Training Deep Learning Models with Norm-Constrained LMOs

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

Why come up with new optimizers?

Adam(W)		At the end of today
• Large memory overhead	→	no additional memory required
• Requires warmup, gradient norm clipping…	→	more stable (no tricks required)
• does not work well for large batches	→	benefit is larger for larger batches
• does not take network structure into account	→	adapts to the geometry (faster)
		hyperparameter transfer

Line of attack

- Review classical algorithms: Conditional gradient / Frank-Wolfe
- Stochastic case: the role of momentum
- Conditional gradient as normalization: why normalize?
- How to adapt to deep learning
- Experiments

Ingredients in optimization for deep learning

- **Gradient descent**: $x^{k+1} = x^k - \gamma \nabla f(x^k)$

Steepest descent direction

- **Dense updates**: $x^{k+1} = x^k - \gamma \operatorname{sign}(\nabla f(x^k))$

Updates *all* parameters *equally much*

- **Weight decay**: $x^{k+1} = (1-\gamma)x^k - \gamma \operatorname{sign}(\nabla f(x^k))$

Regularizes by keeping parameters small

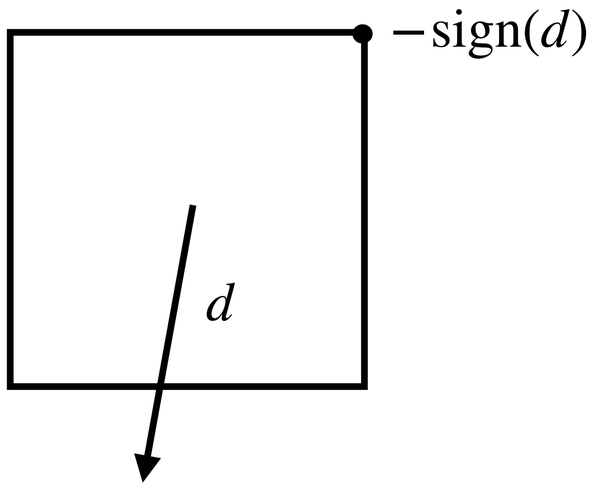
Linearized objective within a trust-region

$$\operatorname{argmin}_{x \in \mathcal{D}} \langle d, x \rangle$$

If we measuring the change in ℓ_∞:

$$\mathcal{D} := \{x \mid \|x\|_\infty \leq 1\}$$

… then all coordinates $|x_i| = 1$.



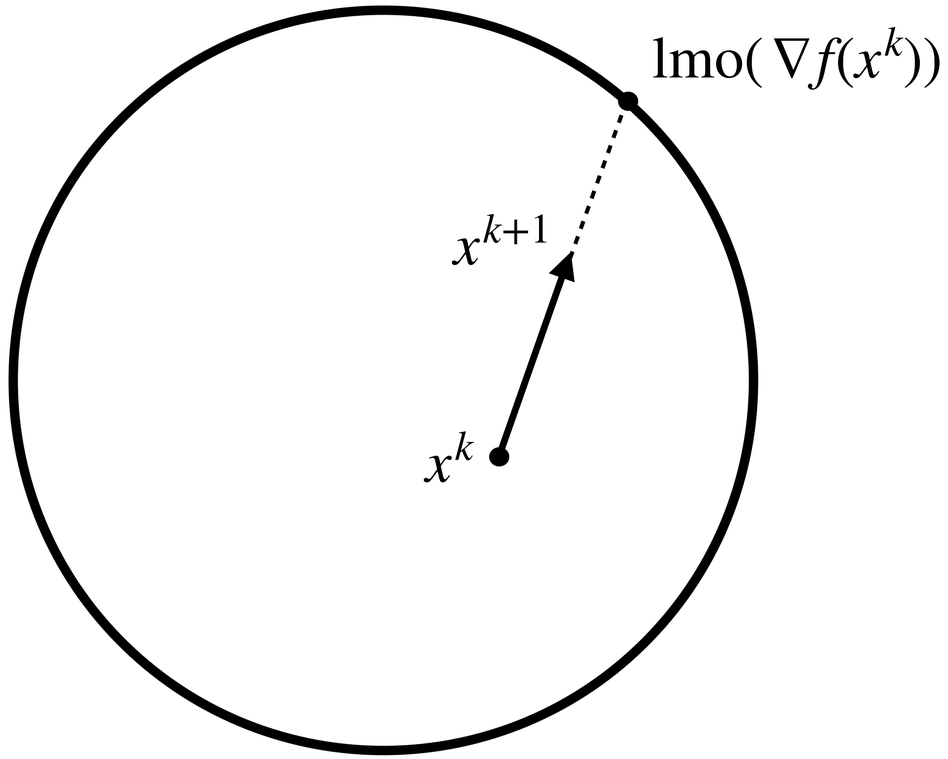
Conditional Gradient (aka Frank-Wolfe)

Linear minimization oracle (LMO)

$$\mathrm{lmo}(d) \in \mathrm{argmin}_{x \in \mathscr{D}} \langle d, x \rangle$$

Conditional Gradient (CG) [Frank & Wolfe 1956]

$$x^{k+1} = (1 - \gamma_k) x^k + \gamma_k \, \mathrm{lmo}(\nabla f(x^k))$$



Norm-ball constraint: $\mathscr{D} := \{x \mid \|x\| \leq \rho\}$

Norm control: $\|x^k\| \leq \rho$ ⇒ avoiding overfitting + numerical stability, weight decay

Adapts to the geometry: distances are controlled in the chosen norm $\|\cdot\|$ ⇒ faster

Dealing with stochastic gradients

Linear minimization oracle (LMO)

$$\mathrm{lmo}(d) \in \operatorname{argmin}_{x \in \mathcal{D}} \langle d, x \rangle$$

Unbiased stochastic oracle:

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

Stochastic Conditional Gradient (SCG) [Mokhtari et al. 2020]

$$x^{k+1} = (1 - \gamma_k) x^k + \gamma_k \, \mathrm{lmo}(d^k)$$

- **Naively:** $d^k = \nabla f(x^k, \xi_k)$

  Problem Even when $\nabla f(x^k, \xi_k)$ is unbiased the (nonlinear) $\mathrm{lmo}(\nabla f(x^k, \xi_k))$ can be biased

- **Momentum based gradient estimator:** reduce the variance

$$d^k = (1 - \alpha_k) d^{k-1} + \alpha_k \nabla f(x^k, \xi_k)$$

Alternative of averaging *after* applying the LMO (ALMOND): still needs increasing batch size

Unconstrained problems

Linear minimization oracle (LMO)

$$\operatorname{lmo}(d) \in \operatorname{argmin}_{x \in \mathcal{D}} \langle d, x \rangle$$

LMOs can be used for unconstrained problems:

Unconstrained Stochastic Conditional Gradient (uSCG)

$$d^k = (1 - \alpha_k) d^{k-1} + \alpha_k \nabla f(x^k, \xi_k)$$

$$x^{k+1} = x^k + \gamma_k \operatorname{lmo}(d^k)$$

Norm control: $\|x^k\| \leq \rho \sum_{0<i<k} \gamma_i$

Examples

ℓ_2	$x^{k+1} = x^k - \gamma_k \frac{d^k}{\|d^k\|_2}$
ℓ_∞	$x^{k+1} = x^k - \gamma_k \operatorname{sign}(d^k)$
$\mathcal{S}_\infty$	$x^{k+1} = x^k - \gamma_k U V^\top$ with $d^k = U \Sigma V^\top$

Many perspectives

- conditional gradient
- trust region method with linear surrogate
- normalized steepest descent

- $\mathcal{S}_\infty$ is the Schatten-∞/Spectral norm
- Dense updates turn out to be particularly useful for deep learning

Why normalize?

Linear minimization oracle (LMO)

$$\operatorname{lmo}(d) \in \operatorname{argmin}_{x \in \mathcal{D}} \langle d, x \rangle$$

LMOs can be used for unconstrained problems:

Unconstrained Stochastic Conditional Gradient (uSCG)

$$d^k = (1 - \alpha_k) d^{k-1} + \alpha_k \nabla f(x^k, \xi_k)$$

$$x^{k+1} = x^k + \gamma_k \operatorname{lmo}(d^k)$$

Constant update magnitude $\| \operatorname{lmo}(d^k) \| = \rho$**:**

PRO Invariance to gradient magnitude ⇒ stepsize independent of Lipschitz constant

Normalization adapts to (local) Hölder smoothness in online learning [Orabona 2023]

Normalization escapes saddle points faster [Levy 2016]

CON Large steps even near solutions ⇒ not a descent method

Not a problem (especially not for stochastic)

since stepsize can be taken decreasing (in practice: linear decay)

What does the theory tell us?

Assumption. Lipschitz continuous under $\|\cdot\|$, i.e.,

$$\|\nabla f(x) - \nabla f(y)\|_* \leq L\|x - y\|$$

Only needs to hold **locally**: $\|x - y\| \leq \gamma$

Theorem. Suppose the above. Then uSCG with $\alpha_k = 1$ satisfies

$$\min_{1\leq k\leq n} \|\nabla f(x^k)\|_* \leq \frac{f(x^1) - f^\star}{\gamma n} + \frac{L\gamma}{2} = \frac{f(x^1) - f^\star + L/2}{\sqrt{n}}.$$

with $\gamma = 1/\sqrt{n}$.

Stepsize decreasing / horizon dependent necessary even for deterministic

How to converge fast? Reduce contribution from:

- … dimension of x
  significant when dimension is large
- … iterations n (for convex)
  acceleration seems difficult [El Halabi 2018]

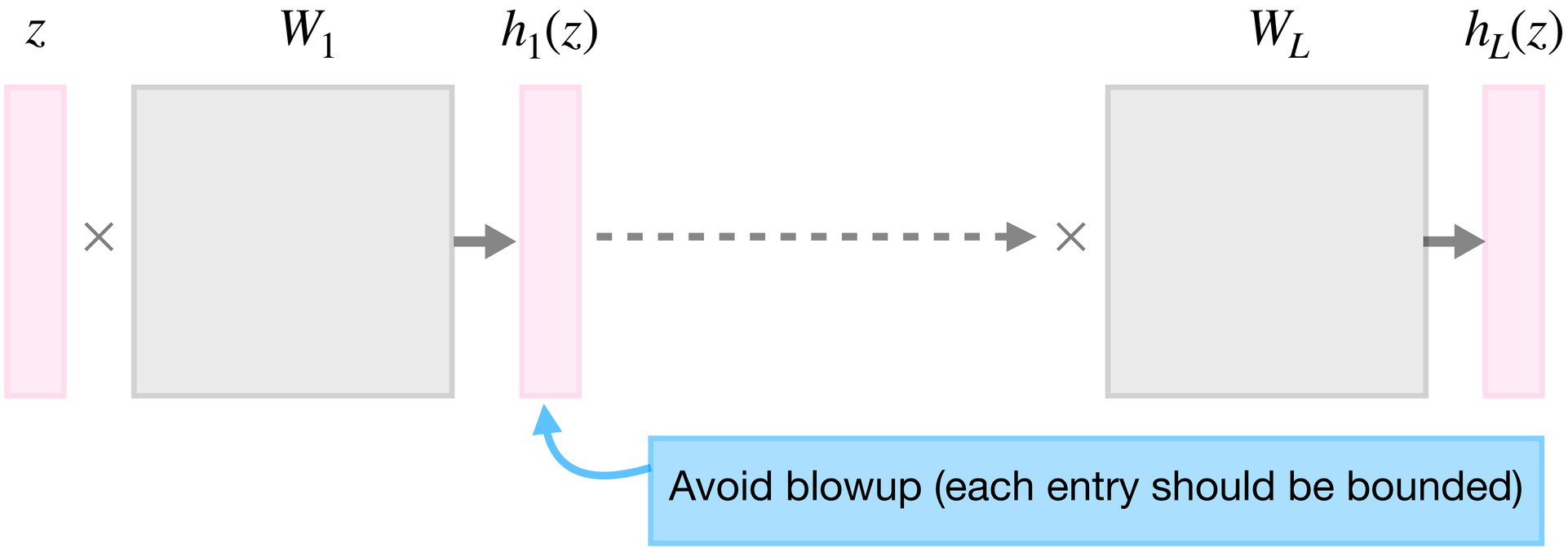
Stochastic case:

- $\mathcal{O}(n^{-1/4})$ rate with $\gamma = 1/n^{3/4}$ and $\alpha_k = 1/\sqrt{k}$
- Noise is inherently Euclidean:
  $\Rightarrow$ *we expect larger benefit for larger batch size*

How to choose the norm in deep learning?

Consider a linear deep neural network

$$\min \mathbb{E}_{z,y}[\mathscr{L}(h_L(z), y)] \qquad \text{with} \qquad h_\ell(z) = W_\ell h_{\ell-1}(z)$$



How to choose the norm in deep learning?

Require that pre-activations are bounded:

$$\|h_\ell(z)\|_\beta \le 1$$

Definition. Operator norm

$$\|A\|_{\alpha\to\beta} = \sup_{\|z\|_\alpha=1} \|Az\|_\beta.$$

Examples

- $\frac{1}{d_{\text{out}}}\|h_\ell(z)\|_1 \le 1$ (the average entry is bounded)
- $\|h_\ell(z)\|_{\text{RMS}} \le 1$ (the typical entry is bounded)
- $\|h_\ell(z)\|_\infty \le 1$ (the maximum entry is bounded)

where $\|z\|_{\text{RMS}} := \frac{1}{\sqrt{d}}\|z\|_2$

- By definition if $\|z\|_\alpha$ and $\|A\|_{\alpha\to\beta}$ are bounded, then the output $\|Az\|_\beta$ is bounded.
- For $\|z\|_{\text{RMS}} \Rightarrow \|A\|_{\alpha\to\beta} = \sqrt{\frac{d_{\text{in}}}{d_{\text{out}}}}\|A\|_{2\to 2}$ (Schatten-∞ / Spectral norm…) [Large et al. 2024]
- same for other norms, e.g. $\|\cdot\|_{\text{RMS}\to\infty}$ (rowwise ℓ_2), but requires conversion to maintain invariance

A layerwise update

Product norm. Consider $x = (W_1, \dots, W_D)$

$$\|x\| = \|(\frac{1}{\rho_1}\|W_1\|_{\alpha_1 \to \beta_1}, \dots, \frac{1}{\rho_D}\|W_D\|_{\alpha_D \to \beta_D})\|_{\mathcal{X}}$$

- For ℓ_1-norm [Flynn 2017]: one layer updated
- For max-norm [Large et al. 2024], lmo decomposes:

$$\mathrm{lmo}(x) = \{\rho_1 \,\mathrm{lmo}_{\alpha_1 \to \beta_1}(W_1), \dots, \rho_D \,\mathrm{lmo}_{\alpha_D \to \beta_D}(W_D)\}$$

Stochastic Conditional Gradient with Operator Norms (Scion)

ScionLight saves this on p.grad

1. $\xi_k \sim \mathcal{P}$
2. $d^k \leftarrow \alpha_k \nabla f(x^k, \xi_k) + (1 - \alpha_k) d^{k-1}$
3. **for** $l = 1, \dots, D$:

$$x_l^{k+1} \leftarrow \begin{cases} x_l^k + \gamma \rho_l \,\mathrm{lmo}_{\|\cdot\|_{\alpha_l \to \beta_l}}(d_l^k) & \textbf{if unconstrained} \\ (1-\gamma) x_l^k + \gamma \rho_l \,\mathrm{lmo}_{\|\cdot\|_{\alpha_l \to \beta_l}}(d_l^k) & \text{otherwise} \end{cases}$$

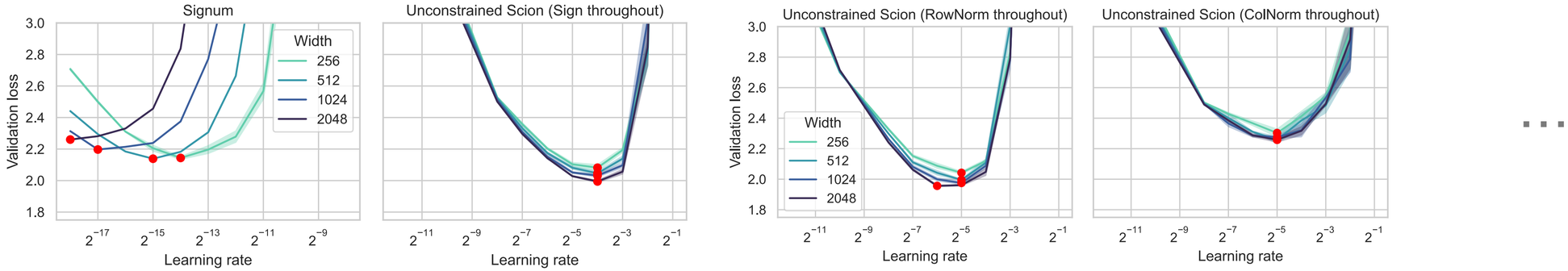
Single stepsize

Layer wise radius

- **Luck**: gradients are low-rank (computing SVD of gradients can be fast), while weights are dense

Different norm choices and hyperparameter transfer

Weight norm (bias norm)		W_1 (1-hot encoded)	W_1 (image domain)	$(W_\ell)_{\ell\in[2,\dots,L-1]}$	W_L	b_ℓ
$\text{RMS} \to \text{RMS}$ (RMS)	lmo	$-\sqrt{d_{\text{out}}}UV^\top$	$-\sqrt{d_{\text{out}}/d_{\text{in}}}UV^\top$			$-\frac{b_\ell}{\|b_\ell\|_{\text{RMS}}}$
$1 \to \text{RMS}$ (RMS)	lmo	$\text{col}_j(W_1) \mapsto -\sqrt{d_{\text{out}}}\frac{\text{col}_j(W_1)}{\|\text{col}_j(W_1)\|_2}$	$\text{col}_j(W_\ell) \mapsto -\frac{\sqrt{d_{\text{out}}}}{d_{\text{in}}}\frac{\text{col}_j(W_\ell)}{\|\text{col}_j(W_\ell)\|_2}$			$-\frac{b_\ell}{\|b_\ell\|_{\text{RMS}}}$
$\text{RMS} \to \infty$ (RMS)	lmo	$\text{row}_i(W_1) \mapsto -\frac{\text{row}_i(W_1)}{\|\text{row}_i(W_1)\|_2}$	$\text{row}_i(W_\ell) \mapsto -\frac{\text{row}_i(W_\ell)}{\sqrt{d_{\text{in}}}\|\text{row}_i(W_\ell)\|_2}$			$-\frac{b_\ell}{\|b_\ell\|_{\text{RMS}}}$
$1 \to \infty$ (∞)	lmo	$-\text{sign}(W_1)$	$-\frac{1}{d_{\text{in}}}\text{sign}(W_\ell)$			$-\text{sign}(b_\ell)$



...

Remark We have reasoned through upper bounds:

Lower bounds are also needed to avoid *lazy regime* where parameters do not move

Definition. Feature learning: $\|h_\ell(z)\|_{\text{RMS}} = \Theta(1)$ and $\|\Delta h_\ell(z)\|_{\text{RMS}} = \Theta(1)$

Experiments

Experimental setup

Muon

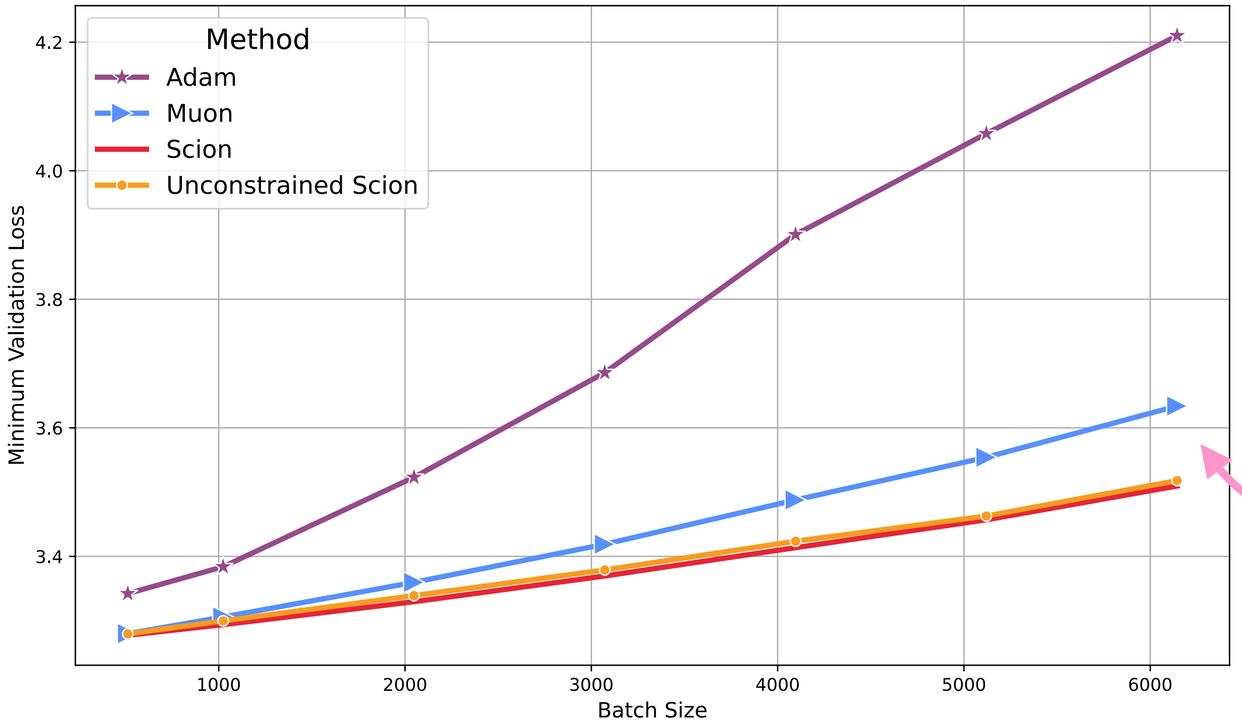
- Spectral LMO for hidden layers (uSCG + Nesterov momentum)
- AdamW for remaining layers

Scion

- 1-hot Language: Sign → Spectral → Sign (weight sharing)
- Image domain: Spectral → Spectral → Sign*

*batch norm parameters uses ℓ_2-norm
learnable pos. embedding uses ℓ_2-norm

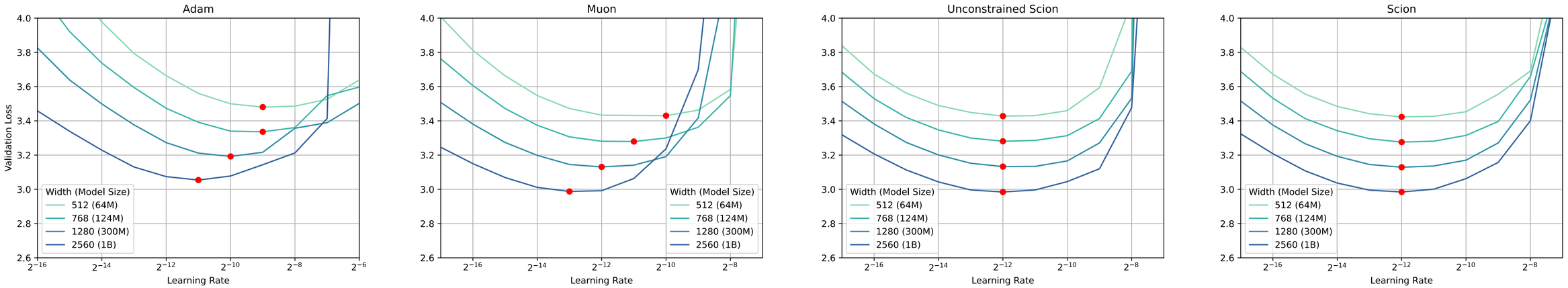
Less sensitive to large batches



Translates into a 25% speedup over Muon for batchsize=6k

- Larger improvement for larger batch
- Why?
  - Noise is inherently Euclidean
  - For single sample: gradients are rank-1 ⇒ all Schatten norms are equivalent

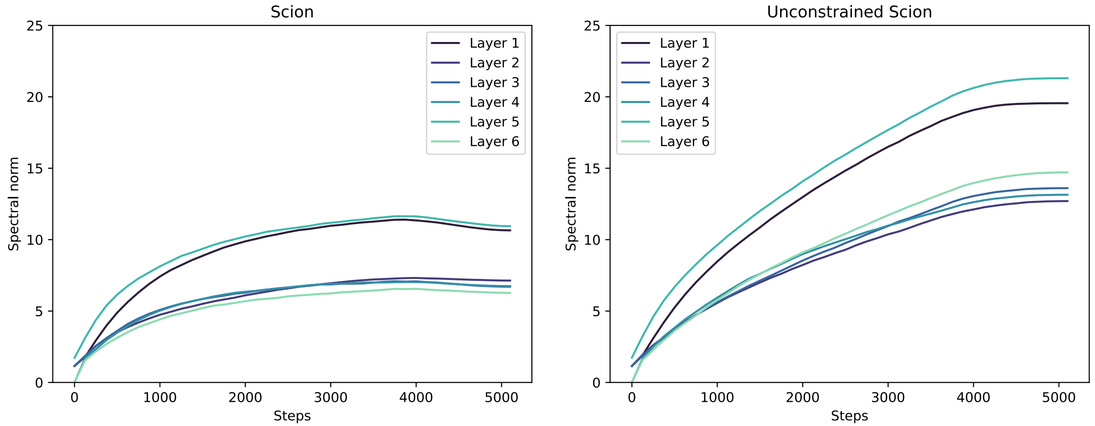
Hyperparameter transfer



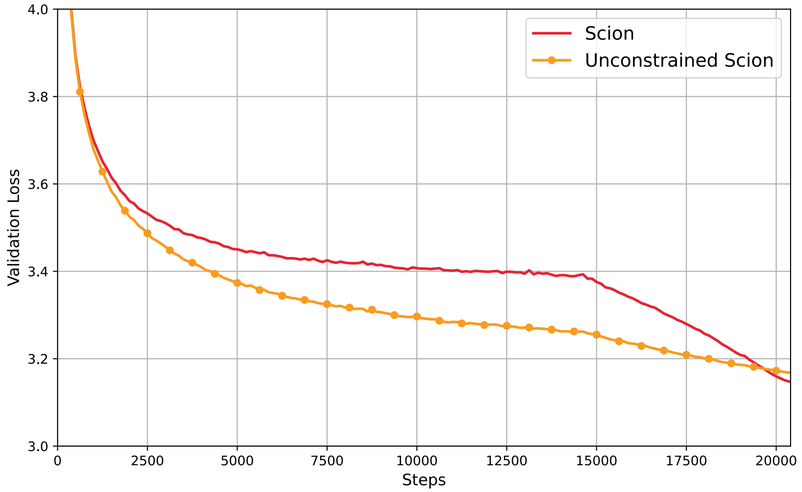
When constraints are useful

Long runs

- Norm of weight on NanoGPT (~5k steps):

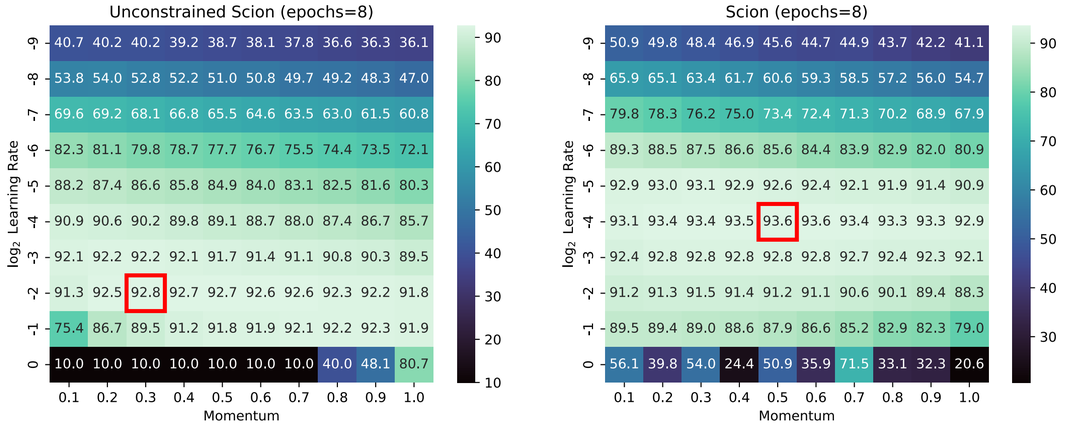


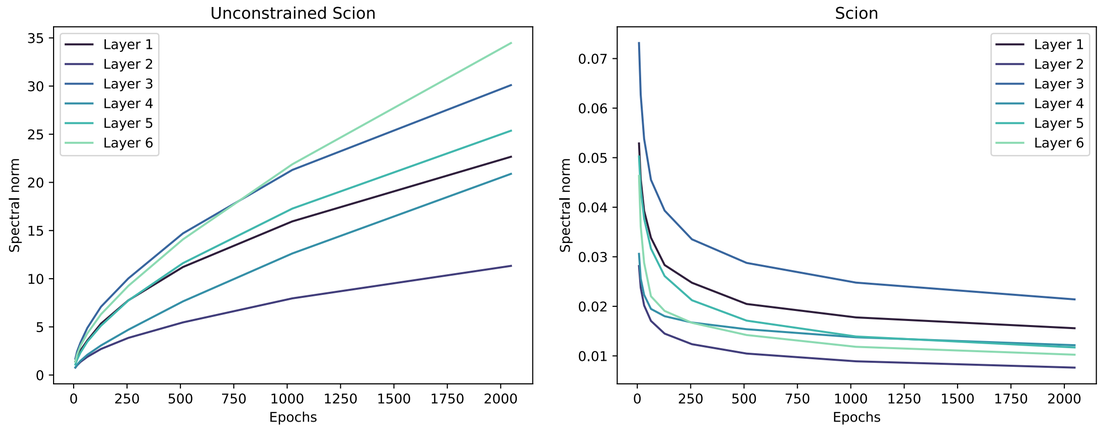
- Important for long runs (~20k):



Multiple epochs Avoid overfitting

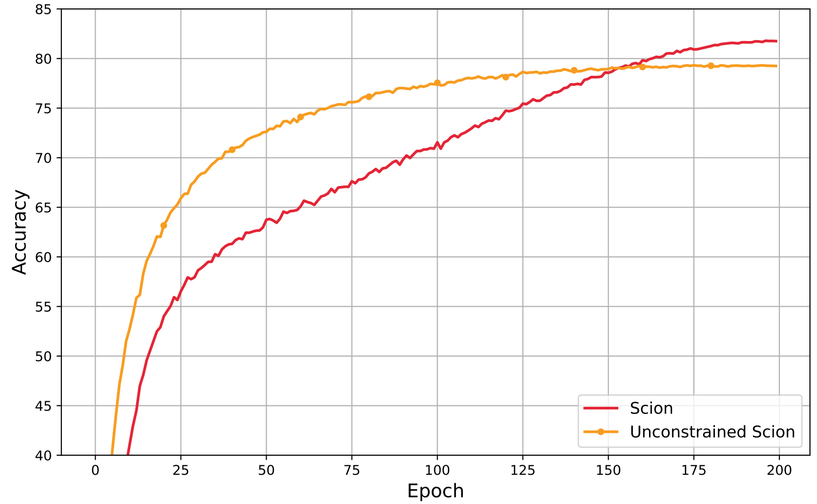
- Even for only ~100 steps runs on CIFAR10





- avoids Frobenius norm., whitening biases present in Muon baseline (see Github for speedrun)

ViTs on ImageNet experiments



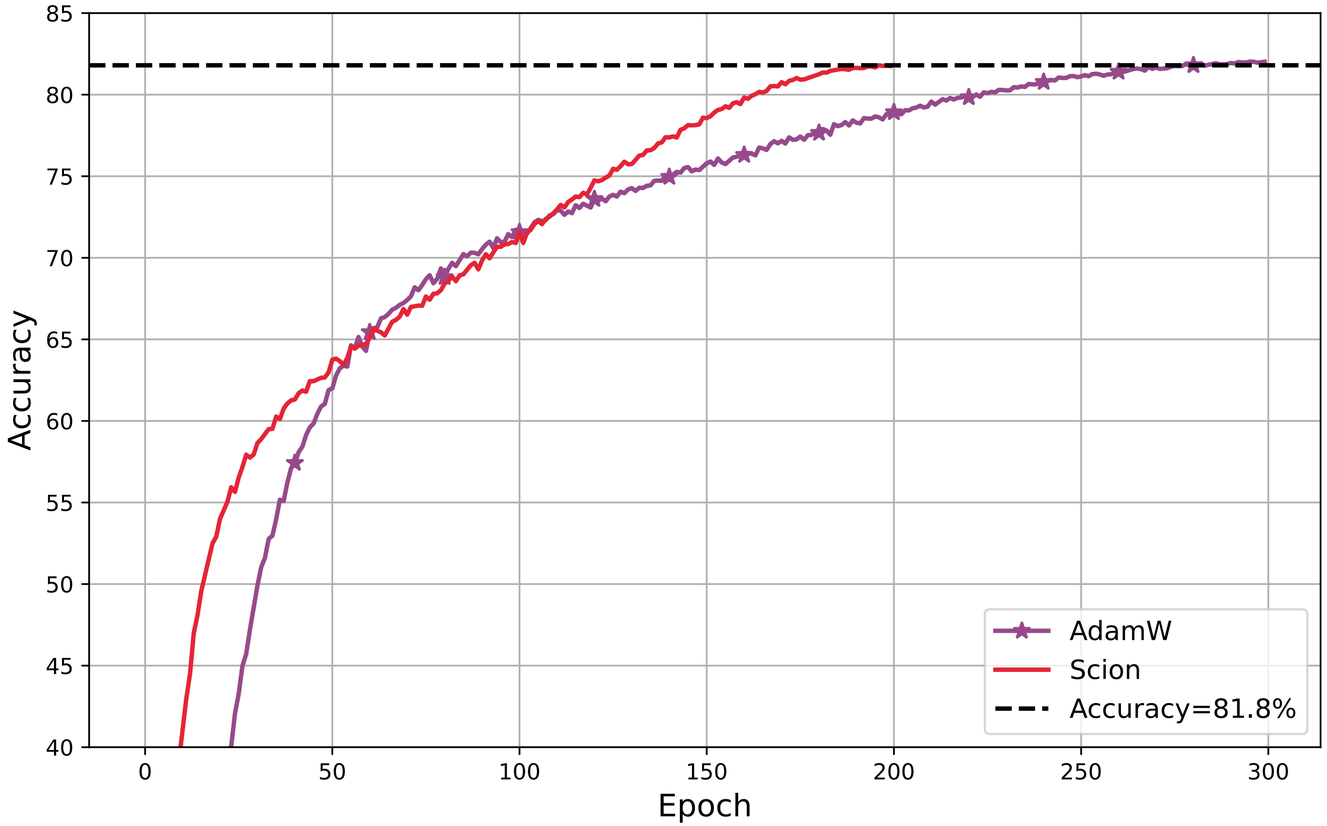
Modifications

- Changes LayerNorm → RMSNorm
- Uses ℓ_2-norm for pos. embeddings
- 4x batch size !

Critical batch size is larger

Speedup (on DeiT-base)

- 30% fewer epochs vs DeiT AdamW
- >80% fewer iterations !
- >40% less wall clock time



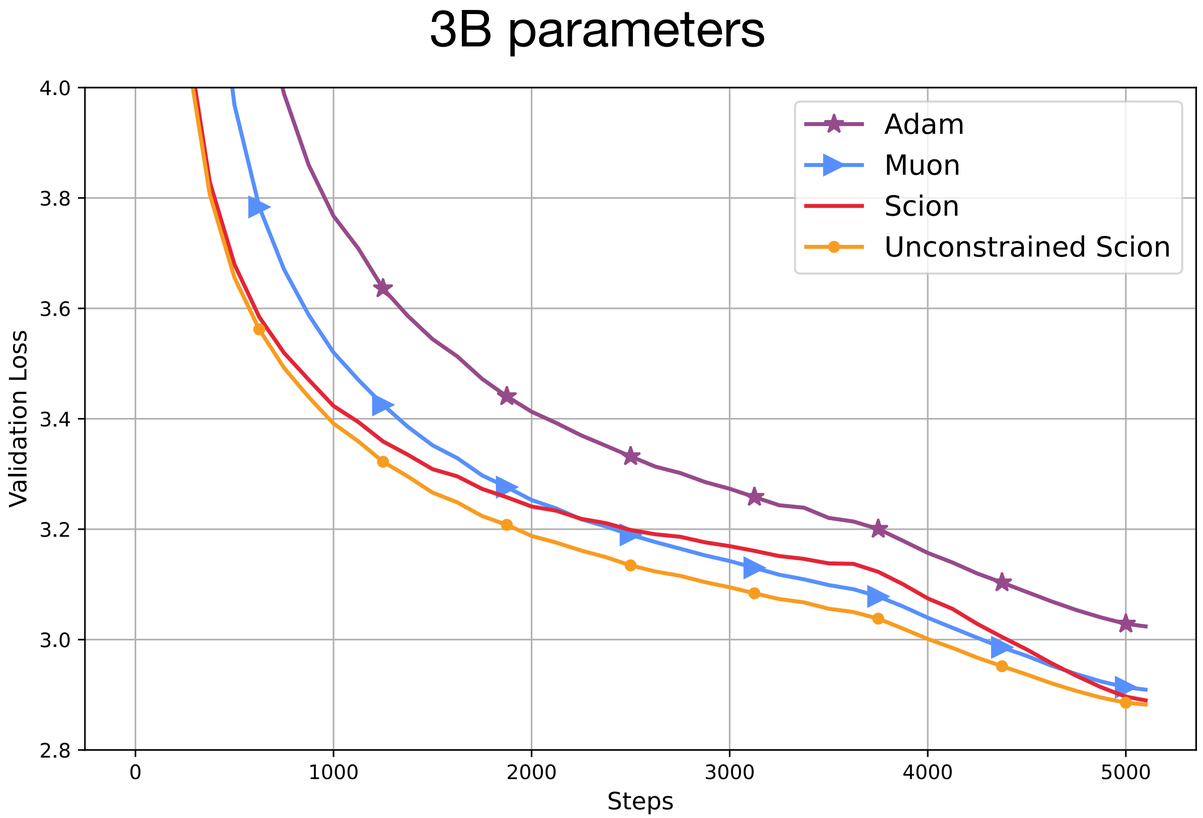
An under-appreciated property: Initial stable behavior

Much more stable initially:

- No learning rate warmup required
- Big improvement initially

Suggests usefulness for non-iid setting:

- RL, continual learning…



Wrap up

Adapt the geometry to the network:

PRO fast

PRO no warmup / initially stable

PRO memory efficient (ScionLight in particular)

PRO large batches

CON can be harder to parallelize \w shared mem.

CON require knowledge of network*

*and sometimes modification

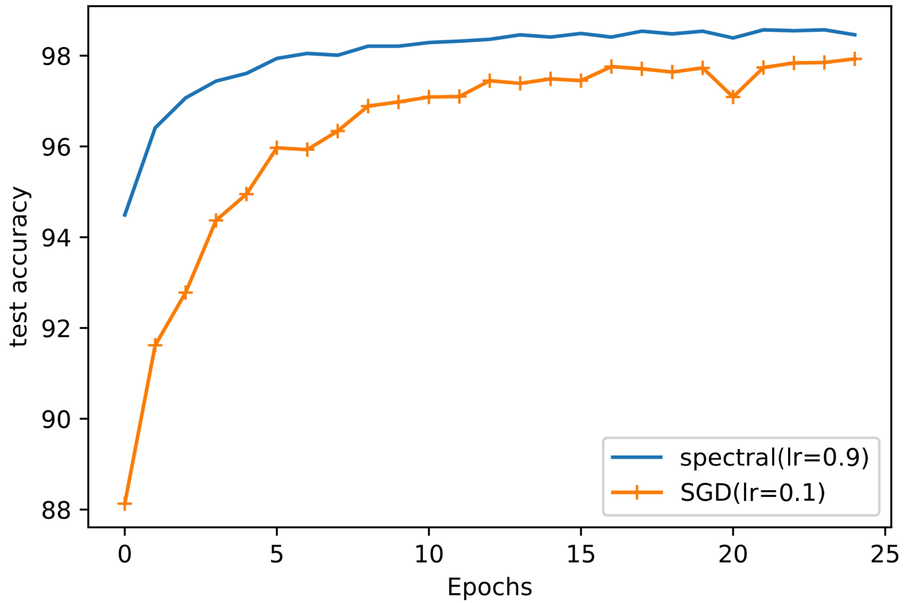
Many open questions:

- How to pick the NN norm (normalization layers, last layer…)
- how to pick the (layerwise) constraint radius ρ_ℓ
- What ideas to import from Frank-Wolfe and normalized gradient literature
- Why unconstrained works despite Lipschitz relying on $\|x\| \leq 1$ [Large et al. 2024]
- …

Appendix

Adam

$$g^k = \nabla f_k(x^{k-1}, \xi_k)$$
$$m^k = \beta_1 m^{k-1} + (1-\beta_1) g^k$$
$$v^k = \beta_2 v^{k-1} + (1-\beta_2)(g^k)^2$$
$$\hat{m}^k = \frac{m^k}{1-\beta_1^k} \quad \text{and} \quad \hat{v}^k = \frac{v^k}{1-\beta_2^k}$$
$$x^k = x^{k-1} - \gamma \frac{\hat{m}^k}{\sqrt{\hat{v}^k} + \epsilon}$$



- MLP on MNIST
- Spectral descent (using SVD)
- Stepsizes have been tuned