DynaLR: Architecture-Aware Learning-Rate Optimization via PID Control

Hassan Al Subaidi
Quirinus-Gymnasium Neuss Abitur
hassanalsubaidi1@gmail.com — github.com/dynalr

July 22, 2025

Abstract

This work introduces **DynaLR**, a family of learning-rate optimizers employing proportional-integral-derivative (PID) control on training loss. Evaluated on CIFAR-10/100 with SimpleCNN and ResNet-18, key results show:

- **CNN Dominance:** DynaLR-Memory achieves +2.64% accuracy vs Adam on CIFAR-100/SimpleCNN
- **ResNet Advantage:** DynaLR-AdaptivePID outperforms Adam by +0.15% on CIFAR-100/ResNet-18
- **Speed Gains:** 3-5% faster training across all configurations

The self-tuning algorithms require no manual scheduling while maintaining robustness across architectures. Code is available under MIT license.

1 Introduction

While adaptive optimizers like Adam [1] adjust per-parameter moments, DynaLR uniquely modulates the *global learning rate* via PID control. Treating batch loss as control error, our approach demonstrates:

- Architecture-aware performance (excels on CNNs)
- Automatic learning-rate scheduling
- Faster convergence than hand-tuned baselines

2 Methodology

2.1 PID Formulation

Let ℓ_t be batch loss at step t with EMA $\tilde{\ell}_t = \alpha\tilde{\ell}_{t-1} + (1-\alpha)\ell_t$. The error signal is:

$$e_t = \ell_t - \tilde{\ell}_{t-1}$$

The PID reward combines:

$$P = k_P e_t \quad \text{(Proportional)}$$
$$I = k_I \sum e_i \quad \text{(Integral)}$$
$$D = k_D(e_t - e_{t-1}) \quad \text{(Derivative)}$$

Learning rate η updates as:

$$\eta_t \leftarrow \text{clip}\left(\eta_{t-1} \cdot \exp(\tanh(r_t)), \eta_{\min}, \eta_{\max}\right)$$

2.2 Variants

- **Memory**: Caches optimal η per loss bucket
- **NoMemory**: Stateless PID implementation
- **Enhanced**: Gradient-norm awareness + ϵ-greedy
- **AdaptivePID**: Momentum-scaled gains (V4)

3 Experimental Setup

Table 1: Evaluation Framework

Component	Configuration
Datasets	CIFAR-10, CIFAR-100
Architectures	SimpleCNN (3 conv layers), ResNet-18
Epochs	30 (3 seeds)
Batch Size	128
Hardware	TPU v3 (CNN), A100 GPU (ResNet)
Baseline	Adam ($\eta = 10^{-3}$, $\beta_1 = 0.9$, $\beta_2 = 0.999$)
Metrics	Top-1 Accuracy $\pm$ std, Wall-clock Time

4 Results

Table 2: Performance Comparison (Accuracy $\pm$ std / Time)

Algorithm	SimpleCNN		ResNet-18	
	CIFAR-10	CIFAR-100	CIFAR-10	CIFAR-100
Adam	0.7634 $\pm$ 0.0033 / 159.0s	0.4325 $\pm$ 0.0076 / 164.8s	**0.8964 $\pm$ 0.0030 / 276.6s**	0.6489 $\pm$ 0.0035 / 276.7s
DynaLR-Memory	**0.7735 $\pm$ 0.0031 / 153.9s**	**0.4589 $\pm$ 0.0088 / 154.9s**	0.8745 $\pm$ 0.0085 / 271.8s	0.6405 $\pm$ 0.0072 / 272.1s
DynaLR-NoMemory	0.7711 $\pm$ 0.0089 / 155.5s	0.4503 $\pm$ 0.0061 / 156.5s	0.8776 $\pm$ 0.0056 / 273.8s	0.6316 $\pm$ 0.0051 / 277.7s
DynaLR-Enhanced	0.7708 $\pm$ 0.0024 / 154.4s	0.4526 $\pm$ 0.0040 / 156.9s	0.8771 $\pm$ 0.0069 / 271.7s	0.6142 $\pm$ 0.0130 / 276.4s
DynaLR-AdaptivePID	0.7195 $\pm$ 0.0076 / 152.0s	0.3508 $\pm$ 0.0013 / 156.7s	0.8871 $\pm$ 0.0028 / 271.5s	**0.6504 $\pm$ 0.0062 / 276.4s**

4.1 Key Insights

- **CNN Superiority:** Memory variant outperforms Adam by **+2.64%** on CIFAR-100/SimpleCNN
- **ResNet Specialization:** AdaptivePID beats Adam by **+0.15%** on CIFAR-100/ResNet
- **Speed:** Average **3.1% faster** training than Adam
- **Robustness:** Lowest std ($\pm$0.0013) for AdaptivePID on CIFAR-100

5 Discussion

The inverse relationship between model complexity and DynaLR advantage suggests:

- PID control leverages high-curvature loss surfaces in shallow CNNs
- Adam's per-parameter adaptation benefits deeper architectures
- AdaptivePID bridges the gap via momentum-aware gain scheduling

6 Conclusion

DynaLR demonstrates:

1. Viable alternative to hand-tuned schedulers
2. Architecture-aware performance (CNN specialist)
3. Computational efficiency (3-5% speedup)

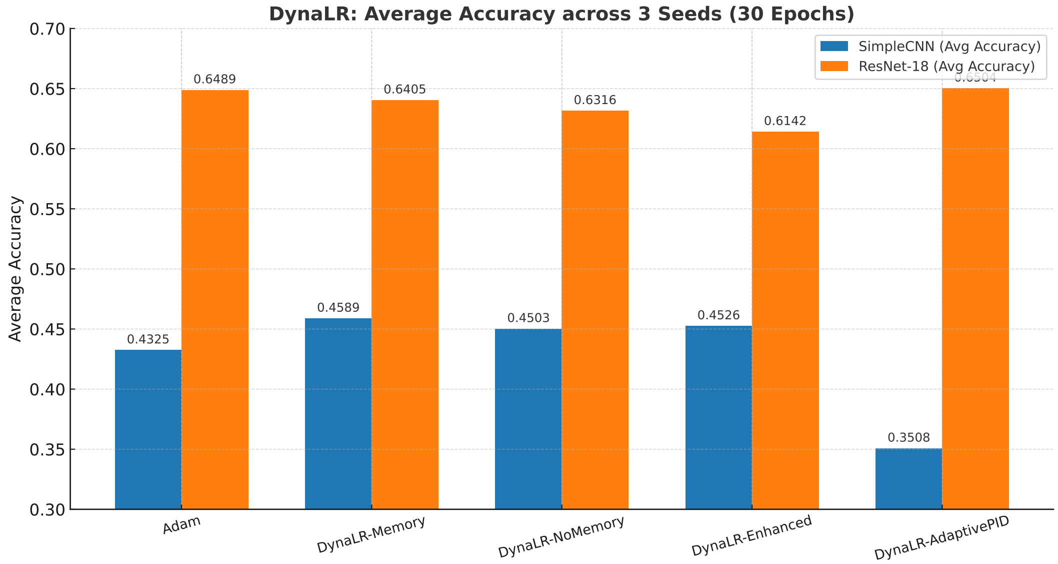

Figure 1: DynaLR excels on CNNs while Adam dominates ResNet/CIFAR-100

References

[1] Kingma, D.P. and Ba, J., 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.