Spectral Optimizers For Deep Learning: Muon, Scion and so on

Tony Silveti-Falls

Based on several joint works with:


Thomas Pethick


Wanyun Xie


Mete Erdogan


Zhenyu Zhu


Kimon Antonakopoulos


Volkan Cevher


Rustem Islamov


Roman Macháček


Eduard Gorbunov


Aurelien Lucchi


Leena Chennuru Vankadara

Lund, Sweden - ELLIIT Focus Period 2026

Training neural networks amounts to solving stochastic optimization problems of the form

$$\min_{x \in \mathcal{X}} f(x) := \mathbb{E}_{\xi}[f(x, \xi)].$$

Template Optimization Problem

Training neural networks amounts to solving stochastic optimization problems of the form

$$\min_{x \in \mathcal{X}} f(x) := \mathbb{E}_{\xi}[f(x, \xi)].$$

We will assume:

- $\mathcal{X}$ is either $\mathbb{R}^d$ (unconstrained) or $\mathcal{D}$ (constrained), with

$$\mathcal{D} := \{x \colon \|x\| \leq \rho\}.$$

Template Optimization Problem

Training neural networks amounts to solving stochastic optimization problems of the form

$$\min_{x \in \mathcal{X}} f(x) := \mathbb{E}_{\xi}[f(x, \xi)].$$

We will assume:

- $\mathcal{X}$ is either $\mathbb{R}^d$ (unconstrained) or $\mathcal{D}$ (constrained), with

$$\mathcal{D} := \{x \colon \|x\| \leq \rho\}.$$

- $\mathbb{E}_{\xi}[f(\cdot, \xi)]$ is Lipschitz-smooth with respect to some norm.

Template Optimization Problem

Training neural networks amounts to solving stochastic optimization problems of the form

$$\min_{x \in \mathcal{X}} f(x) := \mathbb{E}_\xi[f(x, \xi)].$$

We will assume:

- $\mathcal{X}$ is either $\mathbb{R}^d$ (unconstrained) or $\mathcal{D}$ (constrained), with

$$\mathcal{D} := \{x \colon \|x\| \leq \rho\}.$$

- $\mathbb{E}_\xi[f(\cdot, \xi)]$ is Lipschitz-smooth with respect to some norm.
- We have access to a stochastic first-order oracle $\nabla f(\cdot, \xi)$ which is unbiased

$$\mathbb{E}_\xi[\nabla f(\cdot, \xi)] = \nabla f(\cdot)$$

and has bounded variance

$$\mathbb{E}_\xi[\|\nabla f(\cdot, \xi) - \nabla f(\cdot)\|_2^2] \leq \sigma_0^2.$$

What has been done so far?

What has been done so far? (that I can discuss in <5 minutes...)

Stochastic Gradient Descent (SGD):

Input: $x^0 \in \mathcal{X}$, step sizes $\{\gamma_k\}$, horizon $n \in \mathbb{N}^*$
for $k = 0, 1, \ldots, n-1$ **do**
  Sample ξ_k
  $g^k = \nabla f(x^k, \xi_k)$
  $x^{k+1} = x^k - \gamma_k g^k$
Output: x^n

- SGD uses a Euclidean geometry:

$$x^{k+1} = \underset{x \in \mathbb{R}^d}{\operatorname{argmin}} \langle g^k, x - x^k \rangle + \frac{1}{2\gamma_k} \|x - x^k\|_2^2$$

Stochastic Gradient Descent (SGD):

Input: $x^0 \in \mathcal{X}$, step sizes $\{\gamma_k\}$, horizon $n \in \mathbb{N}^*$
for $k = 0, 1, \ldots, n-1$ **do**
  Sample ξ_k
  $g^k = \nabla f(x^k, \xi_k)$
  $x^{k+1} = x^k - \gamma_k g^k$
Output: x^n

- SGD uses a Euclidean geometry:

$$x^{k+1} = \operatorname*{argmin}_{x \in \mathbb{R}^d} \langle g^k, x - x^k \rangle + \frac{1}{2\gamma_k} \|x - x^k\|_2^2$$

Adam [Kingma, 2014]:

Input: $x^0 \in \mathcal{X}$, step size γ, $\epsilon > 0$, momentum β_1, β_2, horizon $n \in \mathbb{N}^*$

for $k = 0, 1, \ldots, n-1$ **do**

Sample ξ_k

$g^k = \nabla f(x^k, \xi_k)$

$m^k = \beta_1 m^{k-1} + (1-\beta_1) g^k$

$v^k = \beta_2 v^{k-1} + (1-\beta_2)(g^k)^2$

$\hat{m}^k = \frac{m^k}{1-\beta_1^k}$

$\hat{v}^k = \frac{v^k}{1-\beta_2^k}$

$x^{k+1} = x^k - \frac{\gamma}{\sqrt{\hat{v}^k}+\epsilon} \odot \hat{m}^k$

Output: x^n

- Adam uses a Mahalanobis geometry:

$$x^{k+1} = \operatorname*{argmin}_{x \in \mathbb{R}^d} \langle \hat{m}^k, x - x^k \rangle + \frac{1}{2\gamma} \|x - x^k\|_{2,H_k}^2$$

with $\|x\|_{2,H_k} := \sqrt{\langle x, H_k x \rangle}$ and $H_k \approx \operatorname{diag}(\hat{v}^k + \varepsilon^2)$

Adam [Kingma, 2014]:

Input: $x^0 \in \mathcal{X}$, step size γ, $\epsilon > 0$, momentum β_1, β_2, horizon $n \in \mathbb{N}^*$

for $k = 0, 1, \ldots, n-1$ **do**

Sample ξ_k

$g^k = \nabla f(x^k, \xi_k)$

$m^k = \beta_1 m^{k-1} + (1-\beta_1) g^k$

$v^k = \beta_2 v^{k-1} + (1-\beta_2)(g^k)^2$

$\hat{m}^k = \frac{m^k}{1-\beta_1^k}$

$\hat{v}^k = \frac{v^k}{1-\beta_2^k}$

$x^{k+1} = x^k - \frac{\gamma}{\sqrt{\hat{v}^k}+\epsilon} \odot \hat{m}^k$

Output: x^n

- Adam uses a Mahalanobis geometry:

$$x^{k+1} = \operatorname*{argmin}_{x \in \mathbb{R}^d} \langle \hat{m}^k, x - x^k \rangle + \frac{1}{2\gamma} \|x - x^k\|_{2,H_k}^2$$

with $\|x\|_{2,H_k} := \sqrt{\langle x, H_k x \rangle}$ and $H_k \approx \operatorname{diag}(\hat{v}^k + \varepsilon^2)$

Adam [Kingma, 2014]:

Input: $x^0 \in \mathcal{X}$, step size γ, $\epsilon > 0$, momentum β_1, β_2, horizon $n \in \mathbb{N}^*$

for $k = 0, 1, \ldots, n-1$ **do**

Sample ξ_k

$g^k = \nabla f(x^k, \xi_k)$

$m^k = \beta_1 m^{k-1} + (1-\beta_1) g^k$

$v^k = \beta_2 v^{k-1} + (1-\beta_2)(g^k)^2$

$\hat{m}^k = \frac{m^k}{1-\beta_1^k}$

$\hat{v}^k = \frac{v^k}{1-\beta_2^k}$

$x^{k+1} = x^k - \frac{\gamma}{\sqrt{\hat{v}^k + \epsilon}} \odot \hat{m}^k$

Output: x^n

- Adam uses a Mahalanobis geometry:

$$x^{k+1} = \operatorname*{argmin}_{x \in \mathbb{R}^d} \langle \hat{m}^k, x - x^k \rangle + \frac{1}{2\gamma} \|x - x^k\|_{2,H_k}^2$$

with $\|x\|_{2,H_k} := \sqrt{\langle x, H_k x \rangle}$ and $H_k \approx \operatorname{diag}(\hat{v}^k + \varepsilon^2)$

- Adagrad, RMSProp, Adam, AdamW, etc, all adapt using a Mahalanobis norm.

Adam [Kingma, 2014]:

Input: $x^0 \in \mathcal{X}$, step size γ, $\epsilon > 0$, momentum β_1, β_2, horizon $n \in \mathbb{N}^*$

for $k = 0, 1, \ldots, n-1$ **do**

Sample ξ_k

$g^k = \nabla f(x^k, \xi_k)$

$m^k = \beta_1 m^{k-1} + (1-\beta_1) g^k$

$v^k = \beta_2 v^{k-1} + (1-\beta_2)(g^k)^2$

$\hat{m}^k = \frac{m^k}{1-\beta_1^k}$

$\hat{v}^k = \frac{v^k}{1-\beta_2^k}$

$x^{k+1} = x^k - \frac{\gamma}{\sqrt{\hat{v}^k + \epsilon}} \odot \hat{m}^k$

Output: x^n

- Adam uses a Mahalanobis geometry:

$$x^{k+1} = \operatorname*{argmin}_{x \in \mathbb{R}^d} \langle \hat{m}^k, x - x^k \rangle + \frac{1}{2\gamma} \|x - x^k\|_{2,H_k}^2$$

with $\|x\|_{2,H_k} := \sqrt{\langle x, H_k x \rangle}$ and $H_k \approx \operatorname{diag}(\hat{v}^k + \varepsilon^2)$

- Adagrad, RMSProp, Adam, AdamW, etc, all adapt using a Mahalanobis norm.
- tl;dr: coordinate-wise adaptive step size using 2nd moment + momentum.

1. Sample ξ_k and compute $g^k = \nabla f(x^k, \xi_k)$ (here, $x^k = [W_1^k, W_2^k, W_3^k]^T$).

What about Muon [Jordan et al., 2024]?

1. Sample ξ_k and compute $g^k = \nabla f(x^k, \xi_k)$ (here, $x^k = [W_1^k, W_2^k, W_3^k]^T$).
2. Compute dual feedback $d^k = (1 - \alpha_k)d^{k-1} + \alpha_k \nabla f(x^k, \xi_k)$.

What about Muon [Jordan et al., 2024]?

1. Sample ξ_k and compute $g^k = \nabla f(x^k, \xi_k)$ (here, $x^k = [W_1^k, W_2^k, W_3^k]^T$).
2. Compute dual feedback $d^k = (1 - \alpha_k)d^{k-1} + \alpha_k \nabla f(x^k, \xi_k)$.
3. Add Nesterov momentum: $\tilde{d}^k = (1 - \beta)d^k + \beta \nabla f(x^k, \xi_k)$.

What about Muon [Jordan et al., 2024]?

1. Sample ξ_k and compute $g^k = \nabla f(x^k, \xi_k)$ (here, $x^k = [W_1^k, W_2^k, W_3^k]^T$).
2. Compute dual feedback $d^k = (1-\alpha_k)d^{k-1} + \alpha_k \nabla f(x^k, \xi_k)$.
3. Add Nesterov momentum: $\tilde{d}^k = (1-\beta)d^k + \beta \nabla f(x^k, \xi_k)$.
4. Compute the update

$$\begin{bmatrix} W_1^{k+1} \\ W_2^{k+1} \\ W_3^{k+1} \end{bmatrix} = \begin{bmatrix} W_1^k \\ W_2^k \\ W_3^k \end{bmatrix} - \gamma_k \begin{bmatrix} \text{Adam} \\ \text{msign}(\tilde{d}_2^k) \\ \text{Adam} \end{bmatrix} \qquad \text{(Compute orthogonalization + Adam)}$$

where $\text{msign}(X) = UV^T, \quad X = U\Sigma V^T$.

1. Sample ξ_k and compute $g^k = \nabla f(x^k, \xi_k)$ (here, $x^k = [W_1^k, W_2^k, W_3^k]^T$).
2. Compute dual feedback $d^k = (1-\alpha_k)d^{k-1} + \alpha_k \nabla f(x^k, \xi_k)$.
3. Add Nesterov momentum: $\tilde{d}^k = (1-\beta)d^k + \beta \nabla f(x^k, \xi_k)$.
4. Compute the update

$$\begin{bmatrix} W_1^{k+1} \\ W_2^{k+1} \\ W_3^{k+1} \end{bmatrix} = \begin{bmatrix} W_1^k \\ W_2^k \\ W_3^k \end{bmatrix} - \gamma_k \begin{bmatrix} \text{Adam} \\ \text{msign}(\tilde{d}_2^k) \\ \text{Adam} \end{bmatrix} \qquad \text{(Compute orthogonalization + Adam)}$$

$$\text{(Optional)} \quad \begin{bmatrix} W_1^{k+1} \\ W_2^{k+1} \\ W_3^{k+1} \end{bmatrix} = \begin{bmatrix} W_1^k \\ W_2^k \\ W_3^k \end{bmatrix} - \gamma_k \begin{bmatrix} \rho_1 \, (\text{Adam}) \\ \rho_2 \left(\text{msign}(\tilde{d}_2^k)\right) \\ \rho_3 \, (\text{Adam}) \end{bmatrix} \qquad \text{(Layerwise learning rate).}$$

where $\text{msign}(X) = UV^T, \quad X = U\Sigma V^T$.

The update of SGD can be written

$$x^{k+1} = \underset{x \in \mathbb{R}^d}{\operatorname{argmin}} \langle g^k, x - x^k \rangle + \frac{1}{2\gamma_k} \|x - x^k\|_2^2.$$

The update of SGD can be written

$$x^{k+1} = \underset{x \in \mathbb{R}^d}{\operatorname{argmin}} \langle g^k, x - x^k \rangle + \frac{1}{2\gamma_k} \|x - x^k\|_2^2.$$

What if we change the norm?

The update of Steepest Descent can be written

$$x^{k+1} = \underset{x \in \mathbb{R}^d}{\operatorname{argmin}} \langle g^k, x - x^k \rangle + \frac{1}{2\gamma_k} \|x - x^k\|^2.$$

What if we change the norm?

The update of Steepest Descent can be written

$$x^{k+1} = \underset{x \in \mathbb{R}^d}{\operatorname{argmin}} \langle g^k, x - x^k \rangle + \frac{1}{2\gamma_k} \|x - x^k\|^2.$$

What if we change the norm?

Assume the update has the form $x - x^k = \alpha s$ with $\|s\| = 1$. Then:

$$\underset{\alpha \geq 0,\ \|s\|=1}{\operatorname{argmin}} \quad \alpha \langle g^k, s \rangle + \frac{\alpha^2}{2\gamma_k}$$

The update of Steepest Descent can be written

$$x^{k+1} = \underset{x \in \mathbb{R}^d}{\operatorname{argmin}} \langle g^k, x - x^k \rangle + \frac{1}{2\gamma_k} \|x - x^k\|^2.$$

What if we change the norm?

Assume the update has the form $x - x^k = \alpha\, s$ with $\|s\| = 1$. Then:

$$\underset{\alpha \geq 0,\ \|s\|=1}{\operatorname{argmin}} \ \alpha \langle g^k, s \rangle + \frac{\alpha^2}{2\gamma_k} \ = \ \underset{\alpha \geq 0}{\operatorname{argmin}} \ -\alpha \, \|g^k\|_* + \frac{\alpha^2}{2\gamma_k}$$

where $-\|g^k\|_* := \min_{\|s\| \leq 1} \langle g^k, s \rangle$ is the *dual norm* and the optimal $s^\star$ solves $\operatorname{argmin}_{\|s\| \leq 1} \langle g^k, s \rangle$.

The update of Steepest Descent can be written

$$x^{k+1} = \underset{x \in \mathbb{R}^d}{\operatorname{argmin}} \langle g^k, x - x^k \rangle + \frac{1}{2\gamma_k} \|x - x^k\|^2.$$

What if we change the norm?

Assume the update has the form $x - x^k = \alpha\, s$ with $\|s\| = 1$. Then:

$$\underset{\alpha \geq 0,\ \|s\|=1}{\operatorname{argmin}} \ \alpha \langle g^k, s \rangle + \frac{\alpha^2}{2\gamma_k} \ = \ \underset{\alpha \geq 0}{\operatorname{argmin}} \ -\alpha \, \|g^k\|_* + \frac{\alpha^2}{2\gamma_k}$$

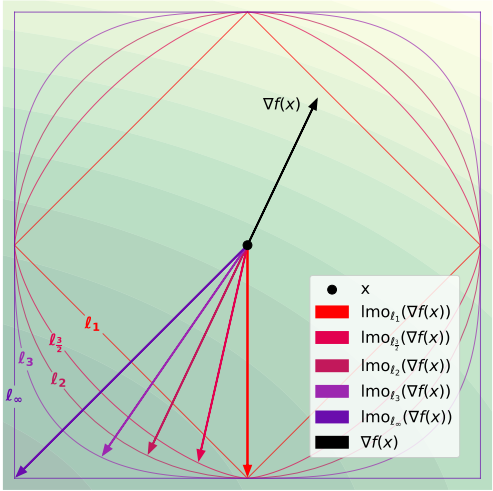
where $-\|g^k\|_* := \min_{\|s\| \leq 1} \langle g^k, s \rangle$ is the *dual norm* and the optimal $s^\star$ solves $\operatorname{argmin}_{\|s\| \leq 1} \langle g^k, s \rangle$.
Optimizing over α gives $\alpha^\star = \gamma_k \|g^k\|_*$

$$\boxed{x^{k+1} = x^k + \gamma_k \|g^k\|_* \, s^\star}$$

Given a norm $\|\cdot\|$, the associated *linear minimization oracle* (lmo) gives back a direction least aligned with its input,

$$\text{lmo}(g) \in \underset{\{s:\ \|s\| \leq 1\}}{\text{argmin}} \ \langle g, s \rangle$$

- The output of the lmo is always on the boundary of the ball.
- The lmo for the scaled ball is the scaled lmo for the unit ball.



Linear Minimization Oracles (lmo) for Norm Balls
If $\mathcal{D}$ is the unit-ball associated to a norm $\|\cdot\|$,
then $\mathrm{lmo}_{\mathcal{D}}(g) = -\partial\|g\|_*$ where $\|\cdot\|_*$ is the *dual norm*.

Norm	**Linear Minimization Oracle (lmo)**
ℓ_2: $\|\cdot\| = \|\cdot\|_2$	$\mathrm{lmo}(g) = -\frac{g}{\|g\|_2}$
Dual Norm	**Steepest Descent ($-\|g\|_* \mathrm{lmo}(g)$)**
$\|\cdot\|_* = \|\cdot\|_2$	$-\|g\|_2 \left(-\frac{g}{\|g\|_2}\right) = g$

Steepest Descent in ℓ^2-norm recovers gradient descent/SGD.

$$x^{k+1} = x^k - \gamma_k g^k$$

Linear Minimization Oracles (lmo) for Norm Balls
If $\mathcal{D}$ is the unit-ball associated to a norm $\|\cdot\|$,
then $\mathrm{lmo}_{\mathcal{D}}(g) = -\partial\|g\|_*$ where $\|\cdot\|_*$ is the *dual norm*.

Norm	**Linear Minimization Oracle (lmo)**
ℓ_∞: $\|\cdot\| = \|\cdot\|_\infty$	$\mathrm{lmo}(g) = -\mathrm{sign}(g)$
Dual Norm	**Steepest Descent ($-\|g\|_* \,\mathrm{lmo}(g)$)**
$\|\cdot\|_* = \|\cdot\|_1$	$-\|g\|_1\,(-\mathrm{sign}(g)) = \left(\sum_i |g_i|\right)\mathrm{sign}(g)$

Steepest Descent in ℓ^∞-norm recovers sign descent (up to step size).

$$x^{k+1} = x^k - \gamma_k \left(\sum_i |g_i^k|\right)\mathrm{sign}(g^k)$$

Examples of Linear Minimization Oracles

Linear Minimization Oracles (lmo) for Norm Balls
If $\mathcal{D}$ is the unit-ball associated to a norm $\|\cdot\|$,
then $\mathrm{lmo}_{\mathcal{D}}(g) = -\partial\|g\|_*$ where $\|\cdot\|_*$ is the *dual norm*.

Norm	**Linear Minimization Oracle (lmo)**
$\ell_2 \to \ell_2$: $\|\cdot\| = \|\cdot\|_{\mathrm{op}}$	$\mathrm{lmo}(g) = -UV^T$ where $g = U\Sigma V^T$ (reduced SVD)
Dual Norm	**Steepest Descent ($-\|g\|_* \,\mathrm{lmo}(g)$)**
$\|\cdot\|_* = \|\cdot\|_{\mathrm{Nuc}}$	$-\|g\|_{\mathrm{Nuc}}\left(-UV^T\right) = \left(\sum_i \sigma_i(g)\right)\left(UV^T\right)$

Steepest Descent in $\|\cdot\|_{\mathrm{op}}$ recovers spectral descent (up to step size).

$$x^{k+1} = x^k - \gamma_k \left(\sum_i \sigma_i(g^k)\right) \mathrm{msign}(g^k)$$

(we can compute this without SVD, using either Newton-Schulz or sketching.)

Instead of Steepest Descent

$$x^{k+1} = \operatorname*{argmin}_{x \in \mathbb{R}^d} \langle g^k, x - x^k \rangle + \frac{1}{2\gamma_k} \|x - x^k\|^2$$

which scales the lmo by $\|g^k\|_*$, we can directly use

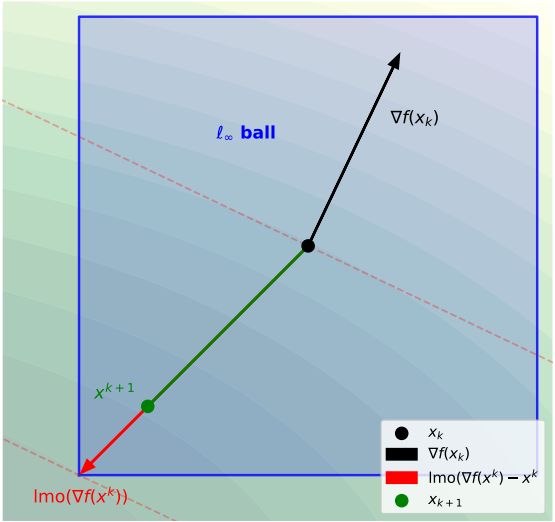
$$\begin{aligned} x^{k+1} &= \operatorname*{argmin}_{x \in \mathbb{R}^d} \langle g^k, x - x^k \rangle + \iota_{\gamma_k \mathcal{D}}(x - x^k) \\ &= \operatorname*{argmin}_{\|x - x_k\| \leq \gamma_k} \langle g^k, x - x^k \rangle \end{aligned}$$

to get

$$x^{k+1} = x^k + \gamma_k \mathrm{lmo}_{\mathcal{D}}(g^k).$$

Take-home Message

In deep learning, this ensures *feature learning*; steepest descent alone does not ensure the spectral conditions for feature learning will hold.

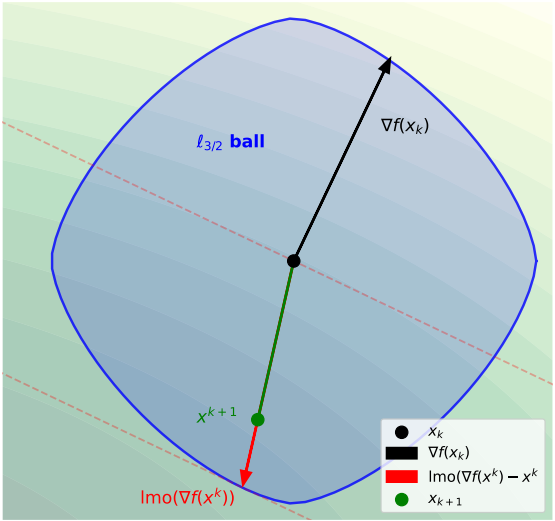


The conditional gradient algorithm (also known as the Frank-Wolfe algorithm [Frank and Wolfe, 1956]) solves constrained optimization problems:

$$\min_{x \in \mathcal{D}} f(x)$$

Conditional Gradient (CG):

Input: $x_0 \in \mathcal{D}$, step sizes $\{\gamma_k\}$ where $\gamma_k \in [0, 1]$, horizon $n \in \mathbb{N}^*$
for $k = 0, 1, \ldots, n-1$ **do**
  $s^k = \text{lmo}(\nabla f(x^k))$
  $v^k = s^k - x^k$
  $x^{k+1} = x_k + \gamma_k v^k$
Output: x^n



The conditional gradient algorithm (also known as the Frank-Wolfe algorithm [Frank and Wolfe, 1956]) solves constrained optimization problems:

$$\min_{x \in \mathcal{D}} f(x)$$

Conditional Gradient (CG):

Input: $x_0 \in \mathcal{D}$, step sizes $\{\gamma_k\}$ where $\gamma_k \in [0, 1]$, horizon $n \in \mathbb{N}^*$
for $k = 0, 1, \ldots, n-1$ **do**
  $s^k = \mathrm{lmo}(\nabla f(x^k))$
  $v^k = s^k - x^k$
  $x^{k+1} = x_k + \gamma_k v^k$
Output: x^n

(Unconstrained) Stochastic Conditional Gradient (uSCG/SCG) [Pethick et al., 2025a]:

Input: $x^1 \in \mathcal{D}$, step sizes $\{\gamma_k\}$, momentum $\{\alpha_k\}$, horizon $K \in \mathbb{N}$

Initialize $d^0 = 0$

for $k = 1, 2, \ldots K-1$ **do**

Sample ξ_k

$g^k = \nabla f(x^k, \xi_k)$

$d^k = (1-\alpha_k)d^{k-1} + \alpha_k g^k$

$s^k = \mathrm{lmo}(d^k)$

$$v^k = \begin{cases} s^k & \text{uSCG} \\ s^k - x^k & \text{SCG} \end{cases}$$

$x^{k+1} = x^k + \gamma_k v^k$

(Unconstrained) Stochastic Conditional Gradient (uSCG/SCG) [Pethick et al., 2025a]:

Input: $x^1 \in \mathcal{D}$, step sizes $\{\gamma_k\}$, momentum $\{\alpha_k\}$, horizon $K \in \mathbb{N}$

Initialize $d^0 = 0$

for $k = 1, 2, \ldots K-1$ **do**

Sample ξ_k

$g^k = \nabla f(x^k, \xi_k)$

$d^k = (1-\alpha_k)d^{k-1} + \alpha_k g^k$

$s^k = \text{lmo}(d^k)$

$$v^k = \begin{cases} s^k & \text{uSCG} \\ s^k - x^k & \text{SCG} \end{cases}$$

$x^{k+1} = x^k + \gamma_k v^k$

- Momentum reduces variance in stochastic setting.

(Unconstrained) Stochastic Conditional Gradient (uSCG/SCG) [Pethick et al., 2025a]:

Input: $x^1 \in \mathcal{D}$, step sizes $\{\gamma_k\}$, momentum $\{\alpha_k\}$, horizon $K \in \mathbb{N}$

Initialize $d^0 = 0$

for $k = 1, 2, \ldots K-1$ **do**

Sample ξ_k

$g^k = \nabla f(x^k, \xi_k)$

$d^k = (1-\alpha_k)d^{k-1} + \alpha_k g^k$

$s^k = \mathrm{lmo}(d^k)$

$$v^k = \begin{cases} s^k & \text{uSCG} \\ s^k - x^k & \text{SCG} \end{cases}$$

$x^{k+1} = x^k + \gamma_k v^k$

- Momentum reduces variance in stochastic setting.
- The direction s^k has fixed norm of our choosing.

(Unconstrained) Stochastic Conditional Gradient (uSCG/SCG) [Pethick et al., 2025a]:

Input: $x^1 \in \mathcal{D}$, step sizes $\{\gamma_k\}$, momentum $\{\alpha_k\}$, horizon $K \in \mathbb{N}$

Initialize $d^0 = 0$

for $k = 1, 2, \ldots K-1$ **do**

Sample ξ_k

$g^k = \nabla f(x^k, \xi_k)$

$d^k = (1-\alpha_k)d^{k-1} + \alpha_k g^k$

$s^k = \mathrm{lmo}(d^k)$

$$v^k = \begin{cases} s^k & \text{uSCG} \\ s^k - x^k & \text{SCG} \end{cases}$$

$x^{k+1} = x^k + \gamma_k v^k$

- Momentum reduces variance in stochastic setting.
- The direction s^k has fixed norm of our choosing.
- SCG is "just" uSCG with weight decay.

(Unconstrained) Stochastic Conditional Gradient (uSCG/SCG) [Pethick et al., 2025a]:

Input: $x^1 \in \mathcal{D}$, step sizes $\{\gamma_k\}$, momentum $\{\alpha_k\}$, horizon $K \in \mathbb{N}$
Initialize $d^0 = 0$
for $k = 1, 2, \ldots K-1$ **do**

Sample ξ_k
$g^k = \nabla f(x^k, \xi_k)$
$d^k = (1-\alpha_k)d^{k-1} + \alpha_k g^k$
$s^k = \text{lmo}(d^k)$
$v^k = \begin{cases} s^k & \text{uSCG} \\ s^k - x^k & \text{SCG} \end{cases}$
$x^{k+1} = x^k + \gamma_k v^k$

- Momentum reduces variance in stochastic setting.
- The direction s^k has fixed norm of our choosing.
- SCG is "just" uSCG with weight decay.
- uSCG solves the problem $\min_{x \in \mathbb{R}^d} f(x)$ while SCG solves the problem $\min_{x \in \mathcal{D}} f(x)$ where $\mathcal{D}$ is a norm ball.

(Unconstrained) Stochastic Conditional Gradient (uSCG/SCG) [Pethick et al., 2025a]:

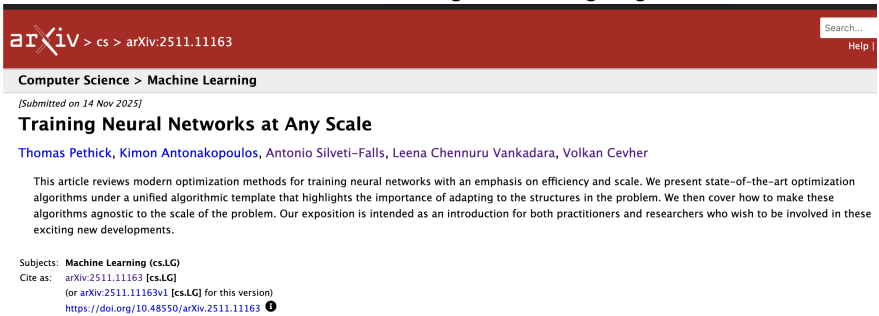
Input: $x^1 \in \mathcal{D}$, step sizes $\{\gamma_k\}$, momentum $\{\alpha_k\}$, horizon $K \in \mathbb{N}$
Initialize $d^0 = 0$
for $k = 1, 2, \ldots K-1$ **do**

Sample ξ_k
$g^k = \nabla f(x^k, \xi_k)$
$d^k = (1-\alpha_k)d^{k-1} + \alpha_k g^k$
$s^k = \text{lmo}(d^k)$
$v^k = \begin{cases} s^k & \text{uSCG} \\ s^k - x^k & \text{SCG} \end{cases}$
$x^{k+1} = x^k + \gamma_k v^k$

- Momentum reduces variance in stochastic setting.
- The direction s^k has fixed norm of our choosing.
- SCG is "just" uSCG with weight decay.
- uSCG solves the problem $\min_{x \in \mathbb{R}^d} f(x)$ while SCG solves the problem $\min_{x \in \mathcal{D}} f(x)$ where $\mathcal{D}$ is a norm ball.

This is a broad framework that covers many other algorithms through choice of norm + momentum (or other dual feedback)

arXiv:2511.11163, also at Signal Processing Magazine



arXiv > cs > arXiv:2511.11163

Search... Help |

Computer Science > Machine Learning

[Submitted on 14 Nov 2025]

Training Neural Networks at Any Scale

Thomas Pethick, Kimon Antonakopoulos, Antonio Silveti-Falls, Leena Chennuru Vankadara, Volkan Cevher

This article reviews modern optimization methods for training neural networks with an emphasis on efficiency and scale. We present state-of-the-art optimization algorithms under a unified algorithmic template that highlights the importance of adapting to the structures in the problem. We then cover how to make these algorithms agnostic to the scale of the problem. Our exposition is intended as an introduction for both practitioners and researchers who wish to be involved in these exciting new developments.

Subjects: **Machine Learning (cs.LG)**
Cite as: arXiv:2511.11163 **[cs.LG]**
(or arXiv:2511.11163v1 **[cs.LG]** for this version)
https://doi.org/10.48550/arXiv.2511.11163

Weight Decay and SCG

Deep learning community argues that Weight Decay should not simply be seen as Tikhonov/L^2 regularization ([Loshchilov and Hutter, 2017]).

$$\text{GD with weight decay (decoupled):} \quad x^{k+1} = (1 - \gamma\lambda)x^k - \gamma\nabla f(x^k)$$

$$\text{GD on Tikhonov problem (coupled):} \quad x^{k+1} = x^k - \gamma\nabla\left(f(x^k) + \lambda\|x^k\|_2^2/2\right)$$

However, these really are equivalent up to a rescaling/renaming of constants (but decoupled is known to work "better" with Adam).

Weight Decay and SCG

Deep learning community argues that Weight Decay should not simply be seen as Tikhonov/L^2 regularization ([Loshchilov and Hutter, 2017]).

$$\text{GD with weight decay (decoupled):} \quad x^{k+1} = (1-\gamma\lambda)x^k - \gamma\nabla f(x^k)$$
$$\text{GD on Tikhonov problem (coupled):} \quad x^{k+1} = x^k - \gamma\nabla\left(f(x^k) + \lambda\|x^k\|_2^2/2\right)$$

However, these really are equivalent up to a rescaling/renaming of constants (but decoupled is known to work "better" with Adam).

In a noneuclidean setting, this point is *critical* because the lmo is nonlinear.

$$\text{uSCG + weight decay} \to \text{SCG:} \quad x^{k+1} = (1-\lambda)x^k + \gamma\,\mathsf{lmo}(\nabla f(x^k))$$
$$= (1-\lambda)x^k + \lambda\,\underset{\gamma/\lambda}{\mathsf{lmo}}(\nabla f(x^k))$$
$$\text{uSCG on Tikhonov problem:} \quad x^{k+1} = x^k + \gamma\,\mathsf{lmo}(\nabla f(x^k) + \lambda x^k)$$

Deep learning community argues that Weight Decay should not simply be seen as Tikhonov/L^2 regularization ([Loshchilov and Hutter, 2017]).

$$\text{GD with weight decay (decoupled):} \quad x^{k+1} = (1-\gamma\lambda)x^k - \gamma\nabla f(x^k)$$
$$\text{GD on Tikhonov problem (coupled):} \quad x^{k+1} = x^k - \gamma\nabla\left(f(x^k) + \lambda\|x^k\|_2^2/2\right)$$

However, these really are equivalent up to a rescaling/renaming of constants (but decoupled is known to work "better" with Adam).

In a noneuclidean setting, this point is *critical* because the lmo is nonlinear.

$$\text{uSCG + weight decay} \to \text{SCG:} \quad x^{k+1} = (1-\lambda)x^k + \gamma\,\mathrm{lmo}(\nabla f(x^k)) = (1-\lambda)x^k + \lambda\,\mathop{\mathrm{lmo}}_{\gamma/\lambda}(\nabla f(x^k))$$
$$\text{uSCG on Tikhonov problem:} \quad x^{k+1} = x^k + \gamma\,\mathrm{lmo}(\nabla f(x^k) + \lambda x^k)$$

Different Interpretation of Weight Decay

Weight Decay in this context transforms your unconstrained optimizer into a constrained optimizer, with implicit radii that are dictated by the chosen combination of step size γ and Weight Decay λ!

Averaged **LMO** direction**N**al **D**escent (ALMOND) [Pethick et al., 2025a]:

Input: $x^0 \in \mathcal{D}$, step sizes $\{\gamma_k\}$, momentum $\{\alpha_k\}$, horizon $n \in \mathbb{N}$
Initialize $d^0 = 0$
for $k = 0, 1, 2, \ldots n-1$ **do**
  $g^k = \nabla f(x^k, \xi_k)$
  $d^k = (1-\alpha_k)d^{k-1} + \alpha_k \mathrm{lmo}(g^k)$
  $x^{k+1} = x^k + \gamma_k d^k$

- The ordering between averaging and taking the LMO matters!
- Not competitive empirically.
- Theoretically, can only show convergence to a noise dominated region.

We can only define our algorithm with respect to *a single norm*.

We can only define our algorithm with respect to *a single norm*.
In the case where $x = [W_1, \ldots, W_L]$ and we want to assign a norm $\|\cdot\|_{\{\ell\}}$ to each W_ℓ for $\ell \in [L]$, we can take the *max*-norm,

$$\|x\| := \max\left\{\|W_1\|_{\{1\}}, \ldots, \|W_L\|_{\{L\}}\right\}$$

We can only define our algorithm with respect to *a single norm*.
In the case where $x = [W_1, \ldots, W_L]$ and we want to assign a norm $\|\cdot\|_{\{\ell\}}$ to each W_ℓ for $\ell \in [L]$, we can take the *max*-norm,

$$\|x\| := \max\left\{\|W_1\|_{\{1\}}, \ldots, \|W_L\|_{\{L\}}\right\}$$

Then, $\mathrm{lmo}(g)$ with respect to this norm is *separable* across the parameters g_ℓ:

$$\mathrm{lmo}(g) = \mathrm{lmo}\left(\begin{bmatrix} g_1 \\ \vdots \\ g_L \end{bmatrix}\right) = \begin{bmatrix} \mathrm{lmo}_{\{1\}}(g_1) \\ \vdots \\ \mathrm{lmo}_{\{L\}}(g_L) \end{bmatrix}$$

with each $\mathrm{lmo}_{\{\ell\}}$ corresponding to the lmo over the ball induced by the norm $\|\cdot\|_{\{\ell\}}$.

Picking a Norm and Initializations

- If we can specify a norm $\|\cdot\|_{\alpha_\ell}$ for the input space and a norm $\|\cdot\|_{\beta_\ell}$ for the output spaces of each layer of our network, then this induces an operator norm for each layer.

Picking a Norm and Initializations

- If we can specify a norm $\|\cdot\|_{\alpha_\ell}$ for the input space and a norm $\|\cdot\|_{\beta_\ell}$ for the output spaces of each layer of our network, then this induces an operator norm for each layer.
- We can specify a norm for the whole set of parameters by taking

$$\|x\| = \max_{\ell \in [L]} \left\{ \|W_\ell\|_{\alpha_\ell \to \beta_\ell} \right\}$$

Picking a Norm and Initializations

- If we can specify a norm $\|\cdot\|_{\alpha_\ell}$ for the input space and a norm $\|\cdot\|_{\beta_\ell}$ for the output spaces of each layer of our network, then this induces an operator norm for each layer.
- We can specify a norm for the whole set of parameters by taking

$$\|x\| = \max_{\ell \in [L]} \left\{ \|W_\ell\|_{\alpha_\ell \to \beta_\ell} \right\}$$

- Spectral conditions for feature learning [Yang et al., 2023] suggest taking the RMS norm on the input and output spaces of intermediary layers.
  $\to$ leads to a scaled $\ell^2 \to \ell^2$ operator norm $\|\cdot\|_{\text{op}}$ on weight matrices

$$\|W\|_{\text{RMS}\to\text{RMS}} = \sqrt{\frac{d_{\text{in}}}{d_{\text{out}}}} \|W\|_{\text{op}}.$$

The lmo associated to the ball for this norm is given by the scaled matrix sign

$$\text{lmo}(g) = -\sqrt{\frac{d_{\text{out}}}{d_{\text{in}}}} UV^T, \quad g = U\Sigma V^T.$$

- If we can specify a norm $\|\cdot\|_{\alpha_\ell}$ for the input space and a norm $\|\cdot\|_{\beta_\ell}$ for the output spaces of each layer of our network, then this induces an operator norm for each layer.
- We can specify a norm for the whole set of parameters by taking

$$\|x\| = \max_{\ell \in [L]} \left\{ \|W_\ell\|_{\alpha_\ell \to \beta_\ell} \right\}$$

- Spectral conditions for feature learning [Yang et al., 2023] suggest taking the RMS norm on the input and output spaces of intermediary layers.
  $\to$ leads to a scaled $\ell^2 \to \ell^2$ operator norm $\|\cdot\|_{\text{op}}$ on weight matrices

$$\|W\|_{\text{RMS}\to\text{RMS}} = \sqrt{\frac{d_{\text{in}}}{d_{\text{out}}}} \|W\|_{\text{op}}.$$

The lmo associated to the ball for this norm is given by the scaled matrix sign

$$\text{lmo}(g) = -\sqrt{\frac{d_{\text{out}}}{d_{\text{in}}}} UV^T, \quad g = U\Sigma V^T.$$

The first and final layers require more thought!

Norms for input layers

The operator norm chosen for the initial layer differs from the intermediary layers, depending on the task (NLP, images, etc).

vs

Large Language Models (LLMs), such as GPT-3 and GPT-4, util ize a process called tokenization. Tokenization involves br eaking down text into smaller units, known as tokens, which the model can process and understand. These tokens can rang e from individual characters to entire words or even larger chunks, depending on the model. For GPT-3 and GPT-4, a Byte Pair Encoding (BPE) tokenizer is used. BPE is a subword tok enization technique that allows the model to dynamically bu ild a vocabulary during training, efficiently representing common words and word fragments. Although the core tokeniza tion process remains similar across different versions of t hese models, the specific implementation can vary based on the model's architecture and training objectives.

For image domains, we use the $\text{RMS} \to \text{RMS}$ norm.

For text domains, we use a scaled $1 \to \infty$ operator norm.

We refer to the instantiation of uSCG and SCG using operator norms as Unconstrained Scion and Scion respectively:

Stochastic **C**onditional grad**I**ent with **O**perator **N**orms

Scion

Example: One step of Scion with (Sign→Spectral→Sign)

1. Sample ξ_k and compute $g^k = \nabla f(x^k, \xi_k)$ (here, $x^k = [W_1^k, W_2^k, W_3^k]^T$).
2. Compute dual feedback $d^k = (1-\alpha_k)d^{k-1} + \alpha_k \nabla f(x^k, \xi_k)$.
3. Compute the update

$$\begin{bmatrix} W_1^{k+1} \\ W_2^{k+1} \\ W_3^{k+1} \end{bmatrix} = (1-\gamma_k) \begin{bmatrix} W_1^k \\ W_2^k \\ W_3^k \end{bmatrix} + \gamma_k \operatorname{lmo} \left(\begin{bmatrix} d_1^k \\ d_2^k \\ d_3^k \end{bmatrix} \right) \qquad \text{(Definition of alg)}$$

Example: One step of Scion with (Sign→Spectral→Sign)

1. Sample ξ_k and compute $g^k = \nabla f(x^k, \xi_k)$ (here, $x^k = [W_1^k, W_2^k, W_3^k]^T$).
2. Compute dual feedback $d^k = (1-\alpha_k)d^{k-1} + \alpha_k \nabla f(x^k, \xi_k)$.
3. Compute the update

$$\begin{bmatrix} W_1^{k+1} \\ W_2^{k+1} \\ W_3^{k+1} \end{bmatrix} = (1-\gamma_k) \begin{bmatrix} W_1^k \\ W_2^k \\ W_3^k \end{bmatrix} + \gamma_k \operatorname{lmo} \left(\begin{bmatrix} d_1^k \\ d_2^k \\ d_3^k \end{bmatrix} \right) \qquad \text{(Definition of alg)}$$

$$\begin{bmatrix} W_1^{k+1} \\ W_2^{k+1} \\ W_3^{k+1} \end{bmatrix} = (1-\gamma_k) \begin{bmatrix} W_1^k \\ W_2^k \\ W_3^k \end{bmatrix} + \gamma_k \begin{bmatrix} \operatorname{lmo}_{\{1\}}(d_1^k) \\ \operatorname{lmo}_{\{2\}}(d_2^k) \\ \operatorname{lmo}_{\{3\}}(d_3^k) \end{bmatrix} \qquad \text{(lmo is separable)}$$

Example: One step of Scion with (Sign→Spectral→Sign)

1. Sample ξ_k and compute $g^k = \nabla f(x^k, \xi_k)$ (here, $x^k = [W_1^k, W_2^k, W_3^k]^T$).
2. Compute dual feedback $d^k = (1-\alpha_k)d^{k-1} + \alpha_k \nabla f(x^k, \xi_k)$.
3. Compute the update

$$\begin{bmatrix} W_1^{k+1} \\ W_2^{k+1} \\ W_3^{k+1} \end{bmatrix} = (1-\gamma_k) \begin{bmatrix} W_1^k \\ W_2^k \\ W_3^k \end{bmatrix} + \gamma_k \operatorname{lmo} \left(\begin{bmatrix} d_1^k \\ d_2^k \\ d_3^k \end{bmatrix} \right) \qquad \text{(Definition of alg)}$$

$$\begin{bmatrix} W_1^{k+1} \\ W_2^{k+1} \\ W_3^{k+1} \end{bmatrix} = (1-\gamma_k) \begin{bmatrix} W_1^k \\ W_2^k \\ W_3^k \end{bmatrix} + \gamma_k \begin{bmatrix} \operatorname{lmo}_{\{1\}}(d_1^k) \\ \operatorname{lmo}_{\{2\}}(d_2^k) \\ \operatorname{lmo}_{\{3\}}(d_3^k) \end{bmatrix} \qquad \text{(lmo is separable)}$$

$$\begin{bmatrix} W_1^{k+1} \\ W_2^{k+1} \\ W_3^{k+1} \end{bmatrix} = (1-\gamma_k) \begin{bmatrix} W_1^k \\ W_2^k \\ W_3^k \end{bmatrix} - \gamma_k \begin{bmatrix} \operatorname{sign}(d_1^k) \\ \operatorname{msign}(d_2^k) \\ \frac{1}{m}\operatorname{sign}(d_3^k) \end{bmatrix} \qquad \text{(Compute lmos)}$$

Example: One step of Scion with (Sign→Spectral→Sign)

1. Sample ξ_k and compute $g^k = \nabla f(x^k, \xi_k)$ (here, $x^k = [W_1^k, W_2^k, W_3^k]^T$).
2. Compute dual feedback $d^k = (1-\alpha_k)d^{k-1} + \alpha_k \nabla f(x^k, \xi_k)$.
3. Compute the update

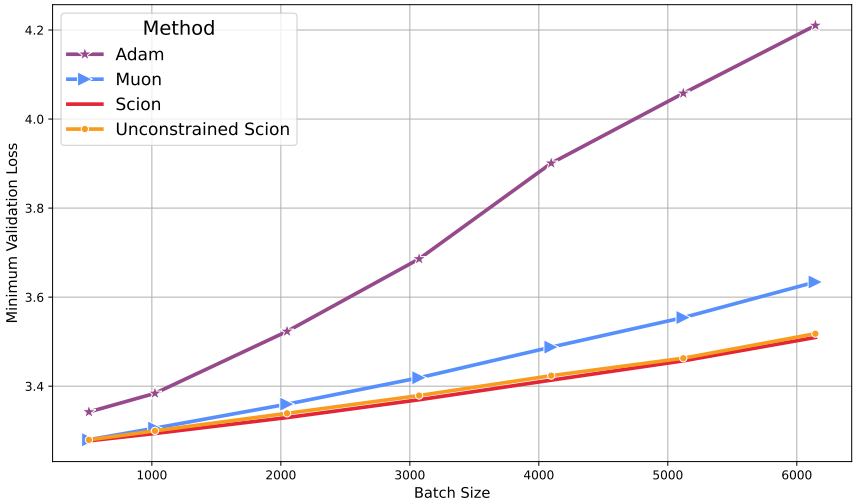
$$\begin{bmatrix} W_1^{k+1} \\ W_2^{k+1} \\ W_3^{k+1} \end{bmatrix} = (1-\gamma_k) \begin{bmatrix} W_1^k \\ W_2^k \\ W_3^k \end{bmatrix} + \gamma_k \operatorname{lmo} \left(\begin{bmatrix} d_1^k \\ d_2^k \\ d_3^k \end{bmatrix} \right) \qquad \text{(Definition of alg)}$$

$$\begin{bmatrix} W_1^{k+1} \\ W_2^{k+1} \\ W_3^{k+1} \end{bmatrix} = (1-\gamma_k) \begin{bmatrix} W_1^k \\ W_2^k \\ W_3^k \end{bmatrix} + \gamma_k \begin{bmatrix} \operatorname{lmo}_{\{1\}}(d_1^k) \\ \operatorname{lmo}_{\{2\}}(d_2^k) \\ \operatorname{lmo}_{\{3\}}(d_3^k) \end{bmatrix} \qquad \text{(lmo is separable)}$$

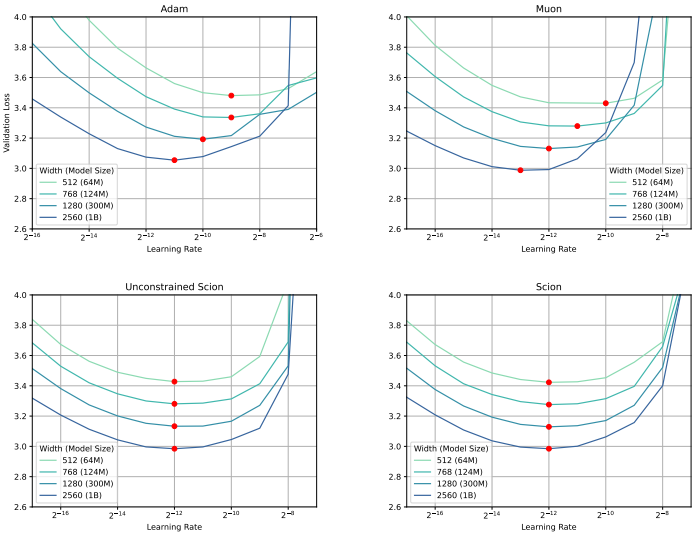
$$\begin{bmatrix} W_1^{k+1} \\ W_2^{k+1} \\ W_3^{k+1} \end{bmatrix} = (1-\gamma_k) \begin{bmatrix} W_1^k \\ W_2^k \\ W_3^k \end{bmatrix} - \gamma_k \begin{bmatrix} \operatorname{sign}(d_1^k) \\ \operatorname{msign}(d_2^k) \\ \frac{1}{m}\operatorname{sign}(d_3^k) \end{bmatrix} \qquad \text{(Compute lmos)}$$

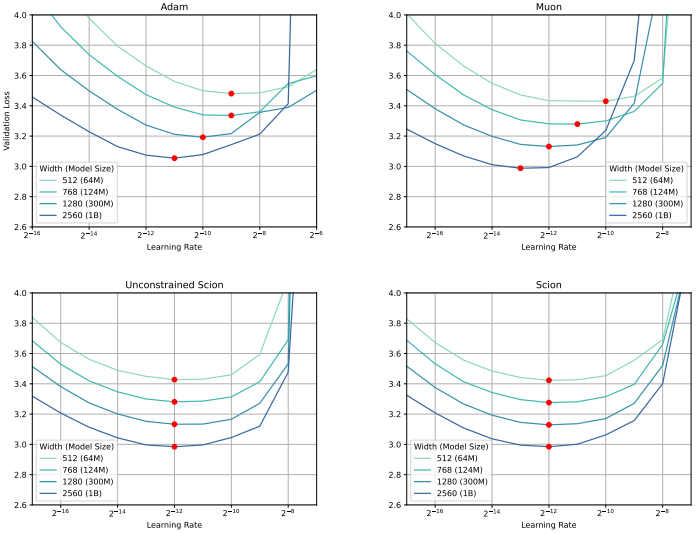
(Optional)
$$\begin{bmatrix} W_1^{k+1} \\ W_2^{k+1} \\ W_3^{k+1} \end{bmatrix} = (1-\gamma_k) \begin{bmatrix} W_1^k \\ W_2^k \\ W_3^k \end{bmatrix} - \gamma_k \begin{bmatrix} \rho_1 \left(\operatorname{sign}(d_1^k) \right) \\ \rho_2 \left(\operatorname{msign}(d_2^k) \right) \\ \rho_3 \left(\frac{1}{m}\operatorname{sign}(d_3^k) \right) \end{bmatrix} \qquad \text{(Layerwise radii).}$$

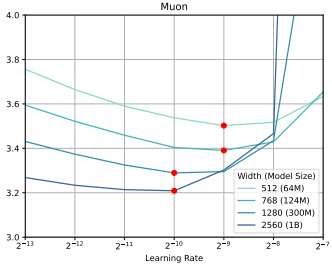
Fix the total tokens and sweep batch sizes, adjusting steps accordingly.



Hyperparameter Transfer: GPT Training

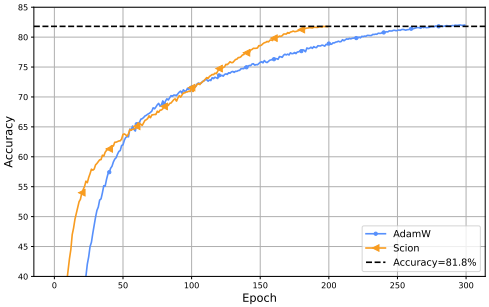


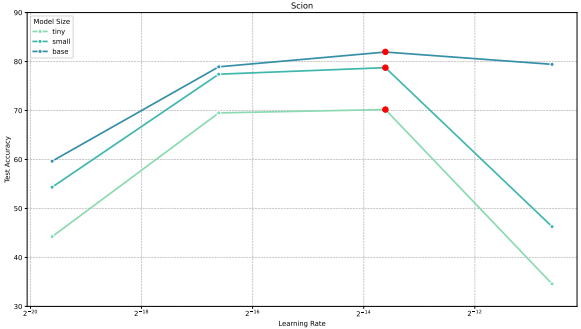




$$(\max\left(1, \sqrt{\frac{d_{\text{out}}}{d_{\text{in}}}}\right) \text{ scaling})$$

30% less epochs, 40% less wallclock time on same resources





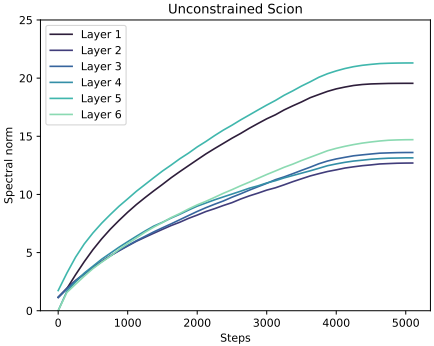
Significant accelerations are possible for imagenet training.

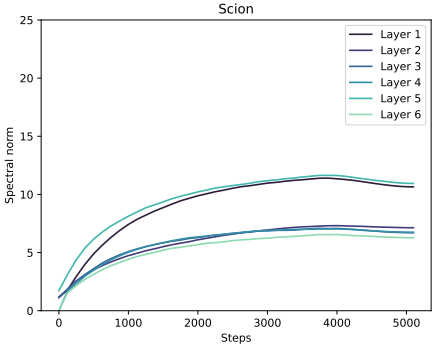
Illustration of norm control: GPT Training

Let ρ be the radius of the set $\mathcal{D}$ that is used to define lmo. Both uSCG and SCG provide control over the norm of the output x^n:

- SCG Guarantees $\|x^n\| \leq \rho$
- uSCG Guarantees $\|x^n\| \leq \rho \sum_{k=0}^{n-1} \gamma_k$

Scion inherits these bounds!





Convergence Results for fixed α independent of horizon n

Let ρ be the radius of the set $\mathcal{D}$ used in the lmo.

Theorem (Convergence rate for uSCG with constant α [Pethick et al., 2025a])

Let $n \in \mathbb{N}^*$ and let $\{x^k\}_{k=1}^n$ be the generated by uSCG with $\alpha \in (0,1)$ and constant step size $\gamma = \frac{1}{\sqrt{n}}$. Then,

$$\min_{1 \le k \le n} \mathbb{E}[\|\nabla f(x^k)\|_*] \in O\left(\frac{L\rho}{\sqrt{n}} + \sigma\right)$$

Theorem (Convergence rate for SCG with constant α [Pethick et al., 2025a])

Let $n \in \mathbb{N}^*$ and let x^n be the output of SCG with $\alpha \in (0,1)$ and constant step size $\gamma = \frac{1}{\sqrt{n}}$. Then, for all $u \in \mathcal{D}$,

$$\min_{1 \le k \le n} \mathbb{E}[\langle \nabla f(x^k), x^k - u \rangle] \in O\left(\frac{L\rho^2}{\sqrt{n}} + \sigma\right)$$

Take-home Message

Guaranteed convergence to a noise-dominated region induced by σ.

Let ρ be the radius of the set $\mathcal{D}$ used in the lmo.

Theorem (Convergence rate for uSCG with vanishing α_k [Pethick et al., 2025a])

Let $n \in \mathbb{N}^*$ and let $\{x^k\}_{k=1}^n$ be generated by uSCG with $\alpha_k = 1/\sqrt{k}$ and constant step size $\gamma = \frac{3}{4n^{3/4}}$. Then,

$$\min_{1 \leq k \leq n} \mathbb{E}[\|\nabla f(x^k)\|_*] \in O\left(\frac{1}{n^{1/4}} + \frac{L\rho}{n^{3/4}}\right)$$

Theorem (Convergence rate for SCG with vanishing α_k [Pethick et al., 2025a])

Let $n \in \mathbb{N}^*$ and let $\{x^k\}_{k=1}^n$ be generated by SCG with $\alpha_k = 1/\sqrt{k}$ and constant step size $\gamma = \frac{3}{4n^{3/4}}$. Then, for all $u \in \mathcal{D}$,

$$\min_{1 \leq k \leq n} \mathbb{E}[\langle \nabla f(x^k), x^k - u \rangle] \in O\left(\frac{1}{n^{1/4}} + \frac{L\rho^2}{n^{3/4}}\right)$$

Take-home Message

Guaranteed convergence to a first-order critical point (in expectation) for either the unconstrained (uSCG) or the constrained (SCG) problem.

arXiv:2502.07529, also at ICML 2025 (Spotlight)

Computer Science > Machine Learning

[Submitted on 11 Feb 2025]

Training Deep Learning Models with Norm-Constrained LMOs

Thomas Pethick, Wanyun Xie, Kimon Antonakopoulos, Zhenyu Zhu, Antonio Silveti-Falls, Volkan Cevher

In this work, we study optimization methods that leverage the linear minimization oracle (LMO) over a norm-ball. We propose a new stochastic family of algorithms that uses the LMO to adapt to the geometry of the problem and, perhaps surprisingly, show that they can be applied to unconstrained problems. The resulting update rule unifies several existing optimization methods under a single framework. Furthermore, we propose an explicit choice of norm for deep architectures, which, as a side benefit, leads to the transferability of hyperparameters across model sizes. Experimentally, we demonstrate significant speedups on nanoGPT training without any reliance on Adam. The proposed method is memory-efficient, requiring only one set of model weights and one set of gradients, which can be stored in half-precision.

Subjects: **Machine Learning (cs.LG)**; Optimization and Control (math.OC)
Cite as: arXiv:2502.07529 **[cs.LG]**
(or arXiv:2502.07529v1 **[cs.LG]** for this version)
https://doi.org/10.48550/arXiv.2502.07529

https://github.com/LIONS-EPFL/scion

Instead of Steepest Descent

$$x^{k+1} = \operatorname*{argmin}_{x \in \mathbb{R}^d} \langle d^k, x - x^k \rangle + \frac{1}{2\gamma_k} \|x - x^k\|^2$$

which scales the lmo by $\|d^k\|_*$, and instead of using

$$x^{k+1} = \operatorname*{argmin}_{x \in \mathbb{R}^d} \langle d^k, x - x^k \rangle + \iota_{\gamma_k \mathcal{D}}(x - x^k)$$

we can combine them to get

$$x^{k+1} = \operatorname*{argmin}_{\|x - x^k\| \leq \gamma_k} \langle d^k, x - x^k \rangle + \frac{1}{2} \|x - x^k\|^2$$

which gives the *clipped* update

$$x^{k+1} = x^k + \min\{\gamma_k, \|d^k\|_*\} \mathrm{lmo}_{\mathcal{D}}(d^k).$$

The gradient ∇f is said to be (L_0, L_1)-smooth with $L_0, L_1 \in [0, \infty)$ if, for all $x, y \in \mathcal{X}$ with $\|x - y\| \leq \frac{1}{L_1}$, it holds

$$\|\nabla f(x) - \nabla f(y)\|_* \leq (L_0 + L_1 \|\nabla f(x)\|_*)\|x - y\|. \tag{1}$$

Let ρ be the radius of the set $\mathcal{D}$ used in the lmo.

Theorem (Convergence rate for Clipped Scion [Pethick et al., 2025b])

Let $n \in \mathbb{N}^*$. Consider the iterates $\{x^k\}_{1 \leq k \leq n}$ generated by Clipped Scion with a constant stepsize $\gamma \leq 1/L_0$ and $\gamma\rho \leq 1/2L_1$. Then,

$$\min_{1 \leq k \leq n} \mathbb{E}[\|\nabla f(x^k)\|_*] \in O\left(\frac{1}{n^{1/4}}\right)$$

Take-home Message

This matches the worst-case rate of uSCG/SCG (Scion).

To clip or not to clip

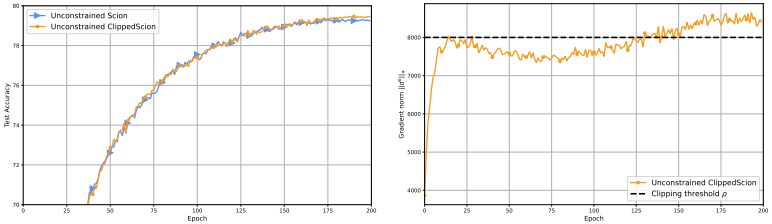


Figure: Clipping improves over Scion by a 15% speedup on DeiT-base.

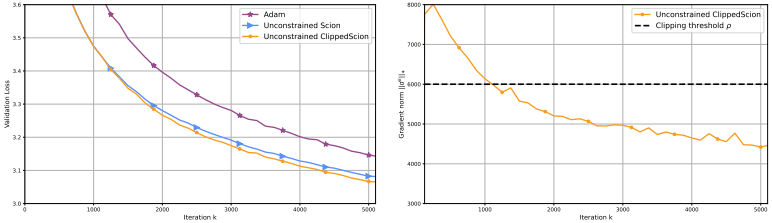


Figure: Clipping can improve uScion by 10% on NanoGPT (1B). Effectively, uScion with clipping is Stoch. Spectral Desc. with momentum.

arXiv:2506.01913, also at NeurIPS 2025 (Oral)

Computer Science > Machine Learning

[Submitted on 2 Jun 2025]

Generalized Gradient Norm Clipping & Non-Euclidean (L_0, L_1)-Smoothness

Thomas Pethick, Wanyun Xie, Mete Erdogan, Kimon Antonakopoulos, Tony Silveti-Falls, Volkan Cevher

This work introduces a hybrid non-Euclidean optimization method which generalizes gradient norm clipping by combining steepest descent and conditional gradient approaches. The method achieves the best of both worlds by establishing a descent property under a generalized notion of (L_0, L_1)-smoothness. Weight decay is incorporated in a principled manner by identifying a connection to the Frank-Wolfe short step. In the stochastic case, we show an order optimal $O(n^{-1/4})$ convergence rate by leveraging a momentum based gradient estimator. We discuss how to instantiate the algorithms for deep learning and demonstrate their properties on image classification and language modeling.

Subjects: **Machine Learning (cs.LG)**; Machine Learning (stat.ML)
Cite as: arXiv:2506.01913 **[cs.LG]**
(or arXiv:2506.01913v1 **[cs.LG]** for this version)
https://doi.org/10.48550/arXiv.2506.01913

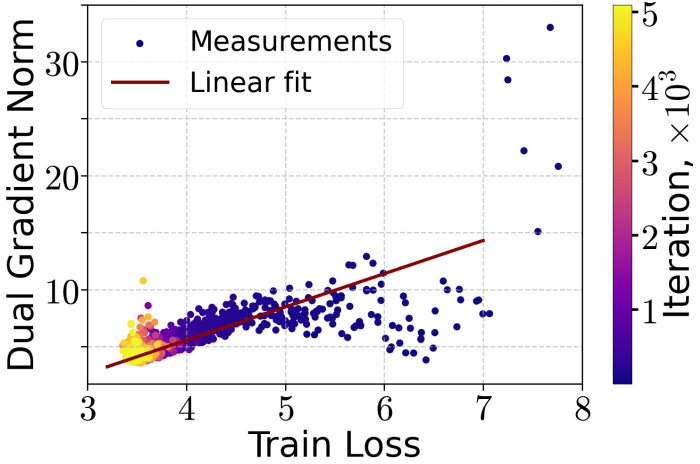
`https://github.com/LIONS-EPFL/ClippedScion`

Extension to (L_0, L_1) smoothness with clipping.

The objective function $f(x)$ is μ-KL for some $\mu > 0$, if the following inequality holds:

$$\|\nabla f(x)\|_* \geq \mu\,(f(x) - f^\star) \qquad \text{for all } x \in \mathcal{X}.$$

where $f^\star = \min_{x \in \mathcal{X}} f(x)$.



Theorem (Convergence of SCG)

Suppose f satisfies the μ-KL condition. Let $d^0 = g(x_0; \xi_0)$ and $\|x_0\| \leq \|x^\star\| = R_\star$ Let the parameters of Scion be chosen as follows, where η is the norm-equivalence constant and σ is the gradient noise:

$$K = \max\left\{2, \tilde{\mathcal{O}}\left(\max\left\{\frac{L}{\varepsilon\mu^2}, \frac{\eta\sigma}{\varepsilon\mu}, \frac{L(\eta\sigma)^2}{\mu(\varepsilon\mu)^3}, \frac{(\eta\sigma)^3}{(\varepsilon\mu)^3}\right\}\right)\right\},$$

$$\beta = \min\left\{\frac{1}{K}, \frac{1}{R.\mu} \cdot \mathcal{O}\left(\frac{\varepsilon\mu^2}{L}, \frac{\varepsilon\mu}{\eta\sigma}, \frac{\mu(\varepsilon\mu)^3}{L(\eta\sigma)^2}, \frac{(\varepsilon\mu)^3}{(\eta\sigma)^3}\right)\right\}$$

$$\rho = 2R_\star, \text{ and } \alpha = \min\left\{1, \mathcal{O}\left(\frac{(\varepsilon\mu)^2}{(\eta\sigma)^2}\right)\right\},$$

where $\mathcal{O}$ hides all numerical constants and $\tilde{\mathcal{O}}$ hides all numerical and logarithmic factors. Then, under mild conditions, the output of SCG after K iterations satisfies $\mathbb{E}[f(x_K) - f^\star] \leq \varepsilon$.

Corollary

Let the token budget be large enough: $T \geq 2BS$. Then, running the algorithm with parameters from Theorem 1 for $\frac{T}{BS}$ iterations, we achieve the loss

$$\varepsilon = \tilde{\mathcal{O}}\left(\max\left\{\underbrace{\frac{\rho\sigma_0}{\mu\, T^{1/3}(BS)^{1/6}}}_{\text{small-batch noise}}, \underbrace{\frac{L^{1/3}\rho^{2/3}\sigma_0^{2/3}}{\mu^{4/3}T^{1/3}}}_{\text{batch-independent floor}}, \underbrace{\frac{LBS}{\mu^2 T}}_{\text{iteration-starved}}\right\}\right). \qquad (*)$$

Remarks:

- With BS small, the first term in (*) dominates, ε diminishes with the increasing of BS.
- When $BS > \left(\frac{L}{\mu\eta\sigma_0}\right)^2$, the second term dominates, ε is independent of BS.
- Further increasing BS, the error deteriorates with BS.
  - Large BS is preferable due to higher parallelism.
  - The above motivates us to stay between the second and the third regime, which yields:

$$\frac{L}{\mu^2}\frac{BS}{T} = \left(\frac{L\eta^2\sigma_0^2}{\mu^4 T}\right)^{1/3} \Leftrightarrow BS = \left(\frac{T\mu\eta\sigma_0}{L}\right)^{2/3}.$$

Example

You've tuned on a base run with budget T_0 and found optimal (B_0, S_0, β_0). Now you want to train with budget $T_1 = r \cdot T_0$.
Example: Suppose your base 124M run uses

$$T_0 = 1.3\text{B tokens}, \qquad B_0 = 256, \qquad S_0 = 1024, \qquad \beta_0 = 3.6 \cdot 10^{-4}.$$

Now you want a roughly $8\times$ larger token budget, say $T_1 \approx 10.6$B. Then $r \approx 8.15$, so:

$$r^{2/3} \approx 4.05, \qquad r^{-1/3} \approx 0.50.$$

Therefore:

$$B_1 S_1 \approx 4.05 \cdot B_0 S_0, \qquad \beta_1 \approx 1.8 \cdot 10^{-4}.$$

If you keep $S = 1024$, this gives $B \approx 1037$, which you would round to $B = 1024$. Alternatively, $B = 512$ and $S = 2048$ gives nearly the same token batch while also extending context length.

$B = 416$ for $T = 2.7$B, $B = 672$ for $T = 5.3$B, and $B = 896$ for $T = 8.0$B.
Momentum and sequence length are set to $\alpha = 0.1$ and $S = 1024$, respectively.

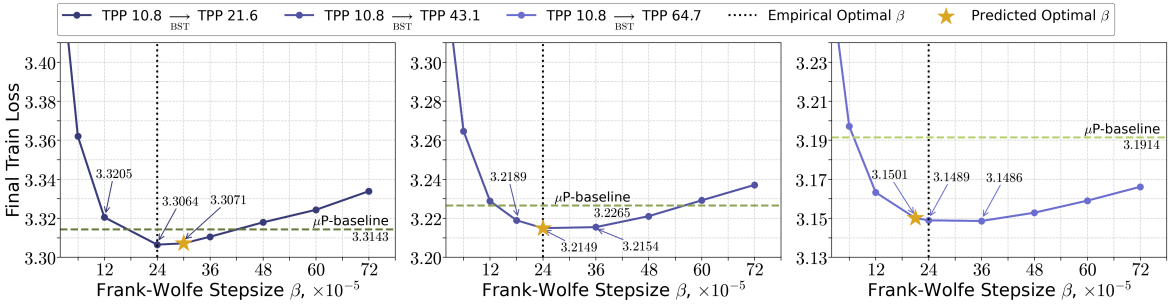


Figure: The final performance of the 124M model when varying the Frank–Wolfe stepsize β under different token budgets (**left:** 2.7B, **center:** 5.3B, **right:** 8.0B). We average the train loss over 3 random seeds and report the moving average in the window of size 500. We observe that the BST scaling rule predicts a good estimate for the optimal β when increasing the token budget. Moreover, the difference in performance between BST and μP baselines grows with a token budget.

arXiv:2506.01913, also at ICML 2026

Computer Science > Machine Learning

[Submitted on 22 Mar 2026]

On the Role of Batch Size in Stochastic Conditional Gradient Methods

Rustem Islamov, Roman Machacek, Aurelien Lucchi, Antonio Silveti-Falls, Eduard Gorbunov, Volkan Cevher

We study the role of batch size in stochastic conditional gradient methods under a μ-Kurdyka-Łojasiewicz (μ-KL) condition. Focusing on momentum-based stochastic conditional gradient algorithms (e.g., Scion), we derive a new analysis that explicitly captures the interaction between stepsize, batch size, and stochastic noise. Our study reveals a regime-dependent behavior: increasing the batch size initially improves optimization accuracy but, beyond a critical threshold, the benefits saturate and can eventually degrade performance under a fixed token budget. Notably, the theory predicts the magnitude of the optimal stepsize and aligns well with empirical practices observed in large-scale training. Leveraging these insights, we derive principled guidelines for selecting the batch size and stepsize, and propose an adaptive strategy that increases batch size and sequence length during training while preserving convergence guarantees. Experiments on NanoGPT are consistent with the theoretical predictions and illustrate the emergence of the predicted scaling regimes. Overall, our results provide a theoretical framework for understanding batch size scaling in stochastic conditional gradient methods and offer guidance for designing efficient training schedules in large-scale optimization.

Subjects: **Machine Learning (cs.LG)**; Optimization and Control (math.OC); Machine Learning (stat.ML)
Cite as: arXiv:2603.21191 **[cs.LG]**
(or arXiv:2603.21191v1 **[cs.LG]** for this version)
https://doi.org/10.48550/arXiv.2603.21191

- Frank-Wolfe [Frank and Wolfe, 1956], Stochastic FW [Mokhtari et al., 2020], etc
- Stochastic Spectral Descent [Carlson et al., 2015]
- Duality Structure of Gradient Descent [Flynn, 2017]
- PSGD [Li, 2017], [Pooladzandi and Li, 2024]
- Shampoo [Gupta et al., 2018, Anil et al., 2020]
- Muon blogpost [Jordan et al., 2024]
- Modular Duality [Bernstein and Newhouse, 2024]
- Kimi Moonlight [Liu et al., 2025]

Adaptive methods:

- Gluon [Riabinin et al., 2025]
- AdaMuon [Si et al., 2025]
- NorMuon [Li et al., 2025]
- AdaGo [Zhang et al., 2025]

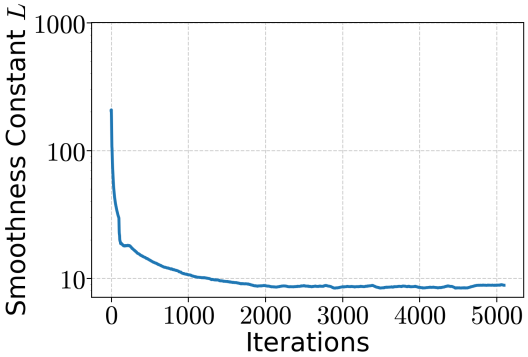
Inexact variants, Period or Block LMO:

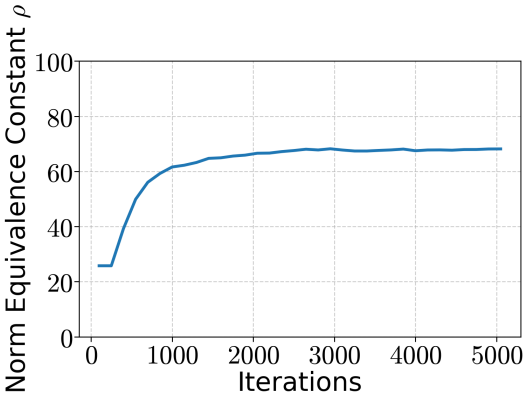
- Beyond the ideal [Shulgin et al., 2025]
- MuonBP (Block-Periodic orthogonalization) [Khaled et al., 2025]
- DropMuon [Gruntkowska et al., 2025]

Distributed/Federated versions:

- Dion [Ahn et al., 2025]
- Disco [Filatov et al., 2025]
- Federated Muon [Takezawa et al., 2025]

More every single day...


Smoothness Constant L
1000
100
10
0
1000
2000
3000
4000
5000
Iterations


Norm Equivalence Constant ρ
100
80
60
40
20
0
0
1000
2000
3000
4000
5000
Iterations

References I

Kwangjun Ahn, Byron Xu, Natalie Abreu, Ying Fan, Gagik Magakyan, Pratyusha Sharma, Zheng Zhan, and John Langford. Dion: Distributed orthonormalized updates. *arXiv preprint arXiv:2504.05295*, 2025.

Rohan Anil, Vineet Gupta, Tomer Koren, Kevin Regan, and Yoram Singer. Second order optimization made practical. *arXiv preprint arXiv:2002.09018*, 2020.

Jeremy Bernstein and Laker Newhouse. Modular duality in deep learning. *arXiv preprint arXiv:2410.21265*, 2024.

David Carlson, Volkan Cevher, and Lawrence Carin. Stochastic spectral descent for restricted Boltzmann machines. In *Artificial Intelligence and Statistics*, pages 111–119. PMLR, 2015.

Oleg Filatov, Jiangtao Wang, Jan Ebert, and Stefan Kesselheim. Optimal scaling needs optimal norm. *arXiv preprint arXiv:2510.03871*, 2025.

Thomas Flynn. The duality structure gradient descent algorithm: analysis and applications to neural networks. *arXiv preprint arXiv:1708.00523*, 2017.

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

Kaja Gruntkowska, Yassine Maziane, Zheng Qu, and Peter Richtárik. Drop-muon: Update less, converge faster. *arXiv preprint arXiv:2510.02239*, 2025.

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

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

Ahmed Khaled, Kaan Ozkara, Tao Yu, Mingyi Hong, and Youngsuk Park. Muonbp: Faster muon via block-periodic orthogonalization. *arXiv preprint arXiv:2510.16981*, 2025.

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

Xi-Lin Li. Preconditioned stochastic gradient descent. *IEEE transactions on neural networks and learning systems*, 29(5): 1454–1466, 2017.

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

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*, 2025.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.

Aryan Mokhtari, Hamed Hassani, and Amin Karbasi. Stochastic conditional gradient methods: From convex minimization to submodular maximization. *Journal of machine learning research*, 21(105):1–49, 2020.

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*, 2025a.

Thomas Pethick, Wanyun Xie, Mete Erdogan, Kimon Antonakopoulos, Antonio Silveti-Falls, and Volkan Cevher. Generalized gradient norm clipping & non-euclidean (l_0, l_1)-smoothness. *arXiv preprint arXiv:2506.01913*, 2025b.

Omead Pooladzandi and Xi-Lin Li. Curvature-informed sgd via general purpose lie-group preconditioners. *arXiv preprint arXiv:2402.04553*, 2024.

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.

Egor Shulgin, Sultan AlRashed, Francesco Orabona, and Peter Richtárik. Beyond the ideal: Analyzing the inexact muon update. *arXiv preprint arXiv:2510.19933*, 2025.

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

Yuki Takezawa, Anastasia Koloskova, Xiaowen Jiang, and Sebastian U Stich. Fedmuon: Federated learning with bias-corrected lmo-based optimization. *arXiv preprint arXiv:2509.26337*, 2025.

Greg Yang, James B Simon, and Jeremy Bernstein. A spectral condition for feature learning. *arXiv preprint arXiv:2310.17813*, 2023.

Minxin Zhang, Yuxuan Liu, and Hayden Schaeffer. Adagrad meets muon: Adaptive stepsizes for orthogonal updates. *arXiv preprint arXiv:2509.02981*, 2025.