A Brief Summary of Optimization in Deep Learning in New Era

Yifeng Liu
University of California, Los Angeles
liuyifeng@g.ucla.edu

Abstract

Newton's method provides one of the earliest insight in optimization theories. Based on gradient descent, a lot of optimization theories including momentum, adaptive learning, sign of gradient, second-order optimization, variance reduction and scheduler-free optimization have been proposed. However, there is a lack of comprehensive and clear summary of these approaches with unified notation system. This paper attempts to give a systematic, explicit and concise formulation of over 100 optimization methods in deep learning with citation, which is potential for promoting innovation in optimization theory in deep learning, while facilitating relevant researchers to search for references. And the related materials can be found in `https://github.com/lauyikfung/A-Summary-Sheet-of-Optimization-in-Deep-Learning`.

1 Introduction

Optimization lies at the heart of deep learning, facilitating the training process of deep neural networks nowadays, including large language models (Team et al., 2025; Liu et al., 2024a; Grattafiori et al., 2024)(Figure 1), and computer vision models (Ramesh et al., 2022; Liu et al., 2023). From the foundational principles of Newton's method, numerous optimization theories has emerged, significantly enhancing the efficiency and effectiveness of training deep neural networks. These advancements encompass a wide array of techniques, including momentum-based methods, adaptive learning rate strategies, approaches leveraging the sign of gradients, second-order optimization techniques, variance reduction schemes, and even scheduler-free optimization paradigms. Despite the proliferation of these innovative methods, a comprehensive and clearly structured summary, particularly one employing a unified notation system, has been notably absent. This gap often poses a challenge for researchers seeking to navigate the vast and rapidly evolving field of deep learning optimization, hindering both the promotion of new innovations and the efficient discovery of relevant references.

This paper aims to bridge this gap by presenting a systematic, explicit, and unified formulation of more than 100 optimization methods for deep learning. Each method is documented with appropriate citations, aiming to provide a singular, accessible resource for the research community. By offering a unified framework and clear descriptions, this work seeks to foster further innovation in optimization theory within deep learning and to significantly streamline the process for researchers to identify and utilize relevant methodologies.

2 From Gradient Descent to Adaptive Learning

2.1 Notations

In the paper, $f(\theta_t)$ denotes the deep learning network at t-th iteration with parameter θ_t, which is usually regarded as a vector except for second-order methods. The objective is $J(\theta_t)$ and the

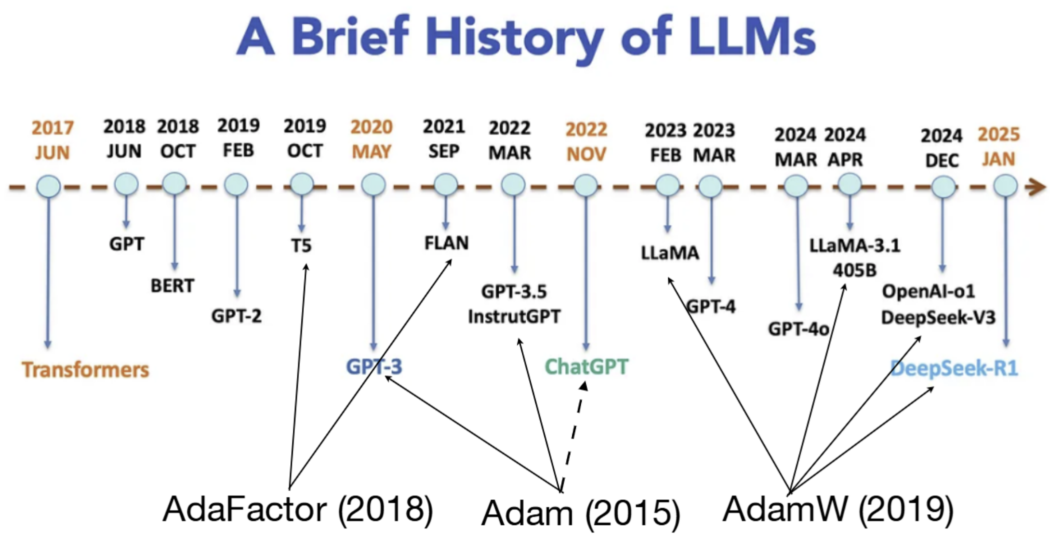

Figure 1: A brief history of large language models. Some optimizers are used in training well-known large language models. The base figure is from `https://medium.com/@lmpo/a-brief-history-of-lmms-from-transformers-2017-to-deepseek-r1-2025-dae75dd3f59a`

gradient is $\nabla_\theta J(\theta_t) = \nabla f(\theta_t, \boldsymbol{\xi}_t) = g_t$, where the input is $\boldsymbol{\xi}_t$ by default. Moreover, $\|\mathbf{x}\| = \|\mathbf{x}\|_2$ denotes the 2-norm, while $|\mathbf{x}|$ denotes taking the absolute value element-wisely. η is the learning rate or step size, and it may change over iterations unless specific explanation. We just omit the initialization of parameters and state variables for simplicity.

2.2 Newton's Method

For Newton's method, $f(\theta + \epsilon) = f(\theta) + \epsilon^\top \nabla f(\theta) + \frac{1}{2}\epsilon^\top \mathbf{H}\epsilon + \mathcal{O}(\|\epsilon\|^3)$, where $\mathbf{H} = \nabla^2 f(\theta)$ is the Hessian matrix (Zhang et al., 2021).

For first-order method, the second-order and higher-order terms $(\frac{1}{2}\epsilon^\top \mathbf{H}\epsilon + \mathcal{O}(\|\epsilon\|^3))$ are ignored since they have relatively lower influence on the convergence of training and much harder to compute, and the update rule comes to:

$$\theta_{t+1} = \theta_t - \eta \nabla f(\theta_t),$$

where $\eta = -\epsilon$ is the step size.

While in second-order methods, only the higher-order terms $(\mathcal{O}(\|\epsilon\|^3))$ are ignored. Since at the minimum of f, $\nabla_\theta f(\theta) = 0$, then $\epsilon = -\mathbf{H}^{-1}\nabla f(\theta)$, and it comes to:

$$\theta_{t+1} = \theta_t - \eta \mathbf{H}^{-1} \nabla f(\theta_t),$$

where η is the step size.

2.3 Gradient Descent (GD)

The gradient descent method is the first-order approximation of Newton method. **Full GD**:

- Using full datasets for gradient descent, 1. $\theta_{t+1} = \theta_t - \eta \nabla_\theta J(\theta_t) = \theta_t - \eta g_t$. Here and below, g_t is defined the gradient of full data/mini batch data on θ_t.
- Convergence $O(\frac{1}{T})$. Here the convergence is defined by $f(x^T) - f^*$

Stochastic GD (SGD): Using one sample per step, convergence $O(\frac{1}{\sqrt{T}})$

Batch GD (BGD): Using small batch (size=b) per steps, convergence $O(\frac{1}{\sqrt{bT}} + \frac{1}{T})$

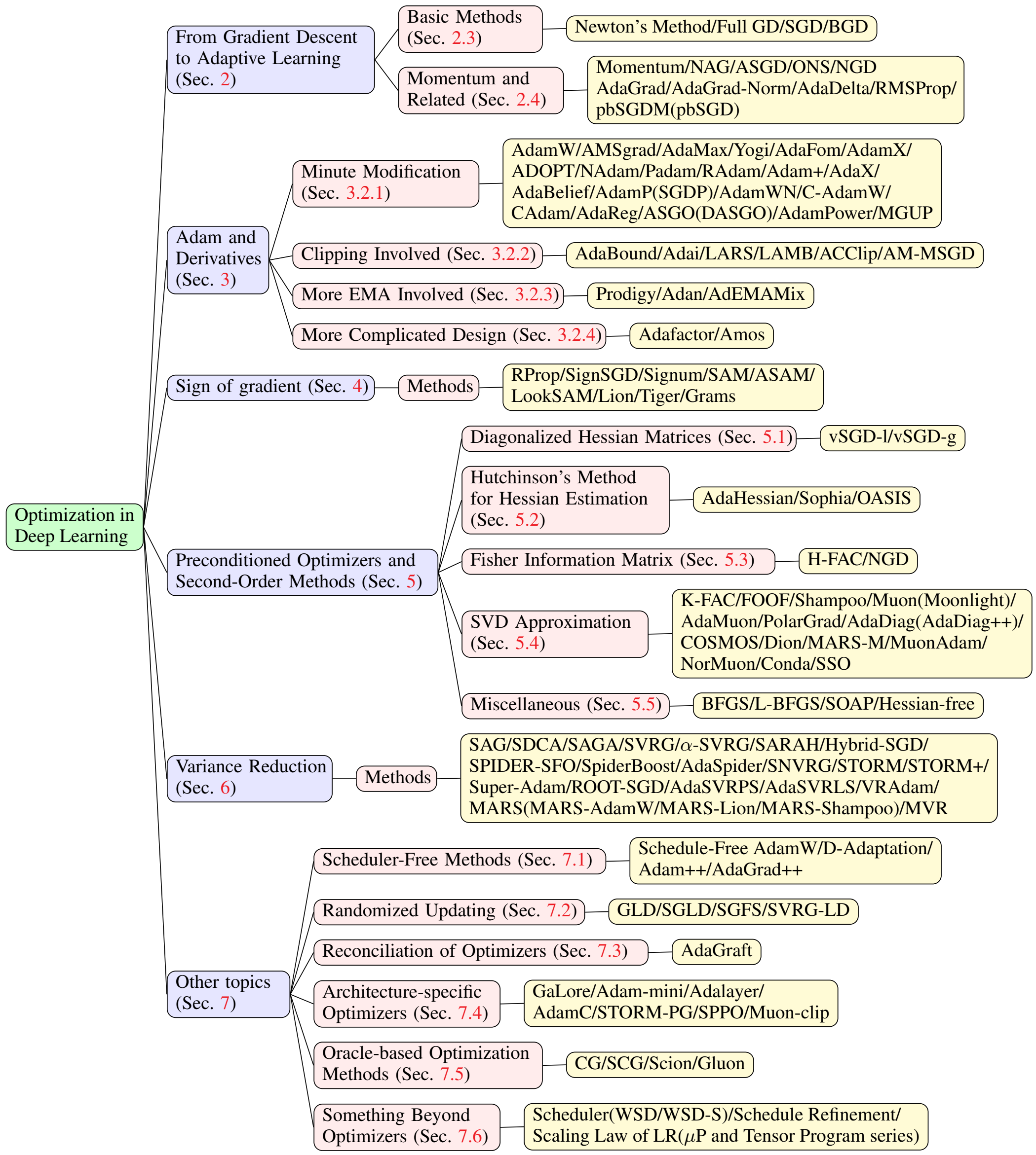

Figure 2: Summary of all the optimizers covered in this paper.

2.4 Momentum and Related

Momemtum (Grum, 2023) $m_{t+1} = \gamma m_t + \eta \nabla_\theta J(\theta_t)$, $\theta_{t+1} = \theta_t - m_{t+1}$

Nestorov's Accelerated Gradient (NAG) (Nesterov, 1983) $m_{t+1} = \gamma m_t + \eta \nabla_\theta J(\theta_t - \gamma m_t)$, $\theta_{t+1} = \theta_t - m_{t+1}$

ASGD (Polyak & Juditsky, 1992) $\theta_{t+1} = \theta_t - \eta \cdot \frac{1}{t+1} \sum_{i=1}^{t} g_i$

Online Newton Step (ONS) (Hazan et al., 2007) $r_t = r_{t-1} + g_t^2$, $\theta_{t+1} = \theta_t - \frac{\eta}{r_t + \epsilon} g_t$

Normalized Gradient Descent (NGD) (Hazan et al., 2015) $\theta_{t+1} = \theta_t - \eta \frac{g_t}{||g_t||}$.

AdaGrad (Duchi et al., 2011) $r_t = r_{t-1} + g_t^2$, $\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{r_t + \epsilon}} g_t$

AdaGrad-Norm (Ward et al., 2020) $r_t = r_{t-1} + \|g_t\|^2$, $\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{r_t + \epsilon}} g_t$

AdaDelta (Zeiler, 2012) $v_t = \rho v_{t-1} + (1-\rho)g_t^2, \theta_t = \theta_{t-1} - \eta\frac{\sqrt{u_{t-1}}}{\sqrt{v_t+\epsilon}}g_t, u_t = \rho u_{t-1} + (1-\rho)\Delta\theta_t^2$, where $\Delta\theta_t = \theta_t - \theta_{t-1}$

RMSProp (Tieleman & Hinton, 2012) $v_t = \rho v_{t-1} + (1-\rho)g_t^2$ (EMA (Exponential Moving Average) of the squared gradient), $\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{v_t+\epsilon}}g_t$

pbSGDM (Zhou et al., 2020a) $v_{t+1} = \beta v_t - \eta \text{sign}(g_t)|g_t|^\gamma, \theta_{t+1} = \theta_t + v_{t+1}$, when $\beta = 0$, it reduces to **pbSGD**, and when $\gamma = 0$, it reduces to SGD.

3 Adam and Derivatives

Adam (Kingma & Ba, 2015)

- $m_t = \beta_1 m_{t-1} + (1-\beta_1)g_t$ (EMA of gradient, use m_t below if have same expression)
- $v_t = \beta_2 v_{t-1} + (1-\beta_2)g_t^2$ (EMA of squared gradient, use v_t below if have same expression)
- $\theta_{t+1} = \theta_t - \eta\frac{\widehat{m}_t}{\sqrt{\widehat{v}_t+\epsilon}}$, where $\widehat{m}_t = \frac{m_t}{1-\beta_1^t}$, $\widehat{v}_t = \frac{v_t}{1-\beta_2^t}$ (Use $\widehat{m_t}$ or $\widehat{v}_t$ below if have same expression)

3.1 Other forms for Adam

3.1.1 Hessian Matrices

$\theta_{t+1} = \arg\min_\theta\{\eta\langle m_t, \theta\rangle + \frac{1}{2}||\theta - \theta_t||^2_{\mathbf{H}_t}\}$. In closed form: $\theta_{t+1} = \theta_t - \eta\mathbf{H}_t^{-1}m_t$. (In some works, $P_t = \mathbf{H}_t^{-1} \in \mathbb{S}_{++}$ is the *preconditioning matrix* or *preconditioner* (Lau et al., 2025).)

- $\mathbf{H}_t = \sqrt{\text{diag}(v_t)} \cdot \frac{1-\beta_1^t}{\sqrt{1-\beta_2^t}}$ for AdamW;
- $\mathbf{H}_t = \sqrt{\text{diag}(m_t^2)}$ for Lion;
- $\mathbf{H}_t = (\sum_{\tau=1}^t \mathbf{G}_t\mathbf{G}_t^\top)^{1/4} \otimes (\sum_{\tau=1}^t \mathbf{G}_t^\top\mathbf{G}_t)^{1/4}$ for Shampoo.

3.1.2 Matrix form of v_t

$\theta_{t+1} = \prod_{\mathcal{F},\sqrt{V_t}}(\theta_t - \eta_t m_t/\sqrt{\widehat{v}_t})$, where $\theta \in \mathcal{F}, V_t = \text{diag}(\widehat{v}_t)$

3.2 Derivatives of Adam

In the following algorithms, if not specified, m_t, v_t, $\widehat{m}_t$, $\widehat{v}_t$ and the updating rule are the same with Adam, or AdamW with decoupled weight decay.

3.2.1 Minute Modification

AdamW (Loshchilov & Hutter, 2019) $\theta_{t+1} = \theta_t - \eta(\frac{\widehat{m}_t}{\sqrt{\widehat{v}_t+\epsilon}} + \lambda\theta_t)$, where λ is called decoupled weight decay (hyper-parameter). And The decoupled weight decay can be applied to the algorithms below.

AMSgrad (Reddi et al., 2018) $\widetilde{v}_t = \max(\widetilde{v}_{t-1}, v_t)$, $\widehat{v}_t = \frac{\widetilde{v}_t}{1-\beta_2^t}$.

AdaMax (Loshchilov & Hutter, 2019) $u_t = \max(\beta_2 u_{t-1}, |g_t|)$, $\theta_{t+1} = \theta_t - \eta\frac{\widehat{m}_t}{u_t}$.

Yogi (Zaheer et al., 2018) $v_t = v_{t-1} - (1-\beta_2)\text{sign}(v_{t-1} - g_t^2)g_t^2$.

AdaFom (Chen et al., 2018) $v_t = (1-1/t)v_{t-1} + (1/t)g_t^2$, $\theta_{t+1} = \theta_t - \eta\frac{m_t}{\sqrt{v_t}}$.

AdamX (Tran et al., 2019) $\widehat{m}_t = m_t$, $\widehat{v}_t = \max\{\frac{(1-\beta_{1,t})^2}{(1-\beta_{1,t-1})^2}\widehat{v}_{t-1}, v_t\}$.

ADOPT (Taniguchi et al., 2024) $m_t = \beta_1 m_{t-1} + (1-\beta_1)\frac{g_t}{\max\sqrt{v_t}, \epsilon}$, $\theta_{t+1} = \theta_t - \eta m_t$.

NAdam (Dozat, 2016) $\theta_{t+1} = \theta_t - \eta\frac{1}{\sqrt{\widehat{v}_t+\epsilon}}(\beta_1\widehat{m}_t + \frac{1-\beta_1}{1-\beta_1^t}g_t)$, by taking $\psi = 0$ in formula below

- In PyTorch implementation, $\theta_{t+1} = \theta_t - \eta \frac{1}{\sqrt{\widehat{v}_t}+\epsilon}(\mu_{t+1}\frac{m_t}{1-\Pi_{i=1}^{t+1}\mu_i} + \frac{(1-\mu_t)}{1-\Pi_{i=1}^{t}\mu_i} g_t)$, where $\mu_t = \beta_1(1 - \frac{0.96^{t\psi}}{2})$.

Padam (Chen & Gu, 2018) $\widetilde{v}_t = \max(\widetilde{v}_{t-1}, v_t)$, $\theta_{t+1} = \theta_t - \eta_t \frac{m_t}{\widetilde{v}_t^p}$. And the output is chosen from $\{\theta_t\}$ with $P(\theta_{out} = \theta_t) = \frac{\eta_{t-1}}{\sum_{i=1}^{T-1} \eta_i}$.

RAdam (Liu et al., 2020a) $\rho_\infty = \frac{2}{1-\beta_2} - 1$, $\rho_t = \rho_\infty - \frac{2t\beta_2^t}{1-\beta_2^t}$. If $\rho_t > 4$, $l_t = \sqrt{(1-\beta_2^t)/v_t}$, $r_t = \sqrt{\frac{(\rho_t-4)(\rho_t-2)\rho_\infty}{(\rho_\infty-4)(\rho_\infty-2)\rho_t}}$, $\theta_{t+1} = \theta_t - \eta r_t \widehat{m}_t l_t$; otherwise $\theta_{t+1} = \theta_t - \eta\widehat{m}_t$.

Adam$^+$ (Liu et al., 2020b) $\eta_t = \frac{\alpha\beta^\gamma}{\max(||z_t||^{1/2}, \epsilon_0)}$, $\theta_{t+1} = \theta_t - \eta z_t$, $z_{t+1} = (1-\beta)z_t + \beta\nabla_\theta J((1-\frac{1}{\beta})\theta_t + \frac{1}{\beta}\theta_{t+1})$, where I replace a in the original papar with γ for clarity

AdaX (Li et al., 2020) $v_t = (1+\beta_2)v_{t-1} + \beta_2 g_t^2$, $\widehat{v}_t = \frac{v_t}{(1+\beta_2)^t - 1}$

AdaBelief (Zhuang et al., 2020) $v_t = \beta_2 v_{t-1} + (1-\beta_2)(g_t - m_t)^2 + \epsilon$

AdamP (Heo et al., 2021) $p_t = \frac{m_t}{\sqrt{v_t}+\epsilon}$. If $\cos(\theta_t, g_t) < \delta/\sqrt{\dim(\theta_t)}$, $q_t = p_t - \frac{(\theta_t \cdot p_t)\theta_t}{||\theta_t||_2^2}$, else $q_t = p_t$. $\theta_{t+1} = \theta_t - \eta q_t$.

- **SGDP** Similar to AdamP, but $p_t = \beta p_{t-1} + g_t$.

AdamWN (Loshchilov, 2023) $\widehat{\theta}_t = \theta_t - \eta\frac{\widehat{m}_t}{\sqrt{\widehat{v}_t}+\epsilon}$, $\theta_{t+1} = \widehat{\theta}_t - k_t(1 - \frac{r_t\|\theta_0\|}{\|\widehat{\theta}_t\|})\widehat{\theta}_t$, $k_t \in [0,1]$, $r_t\|\theta_0\|$ is the target weight norm for θ_t.

C-AdamW (Liang et al., 2024) $u_t = \frac{\widehat{m}_t}{\sqrt{\widehat{v}_t}+\epsilon}$, $\phi_t = \mathbf{1}_{u_t \odot g_t \geq 0}$, $\bar{\eta}_t = \eta\frac{d}{\|\phi_t\|_0 + 1}$, $\theta_{t+1} = \theta_t - \bar{\eta}_t(\phi_t \circ u_t + \lambda\theta_t)$.

CAdam (Wang et al., 2024) $\theta_{t+1} = \theta_t - \eta\frac{\widehat{m}_t}{\sqrt{\widehat{v}_t}+\epsilon} \odot \mathbf{1}_{m_t \odot g_t > 0}$.

AdaReg (Gupta et al., 2017) $r_t = r_{t-1} + g_t g_t^\top$, $H_t = \arg\min_H\{\mathrm{Tr}(r_t^\top H) + \Phi(H)\}$, $\theta_{t+1} = \theta_t - H_t g_t$.

- $\Phi(H) = \mathrm{Tr}(H^{-1})$ ($H_t = (\sum_{s=1}^t g_s g_s^\top)^{-1/2}$) for AdaGrad (Duchi et al., 2011);
- $\Phi(H) = -\log|H|$ ($H_t = (\sum_{s=1}^t g_s g_s^\top)^{-1}$) for ONS (Hazan et al., 2007).

ASGO (An et al., 2025) For 2-dimension θ_t, g_t, $r_t = r_{t-1} + g_t g_t^\top$, $\Lambda_t = r_t^{1/2} + \epsilon I_m$, $\theta_{t+1} = \theta_t - \eta\Lambda_t^{-1} g_t$.

- **DASGO** For 2-dimension θ_t, g_t, $v_t = \beta_2 v_{t-1} + (1-\beta_2)\mathrm{diag}(g_t^\top g_t)$, $\theta_{t+1} = \theta_t - \eta\frac{m_t}{\sqrt{v_t}+\epsilon}$.

AdamPower Use powered gradient (Wang et al., 2025b) $\widetilde{g}_t = |g_t|^p \mathrm{sign}(g_t)$ to replace the gradient in AdamW.

MGUP-AdamW (Da Chang134) $\eta_t = \eta\frac{\sqrt{1-\beta_2^t}}{1-\beta_1^t}$, $u_t = \frac{m_t}{\sqrt{v_t}+\epsilon}$, $\phi_t = \mathrm{MGUP}(u_t \odot g_t)$, $\theta_{t+1} = \theta_t - \eta_t(\phi_t \odot u_t + \lambda\theta_t)$, where MGUP outputs $\phi_{t,i} = 1/\tau$ for $i \in \mathcal{I}_{\text{topK}}$ with $\mathcal{I}_{\text{topK}}$ as the index set of the largest K elements of $u_t \cdot g_t$ with $K = \lfloor\tau \cdot d\rfloor$, otherwise $\phi_{t,i} = \tau$.

3.2.2 Clipping Involved

AdaBound (Luo et al., 2019) $\eta_t = \mathrm{Clip}(\alpha/\sqrt{\mathrm{diag}(v_t)}, \eta_l, \eta_u)/\sqrt{t}$, where, η_l and η_u are lower bound and upper bound of learning rates, $\theta_{t+1} = \theta_t - \eta_t m_t$.

Adai (Xie et al., 2022) $\bar{v}_t = \mathrm{mean}(\widehat{v}_t)$, $\beta_{1,t} = \mathrm{Clip}(1 - \frac{\beta_0}{\bar{v}_t}\widehat{v}_t, 0, 1-\epsilon)$, $\widehat{m}_t = \frac{m_t}{1-\Pi_{s=1}^t \beta_{1,t}}$, $\theta_{t+1} = \theta_t - \eta\widehat{m}_t$.

LARS (You et al., 2017) $m_t = \beta_1 m_{t-1} + (1-\beta_1)(g_t + \lambda\theta_t)$, $\theta_{t+1} = \theta_t - \eta\frac{\mathrm{Clip}(||\theta_t||, \gamma_l, \gamma_u)}{||\theta_t|| + \epsilon} m_t$, where $\mathrm{Clip}(x, \gamma_l, \gamma_u) = \min(\max(x, \gamma_l), \gamma_u)$, γ_l is default to 0 if only 1 bound given

LAMB (You et al., 2020) $r_t = \frac{\widehat{m}_t}{\sqrt{\widehat{v}_t}+\epsilon}$, $\theta_{t+1} = \theta_t - \eta\frac{\text{Clip}(||\theta_t||,\gamma_l,\gamma_u)}{||r_t+\lambda\theta_t||+\epsilon}(r_t + \lambda\theta_t)$

ACClip (Zhang et al., 2020) $m_t = \beta_1 m_{t-1} + (1-\beta_1)g_t$, $v_t^\alpha = \beta_2 v_t^\alpha + (1-\beta_2)|g_t|^\alpha$, $\theta_t = \theta_{t-1} - \eta m_t \cdot \min\{\frac{v_t}{|m_t|+\epsilon}, 1\}$.

AM-MSGD $\beta_t = \text{Clip}(\frac{(\lambda+1)g_t^\top d_t - \langle d_t - g_t, g_t + \lambda d_t\rangle}{\|d_t - g_t\|_2^2}, 0, \beta_{\max,t})$, $d_{t+1} = \frac{\beta_t+\lambda}{1+\lambda}d_t + \frac{1-\beta_t}{1+\lambda}g_t$, $\theta_{t+1} = \theta_t - \eta d_{t+1}$.

3.2.3 More EMA Involved

Prodigy (Bernstein & Newhouse, 2024) $m_t = \beta_1 m_{t-1} + (1-\beta_1)\eta_t g_t$, $v_t = \beta_2 v_{t-1} + (1-\beta_2)\eta_t^2 g_t^2$, $r_t = \sqrt{\beta_2} r_{t-1} + (1-\sqrt{\beta_2})\eta_t^2 g_t^\top(\theta_0 - \theta_t)$, $s_t = \sqrt{\beta_2} s_{t-1} + (1-\sqrt{\beta_2})\eta_t^2 g_t$, $\eta_{t+1} = \max(\eta_t, \frac{r_t}{\|s_t\|_1})$, $\theta_{t+1} = \theta_t - \eta_t \frac{m_t}{\sqrt{v_t}}$

Adan (Xie et al., 2024) $v_t = (1-\beta_2)v_{t-1} + \beta_2(g_t - g_{t-1})$, $n_t = (1-\beta_3)n_{t-1} + \beta_3[g_t + (1-\beta_2)(g_t - g_{t-1})]^2$, $\theta_t = \frac{1}{1+\lambda\eta}[\theta - \frac{\eta}{\sqrt{n_t}+\epsilon} \circ (m_t + (1-\beta_2)v_t)]$

AdEMAMix (Pagliardini et al., 2024) $r_t = \beta_3 r_{t-1} + (1-\beta_3)g_t$, $\theta_t = \theta_{t-1} - \eta(\frac{\widehat{m}_t + \alpha_t r_t}{\sqrt{\widehat{v}_t}+\epsilon} + \lambda\theta_{t-1})$

3.2.4 More complicated Design

Adafactor (Shazeer & Stern, 2018) $u_t = \frac{g_t}{\sqrt{\widehat{v}_t}}$, $\widehat{u}_t = \frac{u_t}{\max(1, \text{RMS}(u_t)/d)}$, $\theta_{t+1} = \theta_t - \eta_t\widehat{u}_t$, where $\alpha_t = \eta \cdot \max(\epsilon_2, \text{RMS}(\theta_t))$, $\beta_{2,t} = 1 - t^\tau$, and

- For weight vector $\theta_t \in \mathbb{R}^n$, $\widehat{v}_t = \beta_{2,t}\widehat{v}_{t-1} + (1-\beta_{2,t})(g_t^2 + \epsilon_1 1_n)$;
- For weight matrix $\theta_t \in \mathbb{R}^{m\times n}$, $r_t = \beta_{2,t} r_{t-1} + (1-\beta_{2,t})(g_t^2 + \epsilon_1 1_m 1_n^\top)1_n$, $v_t = \beta_{2,t} v_{t-1} + (1-\beta_{2,t})1_m^\top(g_t^2 + \epsilon_1 1_m 1_n^\top)$, $\widehat{v}_t = r_t v_t / 1_m^\top r_t$.

Amos (Tian & Parikh, 2022) $c_t = (1 + \frac{1}{4}\sqrt{\eta}b_t)^{-1/2}$, $d_t = (1 + \frac{1}{4}\sqrt{\eta\widetilde{\eta}}b_t)^{-1}$, with $\widetilde{\eta}$ the expected scale for model weights θ, (Optional) $\widetilde{g}_t = \frac{\chi}{\max(\chi, \|g_t\|)}g_t$, $v_t = \beta v_{t-1} + (1-\beta)\text{M}_2(\widetilde{g}_t)^2$, where $\text{M}_2(a) := \sqrt{\frac{1}{k}\sum_{i=1}^k a_i^2}$ is the quadratic mean of the entries. $\widehat{v}_t = \frac{v_t}{1-\beta^t}$, $\gamma_t = c_t\frac{\eta^2}{\widehat{v}_t}\text{M}_2(g_t)^2$, $\delta_t = d_t(\frac{\eta\widetilde{\eta}}{\sqrt{\widehat{v}_t}}g_t + \frac{\gamma_t}{2}\theta_t)$. $b_{t+1} = b_t + \gamma_t(1 + b_t)$, (Optional) $\widetilde{\delta}_t = m_{t+1} = \mu m_t + (1-\mu)\delta_t$, $\theta_{t+1} = \theta_t - \widetilde{\delta}_t$

4 Sign of Gradient

RProp (Riedmiller & Braun, 1993) for $0 < \eta^- < 1 < \eta^+$,

- If $g_t \cdot g_{t-1} > 0$, $\eta_t = \min(\eta^+\eta_{t-1}, \eta_{max})$, $\theta_{t+1} = \theta_t - \eta_t\text{Sign}(g_t)$
- If $g_t \cdot g_{t-1} < 0$, $\eta_t = \max(\eta^-\eta_{t-1}, \eta_{min})$, $\theta_{t+1} = \theta_t - \Delta\theta_t$, where $\Delta\theta_t = \theta_t - \theta_{t-1}$
- Otherwise $\eta_t = 1$, $\theta_{t+1} = \theta_t - \eta_t\text{Sign}(g_t)$

SignSGD (Bernstein et al., 2018) $\theta_{t+1} = \theta_t - \eta\text{Sign}(g_t)$.

Signum (Bernstein et al., 2018) $m_t = \beta m_{t-1} + (1-\beta)g_t$,$\theta_{t+1} = \theta_t - \eta\text{Sign}(m_t)$.

Sharpness-Aware Minimization (SAM) (Foret et al., 2021) $\widehat{\epsilon}_t = \rho\ \text{Sign}(g_t)|g_t|^{q-1}/||g_t||_q^{q/p}$, $\theta_{t+1} = \theta_t - \eta\nabla_\theta J(\theta_t + \widehat{\epsilon}_t)$.

ASAM (Kwon et al., 2021) $\theta_{t+1} = \theta_t - \alpha(\nabla_\theta J(\theta_t + \rho\frac{T_{\theta_t}^2 g_t}{||T_{\theta_t}^2 g_t||_2}) + \lambda\theta_t)^\dagger$, where T_θ is some invertible linear operator such as $T_\theta = \text{diag}(|\theta|)$.

LookSAM (Liu et al., 2022) If $t \mod k = 0$, $\bar{g}_t = \nabla_\theta J(\theta + \rho g_t/||g_t||)$, $\widetilde{g}_t = \bar{g}_t - ||\bar{g}_t||\frac{(g_t\cdot\bar{g}_t)g_t}{||g_t||^2||\bar{g}_t||}$; else $\bar{g}_t = g_t + \alpha\frac{||g_t||}{||\widetilde{g}_{t-1}||}\widetilde{g}_{t-1}$, $\widetilde{g}_t = \widetilde{g}_{t-1}$. $\theta_{t+1} = \theta_t - \eta\bar{g}_t$.

Lion (Chen et al., 2023) $c_t = \beta_1 m_{t-1} + (1-\beta_1)g_t$, $m_t = \beta_2 m_{t-1} + (1-\beta_2)g_t$, $\theta_{t+1} = \theta_t - \eta(\text{Sign}(c_t) + \lambda\theta_t)$

- **Tiger** (Su, 2023) (A special case of Lion when $\beta_1 = \beta_2 = \beta$) $m_t = \beta m_{t-1} + (1-\beta)g_t$, $\theta_{t+1} = \theta_t - \eta(\text{Sign}(m_t) + \lambda\theta_t)$. For bias and normalization parameters, $\eta_i = 0.5\eta$, $\lambda = 0$, otherwise $\eta_i = \eta \times RMS(\theta_{t,i})$, $\lambda =$ Constant> 0

Grams (Cao et al., 2024) (Based on AdamW) $\theta_{t+1} = \theta_t - \eta(\text{Sign}(g_t) \circ |\frac{\widehat{m}_t}{\sqrt{\widehat{v}_t}+\epsilon}| + \lambda\theta_t)$

5 Preconditioned Optimizers and Second-Order Methods

Preconditioned approaches usually use a preconditioner $\mathbf{H}_t$ to adjust the updates by involving the inter-dependencies among model parameters. According to the Newton's Method,

$$\theta_{t+1} = \theta_t - (\nabla^2_{\theta_t} J(\theta))^{-1} \nabla_{\theta_t} J(\theta_t) = \theta_t - \mathbf{H}_t^{-1} g_t.$$

Therefore, a direct motivation for preconditioned approaches is to directly involve or approximate the second-order information in optimization process. However, direct computation needs $\mathcal{O}(n^2)$ time, therefore, adequate approximation or estimation is needed.

5.1 Diagonalized Hessian Matrices

Usually, diagonalized Hessian matrices and updates are expected. Becker (1988) first use diagonal Hessian as the pre-conditioner by ignoring off-diagonal entries:

$$\theta_{t+1} = \theta_t - \eta \frac{g_t}{|\text{diagonal}(\mathbf{H}_t)| + \epsilon},$$

where we use "diagonal" to indicate taking diagonal entries instead of forming diagonal matrix with entries from another vector (denoted as "diag").

vSGD-l (Schaul et al., 2013) $m_t = (1-\tau_t^{-1})m_{t-1} + \tau_t^{-1} g_t$, $v_t = (1-\tau_t^{-1})v_{t-1} + \tau_t^{-1} g_t^2$, $h_t = (1-\tau_t^{-1})h_{t-1} + \tau_t^{-1}|\text{diagonal}(\mathbf{H}_t)|$, $\eta_t = \frac{m_t^2}{h_t v_t}$, $\tau_{t+1} = (1 - \frac{m_t^2}{v_t})\tau_t + 1$, $\theta_{t+1} = \theta_t - \eta_t g_t$.

vSGD-g, same as vSGD-l, but $\eta_t = \frac{\sum_i (m_t)_i^2}{h_t \max_i (h_t)_i}$.

5.2 Hutchinson's Method for Hessian Estimation

To estimate $\mathbf{H}_t$, Hutchinson's method (Hutchinson, 1989) can be used: $z \sim$ Rademacher(0.5), $\frac{\partial g^\top z}{\partial \theta} = \frac{\partial g^\top}{\partial \theta} z + g^\top \frac{\partial z}{\partial \theta} = \frac{\partial g^\top}{\partial \theta} z = \mathbf{H}_t z$, $\mathbf{D} = \text{diag}(\mathbf{H}) = \mathbb{E}[z \odot (\mathbf{H}z)]$.

AdaHessian (Yao et al., 2021) (Based on Adam) $v_t = \beta_2 v_{t-1} + (1-\beta_2)\mathbf{D}_t^2$, $\theta_{t+1} = \theta_t - \eta \frac{\widehat{m}_t}{(\sqrt{\widehat{v}_t})^k + \epsilon}$, where k is the Hessian power.

Sophia (Liu et al., 2024b) $m_t = \beta_1 m_{t-1} + (1-\beta_1)g_t$, if t mod $k = 1$ for the size of the step group k, $h_t = \beta_2 h_{t-k} + (1-\beta_2)\widehat{h}_t$, where $\widehat{h}_t$ is calculated using one of the following Hessian estimators:

- Hutchinson: Draw $u \sim \mathcal{N}(0, I_d)$, $\widehat{h}_t = u \odot \nabla(\langle g_t, u\rangle)$;
- Gauss-Newton-Bartlett: Compute logits on mini-batch $\{l(\theta_t, \boldsymbol{\xi}_{t,b})\}_{b=1}^B$, sample $\widehat{y}_{t,b} \sim$ Softmax$(l(\theta_t, \boldsymbol{\xi}_{t,b})), \forall b \in [B]$. $\widehat{g} = \frac{1}{B}\nabla L(l(\theta_t, \boldsymbol{\xi}_{t,b}), \widehat{y}_{t,b})$ with loss function l, $\widehat{h} = B \cdot \widehat{g} \odot \widehat{g}$,

Otherwise, $h_t = h_{t-1}$. $\theta_{t+1} = \theta_t - \eta(\text{Clip}(\frac{m_t}{\max\{\gamma h_t, \epsilon\}}, 1) - \lambda\theta_t)$.

OASIS (Jahani et al.) $v_t = \beta_2 v_{t-1} + (1-\beta_2)\mathbf{D}_t^2$, $\widehat{v}_t = \max\{v_t, \alpha\}$ for a positive truncation value α; $\eta_t = \min\{\sqrt{1+S_{t-1}}\eta_{t-1}, \frac{||\theta_t - \theta_{t-1}||_{\widehat{v}_t}}{2||g_t - g_{t-1}||^*_{\widehat{v}_t}}\}$, where $S_0 = +\infty$, $\theta_{t+1} = \theta_t - \eta_t \frac{g_t}{\widehat{v}_t}$, $S_t = \frac{\eta_t}{\eta_{t-1}}$.

5.3 Fisher Information Matrix

Another way to estimate Hessian matrix is to compute Fisher information matrix (proposed by Amari (1998) and **H-FAC** (Martens & Grosse, 2015)), since $\widehat{v}_t$ is an approximation to the diagonal of Fisher information matrix (Pascanu & Bengio, 2014):

$$F = E[\frac{d \log p(y|x,\theta)}{d\theta} (\frac{d \log p(y|x,\theta)}{d\theta})^\top] = E[\mathcal{D}\theta\mathcal{D}\theta^\top].$$

Natural Gradient Descent (NGD) (Amari, 1998) $\theta_{t+1} = \theta_t - \frac{1}{\lambda}F^{-1}g_t$.

5.4 SVD Approximation

Sometimes, to approximate Hessian matrix, SVD is utilized. To approximate this process, they developed Newton-Schulz iteration (Higham, 2008) methods for computing $UV^\top$ for $g = U\Sigma V^\top$, by setting $X_0 = g/\|g\|_{l_2\to l_2}$ or $X_0 = g/\|g\|_F$, where $\|M\|_{\alpha\to\beta} := \max_x \frac{\|Mx\|_\beta}{\|x\|_\alpha}$ is the induced operator norm. Then $UV^\top$ can be approximated by several iteration of $X_{t+1} = \frac{3}{2}X_t - \frac{1}{2}X_tX_t^\top X_t$.

- Usually, NewtonSchulz5 is often utilized by applying Newton-Schulz methods for 5 iterations with $X_{t+1} = aX_t + bX_tX_t^\top X_t + c(X_tX_t^\top)^2X_t$ for $(a, b, c) = (3.4445, -4, 7750, 2.0315)$.
- **Polar Express** (Amsel et al., 2025) (For 5 steps):

$$\begin{aligned}(a_i, b_i, c_i) = [&(8.28721201814563, -23.595886519098837, 17.300387312530933),\\ &(4.107059111542203, -2.9478499167379106, 0.5448431082926601),\\ &(3.9486908534822946, -2.908902115962949, 0.5518191394370137),\\ &(3.3184196573706015, -2.488488024314874, 0.51004894012372),\\ &(2.300652019954817, -1.6689039845747493, 0.4188073119525673)].\end{aligned}$$

K-FAC (Martens & Grosse, 2015; Grosse & Martens, 2016) For one layer of neural network $\mathbf{u}_l = \theta_l\mathbf{h}_{l-1} + \mathbf{b}_l, \mathbf{h}_l = \phi(\mathbf{u}_l)$, for $G_l = \nabla_{\mathbf{u}_l}J(\theta_l)$, K-FAC updates θ_l by $L_{l,t} = L_{l,t-1} + G_lG_l^\top$, $R_{l,t} = R_{l,t-1} + \mathbf{h}_{l-1}\mathbf{h}_{l-1}^\top$, $\theta_{l,t+1} = \theta_{l,t} - \eta L_{l,t}^{-1}g_tR_{l,t}^{-1}$.

FOOF (Benzing, 2022) Similar to K-FAC, but $\Sigma_{l,t} = \beta\Sigma_{l,t-1} + (1-\beta)G_lG_l^\top$, $L_{l,t} = L_{l,t-1}$ (except for every T steps, $L_{l,t} = \Sigma_{l,t} + \epsilon\mathbf{I}$), $\theta_{l,t+1} = \theta_{l,t} - \eta L_{l,t}^{-1}g_t$.

Shampoo (Gupta et al., 2018) $L_t = L_{t-1} + g_tg_t^\top$, $R_t = R_{t-1} + g_t^\top g_t$, $\theta_{t+1} = \theta_t - \eta L_t^{-1/4}g_tR_t^{-1/4}$.

- Proposed by Bernstein & Newhouse (2024), $\Delta\theta = -\eta(gg^\top)^{-1/4}g(g^\top g)^{-1/4} = -\eta UV^\top$, where $g = U\Sigma V^\top$ is the SVD decomposition of the gradient.

CASPR (Duvvuri et al., 2024) (Only for 2D parameters $\theta_t \in \mathbb{R}^{m\times n}$) $L_t = L_{t-1} + g_tg_t^\top$, $R_t = R_{t-1} + g_t^\top g_t$, $\widetilde{L}_t^{-1/4} = (L_t + \epsilon I_m)^{-1/4}$, $\widetilde{R}_t^{-1/4} = (R_t + \epsilon I_n)^{-1/4}$, $U_t = \widetilde{L}_t^{-1/4}g_t + g_t\widetilde{R}_t^{-1/4}$, $\theta_{t+1} = \theta_t - \eta(\widetilde{L}_t^{-1/4}U_t + U_t\widetilde{R}_t^{-1/4})$.

Muon (Jordan et al., 2024) (Only for 2D parameters $\theta_t \in \mathbb{R}^{m\times n}$) $M_t = \mu M_{t-1} + g_t$, $O_t = \text{NewtonSchulz5}(M_t)$, $\theta_{t+1} = \theta_t - \eta O_t$. (In practice, $\mu M_t + g_t$ is used in NewtonSchulze5 instead of M_t.)

- **Moonlight** (Liu et al., 2025a) $\theta_{t+1} = \theta_t - \eta(0.2\sqrt{\max(m,n)}O_t + \lambda\theta_t)$

AdaMuon (Si et al., 2025) (Based on Muon) $v_t = \mu v_{t-1} + (1-\mu)O_t \odot O_t$, $\widehat{O}_t = \frac{O_t}{\sqrt{v_t+\epsilon}}$, $\theta_{t+1} = \theta_t - \eta(\frac{0.2\sqrt{mn}}{||\widehat{O}_t||_F}\widehat{O}_t + \lambda\theta_t)$.

PolarGrad (Lau et al., 2025) $U_tH_t = \text{polar}(g_t)$, $\theta_{t+1} = \theta_t - \eta\,\text{tr}(H_t)U_t$.

- *Polar Decomposition*: For any matrix $M \in \mathbb{R}^{m\times n}$, it has a polar decomposition $A = U_pH$ (if $m \geq n$) or $A = HU_p$ (if $m < n$) for semi-orthogonal matrix $U_p \in \mathbb{R}^{m\times n}$ and Hermitian matrix $H \in \mathbb{S}_+^n$ (if $m \geq n$) or $\mathbb{S}_+^m$ (if $m < n$). For $U\Sigma V^\top = \text{SVD}(A)$, $U_p = UV^\top$, $H = V\Sigma V^\top$ (if $m \geq n$) or $H = U\Sigma U^\top$ (if $m < n$).
- **PolarMuon**: $m_t = \beta m_{t-1} + (1-\beta)g_t$, $U_tH_t = \text{polar}(m_t)$, $\theta_{t+1} = \theta_t - \eta(\text{tr}(H_t)U_t + \lambda\theta_t)$.
- **Polar-first**: $U_tH_t = \text{polar}(g_t)$, $m_t = \beta m_{t-1} + (1-\beta)U_t$, $\theta_{t+1} = \theta_t - \eta(\text{tr}(H_t)m_t + \lambda\theta_t)$.

AdaDiag (Nguyen et al., 2025) (Only for 2D parameters $\theta \in \mathbb{R}^{m\times n}$)) If $t \mod T = 0$, $P_t, Q_t^\top = SVD(g_t)$ else $P_t, Q_t^\top = P_{t-1}, Q_{t-1}^\top$. $\widetilde{g}_t = P_t^\top g_t$, $m_t = \beta_1m_{t-1} + (1-\beta_1)\widetilde{g}_t$, $v_t = \beta_2v_{t-1} + (1-\beta_2)\widetilde{g}_t^2$, $\theta_{t+1} = \theta_t - \eta_t(P_t\frac{m_t}{\sqrt{v_t+\epsilon}} + \lambda\theta_t)$

- **AdaDiag++**: similarly but $\widetilde{g}_t = P_t^\top g_t Q_t$, $\theta_{t+1} = \theta_t - \eta_t(P_t \frac{m_t}{\sqrt{v_t+\epsilon}} Q_t^\top + \lambda\theta_t)$

COSMOS (Liu et al., 2025b) (Only for 2D parameters $\theta \in \mathbb{R}^{m\times n}$)): $m_t = \beta_1 m_{t-1} + (1-\beta_1)g_t$, $u_t = \text{QR}(\beta_2 u_{t-1}s_{t-1} + (1-\beta_2)g_t^\top g_t u_{t-1})$, $s_t = u_t^\top(\beta_2 u_{t-1}s_{t-1}u_{t-1}^\top + (1-\beta_2)g_t^\top g_t)u_t$, $v_t = \beta_2 v_{t-1} + (1-\beta_2)(g_t u_t) \odot (g_t u_t)$, $a_t = (\frac{m_t u_t/(1-\beta_1^t)}{\sqrt{(v_t+\epsilon)/(1-\beta_2^t)}})u_t^\top$, $b_t = \text{Norm}(\text{NewtonSchulz5}(\frac{m_t - m_t u_t u_t^\top}{||m_t - m_t u_t u_t^\top||_F}))$, $\theta_{t+1} = \theta_t - \eta\text{Norm}(a_t + \gamma b_t\sqrt{m})$, where $\text{Norm}(X) = \frac{\sqrt{n}X}{||X||_F}$.

Dion (Ahn et al., 2025) (Centralized version. Only for 2D parameters $\theta \in \mathbb{R}^{m\times n}$) $B_t = m_{t-1} + g_t$. Then do Power Iteration for 1 iteration: $P_t = \text{Orthogonalize}(B_t Q_{t-1})$ (orthogonalize P_t using Cholesky decomposition), $R_t = B_t^\top P_t$. $m_t = B_t - (1-\mu)P_t R_t^\top$, $Q_t = \text{ColumnNormalize}(R_t)$ (Normalize each column of R_t), $\theta_t = \theta_{t-1} - \eta\sqrt{m/n}P_t Q_t^\top$.

MARS-M (Liu et al., 2025c) (Only for 2D parameters $\theta \in \mathbb{R}^{m\times n}$) $c_t = g_t + \gamma_t \frac{\beta_1}{1-\beta_1}(g_t - \nabla f(\theta_{t-1}, \boldsymbol{\xi}_t))$, $\widetilde{c}_t = \text{Clip}(c_t, 1)$, $m_t = \beta_1 m_{t-1} + (1-\beta_1)\widetilde{c}_t$, $O_t = \text{NewtonSchulz5}(m_t)$, $\theta_{t+1} = \theta_t - \eta_t(0.2\sqrt{\max(m,n)}O_t + \lambda\theta_t)$.

NorMuon (Li et al., 2025) (Only for 2D parameters $\theta \in \mathbb{R}^{m\times n}$) $m_t = \beta_1 m_{t-1} + (1-\beta_1)g_t$. $O_t = \text{NewtonSchulze5}(m_t)$, $v_t = \beta_2 v_{t-1} + (1-\beta_2)\text{mean}_{\text{col}}(O_t \cdot O_t)$, $V_t = \text{ExpandRows}(v_t)$, $\widehat{O}_t = O_t/(\sqrt{V}_t + \epsilon)$, $\theta_{t+1} = \theta_t - \eta(\lambda\theta_t + 0.2\sqrt{mn}\widehat{O}_t/\|\widehat{O}_t\|_F$.

MuonAdam (Crawshaw et al., 2025) For 2D factors $m_t = \beta_1 m_{t-1} + (1-\beta_1)g_t$, $\theta_{t+1} = \theta_t - \eta_m\text{Polar}(m_t)$ with polar decomposition. For 1D parameters, use Adam(W) with different learning rate.

Conda (Wang et al., 2025a) (Column-Normalized Adam) (Only for 2D parameters $\theta \in \mathbb{R}^{m\times n}$, assume $m \le n$) $m_t = \beta_1 m_{t-1} + (1-\beta_1)g_t$. If $t \mod T = 0$, $U_t, \Sigma_t, V_t^\top = \text{SVD}(m_t)$, $\bar{U}_t = U_t$; else $\bar{U}_t = \bar{U}_{t-1}$. $m_t' = \bar{U}_t^\top m_t$, $n_t = \beta_2 V_{t-1} + (1-\beta_2)(\bar{U}_t^\top g_t)^2$, $\theta_{t+1} = \theta_t + \eta\bar{U}_t \frac{m_t'}{\sqrt{n_t}+\epsilon}\frac{1-\beta_2}{1-\beta_1}$.

SSO (Xie et al., 2026) (Only for 2D parameters $\theta \in \mathbb{R}^{m\times n}$) $m_t = \beta m_{t-1} + (1-\beta)g_t$, $\widehat{m}_t = m_t/\|m_t\|_F$. $(\sigma_t, u_t, v_t) = \text{PowerIteration}(\theta_t)$ (to get the top singular value and vectors), $\Theta_t = u_t v_t^\top$. Define $R = \sqrt{d_{out}/d_{in}}$ as μP scaler. $\lambda_t^* = \arg\min_\lambda\langle\Theta_t, \text{msign}(\widehat{m}_t + \lambda\Theta_t)\rangle$ (using Bisection search with tolerance ϵ). $\theta_{t+1} = \theta_t \cdot R/\sigma_t - \eta R \cdot \text{msign}.(\widehat{m}_t + \lambda_t^*\Theta_t)$

5.5 Miscellaneous

BFGS (Broyden–Fletcher–Goldfarb–Shanno) (Fletcher, 1987) $d_t = -H_t g_t$, $\alpha_t = \arg\min_\alpha f(\theta - \alpha d_t)$, $\theta_{t+1} = \theta_t + \alpha_t d_t$, $H_{t+1} = (I - \frac{s_t y_t}{y_t^\top s_t})^\top H_t(I - \frac{y_t s_t^\top}{y_t^\top s_t}) + \frac{s_t s_t^\top}{y_t^\top s_t}$, where $s_t = \theta_{t+1} - \theta_t$, $y_t = g_{t+1} - g_t$.

L-BFGS (Liu & Nocedal, 1989) Based on BFGS, but choose α_t satisfying Wolfe conditions (try $\alpha_t = 1$ first): $f(\theta_t + \alpha_t d_t) \le f(\theta_t) + \beta'\alpha_t g_t^\top d_t$. Moreover, for $\widehat{m} = \min\{t, m-1\}$,

$$H_{t+1} = (\prod_{i=t,inv}^{t-\widehat{m}} V_i^\top)H_0(\prod_{i=t-\widehat{m}}^{t} V_i) + \sum_{j=t-\widehat{m}}^{t}\rho_j(\prod_{i=t,inv}^{j+1} V_i^\top)s_j s_j^\top(\prod_{i=j+1}^{t} V_i),$$

where $\prod_{i=t,inv}^{t'}$ denotes the product of matrices with indices from t to $t' < t$, and $\rho_t = 1/(y_t^\top s_t)$.

SOAP (Vyas et al., 2024) (Only for 2D parameters) $g_t' = Q_L^\top g Q_R$, $m_t = \beta_1 m_{t-1} + (1-\beta_1)g_t$, $m_t' = Q_L^\top m_t Q_R$, $v_t = \beta_2 v_{t-1} + (1-\beta_2)(g_t')^2$, $\theta_{t+1} = \theta_t - \eta Q_L \frac{\widehat{m}_t'}{\sqrt{\widehat{v}_t}+\epsilon}Q_R^\top$, where $\widehat{m}_t' = \frac{m_t'}{1-\beta_1^t}$, $\widehat{v}_t = \frac{v_t}{1-\beta_2^t}$. $L_t = \beta_2 L_{t-1} + (1-\beta_2)gg^\top$, $R_t = \beta_2 R_{t-1} + (1-\beta_2)g^\top g$. For every k steps, obtain Q_L =QR-Eigenvectors(LQ_L), Q_R =QR-Eigenvectors(RQ_R), where QR-Eigenvectors returns the eigenvector matrix of the QR decomposition for the input matrix.

Hessian-free (Martens et al., 2010) Define the function $B_n(d) = \mathbf{H}(\theta_t)d + \lambda d$, where $\mathbf{H}(\theta_t)d = \lim_{\epsilon\to 0}\frac{\nabla f(\theta_t + \epsilon d) - g_t}{\epsilon}$. $p_t = \text{CG-Minimize}(B_t, -g_t)$, where CG-Minimize is the linear conjugate gradient algorithm, $\theta_{t+1} = \theta_t + p_t$. λ can be adjusted by Levenberg-Marquardt style heuristic:

for $\rho_t = \frac{f(\theta_t+p)-f(\theta_t)}{q_{\theta_t}(p)-q_{\theta_t}(0)}$ with $q_{\theta_t}(\cdot)$ the minimization objective of CG, if $\rho_t < \frac{1}{4}, \lambda \to \frac{3}{2}\lambda$, else if $\rho_t > \frac{3}{4}, \lambda \to \frac{2}{3}\lambda$, otherwise keep λ unchanged.

6 Variance Reduction

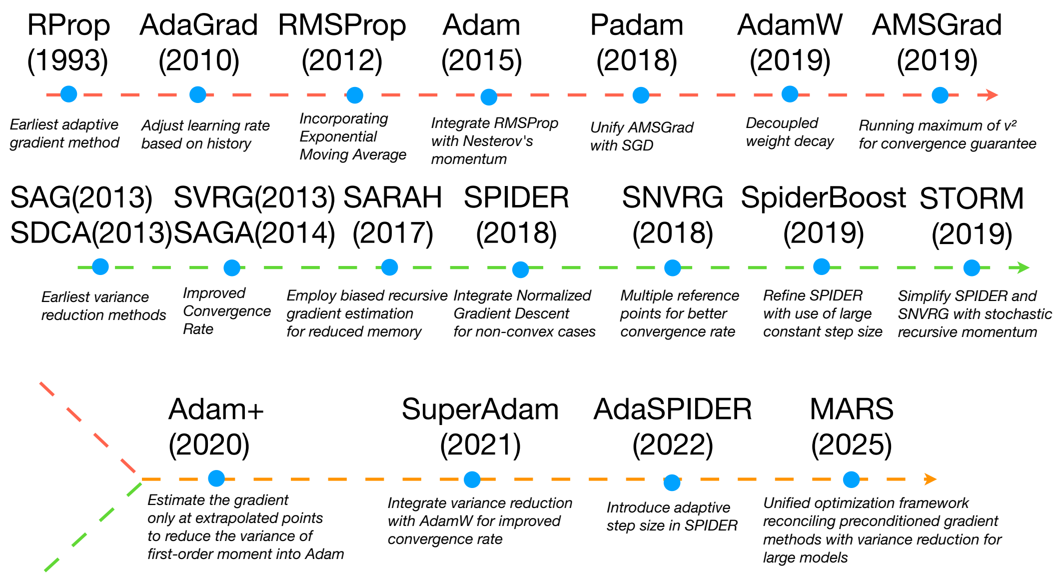

Figure 3: A brief history of variance reduction optimization methods. Some optimizers are omitted.

Since adaptive gradient methods may have risks of high stochastic gradient variance, some researchers consider variance reduction techniques to address this challenge. The core idea, first proposed by Johnson & Zhang (2013), is as follows:

$$\mathbf{m}_t = \nabla f(\theta_t, \boldsymbol{\xi}_t) - \nabla f(\widetilde{\theta}_t, \boldsymbol{\xi}_t) + \nabla F(\widetilde{\mathbf{x}}),$$

where $\widetilde{\mathbf{x}}$ is some reference point (anchoring point) that is periodically updated. By differentiating the gradient for different points on the same data point, the variance in optimization can be reduced significantly.

SAG (Roux et al., 2012) $\theta_{t+1} = \theta_t - \frac{\eta}{n}\sum_{i=1}^{n} y_{i,t}$, where at each t, one $\boldsymbol{\xi}_{i_t}$ is chosen and $y_{i,t} = \nabla f(\theta_t, \boldsymbol{\xi}_{i,t})$ and for other samples, keep $y_{\cdot,t}$ unchanged.

SDCA (Shalev-Shwartz & Zhang, 2013) (For machine learning task $L(\theta) = \frac{1}{n}\sum_{i=1}^{n}\phi_i(\theta^\top \boldsymbol{\xi}_i) + \frac{\lambda}{2}\|\theta\|^2$ with scalar convex function ϕ_i), for $\boldsymbol{\xi}_t = \boldsymbol{\xi}_i$, $\Delta\alpha_t = \arg\max_{\Delta\alpha} -\phi_i^*(-(\alpha_{t-1}+\Delta\alpha), \boldsymbol{\xi}_t) - \frac{\lambda n}{2}\|\theta_{t-1} + \frac{1}{\lambda n}\boldsymbol{\xi}_t\Delta\alpha\|^2$, $\alpha_t = \alpha_{t-1} + \Delta\alpha_t e_i$, $\theta_t = \theta_{t-1} + (\lambda n)^{-1}\boldsymbol{\xi}_t\Delta\alpha_t$. Output $\frac{1}{T}\sum_{i=1}^{T}\theta_{t-1}$ or randomly chosen from $\{\theta_t\}_{t=1}^{T}$.

SAGA (Defazio et al., 2014) Keep a $n \times d$ matrix $\phi = \{\nabla f(\theta_0, \boldsymbol{\xi}_i)\}_{i=1}^{n}$ storing the parameters, for $\boldsymbol{\xi}_t = \boldsymbol{\xi}_j$, $\theta_{t+1} = \arg\min_\theta\{h(\theta) + \frac{1}{2\gamma}\|\theta - (\theta_t - \gamma[\nabla f(\theta_t, \boldsymbol{\xi}_t) - \phi_j + \frac{1}{n}\sum_{i=1}^{n}\phi_i])\|^2\}$, where $h(\theta)$ is the regularization function, then update $\phi_j = \theta_t$ and keep other ϕ_i unchanged.

SVRG (Johnson & Zhang, 2013) For each epoch s, calculate the full gradient g_s for θ_s. For each iteration t, $\theta_{s,t+1} = \theta_{s,t} - \eta(\nabla f(\theta_{s,t}, \boldsymbol{\xi}_{s,t}) - \nabla f(\theta_s, \boldsymbol{\xi}_{s,t}) + g_s)$, set $\theta_s = \theta_{s,T}$ or randomly chosen from $\{\theta_{s,t}\}_{t=1}^{T}$

- α-**SVRG** (Yin et al., 2025) $\theta_{s,t+1} = \theta_{s,t} - \eta(\alpha_t(\nabla f(\theta_{s,t}, \boldsymbol{\xi}_{s,t}) - \nabla f(\theta_s, \boldsymbol{\xi}_{s,t})) + g_s)$, where $\alpha_t = \frac{\mathrm{Cov}(\nabla f(\theta_s, \boldsymbol{\xi}_{s,t}), \nabla f(\theta_{s,t}, \boldsymbol{\xi}_{s,t}))}{\mathrm{Var}(\nabla f(\theta_s, \boldsymbol{\xi}_{s,t}))}$.

SARAH (Nguyen et al., 2017) For each epoch s, calculate the full gradient $v_{s,0} = g_s$ for θ_s, $\theta_{s,1} = \theta_{s,0} - \eta v_{s,0}$. For each iteration t, $v_{s,t} = \nabla f(\theta_{s,t}, \boldsymbol{\xi}_{s,t}) - \nabla f(\theta_s, \boldsymbol{\xi}_{s,t}) + v_{s,t-1}$, $\theta_{s,t+1} = \theta_{s,t} - \eta v_{s,t}$, set θ_s randomly chosen from $\{\theta_{s,t}\}_{t=1}^T$.

Hybrid-SGD (Tran-Dinh et al., 2019) (Hybrid SGD with SARAH) $m_t = \beta(m_{t-1} + g_t - \nabla f(\theta_{t-1}, \boldsymbol{\xi}_t)) + (1-\beta) g_t$, $\theta_{t+1} = \theta_t - \eta m_t$

SPIDER-SFO (Fang et al., 2018) $v_t = g_t$ for every q steps with batch size S_1, otherwise $v_t = g_t - \nabla f(\theta_{t-1}, \boldsymbol{\xi}_t) + v_{t-1}$. Update parameters with Option I: $\theta_{t+1} = \theta_t - \eta v_t / \|v_t\|$ until $\|v_t\|$ less than some threshold, or Option II: $\theta_{t+1} = \theta_t - \eta_t v_t$ with $\eta_t = \min(\eta/\|v_t\|, \eta/(2\epsilon))$. Output θ_T or randomly choose from $\{\theta_t\}_{t=1}^T$.

SpiderBoost (Wang et al., 2019) $v_t = g_t - \nabla f(\theta_{t-1}, \boldsymbol{\xi}_t) + v_{t-1}$, $\theta_{t+1} = \theta_t - \eta v_t / \|v_t\|$

AdaSpider (Kavis et al., 2022) If $t \mod n = 0$, $v_t = g_t$, else $v_t = g_t - \nabla f(\theta_{t-1}, \boldsymbol{\xi}_t) + v_{t-1}$. $\gamma_t = 1/\left(n^{1/4}\beta_0\sqrt{n^{1/2}G_0^2 + \sum_{s=0}^t ||v_s||^2}\right)$, $\theta_{t+1} = \theta_t - \gamma_t v_t$.

SNVRG (Zhou et al., 2020b) For loop parameters $\{T_l\}$, set $T = \prod_{l=1}^K T_l$ or $T \sim \text{Geom}(1/(1 + \prod_{l=1}^K T_l))$ as the total number of steps. For each step t, $r_t = \min\{j : 0 = (t \bmod \prod_{l=j+1}^K T_l), 0 \leq j \leq K\}$. For $0 \leq l \leq r_t - 1$, $\theta_t^l = \theta_{t-1}^l$; otherwise $\theta_t^l = \theta_t$. For $0 \leq l \leq r_t - 1$, $g_t^l = g_{t-1}^l$; for $r_t + 1 \leq l \leq K$, $g_t^l = 0$. Then uniformly generate index set I_t with size B_{r_t}, if $r_t > 0$, $g_t^{r_t} = \frac{1}{B_{r_t}} \sum_{i \in I_t} [\nabla f(\theta_t^{r_t}, \boldsymbol{\xi}_i) - \nabla f(\theta_t^{r_t - 1}, \boldsymbol{\xi}_i)]$, otherwise $g_t^0 = \frac{1}{B_0} \sum_{i \in I_t} \nabla f(\theta_t^0, \boldsymbol{\xi}_i)$. And $\theta_{t+1} = \theta_t - \eta \sum_{l=0}^K g_t^l$. The output parameter is randomly chosen from $\{\theta_t\}_{t=1}^T$.

STORM$^+$ (Levy et al., 2021) (Based on STORM) $a_{t+1} = \frac{1}{(1+\sum_{i=1}^t \|g_i\|^2)^{2/3}}$, $\eta_t = \frac{1}{(\sum_{i=1}^t \|d_i\|^2 / a_{i+1})^{1/3}}$

Super-Adam (Huang et al., 2021) (Based on Adam) $c_t = \alpha_t g_t + (1-\alpha_t)[c_{t-1} + \tau(g_t - \nabla f(\theta_{t-1}, \boldsymbol{\xi}_t))]$, $\tau \in \{0, 1\}$, $\widetilde{\theta}_t = \arg\min_\theta \{\eta \langle c_t, \theta \rangle + \frac{1}{2}||\theta - \theta_t||_{\mathbf{H}_t}^2\}$, $\theta_{t+1} = (1-\mu_t)\theta_t + \mu_t \widetilde{\theta}_t$, where $\mathbf{H}_t$ is defined by one of the following cases:

- Case 1: $\mathbf{H}_t = \text{diag}(\sqrt{v_t} + \lambda)$
- Case 2: $v_t = \beta v_{t-1} + (1-\beta)\|g_t\|$, $\mathbf{H}_t = (v_t + \lambda) I_d$
- Case 3 (Barzilai-Borwein technique): $b_t = \frac{\|\langle g_t - \nabla f(\theta_{t-1}, \boldsymbol{\xi}_t), \theta_t - \theta_{t-1}\rangle\|}{\|\theta_t - \theta_{t-1}\|}$, $\mathbf{H}_t = (b_t + \lambda) I_d$
- Case 4-1: $v_t = \beta_2 v_{t-1} + (1-\beta_2)(g_t - m_t)^2$, $\mathbf{H}_t = \text{diag}(\sqrt{v_t} + \lambda)$
- Case 4-2: $v_t = \beta_2 v_{t-1} + (1-\beta_2)\|g_t - m_t\|$, $\mathbf{H}_t = (v_t + \lambda) I_d$

ROOT-SGD (Li et al., 2022) $v_t = g_t + \frac{t-1}{t}(v_{t-1} - \nabla f(\theta_{t-1}, \boldsymbol{\xi}_t))$, $\theta_{t+1} = \theta_t - \eta v_t$

AdaSVRPS/AdaSVRLS (Jiang & Stich, 2024) $F_{\boldsymbol{\xi}_t}(\theta) = f(\theta, \boldsymbol{\xi}_t) + \mathbf{w}^\top(\nabla f(\mathbf{w}_t) - \nabla f(\mathbf{w}_t, \boldsymbol{\xi}_t)) + \frac{\mu_F}{2}\|\theta - \theta_t\|^2$, $\theta_{t+1} = \theta_t - \eta_t \nabla_\theta F_{\boldsymbol{\xi}_t}(\theta_t)$. With probability p_{t+1}, $\mathbf{w}_{t+1} = \theta_t$, otherwise $\mathbf{w}_{t+1} = \mathbf{w}_t$. Output $\frac{1}{T}\sum_{t=0}^T \theta_t$. Here

- $\eta_t = \min\{\frac{F_{\boldsymbol{\xi}_t}(\theta_t) - F_{\boldsymbol{\xi}_t}^*}{c_p \|\nabla F_{\boldsymbol{\xi}_t}(\theta_t)\|^2 \sqrt{\sum_{s=0}^t F_{\boldsymbol{\xi}_s}(\theta_s) - F_{\boldsymbol{\xi}_s}^*}}, \eta_{t-1}\}$ for AdaSVRPS
- $\eta_t = \min\{\gamma_t \frac{1}{c_l \sqrt{\sum_{s=0}^t \gamma_s \|\nabla F_{\boldsymbol{\xi}_s}(\theta_s)\|^2}}, \eta_{t-1}\}$ for AdaSVRLS, where γ_t can be obtained by Armijo Backtracking line-search (Armijo, 1966; Nocedal & Wright, 2006): do $\gamma = \beta\gamma$ until $f(\theta_t - \gamma \nabla f(\theta_t, \boldsymbol{\xi}_t), \boldsymbol{\xi}_t) \leq f(\theta_t, \boldsymbol{\xi}_t) - \rho\gamma \|\nabla f(\theta_t, \boldsymbol{\xi}_t)\|^2$
- For AdaSPS and AdaSLS, just set $F_{\boldsymbol{\xi}_t}(\theta) = f(\theta, \boldsymbol{\xi}_t)$ and set $F_{\boldsymbol{\xi}_t}^*$ as a predefined lower bound.

VRAdam (Li et al., 2023) (Based on Adam) $m_t = \beta_1 m_{t-1} + (1-\beta_1)(g_t + \frac{\beta_1}{1-\beta_1}(g_t - \nabla f(\theta_{t-1}, \boldsymbol{\xi}_t))$, $\theta_{t+1} = \theta_t - \eta \frac{m_t}{\sqrt{\hat{v}_t} + \epsilon}$

MARS (Yuan et al., 2025) (Based on AdamW/Lion/Shampoo) $c_t = g_t + \gamma_t \frac{\beta_1}{1-\beta_1}(g_t - \nabla f(\theta_{t-1}, \boldsymbol{\xi}_t))$, $\widetilde{c}_t = \text{Clip}(c_t, 1)$, $m_t = \beta_1 m_{t-1} + (1-\beta_1)\widetilde{c}_t$, $\theta_{t+1} = \arg\min_\theta\{\eta \langle m_t, \theta\rangle + \frac{1}{2}\|\theta - \theta_t\|^2_{\mathbf{H}_t}\}$, or $\theta_{t+1} = \arg\min_\theta\{\eta \langle m_t, \theta\rangle + \frac{1}{2}\|\theta - (1-\eta\lambda)\theta_t\|^2_{\mathbf{H}_t}\}$ with weight decay.

- MARS-AdamW: $v_t = \beta_2 v_{t-1} + (1-\beta_2)\widetilde{c}_t^2$, $\mathbf{H}_t := \sqrt{\text{diag}(v_t)} \cdot \frac{1-\beta_1^t}{\sqrt{1-\beta_2^t}}$, (equivalently, $\theta_{t+1} = \theta_t - \eta(\frac{\widehat{m}_t}{\sqrt{\widehat{v}_t+\epsilon}} + \lambda\theta_t)$)
- MARS-Lion: $\mathbf{H}_t = \sqrt{\text{diag}(m_t^2)}$, (equivalently, $\theta_{t+1} = \theta_t - \eta(\text{Sign}(m_t) + \lambda\theta_t)$)
- MARS-Shampoo: $\mathbf{H}_t = (\sum_{\tau=1}^t g_\tau g_\tau^\top)^{1/4} \otimes (\sum_{\tau=1}^t g_\tau^\top g_\tau)^{1/4}$, (equivalently, $U_t, \Sigma_t, V_t = \text{SVD}(m_t)$ $\theta_{t+1} = \theta_t - \eta(U_t V_t^\top + \lambda\theta_t)$), and use Newton-Schulz iteration methods to approximate SVD decomposition (See Section 5.4 for detail).
- MARS-Approximate: $c_t = g_t + \gamma_t \frac{\beta_1}{1-\beta_1}(g_t - g_{t-1})$.
- MVR1 (Chang et al., 2025) $\theta_{t+1} = \theta_t - \eta_t O_t$, where $O_t \in \arg\min_O \|O - m_t\|_F$ such that $O^\top O = I_n$. And MVR2 is the approximate version of MVR1.

7 Other Topics

7.1 Scheduler-Free Methods

Schedule-Free AdamW (Defazio et al., 2024)

- (Form 1) $y_t = (1-\beta_1)z_t + \beta_1\theta_t$, $g_t = \nabla_\theta J(y_t)$, $\eta_t = \eta\sqrt{1-\beta_2^t}\min(1, t/T_{\text{warmup}})$, $z_{t+1} = z_t - \eta_t(\frac{g_t}{\sqrt{v_t+\epsilon}} + \lambda y_t)$, $\theta_{t+1} = (1-c_{t+1})\theta_t + c_{t+1}z_{t+1}$, where $c_{t+1} = \frac{\eta_t^2}{\sum_{i=1}^t \eta_i^2}$
- (Form 2) $\eta_t = \eta\sqrt{1-\beta_2^t}\min(1, t/T_{\text{warmup}})$, $g_t = \nabla_\theta J(y_t)$, $\Delta_t = \eta_t(\frac{g_t}{\sqrt{v_t+\epsilon}} + \lambda y_t)$, $y_{t+1} = y_t + \frac{\beta_1 c_{t+1}}{1-\beta_1}(y_t - \theta_t) - [\beta_1 c_{t+1} + (1-\beta_1)]\Delta_t$, $\theta_{t+1} = \theta_t + \frac{c_{t+1}}{1-\beta_1}(y_t - \theta_t) - c_{t+1}\Delta_t$, where $c_{t+1} = \frac{\eta_t^2}{\sum_{i=1}^t \eta_i^2}$

D-Adaptation (Defazio & Mishchenko, 2023)

- Dual Averaging: $m_t = m_{t-1} + d_{t-1}g_t$, $\eta_t = \frac{1}{\sqrt{\sum_{i=0}^k \|g_i\|^2}}$. For $\widehat{d}_t$, option I: $\widehat{d}_t = \frac{\eta_t\|m_t\|^2 - \sum_{i=0}^{t-1}\eta_i d_i^2\|g_{i+1}\|^2}{2\|m_t\|}$ or option II: $\widehat{d}_t = \frac{\sum_{i=0}^{t-1} d_i\eta_i\langle g_{i+1}, m_i\rangle}{\|m_t\|}$, $d_t = \max(d_{t-1}, \widehat{d}_t)$, $\theta_{t+1} = \theta_t - \eta_t m_t$. Output $\frac{1}{\sum_{t=0}^{T-1} d_t}\sum_{t=1}^T d_{t-1}\theta_t$
- Gradient Descent: $\eta_t = \frac{d_{t-1}}{\sqrt{G^2 + \sum_{i=0}^t \|g_i\|^2}}$, $m_t = m_{t-1} + \eta_t g_t$, $\widehat{d}_t = \frac{\|m_t\|^2 - \sum_{i=0}^t \eta_i^2\|g_{i+1}\|^2}{2\|m_t\|}$, $d_t = \max(d_{t-1}, \widehat{d}_t)$, $\theta_{t+1} = \theta_t - \eta_t m_t$. Output $\frac{1}{\sum_{t=0}^T \eta_t}\sum_{t=0}^T \eta_t\theta_t$
- AdaGrad: $m_t = m_{t-1} + d_{t-1}g_t$, $a_t^2 = a_{t-1}^2 + g_t^2$, $A_t = \text{diag}(a_t)$, $\widehat{d}_t = \frac{\|m_t\|^2_{A_t^{-1}} - \sum_{i=0}^{t-1} d_i^2\|g_{i+1}\|^2_{A_i^{-1}}}{2\|m_t\|_1}$, $d_t = \max(d_{t-1}, \widehat{d}_t)$, $\theta_{t+1} = \theta_t - A_t^{-1}m_t$. Output $\frac{1}{\sum_{t=0}^{T-1} d_t}\sum_{t=1}^T d_{t-1}\theta_t$
- SGD: $\eta_k = \eta d_{k-1}/G$, $m_t = m_{t-1} + \eta_t g_t$, $z_t = z_{t-1} - \eta_t g_t$, $\theta_{t+1} = \beta\theta_t + (1-\beta)z_t$, $\widehat{d}_t = \frac{2\sum_{i=0}^{t-1}\eta_{i+1}\langle g_{i+1}, m_i\rangle}{\|m_t\|}$, $d_t = \max(d_{t-1}, \widehat{d}_t)$.
- Adam version (Based on Adam) $m_t = \beta_1 m_{t-1} + (1-\beta_1)\eta d_{t-1}g_t$, $\theta_{t+1} = \theta_t - \frac{m_t}{\sqrt{v_t+\epsilon}}$, $s_t = \sqrt{\beta_2}s_{t-1} + (1-\sqrt{\beta_2})\eta d_{t-1}g_t$, $r_t = \sqrt{\beta_2}r_{t-1} + (1-\sqrt{\beta_2})\eta d_{t-1}\langle g_t, s_{t-1}\rangle_{(\text{diag}(\sqrt{v_t}+\epsilon))^{-1}}$, $\widehat{d}_t = \frac{r_t}{(1-\sqrt{\beta_2})\|s_t\|_1}$, $d_t = \max(d_{t-1}, \widehat{d}_t)$

Adam++ (Tao et al., 2024) $\eta_t = \max(\eta_{t-1}, \|\theta_t - \theta_0\|/\sqrt{d})$, $\beta_{1,t} = \beta_1\lambda^{t-1}$, $m_t = \beta_{1,t}m_{t-1} + (1-\beta_{1,t})g_t$, $\theta_{t+1} = \theta_t - \eta_t \cdot \mathbf{H}_t^{-1}m_t$, where $\mathbf{H}_t = \epsilon + \text{diag}(s_t)$, and s_t is calculated by either of the following ways:

- Case I: $s_t = \sqrt{\sum_{i=0}^{t} g_i^2}$
- Case II: $v_t = \beta_2 v_{t-1} + (1-\beta_2) g_t^2$, $s_t = \sqrt{(t+1)\max_{t' \le t}(v_{t'})}$

AdaGrad++ (Tao et al., 2024) Similar to Adam++, but only choose Case I of s_t, and $\lambda = 0$.

7.2 Randomized Updating

GLD (Gradient Langevin Dynamics) (Durmus & Moulines, 2017; Dalalyan, 2017a,b) $\theta_0 = \mathbf{0}$, $\epsilon_t \sim N(0, I_{d\times d})$, $\theta_{t+1} = \theta_t - \eta g_t + \sqrt{2\eta/\beta}\epsilon_t$, where g_t is the full gradient.

SGLD (Welling & Teh, 2011) Same as GLD, but g_t is the average gradient over samples in small batch.

SGFS (Ahn et al., 2012) For total number of samples N and batch size B, $\gamma = \frac{B+N}{B}$. For small batch $\{\boldsymbol{\xi}_{t,i}\}_{i=1}^{B}$, $v_t = (1-\beta)v_{t-1} + \beta\frac{1}{n-1}\sum_{i=1}^{B}(\nabla f(\theta_t, \boldsymbol{\xi}_{t,i}) - g_t)(\nabla f(\theta_t, \boldsymbol{\xi}_{t,i}) - g_t)^\top$, $\epsilon_t \sim \mathcal{N}(0, \frac{4C}{\epsilon})$, $\theta_{t+1} = \theta_t + 2(\gamma N v_t + \frac{4C}{\epsilon})^{-1}(g_t + \epsilon_t)$.

SVRG-LD (Xu et al., 2018) For each epoch s, first compute the full gradient g_s for θ_s, $\widetilde{g}_{s,t} = g_{s,t} - \nabla f(g_s, \boldsymbol{\xi}_{s,t}) + g_s$, $\epsilon_t \sim N(0, I_{d\times d})$, $\theta_{t+1} = \theta_t - \eta\widetilde{g}_{s,t} + \sqrt{2\eta/\beta}\epsilon_t$. At the last iteration T for each epoch, $g_s = g_{s,T}$.

7.3 Reconciliation of Optimizers

AdaGraft (Agarwal et al., 2020) For optimizers $\mathcal{M}, \mathcal{D}$, $\theta_{t,\mathcal{M}} = \mathcal{M}(\theta_t, g_t)$, $\theta_{t,\mathcal{D}} = \mathcal{D}(\theta_t, g_t)$, $\theta_{t+1} = \theta_t + \frac{\|\theta_{t,\mathcal{M}} - \theta_t\|}{\|\theta_{t,\mathcal{D}} - \theta_t\| + \epsilon} \cdot (\theta_{t,\mathcal{D}} - \theta_t)$

7.4 Architecture-specific Optimizers

GaLore (Zhao et al., 2024a) (Adam for LLM layer weight matrix): $\theta_t \in \mathbb{R}^{m\times n}$, if $t \bmod T = 0$ $U, S, V = \mathrm{SVD}(g_t)$, $P_t = U[:, : r]$ (low-rank projection), otherwise $P_t = P_{t-1}$. Then estimate the low-rank gradient $R_t = P_t^\top g_t$, and use Adam to optimize: $\theta_{t+1} = \theta_t + \eta\alpha P_t\frac{m_t}{\sqrt{v_t}+\epsilon}$ with scale factor α

Adam-mini (Zhang et al., 2024) (Based on Adam) (For each parameter in parameter blocks) $v_t = \beta_2 v_{t-1} + (1-\beta_2) * \mathrm{Mean}(g \odot g)$. For Transformers, partition the parameters of embedding and output layers by tokens, partition the parameters of query and key matrices by heads, and partition the parameters of value matrices, attention projection matrices and MLP layers by output neurons.

Adalayer (Zhao et al., 2024b) (Based on Adam, for language model) For each layer: $v_t = \beta_2 v_{t-1} + (1-\beta_2)\|g_t\|_2^2/\sqrt{p}$ with p the number of parameters in each layer, $\theta_{t+1} = \theta_t - \eta\frac{m_t}{\sqrt{v_t}+\epsilon}$

AdamC (Based on AdamW) (Only for the layer immediately followed by a normalization operation including LayerNorm or BatchNorm, otherwise use AdamW): $\theta_{t+1} = \theta_t - \eta(\frac{\widehat{m}_t}{\sqrt{\widehat{v}_t}+\epsilon} + \frac{\eta}{\eta_{\max}}\lambda\theta_t)$. (In Transformers, AdamC is applied to every linear layer except the output layer.)

STORM-PG (Yuan et al., 2020) (Based on Reinforce Learning tasks) $\theta_{t+1} = \theta_t + \eta\widehat{g}_t$, $\widehat{g}_{t+1} = (1-\gamma)g'_t + g_t$, where $g_t = \frac{1}{B}\sum_{i\in\mathcal{B}} d_i(\theta_{t+1})$, $g'_t = \frac{1}{B}\sum_{i\in\mathcal{B}}[\widehat{g}_t - d_i^{\theta_{t+1}}(\theta_t)]$ is the gradient estimation with different parameters on small batch of trajectories $\{\tau_i\}_{i\in\mathcal{B}}$ with size B, and $d_i(\theta) = \sum_{h=0}^{H-1} d_{i,h}(\theta)$, $d_i^{\theta'}(\theta) = \sum_{h=0}^{H-1}\frac{p(\tau_{i,h}|\theta)}{p(\tau_{i,h}|\theta')}d_{i,h}(\theta)$, where $d_{i,h}(\theta) = (\sum_{t=0}^{h}\nabla\log\pi_\theta(a_t|s_t))(\gamma^h r(s_h, a_h) - b_h)$.

SPPO (Wu et al., 2024) (Based on Reinforce Learning tasks)

$$\theta_{t+1} = \arg\min_\theta \mathbb{E}_{(\boldsymbol{\xi}_t, y_t, \widehat{P}(y_t \succ \pi_t|\boldsymbol{\xi}_t))}\big(\log\big(\frac{\pi_\theta(y_t|\boldsymbol{\xi}_t)}{\pi_t(y_t|\boldsymbol{\xi}_t)}\big) - \eta(\widehat{P}(y_t \succ \pi_t|\boldsymbol{\xi}_t) - \frac{1}{2})\big)^2,$$

where $\pi_t = \pi_{\theta_t}$ is the policy model with the parameter θ_t.

Muon-clip (QK-Clip) (Moonshot AI, 2025) (Based on Muon, for Attention mechanism) In each layer, for input $\{x_i\}$, query and key weight $\mathbf{W}_q, \mathbf{W}_k$, $S_{\max} = (\max_{ij} x_i\mathbf{W}_q)\cdot(\mathbf{x}_j\mathbf{W}_k)$. First update all parameters using Muon, then if $S_{\max} > \tau$:

- MHA/MQA/GQA: multiply $\sqrt{\tau / S_{\max}}$ to $\mathbf{W}_q, \mathbf{W}_k$;
- MLA: multiply $\sqrt{\tau / S_{\max}}$ to $\mathbf{W}_{qc}, \mathbf{W}_{kc}$ and multiply $\tau / S_{\max}$ to $\mathbf{W}_{qr}$.

7.5 Oracle-based Optimization Methods

Linear Minimization Oracle (LMO): For norm-ball $\mathcal{D} := \{x| \; ||x|| \leq \rho\}$ for some norm $|| \cdot ||$ (Euclidean norm in default), $\mathrm{lmo}(s) \in \arg\min_{x \in \mathcal{D}} \langle s, x \rangle$: for $s \in \mathbb{R}^{d_{\text{out}} \times d_{\text{in}}}$:

- 1→RMS (ColNorm, such as Embedding): $\mathrm{col}_j(s) \to -\sqrt{d_{\text{out}}} \frac{\mathrm{col}_j(s)}{\|\mathrm{col}_j(s)\|_2}$;
- 1→ ∞ (Sign): $s \to -\mathrm{sign}(s)$;
- RMS→RMS (Spectral, such as Linear module): $s \to -\sqrt{\frac{d_{\text{out}}}{d_{\text{in}}}} UV^\top$ for $U, \Sigma, V^\top = \mathrm{SVD}(s)$;
- RMS→ ∞ (RowNorm, such as LM Head): $\mathrm{row}_i(s) = -\frac{1}{\sqrt{d_{\text{in}}}} \frac{\mathrm{row}_i(s)}{\|\mathrm{row}_i(s)\|_2}$

Conditional Gradient Method (CG) (Frank et al., 1956; Clarkson, 2010; Jaggi, 2013) $\theta_{t+1} = (1-\eta)\theta_t + \eta \cdot \mathrm{lmo}(g_t)$

Stochastic Conditional Gradient (SCG) (Pethick et al., 2025) $m_t = \beta m_{t-1} + (1-\beta) g_t$, $\theta_{t+1} = (1-\eta)\theta_t + \eta \cdot \mathrm{lmo}(m_t)$ ($\theta_{t+1} = \theta_t + \eta \cdot \mathrm{lmo}(m_t)$ for Unconstrained SCG (uSCG)). For different base optimization method, the LMOs are different:

- Normalized SGD (Hazan et al., 2015) and Momentum Normalized SGD (Cutkosky & Mehta, 2020): $-\rho \frac{m_t}{||m_t||_2}$;
- SignSGD (Bernstein et al., 2018) and Signum (Bernstein et al., 2018): $-\rho \mathrm{sign}(m_t)$;
- Muon (Jordan et al., 2024) (with non-Nesterov based momentum): $-\rho UV^\top$ for $U\Sigma V^\top = \mathrm{SVD}(m_t)$.

Scion (Pethick et al., 2025) $m_t = \beta m_{t-1} + (1-\beta) g_t$, $\theta_{t+1} = (1-\eta)\theta_t + \eta \cdot \mathrm{lmo}_{||\cdot||_{\alpha \to \beta}}(m_t)$ ($\theta_{t+1} = \theta_t + \eta \cdot \mathrm{lmo}_{||\cdot||_{\alpha \to \beta}}(m_t)$ for unconstrained version), where $||A||_{\alpha \to \beta} = \sup_{||z||_\alpha = 1} ||Az||_\beta$.

Gluon (Riabinin et al., 2025) $m_t = \beta m_{t-1} + (1-\beta) g_t$, $\theta_{t+1} = \arg\min_{||\theta_{t+1} - \theta_t|| \leq p_t} \langle m_t, \theta_t \rangle$ (p_t can be $\frac{||g_t||}{L_0 + L_1 ||g_t||}$ or $\frac{L_0}{(t+1)^{3/4}}$.

7.6 Something Beyond Optimizers

In the previous sections, I did not emphasize the changing of the learning rate η. In this section, I collected some interesting research related to the changing rule of learning rate (Scheduler).

7.6.1 Scheduler

Warmup-Stable-Decay (WSD) Scheduler (Hu et al., 2024) Different from the Cosine Learning Rate Scheduler, in MiniCPM, the researchers utilized WSD scheduler (linear warmup-stable learning rate-decay). It is widely used in industry because it can support continual training, and decaying schedule in the end is also compatible for downstream tasks fine-tuning.

For intermediate checkpoints, WSD performs decay schedule and starts from the state before decay when continuing training. However, in **WSD-S** (Wen et al., 2024), the researchers find that the dispose of training in decay phase is not necessary, and just starts from the state after decay with the max learning rate is fine.

7.6.2 Schedule Refinement

Schedule Refinement (Defazio et al., 2023) The researchers perform comprehensive evaluation of learning rate schedules, give proofs on the convergence of some common optimization approaches, and proposed some important conclusions and algorithms:

- Warm-up followed by linear decay is the best overall non-adaptive schedule, outperforming cosine decay.
- **Schedule Refinement for SGD** $\widehat{g}_t = \text{Median-filter}(\|g_t\|, \text{width} = \tau T, \text{pad} = (\text{nearest}, \text{reflect}))$, $w_t = \widehat{g}_t^{-2}$, $\eta'_t = w_t \sum_{p=t+1}^{T} w_p$, $\eta_t = \eta'_t / \max_p(\eta'_p)$.
- **Schedule Refinement for Adam** $\widehat{g}_t = \text{Median-filter}(\| \sum_{i=1}^{d} \frac{g_{t,i}^2}{\sqrt{v_{t,i}}} \|, \text{width} = \tau T, \text{pad} = (\text{nearest}, \text{reflect}))$, $w_t = \widehat{g}_t^{-1}$, $\eta'_t = w_t \sum_{p=t+1}^{T} w_p$, $\eta_t = \eta'_t / \max_p(\eta'_p)$.

7.6.3 "Scaling Law" of Learning Rate

μP and Tensor Program series (Yang et al., 2022) In Tensor Program series, the researchers led by Greg Yang[1] provides theoretical foundation for the "Scaling Law" of hyper-parameter to the model size. In Tensor Program V (μP), they researched how the best hyper-parameters (including initialization variances and learning rates for different optimizers of different components in deep learning models) change with respect to the size of these components.

8 Acknowledgement

Thank Prof. Quanquan Gu, and other students in UCLA AGI Lab including Huizhuo Yuan. Thank former colleagues from ByteDance Inc. and Moonshot Inc. including Jianlin Su for his blogs at `https://kexue.fm/`.

References

Naman Agarwal, Rohan Anil, Elad Hazan, Tomer Koren, and Cyril Zhang. Disentangling adaptive gradient methods from learning rates. *arXiv preprint arXiv:2002.11803*, 2020.

Kwangjun Ahn, Byron Xu, Natalie Abreu, and John Langford. Dion: Distributed orthonormalized updates. *arXiv preprint arXiv:2504.05295*, 2025.

Sungjin Ahn, Anoop Korattikara Balan, and Max Welling. Bayesian posterior sampling via stochastic gradient fisher scoring. In *Proceedings of the 29th International Conference on Machine Learning, ICML 2012, Edinburgh, Scotland, UK, June 26 - July 1, 2012*. icml.cc / Omnipress, 2012. URL `http://icml.cc/2012/papers/782.pdf`.

Shun-Ichi Amari. Natural gradient works efficiently in learning. *Neural computation*, 10(2):251–276, 1998.

Noah Amsel, David Persson, Christopher Musco, and Robert Gower. The polar express: Optimal matrix sign methods and their application to the muon algorithm. *arXiv preprint arXiv:2505.16932*, 2025.

Kang An, Yuxing Liu, Rui Pan, Shiqian Ma, Donald Goldfarb, and Tong Zhang. Asgo: Adaptive structured gradient optimization. *arXiv preprint arXiv:2503.20762*, 2025.

Larry Armijo. Minimization of functions having lipschitz continuous first partial derivatives. *Pacific Journal of mathematics*, 16(1):1–3, 1966.

Sue Becker. Improving the convergence of backpropagation learning with second order method. In *Proceedings of the 1988 Connectionist Models Summer School, San Mateo, CA*. Morgan Kaufmann, 1988.

Frederik Benzing. Gradient descent on neurons and its link to approximate second-order optimization. In *International conference on machine learning*, pp. 1817–1853. PMLR, 2022.

Jeremy Bernstein and Laker Newhouse. Old optimizer, new norm: An anthology. *arXiv preprint arXiv:2409.20325*, 2024.

[1] `https://thegregyang.com/`

Jeremy Bernstein, Yu-Xiang Wang, Kamyar Azizzadenesheli, and Animashree Anandkumar. signsgd: Compressed optimisation for non-convex problems. In *International Conference on Machine Learning*, pp. 560–569. PMLR, 2018.

Yang Cao, Xiaoyu Li, and Zhao Song. Grams: Gradient descent with adaptive momentum scaling. *arXiv preprint arXiv:2412.17107*, 2024.

Da Chang, Yongxiang Liu, and Ganzhao Yuan. On the convergence of muon and beyond. *arXiv preprint arXiv:2509.15816*, 2025.

Jinghui Chen and Quanquan Gu. Padam: Closing the generalization gap of adaptive gradient methods in training deep neural networks. 2018.

Xiangning Chen, Chen Liang, Da Huang, Esteban Real, Kaiyuan Wang, Hieu Pham, Xuanyi Dong, Thang Luong, Cho-Jui Hsieh, Yifeng Lu, and Quoc V. Le. Symbolic discovery of optimization algorithms. In Alice Oh, Tristan Naumann, Amir Globerson, Kate Saenko, Moritz Hardt, and Sergey Levine (eds.), *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, 2023.

Xiangyi Chen, Sijia Liu, Ruoyu Sun, and Mingyi Hong. On the convergence of a class of adam-type algorithms for non-convex optimization. *arXiv preprint arXiv:1808.02941*, 2018.

Kenneth L Clarkson. Coresets, sparse greedy approximation, and the frank-wolfe algorithm. *ACM Transactions on Algorithms (TALG)*, 6(4):1–30, 2010.

Michael Crawshaw, Chirag Modi, Mingrui Liu, and Robert M Gower. An exploration of non-euclidean gradient descent: Muon and its many variants. *arXiv preprint arXiv:2510.09827*, 2025.

Ashok Cutkosky and Harsh Mehta. Momentum improves normalized sgd. In *International conference on machine learning*, pp. 2260–2268. PMLR, 2020.

Ganzhao Yuan21 Da Chang134. Mgup: A momentum-gradient alignment update policy for stochastic optimization.

Arnak Dalalyan. Further and stronger analogy between sampling and optimization: Langevin monte carlo and gradient descent. In *Conference on Learning Theory*, pp. 678–689. PMLR, 2017a.

Arnak S Dalalyan. Theoretical guarantees for approximate sampling from smooth and log-concave densities. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 79(3):651–676, 2017b.

Aaron Defazio and Konstantin Mishchenko. Learning-rate-free learning by d-adaptation. In *International Conference on Machine Learning*, pp. 7449–7479. PMLR, 2023.

Aaron Defazio, Francis Bach, and Simon Lacoste-Julien. Saga: A fast incremental gradient method with support for non-strongly convex composite objectives. *Advances in neural information processing systems*, 27, 2014.

Aaron Defazio, Ashok Cutkosky, Harsh Mehta, and Konstantin Mishchenko. When, why and how much? adaptive learning rate scheduling by refinement. *arXiv preprint arXiv:2310.07831*, 2023.

Aaron Defazio, Xingyu Alice Yang, Harsh Mehta, Konstantin Mishchenko, Ahmed Khaled, and Ashok Cutkosky. The road less scheduled. *arXiv preprint arXiv:2405.15682*, 2024.

Timothy Dozat. Incorporating nesterov momentum into adam. 2016.

John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of machine learning research*, 12(7), 2011.

Alain Durmus and Éric Moulines. Nonasymptotic convergence analysis for the unadjusted langevin algorithm. *The Annals of Applied Probability*, pp. 1551–1587, 2017.

Sai Surya Duvvuri, Fnu Devvrit, Rohan Anil, Cho-Jui Hsieh, and Inderjit S Dhillon. Combining axes preconditioners through kronecker approximation for deep learning. In *The Twelfth International Conference on Learning Representations*, 2024.

Cong Fang, Chris Junchi Li, Zhouchen Lin, and Tong Zhang. Spider: Near-optimal non-convex optimization via stochastic path-integrated differential estimator. *Advances in neural information processing systems*, 31, 2018.

Roger Fletcher. *Practical Methods of Optimization*. John Wiley & Sons, New York, 2 edition, 1987. ISBN 978-0-471-91547-8.

Pierre Foret, Ariel Kleiner, Hossein Mobahi, and Behnam Neyshabur. Sharpness-aware minimization for efficiently improving generalization. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*, 2021.

Marguerite Frank, Philip Wolfe, et al. An algorithm for quadratic programming. *Naval research logistics quarterly*, 3(1-2):95–110, 1956.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

Roger Grosse and James Martens. A kronecker-factored approximate fisher matrix for convolution layers. In *International Conference on Machine Learning*, pp. 573–582. PMLR, 2016.

Marcus Grum. Learning representations by crystallized back-propagating errors. In Leszek Rutkowski, Rafal Scherer, Marcin Korytkowski, Witold Pedrycz, Ryszard Tadeusiewicz, and Jacek M. Zurada (eds.), *Artificial Intelligence and Soft Computing - 22nd International Conference, ICAISC 2023, Zakopane, Poland, June 18-22, 2023, Proceedings, Part I*, volume 14125 of *Lecture Notes in Computer Science*, pp. 78–100. Springer, 2023. doi: 10.1007/978-3-031-42505-9_8. URL `https://doi.org/10.1007/978-3-031-42505-9_8`.

Vineet Gupta, Tomer Koren, and Yoram Singer. A unified approach to adaptive regularization in online and stochastic optimization. *arXiv preprint arXiv:1706.06569*, 2017.

Vineet Gupta, Tomer Koren, and Yoram Singer. Shampoo: Preconditioned stochastic tensor optimization. In *International Conference on Machine Learning*, pp. 1842–1850. PMLR, 2018.

Elad Hazan, Amit Agarwal, and Satyen Kale. Logarithmic regret algorithms for online convex optimization. *Machine Learning*, 69(2):169–192, 2007.

Elad Hazan, Kfir Levy, and Shai Shalev-Shwartz. Beyond convexity: Stochastic quasi-convex optimization. *Advances in neural information processing systems*, 28, 2015.

Byeongho Heo, Sanghyuk Chun, Seong Joon Oh, Dongyoon Han, Sangdoo Yun, Gyuwan Kim, Youngjung Uh, and Jung-Woo Ha. Adamp: Slowing down the slowdown for momentum optimizers on scale-invariant weights. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*, 2021.

Nicholas J Higham. Functions of matrices. society for industrial and applied mathematics, 2008.

Shengding Hu, Yuge Tu, Xu Han, Chaoqun He, Ganqu Cui, Xiang Long, Zhi Zheng, Yewei Fang, Yuxiang Huang, Weilin Zhao, et al. Minicpm: Unveiling the potential of small language models with scalable training strategies. *arXiv preprint arXiv:2404.06395*, 2024.

Feihu Huang, Junyi Li, and Heng Huang. Super-adam: faster and universal framework of adaptive gradients. *Advances in Neural Information Processing Systems*, 34:9074–9085, 2021.

Michael F Hutchinson. A stochastic estimator of the trace of the influence matrix for laplacian smoothing splines. *Communications in Statistics-Simulation and Computation*, 18(3):1059–1076, 1989.

Martin Jaggi. Revisiting frank-wolfe: Projection-free sparse convex optimization. In *International conference on machine learning*, pp. 427–435. PMLR, 2013.

Majid Jahani, Sergey Rusakov, Zheng Shi, Peter Richtárik, Michael W Mahoney, and Martin Takac. Doubly adaptive scaled algorithm for machine learning using second-order information. In *International Conference on Learning Representations*.

Xiaowen Jiang and Sebastian U Stich. Adaptive sgd with polyak stepsize and line-search: Robust convergence and variance reduction. *Advances in Neural Information Processing Systems*, 36, 2024.

Rie Johnson and Tong Zhang. Accelerating stochastic gradient descent using predictive variance reduction. *Advances in neural information processing systems*, 26, 2013.

Keller Jordan, Yuchen Jin, Vlado Boza, Jiacheng You, Franz Cecista, Laker Newhouse, and Jeremy Bernstein. Muon: An optimizer for hidden layers in neural networks, 2024. URL `https://kellerjordan.github.io/posts/muon/`.

Ali Kavis, Stratis Skoulakis, Kimon Antonakopoulos, Leello Tadesse Dadi, and Volkan Cevher. Adaptive stochastic variance reduction for non-convex finite-sum minimization. *Advances in Neural Information Processing Systems*, 35:23524–23538, 2022.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In Yoshua Bengio and Yann LeCun (eds.), *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015.

Jungmin Kwon, Jeongseop Kim, Hyunseo Park, and In Kwon Choi. Asam: Adaptive sharpness-aware minimization for scale-invariant learning of deep neural networks. In *International Conference on Machine Learning*, pp. 5905–5914. PMLR, 2021.

Tim Tsz-Kit Lau, Qi Long, and Weijie Su. Polargrad: A class of matrix-gradient optimizers from a unifying preconditioning perspective. *arXiv preprint arXiv:2505.21799*, 2025.

Kfir Levy, Ali Kavis, and Volkan Cevher. Storm+: Fully adaptive sgd with recursive momentum for nonconvex optimization. *Advances in Neural Information Processing Systems*, 34:20571–20582, 2021.

Chris Junchi Li, Wenlong Mou, Martin Wainwright, and Michael Jordan. Root-sgd: Sharp nonasymptotics and asymptotic efficiency in a single algorithm. In *Conference on Learning Theory*, pp. 909–981. PMLR, 2022.

Haochuan Li, Alexander Rakhlin, and Ali Jadbabaie. Convergence of adam under relaxed assumptions. *Advances in Neural Information Processing Systems*, 36:52166–52196, 2023.

Wenjie Li, Zhaoyang Zhang, Xinjiang Wang, and Ping Luo. Adax: Adaptive gradient descent with exponential long term memory. *arXiv preprint arXiv:2004.09740*, 2020.

Zichong Li, Liming Liu, Chen Liang, Weizhu Chen, and Tuo Zhao. Normuon: Making muon more efficient and scalable. *arXiv preprint arXiv:2510.05491*, 2025.

Kaizhao Liang, Lizhang Chen, Bo Liu, and Qiang Liu. Cautious optimizers: Improving training with one line of code. *arXiv preprint arXiv:2411.16085*, 2024.

Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, et al. Deepseek-v3 technical report. *arXiv preprint arXiv:2412.19437*, 2024a.

Dong C Liu and Jorge Nocedal. On the limited memory bfgs method for large scale optimization. *Mathematical programming*, 45(1):503–528, 1989.

Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. Visual instruction tuning. *Advances in neural information processing systems*, 36:34892–34916, 2023.

Hong Liu, Zhiyuan Li, David Leo Wright Hall, Percy Liang, and Tengyu Ma. Sophia: A scalable stochastic second-order optimizer for language model pre-training. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*, 2024b.

Jingyuan Liu, Jianlin Su, Xingcheng Yao, Zhejun Jiang, Guokun Lai, Yulun Du, Yidao Qin, Weixin Xu, Enzhe Lu, Junjie Yan, et al. Muon is scalable for llm training. *arXiv preprint arXiv:2502.16982*, 2025a.

Liming Liu, Zhenghao Xu, Zixuan Zhang, Hao Kang, Zichong Li, Chen Liang, Weizhu Chen, and Tuo Zhao. Cosmos: A hybrid adaptive optimizer for memory-efficient training of llms. *arXiv preprint arXiv:2502.17410*, 2025b.

Liyuan Liu, Haoming Jiang, Pengcheng He, Weizhu Chen, Xiaodong Liu, Jianfeng Gao, and Jiawei Han. On the variance of the adaptive learning rate and beyond. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*, 2020a.

Mingrui Liu, Wei Zhang, Francesco Orabona, and Tianbao Yang. Adam $^+$: A stochastic method with adaptive variance reduction. *arXiv preprint arXiv:2011.11985*, 2020b.

Yifeng Liu, Angela Yuan, and Quanquan Gu. Mars-m: When variance reduction meets matrices. *arXiv preprint arXiv:2510.21800*, 2025c.

Yong Liu, Siqi Mai, Xiangning Chen, Cho-Jui Hsieh, and Yang You. Towards efficient and scalable sharpness-aware minimization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 12360–12370, 2022.

Ilya Loshchilov. Weight norm control. *arXiv preprint arXiv:2311.11446*, 2023.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*, 2019.

Liangchen Luo, Yuanhao Xiong, Yan Liu, and Xu Sun. Adaptive gradient methods with dynamic bound of learning rate. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*, 2019.

James Martens and Roger Grosse. Optimizing neural networks with kronecker-factored approximate curvature. In *International conference on machine learning*, pp. 2408–2417. PMLR, 2015.

James Martens et al. Deep learning via hessian-free optimization. In *Icml*, volume 27, pp. 735–742, 2010.

Moonshot AI. Kimi K2: Open Agentic Intelligence, 2025. URL `https://moonshotai.github.io/Kimi-K2/`. Accessed: 2025-07-17.

Yurii Nesterov. A method for solving the convex programming problem with convergence rate o (1/k2). In *Dokl akad nauk Sssr*, volume 269, pp. 543, 1983.

Lam M Nguyen, Jie Liu, Katya Scheinberg, and Martin Takáč. Sarah: A novel method for machine learning problems using stochastic recursive gradient. In *International conference on machine learning*, pp. 2613–2621. PMLR, 2017.

Son Nguyen, Bo Liu, Lizhang Chen, and Qiang Liu. Improving adaptive moment optimization via preconditioner diagonalization. *arXiv preprint arXiv:2502.07488*, 2025.

Jorge Nocedal and Stephen J. Wright. Numerical optimization. Springer, New York, NY, USA, 2e edition, 2006.

Matteo Pagliardini, Pierre Ablin, and David Grangier. The ademamix optimizer: Better, faster, older. *arXiv preprint arXiv:2409.03137*, 2024.

Razvan Pascanu and Yoshua Bengio. Revisiting natural gradient for deep networks. In Yoshua Bengio and Yann LeCun (eds.), *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*, 2014.

Thomas Pethick, Wanyun Xie, Kimon Antonakopoulos, Zhenyu Zhu, Antonio Silveti-Falls, and Volkan Cevher. Training deep learning models with norm-constrained lmos. *arXiv preprint arXiv:2502.07529*, 2025.

Boris T Polyak and Anatoli B Juditsky. Acceleration of stochastic approximation by averaging. *SIAM journal on control and optimization*, 30(4):838–855, 1992.

Aditya Ramesh, Prafulla Dhariwal, Alex Nichol, Casey Chu, and Mark Chen. Hierarchical text-conditional image generation with clip latents. *arXiv preprint arXiv:2204.06125*, 1(2):3, 2022.

Sashank J. Reddi, Satyen Kale, and Sanjiv Kumar. On the convergence of adam and beyond. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*, 2018.

Artem Riabinin, Egor Shulgin, Kaja Gruntkowska, and Peter Richtárik. Gluon: Making muon & scion great again!(bridging theory and practice of lmo-based optimizers for llms). *arXiv preprint arXiv:2505.13416*, 2025.

M. Riedmiller and H. Braun. A direct adaptive method for faster backpropagation learning: the rprop algorithm. In *IEEE International Conference on Neural Networks*, pp. 586–591 vol.1, 1993. doi: 10.1109/ICNN.1993.298623.

Nicolas Roux, Mark Schmidt, and Francis Bach. A stochastic gradient method with an exponential convergence _rate for finite training sets. *Advances in neural information processing systems*, 25, 2012.

Tom Schaul, Sixin Zhang, and Yann LeCun. No more pesky learning rates. In *International conference on machine learning*, pp. 343–351. PMLR, 2013.

Shai Shalev-Shwartz and Tong Zhang. Stochastic dual coordinate ascent methods for regularized loss minimization. *Journal of Machine Learning Research*, 14(1), 2013.

Noam Shazeer and Mitchell Stern. Adafactor: Adaptive learning rates with sublinear memory cost. In *International Conference on Machine Learning*, pp. 4596–4604. PMLR, 2018.

Chongjie Si, Debing Zhang, and Wei Shen. Adamuon: Adaptive muon optimizer. *arXiv preprint arXiv:2507.11005*, 2025.

Jianlin Su. Tiger: A tight-fisted optimizer. `https://github.com/bojone/tiger`, 2023.

Shohei Taniguchi, Keno Harada, Gouki Minegishi, Yuta Oshima, Seong Cheol Jeong, Go Nagahara, Tomoshi Iiyama, Masahiro Suzuki, Yusuke Iwasawa, and Yutaka Matsuo. Adopt: Modified adam can converge with any β_2 with the optimal rate. *Advances in Neural Information Processing Systems*, 37:72438–72474, 2024.

Yuanzhe Tao, Huizhuo Yuan, Xun Zhou, Yuan Cao, and Quanquan Gu. Towards simple and provable parameter-free adaptive gradient methods, 2024. URL `https://arxiv.org/abs/2412.19444`.

Kimi Team, Angang Du, Bofei Gao, Bowei Xing, Changjiu Jiang, Cheng Chen, Cheng Li, Chenjun Xiao, Chenzhuang Du, Chonghua Liao, et al. Kimi k1. 5: Scaling reinforcement learning with llms. *arXiv preprint arXiv:2501.12599*, 2025.

Ran Tian and Ankur P Parikh. Amos: An adam-style optimizer with adaptive weight decay towards model-oriented scale. *arXiv preprint arXiv:2210.11693*, 2022.

Tijmen Tieleman and Geoffrey Hinton. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural networks for machine learning*, 4(2):26–31, 2012.

Phuong Thi Tran et al. On the convergence proof of amsgrad and a new version. *IEEE Access*, 7: 61706–61716, 2019.

Quoc Tran-Dinh, Nhan H Pham, Dzung T Phan, and Lam M Nguyen. Hybrid stochastic gradient descent algorithms for stochastic nonconvex optimization. *arXiv preprint arXiv:1905.05920*, 2019.

Nikhil Vyas, Depen Morwani, Rosie Zhao, Itai Shapira, David Brandfonbrener, Lucas Janson, and Sham Kakade. Soap: Improving and stabilizing shampoo using adam. *arXiv preprint arXiv:2409.11321*, 2024.

Junjie Wang, Pan Zhou, Yiming Dong, Huan Li, Jia Li, Xun Zhou, Qicheng Lao, Cong Fang, and Zhouchen Lin. Conda: Column-normalized adam for training large language models faster. *arXiv preprint arXiv:2509.24218*, 2025a.

Mingze Wang, Jinbo Wang, Jiaqi Zhang, Wei Wang, Peng Pei, Xunliang Cai, Lei Wu, et al. Gradpower: Powering gradients for faster language model pre-training. *arXiv preprint arXiv:2505.24275*, 2025b.

Shaowen Wang, Anan Liu, Jian Xiao, Huan Liu, Yuekui Yang, Cong Xu, Qianqian Pu, Suncong Zheng, Wei Zhang, Di Wang, et al. Cadam: Confidence-based optimization for online learning. *arXiv preprint arXiv:2411.19647*, 2024.

Zhe Wang, Kaiyi Ji, Yi Zhou, Yingbin Liang, and Vahid Tarokh. Spiderboost and momentum: Faster variance reduction algorithms. *Advances in Neural Information Processing Systems*, 32, 2019.

Rachel Ward, Xiaoxia Wu, and Leon Bottou. Adagrad stepsizes: Sharp convergence over nonconvex landscapes. *Journal of Machine Learning Research*, 21(219):1–30, 2020.

Max Welling and Yee W Teh. Bayesian learning via stochastic gradient langevin dynamics. In *Proceedings of the 28th international conference on machine learning (ICML-11)*, pp. 681–688. Citeseer, 2011.

Kaiyue Wen, Zhiyuan Li, Jason Wang, David Hall, Percy Liang, and Tengyu Ma. Understanding warmup-stable-decay learning rates: A river valley loss landscape perspective. *arXiv preprint arXiv:2410.05192*, 2024.

Yue Wu, Zhiqing Sun, Huizhuo Yuan, Kaixuan Ji, Yiming Yang, and Quanquan Gu. Self-play preference optimization for language model alignment. *arXiv preprint arXiv:2405.00675*, 2024.

Tian Xie, Haoming Luo, Haoyu Tang, Yiwen Hu, Jason Klein Liu, Qingnan Ren, Yang Wang, Wayne Xin Zhao, Rui Yan, Bing Su, Chong Luo, and Baining Guo. Controlled llm training on spectral sphere, 2026. URL `https://arxiv.org/abs/2601.08393`.

Xingyu Xie, Pan Zhou, Huan Li, Zhouchen Lin, and Shuicheng Yan. Adan: Adaptive nesterov momentum algorithm for faster optimizing deep models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2024.

Zeke Xie, Xinrui Wang, Huishuai Zhang, Issei Sato, and Masashi Sugiyama. Adaptive inertia: Disentangling the effects of adaptive learning rate and momentum. In *International conference on machine learning*, pp. 24430–24459. PMLR, 2022.

Pan Xu, Jinghui Chen, Difan Zou, and Quanquan Gu. Global convergence of langevin dynamics based algorithms for nonconvex optimization. *Advances in Neural Information Processing Systems*, 31, 2018.

Greg Yang, Edward J Hu, Igor Babuschkin, Szymon Sidor, Xiaodong Liu, David Farhi, Nick Ryder, Jakub Pachocki, Weizhu Chen, and Jianfeng Gao. Tensor programs v: Tuning large neural networks via zero-shot hyperparameter transfer. *arXiv preprint arXiv:2203.03466*, 2022.

Zhewei Yao, Amir Gholami, Sheng Shen, Mustafa Mustafa, Kurt Keutzer, and Michael Mahoney. Adahessian: An adaptive second order optimizer for machine learning. In *proceedings of the AAAI conference on artificial intelligence*, volume 35, pp. 10665–10673, 2021.

Yida Yin, Zhiqiu Xu, Zhiyuan Li, Trevor Darrell, and Zhuang Liu. A coefficient makes svrg effective. In *The Thirteenth International Conference on Learning Representations*, 2025.

Yang You, Igor Gitman, and Boris Ginsburg. Large batch training of convolutional networks. *arXiv preprint arXiv:1708.03888*, 2017.

Yang You, Jing Li, Sashank J. Reddi, Jonathan Hseu, Sanjiv Kumar, Srinadh Bhojanapalli, Xiaodan Song, James Demmel, Kurt Keutzer, and Cho-Jui Hsieh. Large batch optimization for deep learning: Training BERT in 76 minutes. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*, 2020.

Huizhuo Yuan, Xiangru Lian, Ji Liu, and Yuren Zhou. Stochastic recursive momentum for policy gradient methods. *arXiv preprint arXiv:2003.04302*, 2020.

Huizhuo Yuan, Yifeng Liu, Shuang Wu, Quanquan Gu, et al. Mars: Unleashing the power of variance reduction for training large models. In *Forty-second International Conference on Machine Learning*, 2025.

Manzil Zaheer, Sashank Reddi, Devendra Sachan, Satyen Kale, and Sanjiv Kumar. Adaptive methods for nonconvex optimization. *Advances in neural information processing systems*, 31, 2018.

Matthew D Zeiler. Adadelta: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701*, 2012.

Aston Zhang, Zachary C Lipton, Mu Li, and Alexander J Smola. Dive into deep learning. *arXiv preprint arXiv:2106.11342*, pp. 505, 2021.

Jingzhao Zhang, Sai Praneeth Karimireddy, Andreas Veit, Seungyeon Kim, Sashank Reddi, Sanjiv Kumar, and Suvrit Sra. Why are adaptive methods good for attention models? *Advances in Neural Information Processing Systems*, 33:15383–15393, 2020.

Yushun Zhang, Congliang Chen, Ziniu Li, Tian Ding, Chenwei Wu, Yinyu Ye, Zhi-Quan Luo, and Ruoyu Sun. Adam-mini: Use fewer learning rates to gain more. *arXiv preprint arXiv:2406.16793*, 2024.

Jiawei Zhao, Zhenyu Zhang, Beidi Chen, Zhangyang Wang, Anima Anandkumar, and Yuandong Tian. Galore: Memory-efficient LLM training by gradient low-rank projection. In *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*, 2024a.

Rosie Zhao, Depen Morwani, David Brandfonbrener, Nikhil Vyas, and Sham Kakade. Deconstructing what makes a good optimizer for language models. *arXiv preprint arXiv:2407.07972*, 2024b.

Beitong Zhou, Jun Liu, Weigao Sun, Ruijuan Chen, Claire J Tomlin, and Ye Yuan. pbsgd: Powered stochastic gradient descent methods for accelerated non-convex optimization. In *IJCAI*, pp. 3258–3266, 2020a.

Dongruo Zhou, Pan Xu, and Quanquan Gu. Stochastic nested variance reduction for nonconvex optimization. *Journal of machine learning research*, 21(103):1–63, 2020b.

Juntang Zhuang, Tommy Tang, Yifan Ding, Sekhar C Tatikonda, Nicha Dvornek, Xenophon Papademetris, and James Duncan. Adabelief optimizer: Adapting stepsizes by the belief in observed gradients. *Advances in neural information processing systems*, 33:18795–18806, 2020.