Optimizers in deep learning

CPE 727 - Deep Learning

Ana Clara Loureiro Cruz **Bruno** **Emre** **Felipe**
(anaclaralcruz@poli.ufrj.br
anaclaralcruz@poli.ufrj.br
anaclaralcruz@poli.ufrj.br

anaclaralcruz@poli.ufrj.br)

October 15, 2025

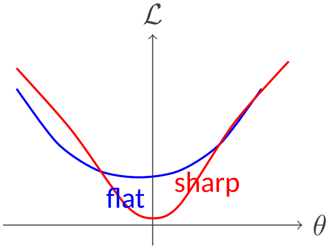
Table of Contents

Why Optimizers Matter

- **Hard landscapes:** non-convex, ill-conditioned; plateaus/saddles/sharp minima.
- **Speed vs. stability:** momentum, adaptivity, curvature cues.
- **Generalization:** optimizer choice influences minima flatness and test accuracy.
- **Scaling:** large batches + mixed precision $\Rightarrow$ LARS/LAMB trust ratios.
- **Anisotropy:** coordinate-wise steps (Adagrad/RMSProp/AdamW).
- **Decay:** AdamW's decoupled weight decay matters.
- **Schedules:** warmup + cosine/one-cycle are often the real win.

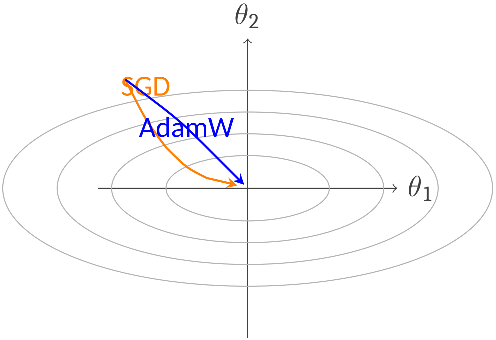
Intuition in Two Pictures

Sharp vs. Flat Minima



SAM/SGD tend to favor flatter minima $\Rightarrow$ better test performance.

Paths on an Anisotropic Bowl



AdamW adapts steps per-coordinate; momentum smooths zig-zagging.

Common notation

Objective: $\min_\theta f(\theta)$
Stochastic gradient at step t: $g_t = \nabla_\theta f_t(\theta_t)$
Base LR: $\eta_t; small \varepsilon > 0 for numerical stability.$

Table of Contents

2 Plain First-Order

SGD (baseline) [1]

$$\theta_{t+1} = \theta_t - \eta_t \, g_t$$

- Sets the stage for momentum, adaptivity, and curvature.
- Sensitive to scale/conditioning; zig-zags in anisotropic valleys.

Example: *SGD (vanilla)*

Table of Contents

Polyak & Nesterov

Polyak Momentum (EMA of gradients):

$$v_t = \beta\, v_{t-1} + (1-\beta)\, g_t, \qquad \theta_{t+1} = \theta_t - \eta_t\, v_t$$

Nesterov Momentum (look-ahead):

$$\tilde{\theta}_t = \theta_t - \eta_t \beta v_{t-1}, \quad v_t = \beta v_{t-1} + g(\tilde{\theta}_t), \quad \theta_{t+1} = \theta_t - \eta_t v_t$$

Example: *SGD + Momentum, Nesterov*

Table of Contents

Per-Coordinate Step Sizes

Adagrad (cumulative second moment):

$$s_t = s_{t-1} + g_t \odot g_t, \qquad \theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{s_t} + \varepsilon} \odot g_t$$

RMSProp (exponential second moment):

$$s_t = \rho s_{t-1} + (1 - \rho)\, g_t \odot g_t, \qquad \theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{s_t} + \varepsilon} \odot g_t$$

Adam/AdamW and Friends

Adam (with bias correction):

$$m_t = \beta_1 m_{t-1} + (1-\beta_1) g_t, \quad v_t = \beta_2 v_{t-1} + (1-\beta_2) g_t \odot g_t$$

$$\hat{m}_t = \frac{m_t}{1-\beta_1^t}, \quad \hat{v}_t = \frac{v_t}{1-\beta_2^t}, \qquad \theta_{t+1} = \theta_t - \eta \frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \varepsilon}$$

AdamW (decoupled weight decay):

$$\theta_{t+1} = (1-\eta\lambda)\,\theta_t \; - \; \eta \frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \varepsilon}$$

Examples: *Adagrad*, *RMSProp*, *Adam*, *AMSGrad*, *AdamW*, *RAdam*, *Adan*, *Lion*

Table of Contents

5 Large-Batch / Layer-Wise Scaling

LARS (Layer-wise Adaptive Rate Scaling)

For layer ℓ with weights $\theta^{(\ell)}$ and gradient $g^{(\ell)}$:

$$r^{(\ell)} = \frac{\|\theta^{(\ell)}\|_2}{\|g^{(\ell)}\|_2 + \varepsilon}, \qquad \Delta\theta^{(\ell)} = -\,\eta\,\phi\,r^{(\ell)}\,g^{(\ell)}$$

$$\theta^{(\ell)}_{t+1} = \theta^{(\ell)}_t + \Delta\theta^{(\ell)} \quad \text{(often with momentum)}$$

LAMB (Layer-wise Adaptive Moments)

Combine Adam-like direction with a LARS-like trust ratio:

$$\Delta^{(\ell)} = \frac{\hat{\boldsymbol{m}}^{(\ell)}}{\sqrt{\hat{\boldsymbol{v}}^{(\ell)}} + \varepsilon} \; (+\, \lambda\, \theta^{(\ell)} \text{ if coupled})$$

$$\boldsymbol{r}^{(\ell)} = \frac{\|\theta^{(\ell)}\|_2}{\|\Delta^{(\ell)}\|_2 + \varepsilon}, \qquad \theta_{t+1}^{(\ell)} = \theta_t^{(\ell)} - \eta\, \boldsymbol{r}^{(\ell)}\, \Delta^{(\ell)}$$

Table of Contents

Sharpness-Aware Minimization (SAM)

Minimax view: $\min_{\theta} \max_{\|\epsilon\| \le \rho} f(\theta + \epsilon)$

$$\epsilon_t = \rho \frac{g_t}{\|g_t\|_2} \quad \Rightarrow \quad g_t^{\text{SAM}} = \nabla f(\theta_t + \epsilon_t), \quad \theta_{t+1} = \theta_t - \eta\, \text{BaseOpt}(g_t^{\text{SAM}})$$

Lookahead (Optimizer Wrapper)

Maintain a slow weight copy ϕ and fast inner updates θ:

$$\theta \leftarrow \text{BaseOptSteps}(\theta), \qquad \phi \leftarrow \phi + \alpha(\theta - \phi), \qquad \theta \leftarrow \phi$$

- Stabilizes training; often improves robustness with negligible overhead.

Example: *SAM*, *Lookahead*

Table of Contents

L-BFGS and Friends

L-BFGS (quasi-Newton): uses limited-memory Hessian inverse approximation H_t:

$$p_t = -H_t\, g_t, \qquad \theta_{t+1} = \theta_t + \eta_t\, p_t$$

AdaHessian (diagonal Hessian):

$$h_t \approx \mathrm{diag}\big(\nabla^2 f(\theta_t)\big), \qquad \theta_{t+1} = \theta_t - \eta \frac{m_t}{\sqrt{h_t} + \varepsilon}$$

Examples: *L-BFGS*, *K-FAC*, *Shampoo*, *AdaHessian*, *Sophia*

Table of Contents

8 Summary

Optimizer Summary (I): Baselines, Adaptive, Large-batch

Method	Strengths / Behavior	Best Use Cases	Pitfalls & Typical HPs
SGD + Nesterov	Low memory; strong generalization; stable with cosine schedule; Nesterov gives look-ahead acceleration	CNNs/vision, medium batches; when you can tune LR/schedule	Can be slow on ill-cond problems; $\eta \in [0.1$ (scale w/ batch), mom 0.9, cosine + warmup
Adagrad	Per-coordinate steps; great on sparse features; no LR tuning once set	Sparse NLP/recsys embeddings; convex-ish problems	Learning rate "dies" (a lator grows); $\eta \in [0.05$ $\varepsilon \sim 10^{-10}$
RMSProp	Controls step via EMA of squared grads; steadier than Adagrad	RNN-ish/online settings; when gradient scales drift	Sensitive to ρ; default $[10^{-3}, 10^{-4}]$, $\rho = 0.$ 10^{-8}
AdamW	Fast convergence; bias correction; *decoupled* weight decay improves generalization vs L2	Transformers/ViTs/LLMs; mixed precision; general default	May overfit vs on some vision $\eta \in [3 \times 10^{-4}, 10^{-4}]$ $(0.9, 0.999)$, wd $= 0.0$
AMSGrad / RAdam	AMSGrad: non-increasing second moment; RAdam: rectifies early variance	When Adam is unstable early or drifts	Slightly slower than A sometimes; use Adan HPs
Lion	Momentum on *sign*; memory-light; competitive on vision/NLP	Resource-constrained training; quick baselines	Tuning can differ from A η often higher; $\beta \approx (0.9$

Optimizer Summary (II): Generalization, Curvature, Schedules

Method	Strengths / Behavior	Best Use Cases	Pitfalls & Typical HPs
SAM (wrapper)	Minimax: avoids sharp minima; often boosts test accuracy	Vision/ViTs where generalization matters; pair with SGD/AdamW	Extra forward/backwar dius $\rho \in [0.05, 0.2]$ weight decay decouplec
Lookahead (wrapper)	Slow–fast weights; stabilizes training; cheap	Add on top of AdamW/SGD for robustness	Sync period/alpha add small but consistent gai
L-BFGS (quasi-Newton)	Fast on smooth/convex-ish, small problems; strong steps	Small models, fine-tuning last layers; classic ML	Not mini-batch-friendl search overhead; mem history
K-FAC	Kronecker-factored curvature; fewer steps to good loss	Deep nets when you can afford extra compute; large-scale setups	Complex to ment/distribute; extra ory/compute
Shampoo	Factored preconditioning per tensor; strong practical results	Large models at scale (when infra supports it)	Higher memory; tunin conditioner update peri
AdaHessian	Diagonal Hessian via stochastic probes; 2nd-order flavor w/ modest cost	When AdamW plateaus and curvature helps	Noisy Hessian diag; tune count; η similar to Adan
Sophia	Light-weight curvature proxy; good for	Large language models; budget-aware	Proxy quality/task-depe

Table of Contents

Referências Bibliográficas

[1] S. Ruder, “An overview of gradient descent optimization algorithms,” 2017.

Optimizers in deep learning

Obrigado pela Atenção!
Alguma Pergunta?