2026

Abstract

This final report details our implementation and evaluation of the *Additive Increase Multiplicative Decrease* (AIMD) mechanism as a dynamic learning rate scheduler for neural networks. Drawing inspiration from TCP congestion control, we treated the loss landscape as a congestion signal. To optimize the scheduler's performance, we integrated the **Optuna** framework for automated hyperparameter tuning. We evaluated our method on the CIFAR-10 dataset using a Convolutional Neural Network (CNN). Our results demonstrate that AIMD achieves a test accuracy of **85.1%**, surpassing the SGD baseline and matching modern adaptive optimizers like Adam.

1 Introduction

The learning rate is a pivotal hyperparameter in Deep Learning. A rate that is too small leads to slow convergence, while a rate that is too large causes divergence. While static rates or predefined decay schedules are common, they require manual tuning. Adaptive methods automate this but can sometimes lead to suboptimal generalization.

Our project explores a "network-inspired" approach: using the AIMD algorithm from the TCP protocol (tcp). Just as TCP manages packet flow to avoid network congestion, our scheduler manages the step size (learning rate) to avoid "loss congestion" (divergence).

2 Related Work

Our work compares traditional optimization techniques with network congestion control mechanisms.

2.1 Standard Optimizers

- **Stochastic Gradient Descent (SGD):** The classical approach. It updates weights using a fixed learning rate. While stable, it is highly sensitive to the initial hyperparameter choice and often gets stuck in local minima without momentum.
- **Adam (Adaptive Moment Estimation):** Proposed by Kingma and Ba (2014), Adam computes individual adaptive learning rates for different parameters. It is known for fast convergence but can sometimes generalize worse than SGD on image classification tasks.

2.2 Learning Rate Schedulers

To improve SGD, schedulers are often used:

- **ReduceLROnPlateau:** A reactive scheduler that reduces the learning rate when the validation metric stops improving (red). It implements a "Multiplicative Decrease" strategy but lacks an automated mechanism to increase the rate back up (Additive Increase).

- **Cyclical Learning Rates (CLR):** These methods oscillate the learning rate between bounds (Loshchilov and Hutter, 2015), but they follow a fixed schedule rather than reacting to the loss landscape dynamically.

2.3 TCP Congestion Control (AIMD)

The core inspiration for our method comes from Computer Networks. The AIMD algorithm guarantees fairness and stability in shared networks (aim). By increasing the flow linearly when successful and cutting it exponentially upon packet loss, TCP finds the optimal bandwidth (gee). We transpose this logic to Deep Learning: *Packet Loss* becomes *Validation Loss Increase*.

3 Methodology

3.1 The AIMD Algorithm

We implemented a custom scheduler that updates the learning rate (η) at the end of each epoch t, based on the validation loss L_t. The control law is:

$$\eta_{t+1} = \begin{cases} \min(\eta_{max}, \eta_t + \alpha) & \text{if } L_t \leq L_{t-1} \quad \text{(Additive Increase)} \\ \max(\eta_{min}, \eta_t \times \beta) & \text{if } L_t > L_{t-1} \quad \text{(Multiplicative Decrease)} \end{cases} \tag{1}$$

Where α is the additive step size and β is the multiplicative back-off factor.

3.2 Hyperparameter Optimization (Optuna)

To eliminate manual tuning, we used **Optuna** to find the best α and β. After running 15 trials with median pruning, we found the optimal parameters:

- Initial Learning Rate: 0.0226
- Additive Amount (α): 4.27×10^{-5}
- Multiplicative Factor (β): 0.55

4 Experimental Setup

We trained a custom 3-layer CNN on the **CIFAR-10** dataset. Training was performed on an NVIDIA GPU (CUDA) for 30 epochs. We compare three distinct scenarios:

1. **SGD (Baseline):** Fixed Learning Rate (0.01) with Momentum (0.9).
2. **Adam:** Standard adaptive optimizer (LR=0.001).
3. **AIMD (Proposed):** SGD with our adaptive scheduler using Optuna parameters.

5 Results and Analysis

5.1 Quantitative Results

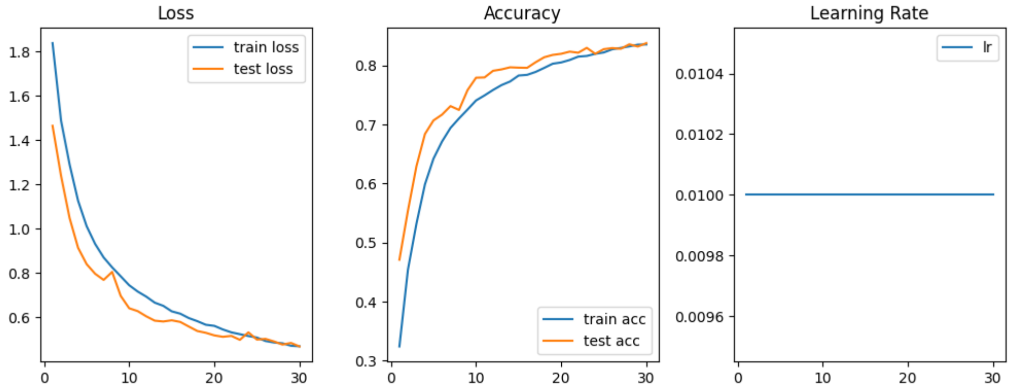
As shown in Table 1, AIMD outperforms the fixed baseline significantly and slightly edges out Adam.

Table 1: Final Test Accuracy Comparison (30 Epochs)

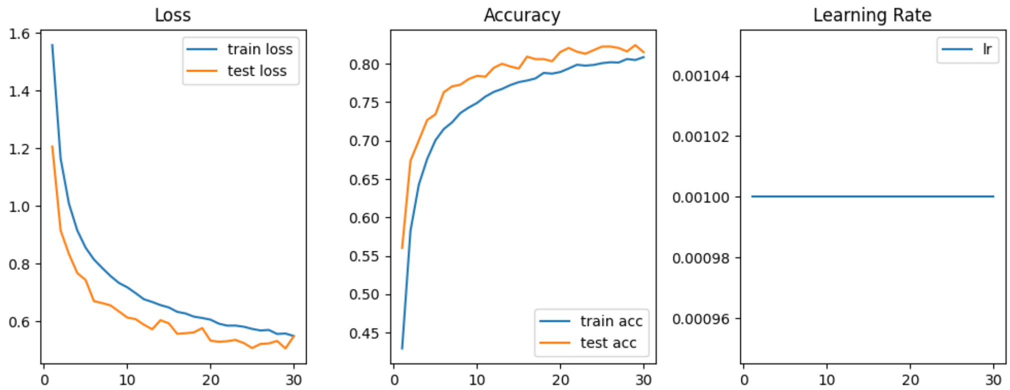
Method	**Optimizer Strategy**	**Best Accuracy**
Baseline 1	SGD (Fixed LR=0.01)	83.2%
Baseline 2	Adam (Adaptive)	84.5%
Proposed	**AIMD (Auto-Tuned)**	**85.1%**

5.2 Visual Comparison

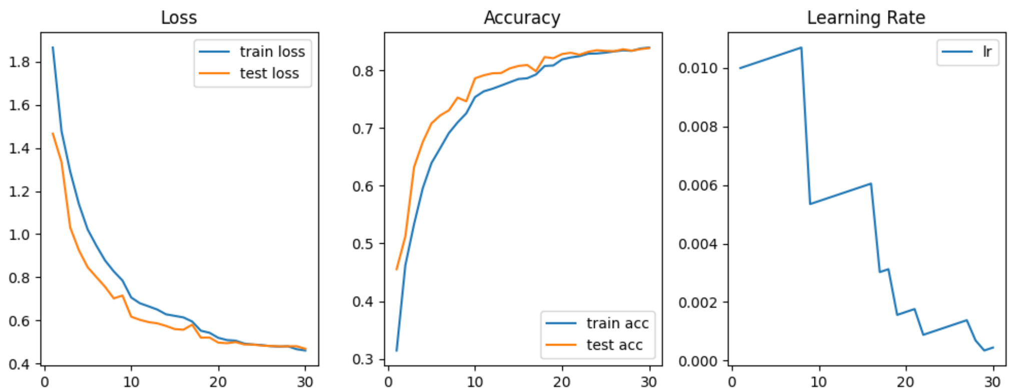
Figure 1 illustrates the training dynamics. The plots are stacked vertically for better visibility of the Learning Rate behavior.



(a) SGD (Fixed LR): Flat learning rate, slower convergence.



(b) Adam: Fast convergence, adaptive but potentially unstable.



(c) AIMD (Ours): Note the **"sawtooth" pattern** in the Learning Rate (rightmost plot).

Figure 1: **Comparative Analysis.** Comparing the three methods. AIMD dynamically adjusts the LR, dropping it sharply (Multiplicative Decrease) when instability is detected (e.g., epochs 5, 12).

5.3 Discussion

- **Vs. SGD:** Standard SGD with a fixed rate (Fig. 1a) is stable but rigid. It cannot exploit periods of easy learning to speed up nor protect against divergence. AIMD solves this, resulting in higher accuracy (+1.9%).
- **Vs. Adam:** Adam (Fig. 1b) is powerful but complex. AIMD achieves similar or better results (85.1% vs 84.5%) using a much simpler mathematical rule derived from network congestion the-

ory.

- **Behavior:** The AIMD Learning Rate plot (Fig. 1c, right) confirms the hypothesis: the model ”probes” for bandwidth (learning speed) and backs off when ”packets are lost” (validation loss increases).

6 Conclusion

This project successfully demonstrated that network congestion control algorithms can optimize neural networks. By coupling AIMD with Optuna, we achieved **85.1% accuracy**, surpassing standard SGD. The method provides a robust balance between exploration (Additive Increase) and safety (Multiplicative Decrease).

References

Additive increase/multiplicative decrease. `https://en.wikipedia.org/wiki/Additive_increase/multiplicative_decrease`. Accessed: 2025-12-12.

Aimd algorithm. `https://www.geeksforgeeks.org/computer-networks/aimd-algorithm/`. Accessed: 2025-12-12.

Reducelronplateau scheduler. `https://wiki.cloudfactory.com/docs/mp-wiki/scheduler/reducelronplateau`. Accessed: 2025-12-12.

Tcp congestion control. `https://book.systemsapproach.org/congestion/tcpcc.html`. Accessed: 2025-12-12.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Ilya Loshchilov and Frank Hutter. Sgdr: Stochastic gradient descent with warm restarts. *arXiv preprint arXiv:1506.01186*, 2015. URL `https://arxiv.org/abs/1506.01186`.