Dimension-adapted Momentum Outscales SGD: Compute-optimality

Courtney Paquette, McGill

Joint work: Elliot Paquette (McGill), Damien Ferbach (UdeM, Mila), Gauthier Gidel (UdeM, Mila) and Katie Everett (Google DeepMind, MIT).

Cargese, August 2025

Momentum Mechanism

$$\min_{\theta \in \mathbb{R}^d} \left\{ \mathscr{P}(\theta) = \mathbb{E}_x[\mathscr{R}(\theta; x)] \right\}, \quad \text{where} \quad d = \textbf{parameters}$$

Momentum Mechanism

$$\min_{\theta \in \mathbb{R}^d} \left\{ \mathscr{P}(\theta) = \mathbb{E}_x[\mathscr{R}(\theta; x)] \right\}, \quad \text{where} \quad d = \textbf{parameters}$$

(Streaming) Stochastic gradient descent (SGD): Generate a new data point x_{t+1} and update

$$\theta_{t+1} = \theta_t - \gamma(t, d) \nabla \mathscr{R}(\theta_t; x_{t+1}), \quad \gamma(t, d) \text{ learning rate}$$

Momentum Mechanism

$$\min_{\theta \in \mathbb{R}^d} \left\{ \mathscr{P}(\theta) = \mathbb{E}_x[\mathscr{R}(\theta; x)] \right\}, \quad \text{where} \quad d = \textbf{parameters}$$

(Streaming) Stochastic gradient descent (SGD): Generate a new data point x_{t+1} and update

$$\theta_{t+1} = \theta_t - \gamma(t, d) \nabla \mathscr{R}(\theta_t; x_{t+1}), \quad \gamma(t, d) \text{ learning rate}$$

Momentum Mechanism: History of past gradients

$$\text{(Momentum)} \quad y_t = (1 - \Delta(t, d)) y_{t-1} + \gamma_1(t, d) \nabla \mathscr{R}(\theta_t; x_{t+1}),$$
$$\theta_{t+1} = \theta_t - \gamma_2(t, d) \underbrace{\nabla \mathscr{R}(\theta_t; x_{t+1})}_{SGD} - \gamma_3(t, d) y_t,$$

- $\Delta(t, d)$ momentum parameter, γ_i learning rates
- Widely used within popular algorithms, e.g., Adam algorithm

Momentum Mechanism

$$\min_{\theta \in \mathbb{R}^d} \left\{ \mathscr{P}(\theta) = \mathbb{E}_x[\mathscr{R}(\theta; x)] \right\}, \quad \text{where} \quad d = \textbf{parameters}$$

(Streaming) Stochastic gradient descent (SGD): Generate a new data point x_{t+1} and update

$$\theta_{t+1} = \theta_t - \gamma(t,d)\nabla\mathscr{R}(\theta_t; x_{t+1}), \quad \gamma(t,d) \text{ learning rate}$$

Momentum Mechanism: History of past gradients

$$\text{(Momentum)} \quad y_t = (1 - \Delta(t,d))y_{t-1} + \gamma_1(t,d)\nabla\mathscr{R}(\theta_t; x_{t+1}),$$
$$\theta_{t+1} = \theta_t - \gamma_2(t,d)\underbrace{\nabla\mathscr{R}(\theta_t; x_{t+1})}_{SGD} - \gamma_3(t,d)y_t,$$

- $\Delta(t,d)$ momentum parameter, γ_i learning rates
- Widely used within popular algorithms, e.g., Adam algorithm

Short-term Question: What does momentum gain?
Dream Goal: Design **new, practical** algorithms that work at **scale** based on analysis of high-dimensional SGD

Why Momentum?

$$\text{(Momentum)} \quad y_t = (1 - \Delta(t))y_{t-1} + \gamma_1 \nabla \mathscr{R}(\theta_t; x_{t+1}),$$
$$\theta_{t+1} = \theta_t - \gamma_2 \underbrace{\nabla \mathscr{R}(\theta_t; x_{t+1})}_{SGD} - \gamma_3 y_t,$$

Deterministic Optimization ($\nabla \mathscr{R}$ is full-gradient)

- Benefit: Larger learning rates are stable
- Heavy-ball/Polyak/Classic Momentum: $\gamma_3 \equiv 1$, $\Delta \equiv \gamma_1$ constant

  GD + M **accelerates** over GD for strongly convex functions
- Nesterov acceleration: Acceleration over GD for non-strongly convex

$$\gamma_1(t) = \frac{\tilde{\gamma}(t+1)^2}{t+2} - \frac{t+1}{t+2}\tilde{\gamma}, \quad \gamma_2(t) = \tilde{\gamma}, \quad \gamma_3(t) = \frac{1}{t+1}, \quad \Delta(t) = \frac{1}{t+2}$$

Why Momentum?

$$\text{(Momentum)} \quad y_t = (1-\Delta(t))y_{t-1} + \gamma_1 \nabla \mathscr{R}(\theta_t; x_{t+1}),$$
$$\theta_{t+1} = \theta_t - \gamma_2 \underbrace{\nabla \mathscr{R}(\theta_t; x_{t+1})}_{SGD} - \gamma_3 y_t,$$

Deterministic Optimization ($\nabla \mathscr{R}$ is full-gradient)

- Benefit: Larger learning rates are stable
- Heavy-ball/Polyak/Classic Momentum: $\gamma_3 \equiv 1$, $\Delta \equiv \gamma_1$ constant

  GD + M **accelerates** over GD for strongly convex functions
- Nesterov acceleration: Acceleration over GD for non-strongly convex

$$\gamma_1(t) = \frac{\tilde{\gamma}(t+1)^2}{t+2} - \frac{t+1}{t+2}\tilde{\gamma}, \quad \gamma_2(t) = \tilde{\gamma}, \quad \gamma_3(t) = \frac{1}{t+1}, \quad \Delta(t) = \frac{1}{t+2}$$

Stochastic Optimization ($\nabla \mathscr{R}$ is stochastic)

- Theory: problems with stability
- SGD+M (classical momentum): $\Delta = \gamma_1 = 0.1$ **Why?**

Scaling

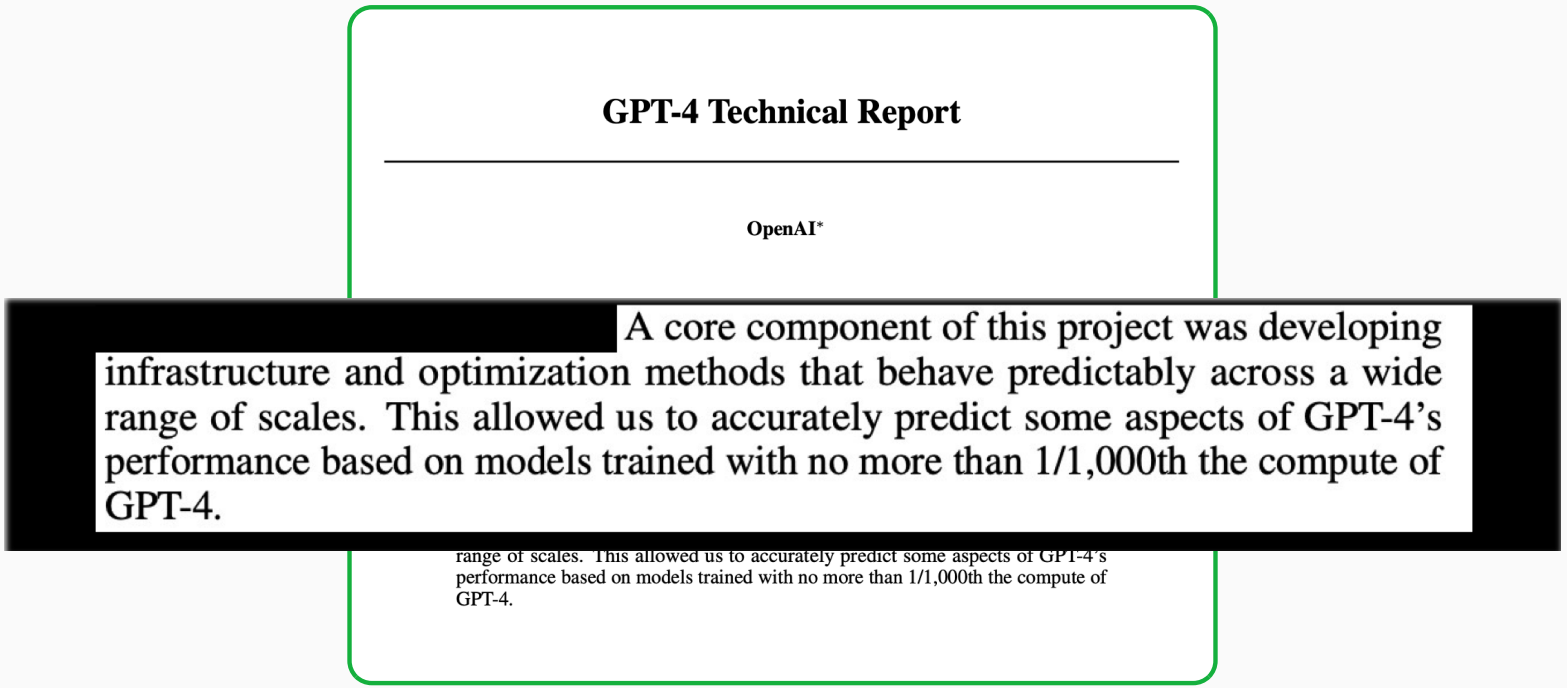
Scaling Laws: precise upper and lower bounds on loss

$\mathscr{P}$ = function(**iterations/samples**, **parameters/# of neurons)**

Scaling

Scaling Laws: precise upper and lower bounds on loss

$\mathscr{P}$ = function(**iterations/samples**, **parameters/# of neurons)**



- Limitation $\rightarrow$ compute (**not** data)
- For given compute budget and infinite data, how do we best allocate resources (e.g. # neurons in a layer / parameters)?

Scaling Law Question

$$\min_{\theta \in \mathbb{R}^d} \left\{ \mathscr{P}(\theta) = \mathbb{E}_x[\mathscr{R}(\theta; x)] \right\}, \quad \text{where} \quad d = \textbf{parameters}$$

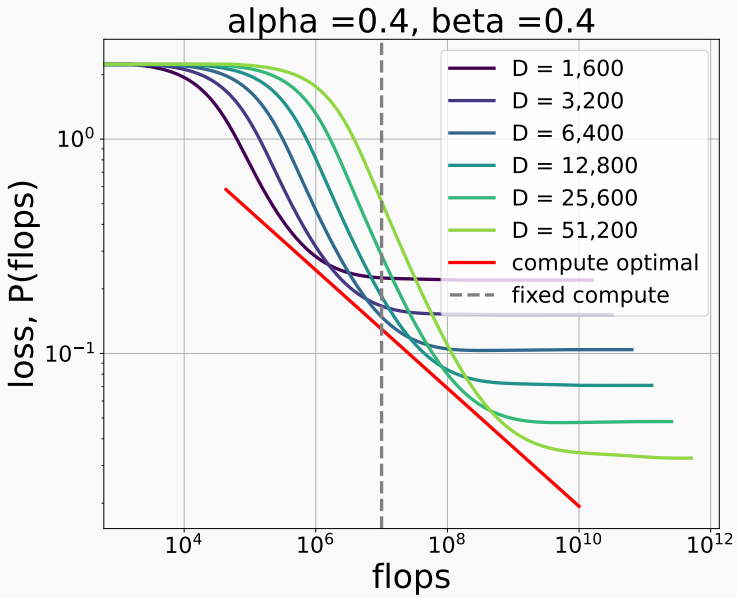
Compute cost (flops ƒ) = iterations of alg. (t) $\times$ **parameters** (d)

Scaling Law Question

$$\min_{\theta \in \mathbb{R}^d} \left\{ \mathscr{P}(\theta) = \mathbb{E}_x[\mathscr{R}(\theta; x)] \right\}, \quad \text{where} \quad d = \textbf{parameters}$$

Compute cost (flops $\mathfrak{f}$) = iterations of alg. (t) $\times$ **parameters** (d)

1. Run SGD for t iterations on different d
2. Plot loss, $\mathscr{P}(\theta_t)$ as a function of flops $\mathfrak{f}$

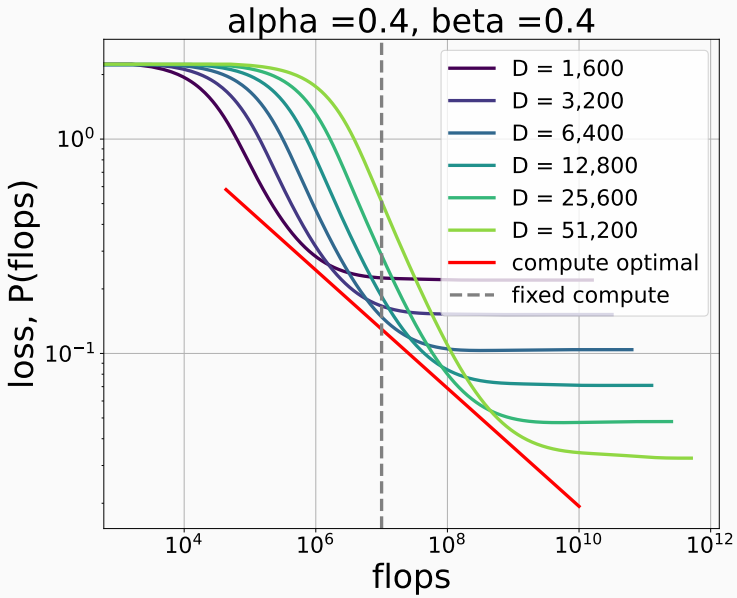


Scaling Law Question

$$\min_{\theta \in \mathbb{R}^d} \left\{ \mathscr{P}(\theta) = \mathbb{E}_x[\mathscr{R}(\theta; x)] \right\}, \quad \text{where} \quad d = \textbf{parameters}$$

Compute cost (flops $\mathfrak{f}$) = iterations of alg. (t) $\times$ **parameters** (d)

1. Run SGD for t iterations on different d
2. Plot loss, $\mathscr{P}(\theta_t)$ as a function of flops $\mathfrak{f}$



Question: Fix # of flops ($\mathfrak{f}$) and infinite data.

How should we choose parameters d so that we get the best loss $\mathscr{P}(\theta_t)$?

$$d^\star(\mathfrak{f}) \in \arg\min_d \mathscr{P}(\tfrac{\mathfrak{f}}{d}; d)$$

For momentum: Can one provable change the exponent σ_1 over SGD?

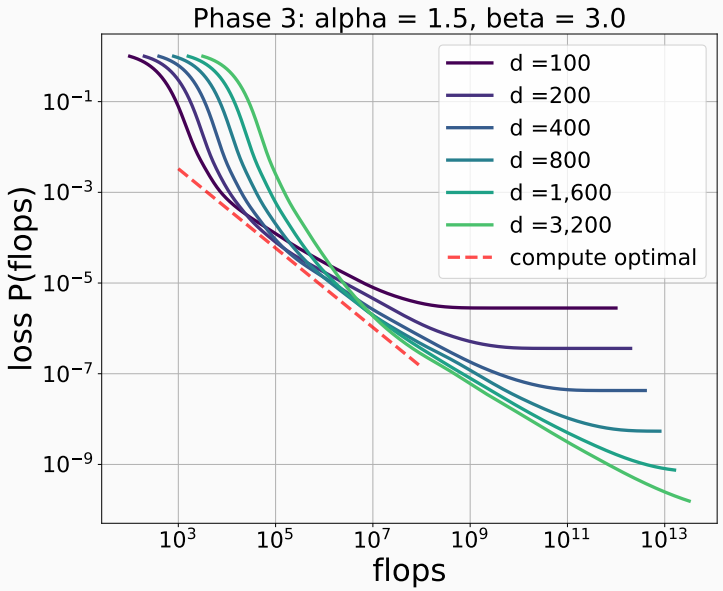
$$d^\star \asymp \mathfrak{f}^{\sigma_1}$$

Scaling Law Question

$$\min_{\theta \in \mathbb{R}^d} \left\{ \mathscr{P}(\theta) = \mathbb{E}_x[\mathscr{R}(\theta; x)] \right\}, \quad \text{where} \quad d = \textbf{parameters}$$

Compute cost (flops $\mathfrak{f}$) = iterations of alg. (t) $\times$ **parameters** (d)

1. Run SGD for t iterations on different d
2. Plot loss, $\mathscr{P}(\theta_t)$ as a function of flops $\mathfrak{f}$



Question: Fix # of flops ($\mathfrak{f}$) and infinite data.

How should we choose parameters d so that we get the best loss $\mathscr{P}(\theta_t)$?

$$d^\star(\mathfrak{f}) \in \arg\min_d \mathscr{P}(\tfrac{\mathfrak{f}}{d}; d)$$

For momentum: Can one provable change the exponent σ_1 over SGD?

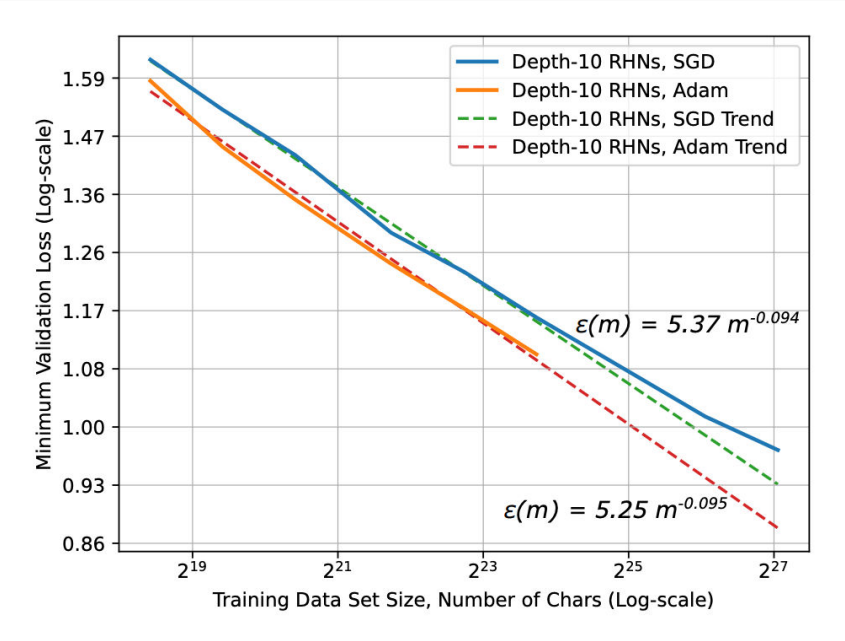
$$d^\star \asymp \mathfrak{f}^{\sigma_1}$$

Can we beat SGD?

Belief: Empirical evidence suggests nothing outscales SGD in the compute-optimality regime.

- Algorithm doesn't matter
- **All** optimizers give the **same exponent!** $\Rightarrow$ Optimizing constants.

Question: Can we devise an algorithm that provable outscales SGD on a simple loss function?



Deep Learning Scaling is Predictable, Empirically, Hestness et al 2017

Momentum framework

General Momentum Algorithm (Gen-Mom-Alg): Generate new sample x_{t+1}

$$\text{(Momentum)} \quad y_t = (1 - \Delta(t))y_{t-1} + \gamma_1 \nabla \mathscr{R}(\theta_t; x_{t+1}),$$

$$\theta_{t+1} = \theta_t - \gamma_2 \underbrace{\nabla \mathscr{R}(\theta_t; x_{t+1})}_{SGD} - \gamma_3 y_t,$$

Hyperparameters

$$\gamma_1(t) \equiv 1, \quad \gamma_2(t,d) = \tilde{\gamma}_2(1+t)^{-\kappa_4} d^{-\kappa_1}, \quad \gamma_3(t,d) = \tilde{\gamma}_3 d^{-\kappa_2}(1+t)^{-\kappa_3},$$

$$\Delta(t) = \delta d^{-\kappa_4}(1+t)^{-\kappa_5}$$

Examples:

- SGD and classical momentum (SGD-M): $\gamma_2 =$ constant, indep. of d
- S-Nesterov acceleration
- Accelerated SGD (Jain, et al. '18, Li et al '23)
- **Dimension-adapted Nesterov Acceleration (DANA):** $\Delta(t) = \delta(1+t)^{-1}$

Momentum framework

General Momentum Algorithm (Gen-Mom-Alg): Generate new sample x_{t+1}

$$\text{(Momentum)} \quad y_t = (1 - \Delta(t))y_{t-1} + \gamma_1 \nabla \mathscr{R}(\theta_t; x_{t+1}),$$
$$\theta_{t+1} = \theta_t - \gamma_2 \underbrace{\nabla \mathscr{R}(\theta_t; x_{t+1})}_{SGD} - \gamma_3 y_t,$$

Hyperparameters

$$\gamma_1(t) \equiv 1, \quad \gamma_2(t,d) = \tilde{\gamma}_2(1+t)^{-\kappa_4} d^{-\kappa_1}, \quad \gamma_3(t,d) = \tilde{\gamma}_3 d^{-\kappa_2}(1+t)^{-\kappa_3},$$
$$\Delta(t) = \delta d^{-\kappa_4}(1+t)^{-\kappa_5}$$

What we need to do: Choose hyperparameters so that they are dimension (d) & time dependent so that we outscale SGD

Simple Model

Power-law Random Features Model (PLRF)

(Maloney, Roberts, and Sully '22)

$$\min_{\theta \in \mathbb{R}^d} \left\{ \mathscr{P}(\theta) \stackrel{\text{def}}{=} \frac{1}{2} \mathbb{E}_x [(x^T W \theta - x^T b)^2] \right\}$$

with *data* $x \in \mathbb{R}^v$ embedded in $\mathbb{R}^d$ by random matrix W, target $b \in \mathbb{R}^v$

Model

- $W \in \mathbb{R}^{v \times d}$, $W_{ij} \sim N(0, 1/d)$
- Model ($\boldsymbol{\alpha}$): $x_j \sim N(0, j^{-2\boldsymbol{\alpha}})$, independent over $1 \leq j \leq v$
- Targets ($\boldsymbol{\beta}$): $b_j = j^{-\boldsymbol{\beta}}$
- $\begin{cases} v \to \infty, & 2\boldsymbol{\alpha} > 1 \\ \frac{v}{d} \to r > 1, & 2\boldsymbol{\alpha} < 1 \end{cases}$
- $\hat{K} \stackrel{\text{def}}{=} DWW^T D$, $D = \text{Diag}(j^{-\alpha} : 1 \leq j \leq v)$

Model continued...

Goal: Compute optimal $= \min_d \mathscr{P}(\frac{\mathrm{f}}{d}, d; \alpha, \beta)$

Find $\sigma_1(\alpha, \beta)$, $\sigma_2(\alpha, \beta)$ s.t. $\mathcal{P}^\star(\frac{\mathrm{f}}{d^\star}, d^\star) \asymp \mathrm{f}^{\sigma_1}$ & $d^\star \in \arg\min_d \mathcal{P}(\frac{\mathrm{f}}{d}, d) \asymp \mathrm{f}^{\sigma_2}$

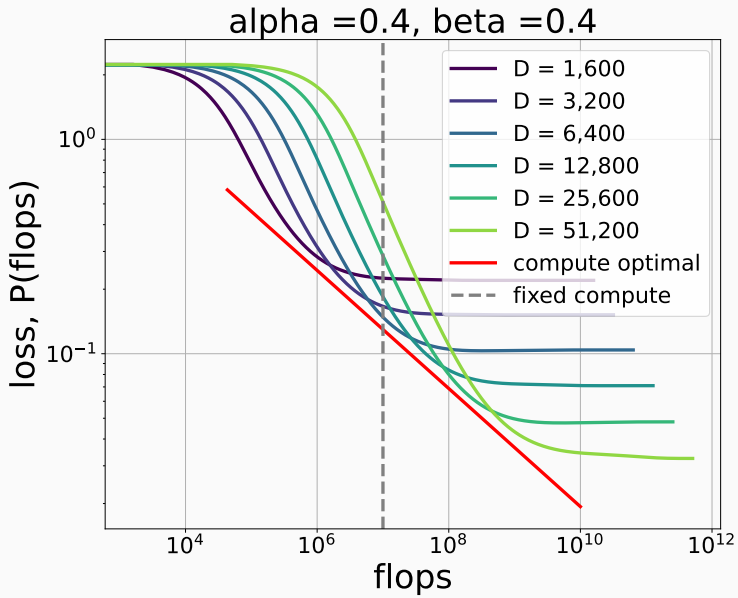
Model continued...

Goal: Compute optimal $= \min_d \mathscr{P}(\frac{\mathfrak{f}}{d}, d; \alpha, \beta)$

Find $\sigma_1(\alpha,\beta)$, $\sigma_2(\alpha,\beta)$ s.t. $\mathcal{P}^\star(\frac{\mathfrak{f}}{d^\star}, d^\star) \asymp \mathfrak{f}^{\sigma_1}$ & $d^\star \in \arg\min_d \mathcal{P}(\frac{\mathfrak{f}}{d}, d) \asymp \mathfrak{f}^{\sigma_2}$

Previous work:

- Bordelon, Atanasov, Pehlevan '24 (Grad. flow version)
- Maloney, Roberts, Sully '22 (This model, no compute constraint)
- Hoffman et al, (Chinchilla Paper) '22 (Compute optimal, empirical)
- Kaplan et al, '20. Bahri et al '21 (this model, no compute constraint).
- Lin, et al, '24 (concurrent). Only proved Phase Ia w/ order 1 target noise for SGD



- Related to 'source' and 'capacity'. Minimax optimal insufficient ($d \to \infty$)

Classical (stochastic) Momentum (SGD-M)

SGD-M: Generate new sample x_{t+1}

$$\text{(Momentum)} \quad y_t = (1-\delta)y_{t-1} + \nabla\mathscr{R}(\theta_t; x_{t+1}),$$

$$\theta_{t+1} = \theta_t - \gamma_2 \underbrace{\nabla\mathscr{R}(\theta_t; x_{t+1})}_{SGD} - \gamma_3 y_t,$$

Hyperparameters

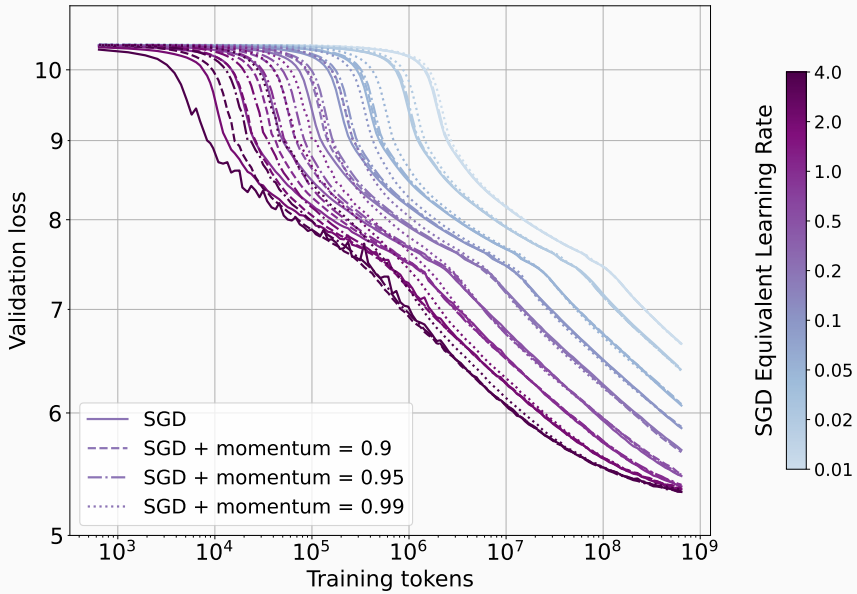
$\gamma_2(t) = \tilde{\gamma}_2 d^{-\kappa_1}, \quad \gamma_3(t) = \tilde{\gamma}_3 d^{-\kappa_2}, \quad \Delta(t) = \delta$ constant, independent of d

Exact **same** scaling law as SGD!

- Equivalent dynamics

$$\gamma^{\text{SGD}} = \gamma_2 + \frac{\gamma_3}{\delta}$$

- Holds on LSTM with C4 data set



Nesterov Acceleration

S-Nesterov: Generate new sample x_{t+1}

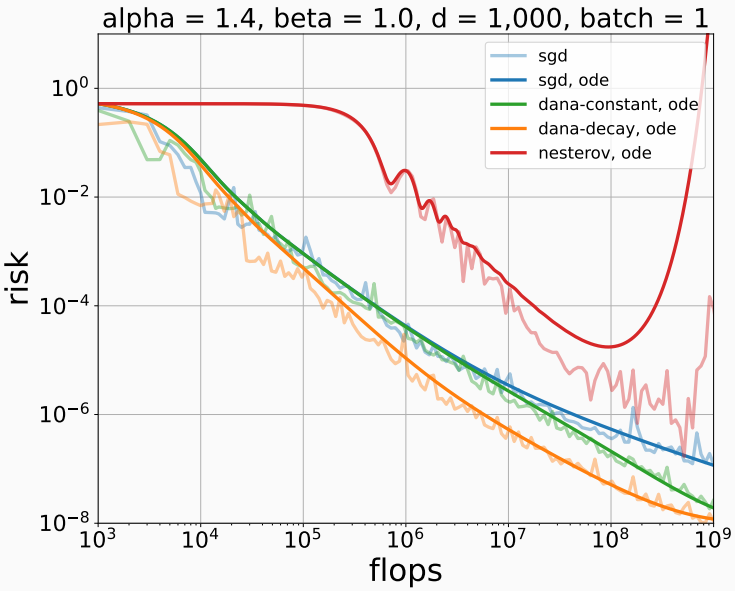
$$\text{(Momentum)} \quad y_t = (1 - \Delta(t))y_{t-1} + \gamma_1(t, d)\nabla\mathscr{R}(\theta_t; x_{t+1}),$$

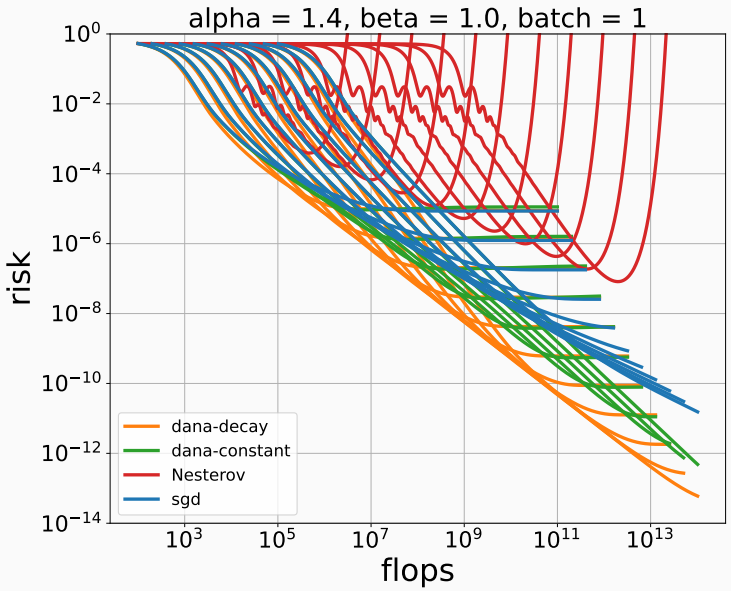
$$\theta_{t+1} = \theta_t - \gamma_2 \underbrace{\nabla\mathscr{R}(\theta_t; x_{t+1})}_{SGD} - \gamma_3(t, d)y_t,$$

Hyperparameters:

$$\gamma_1(t) = \frac{\tilde{\gamma}(t+1)^2}{t+2} - \frac{t+1}{t+2}\tilde{\gamma}, \quad \gamma_2(t) = \tilde{\gamma}, \quad \gamma_3(t) = \frac{1}{t+1}, \quad \Delta(t) = \frac{1}{t+2}$$

Diverges on PLRF for any $\tilde{\gamma}$! see e.g., Theorem 7, (Even, et al., 2021)





DANA-constant: Stable Nesterov

$$\text{(DANA)} \qquad y_t = (1 - \delta(1+t)^{-1})y_{t-1} + \nabla \mathscr{P}(\theta_t; x_{t+1}),$$
$$\theta_{t+1} = \theta_t - \gamma_2(t,d)\nabla \mathscr{P}(\theta_t; x_{t+1}) - \underbrace{c_3 d^{-\kappa_2}(1+t)^{-\kappa_3}}_{\gamma_3(t,d)} y_t,$$

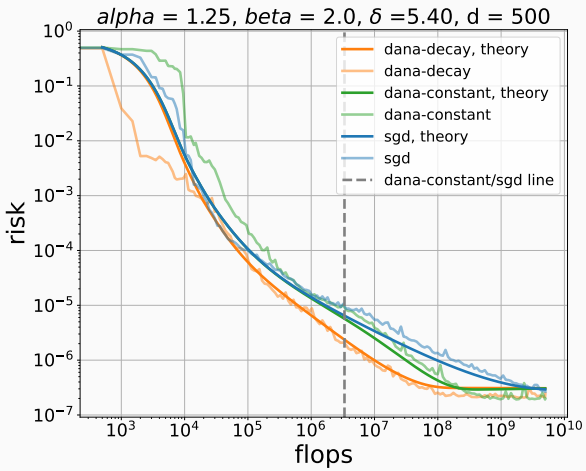
- **Momentum:** exponent is thresholded by δ; big δ = better, $\delta > 5$.
- **Learning rate:** $\gamma_2(t,d) < \frac{1}{\mathrm{tr}}$, $\mathrm{tr} = \sum_{j=1}^{d} j^{-2\alpha}$
- **DANA-constant**[1]: If γ_3 constant ($\kappa_3 = 0$), need $\gamma_3(t,d) \asymp 1/d$

DANA-constant: Stable Nesterov

$$\text{(DANA)} \qquad y_t = (1 - \delta(1+t)^{-1})y_{t-1} + \nabla \mathscr{P}(\theta_t; x_{t+1}),$$
$$\theta_{t+1} = \theta_t - \gamma_2(t,d)\nabla \mathscr{P}(\theta_t; x_{t+1}) - \underbrace{c_3 d^{-\kappa_2}(1+t)^{-\kappa_3}}_{\gamma_3(t,d)} y_t,$$

- **Momentum:** exponent is thresholded by δ; big δ = better, $\delta > 5$.
- **Learning rate:** $\gamma_2(t,d) < \frac{1}{\text{tr}}$, $\text{tr} = \sum_{j=1}^{d} j^{-2\alpha}$
- **DANA-constant**[1]: If γ_3 constant ($\kappa_3 = 0$), need $\gamma_3(t,d) \asymp 1/d$

Problem: Need $t \gtrsim d$ for momentum to have any effect



DANA-constant behaves like SGD for $t \leq d$

[1] Related algorithm (Varre & Flammarion, '22), (Paquette[2], '21)

DANA-decaying: Improving DANA-constant

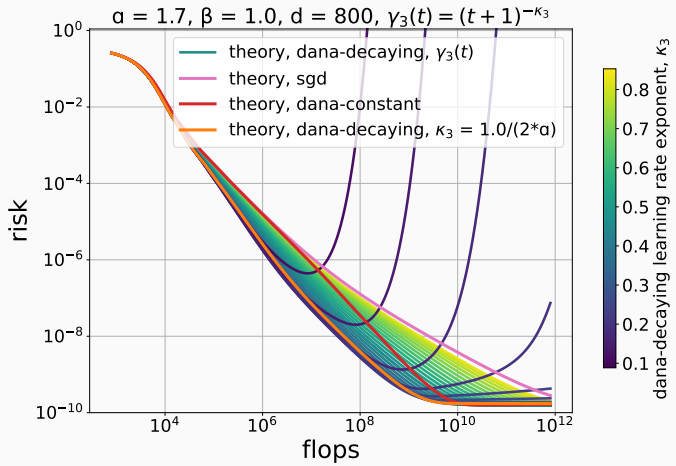
$$\text{(DANA)} \qquad y_t = (1 - \delta(1+t)^{-1})y_{t-1} + \nabla\mathscr{R}(\theta_t; x_{t+1}),$$
$$\theta_{t+1} = \theta_t - \gamma_2(t,d)\nabla\mathscr{R}(\theta_t; x_{t+1}) - \underbrace{c_3 d^{-\kappa_2}(1+t)^{-\kappa_3}}_{\gamma_3(t,d)} y_t,$$

- **Momentum:** exponent is thresholded by δ; big δ = better, $\delta > 5$.
- **Learning rate:** $\gamma_2(t,d) < \frac{1}{\mathrm{tr}}$, $\mathrm{tr} = \sum_{j=1}^{d} j^{-2\alpha}$
- **DANA-decaying^2:** $\gamma_3(t,d) = (1+t)^{-1/(2\alpha)}$ verge of stability

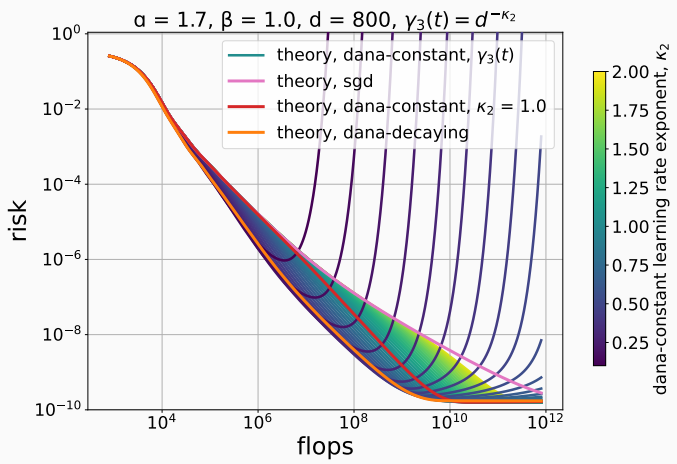
DANA-decaying: Improving DANA-constant

$$\text{(DANA)} \qquad y_t = (1 - \delta(1+t)^{-1})y_{t-1} + \nabla\mathscr{R}(\theta_t; x_{t+1}),$$
$$\theta_{t+1} = \theta_t - \gamma_2(t,d)\nabla\mathscr{R}(\theta_t; x_{t+1}) - \underbrace{c_3 d^{-\kappa_2}(1+t)^{-\kappa_3}}_{\gamma_3(t,d)} y_t,$$

- **Momentum:** exponent is thresholded by δ; big δ = better, $\delta > 5$.
- **Learning rate:** $\gamma_2(t,d) < \frac{1}{\text{tr}}$, $\text{tr} = \sum_{j=1}^{d} j^{-2\alpha}$
- **DANA-decaying**[2]**:** $\gamma_3(t,d) = (1+t)^{-1/(2\alpha)}$ verge of stability



$$\gamma_3(t) \asymp (1+t)^{-\kappa_3}$$



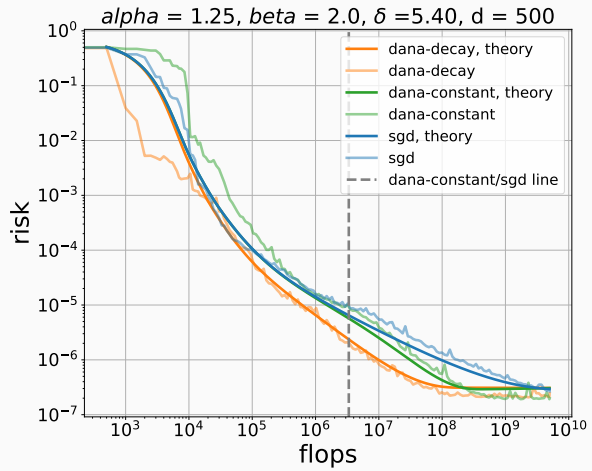
$$\gamma_3(t,d) \asymp d^{-\kappa_2}$$

[2] Related heuristic (Yarotsky & Velikanov, '24)

DANA-decaying: Improving DANA-constant

$$\text{(DANA)} \qquad y_t = (1 - \delta(1+t)^{-1})y_{t-1} + \nabla\mathscr{R}(\theta_t; x_{t+1}),$$
$$\theta_{t+1} = \theta_t - \gamma_2(t,d)\nabla\mathscr{R}(\theta_t; x_{t+1}) - \underbrace{c_3 d^{-\kappa_2}(1+t)^{-\kappa_3}}_{\gamma_3(t,d)} y_t,$$

- **Momentum:** exponent is thresholded by δ; big δ = better, $\delta > 5$.
- **Learning rate:** $\gamma_2(t,d) < \frac{1}{\text{tr}}$, $\text{tr} = \sum_{j=1}^{d} j^{-2\alpha}$
- **DANA-decaying[2]:** $\gamma_3(t,d) = (1+t)^{-1/(2\alpha)}$ verge of stability



- DANA-decaying fixed DANA-constant's problem
- $(\kappa_2, \kappa_3) = (0, 1) \Rightarrow$ Schedule-free SGD (no acceleration)

[2] Related heuristic (Yarotsky & Velikanov, '24)

Summary Statistics

Summary statistics: $(\lambda_j, \omega_j)_{j=1}^d$ eigenvalue-eigenvector pair of $W^T D^2 W \in \mathbb{R}^{d\times d}$; (Θ_t, Y_t) is 'continuized' version of (θ_t, y_t); $D = \text{Diag}(i^{-\alpha}, i = 1, \ldots, d)$, $b = W\check{b} + \dot{b}$ s.t. $W^T D^2 \dot{b} = 0$

$$\rho_j^2(t) = \mathbb{E}[\langle \omega_j^{\otimes 2}, \Theta_t - \check{b}\rangle | W], \quad \xi_j^2(t) = \mathbb{E}[\langle \omega_j^{\otimes 2}, Y_t\rangle | W]$$

$$\text{and} \qquad \chi_j(t) = \mathbb{E}[\langle \omega_j^{\otimes 2}, Y_t \otimes (\Theta_t - \check{b})\rangle | W]$$

Note: $$\mathbb{E}[\mathscr{P}(\Theta_t)|W] = \sum_{j=1}^{d} \rho_j^2(t) + \mathbb{E}[\mathscr{P}(\Theta_\infty)|W]$$

Summary Statistics

Summary statistics: $(\lambda_j, \omega_j)_{j=1}^d$ eigenvalue-eigenvector pair of $W^T D^2 W \in \mathbb{R}^{d\times d}$; (Θ_t, Y_t) is 'continuized' version of (θ_t, y_t);

$D = \text{Diag}(i^{-\alpha}, i = 1, \ldots, d)$, $b = W\check{b} + \dot{b}$ s.t. $W^T D^2 \dot{b} = 0$

$$\rho_j^2(t) = \mathbb{E}[\langle \omega_j^{\otimes 2}, \Theta_t - \check{b}\rangle | W], \quad \xi_j^2(t) = \mathbb{E}[\langle \omega_j^{\otimes 2}, Y_t\rangle | W]$$

$$\text{and} \qquad \chi_j(t) = \mathbb{E}[\langle \omega_j^{\otimes 2}, Y_t \otimes (\Theta_t - \check{b})\rangle | W]$$

$$\textbf{Note:} \quad \mathbb{E}[\mathscr{P}(\Theta_t)|W] = \sum_{j=1}^d \rho_j^2(t) + \mathbb{E}[\mathscr{P}(\Theta_\infty)|W]$$

For each (λ_j, ω_j), there exists a 3×3 system of linear ODEs

$$\frac{d}{dt}\begin{pmatrix}\rho_j^2\\ \xi_j^2\\ \chi_j\end{pmatrix} = A(\gamma_i(t,d), \Delta(t,d), \lambda_j)\begin{pmatrix}\rho_j^2\\ \xi_j^2\\ \chi_j\end{pmatrix} + \begin{pmatrix}\gamma_2^2\lambda_j\\ \gamma_1^2\lambda_j\\ 0\end{pmatrix}\mathbb{E}[\mathscr{P}(\Theta_t)|W]$$

Summary Statistics

Summary statistics: $(\lambda_j, \omega_j)_{j=1}^d$ eigenvalue-eigenvector pair of $W^T D^2 W \in \mathbb{R}^{d\times d}$; (Θ_t, Y_t) is 'continuized' version of (θ_t, y_t); $D = \text{Diag}(i^{-\alpha}, i = 1, \dots, d)$, $b = W\check{b} + \dot{b}$ s.t. $W^T D^2 \dot{b} = 0$

$$\rho_j^2(t) = \mathbb{E}[\langle \omega_j^{\otimes 2}, \Theta_t - \check{b}\rangle | W], \quad \xi_j^2(t) = \mathbb{E}[\langle \omega_j^{\otimes 2}, Y_t\rangle | W]$$

$$\text{and} \qquad \chi_j(t) = \mathbb{E}[\langle \omega_j^{\otimes 2}, Y_t \otimes (\Theta_t - \check{b})\rangle | W]$$

$$\textbf{Note:} \quad \mathbb{E}[\mathscr{P}(\Theta_t)|W] = \sum_{j=1}^d \rho_j^2(t) + \mathbb{E}[\mathscr{P}(\Theta_\infty)|W]$$

For each (λ_j, ω_j), there exists a 3×3 system of linear ODEs

$$\frac{d}{dt}\begin{pmatrix}\rho_j^2\\ \xi_j^2\\ \chi_j\end{pmatrix} = A(\gamma_i(t,d), \Delta(t,d), \lambda_j)\begin{pmatrix}\rho_j^2\\ \xi_j^2\\ \chi_j\end{pmatrix} + \begin{pmatrix}\gamma_2^2\lambda_j\\ \gamma_1^2\lambda_j\\ 0\end{pmatrix}\mathbb{E}[\mathscr{P}(\Theta_t)|W]$$

- **SGD:** Only ρ_j^2 and $A(t, \lambda_j, d) \in \mathbb{R}$ and so explicit
- Much harder for momentum $\Rightarrow$ Duhamel's principle

Volterra Equation

Theorem (**C.P.**-E. Paquette-Xiao-Pennington, '24) Let $\hat{K} = DWW^T D, D = \text{Diag}(j^{-\alpha})$.

Expected loss under Gen-Mom-Alg. satisfies a **Volterra equation**

$$\mathbb{E}[\mathscr{P}(\Theta_t)\,|W] = \underbrace{\mathscr{F}(\Theta_t)}_{full-batch} + \underbrace{\mathscr{K} * \mathbb{E}[\mathscr{P}(\Theta_t)\,|W]}_{\text{variance/noise}}$$

- Forcing function: $\mathscr{F}(t) = (Db)^T f(t, d, \text{eigs. } \hat{K})(Db)$
- Kernel: $\mathscr{K}(t,s) = g(t, s, d, \text{eigs. } \hat{K})$

Volterra Equation

Theorem (**C.P.**-E. Paquette-Xiao-Pennington, '24) Let $\hat{K} = DWW^T D, D = \text{Diag}(j^{-\alpha})$.

Expected loss under Gen-Mom-Alg. satisfies a **Volterra equation**

$$\mathbb{E}[\mathscr{P}(\Theta_t)\,|W] = \underbrace{\mathscr{F}(\Theta_t)}_{full-batch} + \underbrace{\mathscr{K} * \mathbb{E}[\mathscr{P}(\Theta_t)\,|W]}_{\text{variance/noise}}$$

- Forcing function: $\mathscr{F}(t) = (Db)^T f(t, d, \text{eigs. } \hat{K})(Db)$
- Kernel: $\mathscr{K}(t,s) = g(t, s, d, \text{eigs. } \hat{K})$

Idea: Replace $\hat{K}$ w/ **deterministic equivalent** ($\mathbb{E}_W[(\hat{K} - z)^{-1}] \approx \mathcal{R}(z)$, $z \in \mathbb{C}$)

$$m(z) \overset{\text{def}}{=} \frac{1}{1 + \frac{1}{d}\sum_{j=1}^{v} \frac{j^{-2\alpha}}{j^{-2\alpha} m(z) - z}} \quad \text{where} \quad \mathcal{R}(z) \overset{\text{def}}{=} \text{Diag}\left(\frac{1}{j^{-2\alpha} m(z) - z}, 1 \le j \le v\right)$$

Volterra Equation

Theorem (**C.P.**-E. Paquette-Xiao-Pennington, '24) Let $\hat{K} = DWW^T D$, $D = \mathrm{Diag}(j^{-\alpha})$.

Expected loss under Gen-Mom-Alg. satisfies a **Volterra equation**

$$\mathbb{E}[\mathscr{P}(\Theta_t)\,|W] = \underbrace{\mathscr{F}(\Theta_t)}_{full-batch} + \underbrace{\mathscr{K} * \mathbb{E}[\mathscr{P}(\Theta_t)\,|W]}_{\text{variance/noise}}$$

- Forcing function: $\mathscr{F}(t) = (Db)^T f(t, d, \text{eigs. } \hat{K})(Db)$
- Kernel: $\mathscr{K}(t,s) = g(t, s, d, \text{eigs. } \hat{K})$

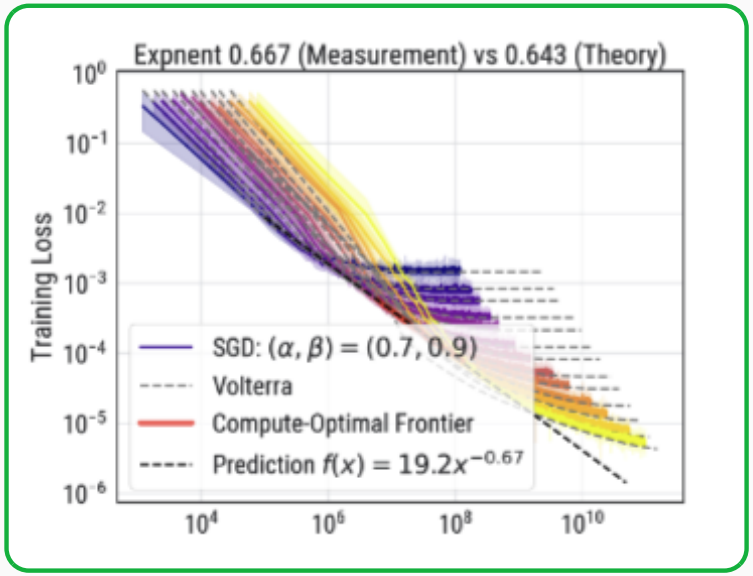
Idea: Replace $\hat{K}$ w/ **deterministic equivalent** ($\mathbb{E}_W[(\hat{K} - z)^{-1}] \approx \mathcal{R}(z)$, $z \in \mathbb{C}$)

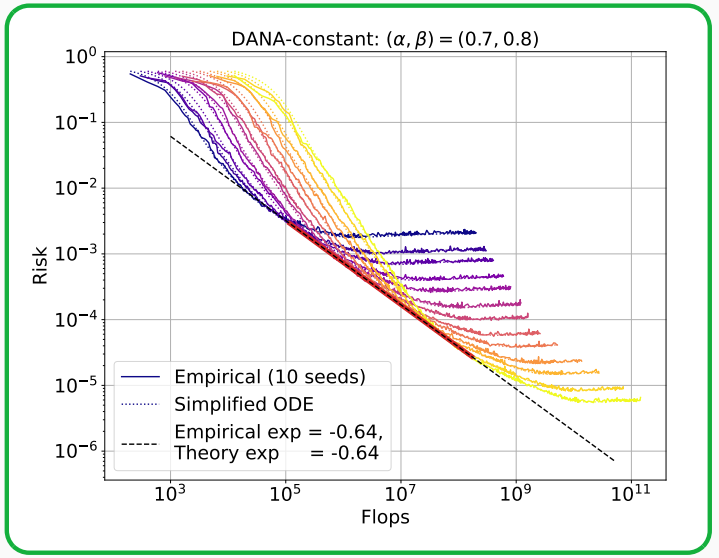
Deterministic equivalent solves convolution Volterra equation

$$\mathcal{P}(t) = \underbrace{\mathcal{F}(t)}_{\text{full-batch}} + \underbrace{(\mathcal{K} * \mathcal{P})(t)}_{\text{variance/noise}}.$$

- Forcing function: $\mathcal{F}(t) \stackrel{\text{def}}{=} -1/(2\pi i) \oint f(t,z)(Db)^T \mathcal{R}(z) Db)\, \mathrm{d}z$
- Kernel: $\mathcal{K}(t,s) \stackrel{\text{def}}{=} -1/(2\pi i) \oint g(t,s,z)\mathcal{R}(z)\, \mathrm{d}z$

Volterra equation vs. Gen-Mom-Alg





flops (x-axis) vs. training loss (y-axis)

Volterra equation is a **good predictor** of Gen-Mom-Alg. behavior!

Bounding the Volterra

Theorem (**C.P.**-E. Paquette-Xiao-Pennington, '24) Suppose stable algorithm. There exists an $M > 0$ large and constants $c = c(\alpha, \beta, M)$, $C = C(\alpha, \beta, M)$, independent of d, so that for all admissible v and d, for all $\gamma t > M$,

$$c \times \left(\mathcal{F}(t) + \frac{1}{\gamma} \cdot \mathcal{K}(t)\right) \leq \mathcal{P}(t) \leq C \times \Big(\overbrace{\mathcal{F}(t)}^{\text{forcing func.}} + \frac{1}{\gamma} \cdot \overbrace{\mathcal{K}(t)}^{\text{kernel func.}} \Big).$$

$$\underbrace{\qquad}_{\text{full batch}} \qquad\qquad \underbrace{\qquad}_{\text{SGD noise}}$$

Remarks:

- $\gamma = \gamma_2$ (DANA) and $\gamma_2 + \frac{\gamma_3}{\delta}$ (SGD-M)
- Suffices to understand forcing function, $\mathcal{F}$, and kernel, $\mathcal{K}$
- Need assumptions on learning rates γ_i and momentum parameter $\Delta(t)$ (e.g., decreasing in t) for Gen-Mom-Alg.

Bounding the Volterra

Theorem (**C.P.**-E. Paquette-Xiao-Pennington, '24) Suppose stable algorithm.

There exists an $M > 0$ large and constants $c = c(\alpha, \beta, M)$, $C = C(\alpha, \beta, M)$, independent of d, so that for all admissible v and d, for all $\gamma t > M$,

$$c \times \left(\mathcal{F}(t) + \frac{1}{\gamma} \cdot \mathcal{K}(t)\right) \leq \mathcal{P}(t) \leq C \times \Big(\overbrace{\mathcal{F}(t)}^{\text{forcing func.}} + \frac{1}{\gamma} \cdot \overbrace{\mathcal{K}(t)}^{\text{kernel func.}} \Big).$$

$$\underbrace{\qquad}_{\text{full batch}} \qquad \underbrace{\qquad}_{\text{SGD noise}}$$

Remarks:

- $\gamma = \gamma_2$ (DANA) and $\gamma_2 + \frac{\gamma_3}{\delta}$ (SGD-M)
- Suffices to understand forcing function, $\mathcal{F}$, and kernel, $\mathcal{K}$
- Need assumptions on learning rates γ_i and momentum parameter $\Delta(t)$ (e.g., decreasing in t) for Gen-Mom-Alg.

Stability conditions:

$$\mathcal{F} \text{ is bounded } \textbf{and } \|\mathcal{K}\|_{\ell_1} < 1.$$

Estimating forcing $\mathcal{F}$ and kernel $\mathcal{K}$

Force: $\mathcal{F}(t) \asymp \hat{\mathcal{F}}_0(\vartheta(t)) + \hat{\mathcal{F}}_{pp}(\vartheta(t)) + \hat{\mathcal{F}}_{ac}(\vartheta(t))$ & **Kernel:** $\mathcal{K}(t) \asymp \hat{\mathcal{K}}_{pp}(\vartheta(t))$

Explicit (w/ simple formulas), and given by $c \times t^{-\tau} d^{-\sigma}$, some $c, \tau, \sigma > 0$

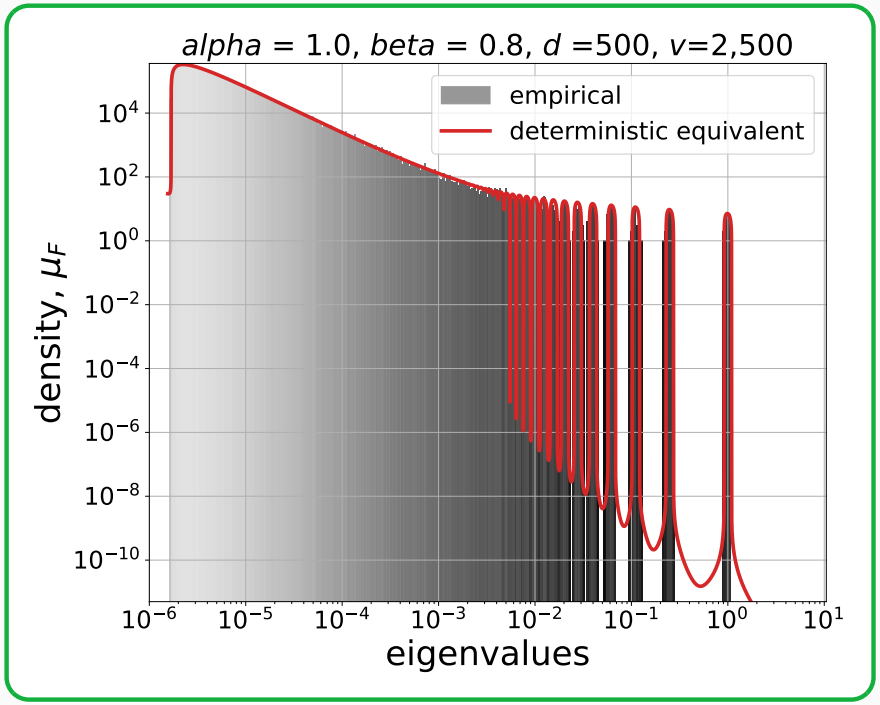
Estimating forcing $\mathcal{F}$ and kernel $\mathcal{K}$

Force: $\mathcal{F}(t) \asymp \hat{\mathcal{F}}_0(\vartheta(t)) + \hat{\mathcal{F}}_{pp}(\vartheta(t)) + \hat{\mathcal{F}}_{ac}(\vartheta(t))$ & **Kernel:** $\mathcal{K}(t) \asymp \hat{\mathcal{K}}_{pp}(\vartheta(t))$

Explicit (w/ simple formulas), and given by $c \times t^{-\tau} d^{-\sigma}$, some $c, \tau, \sigma > 0$

$\langle \mathscr{R}(z; DWW^T D), (D^{1/2} b)^{\otimes 2} \rangle, D = \mathrm{Diag}(j^{-\alpha})$

1. $\hat{\mathcal{F}}_0(t) = \hat{\mathcal{F}}_0(0)$ irreducible loss level
2. $\hat{\mathcal{F}}_{pp}(t)$ spikes (pure point part)
3. $\hat{\mathcal{F}}_{ac}(t)$ result of feature misalignment, matrix W causes leading features to be distorted

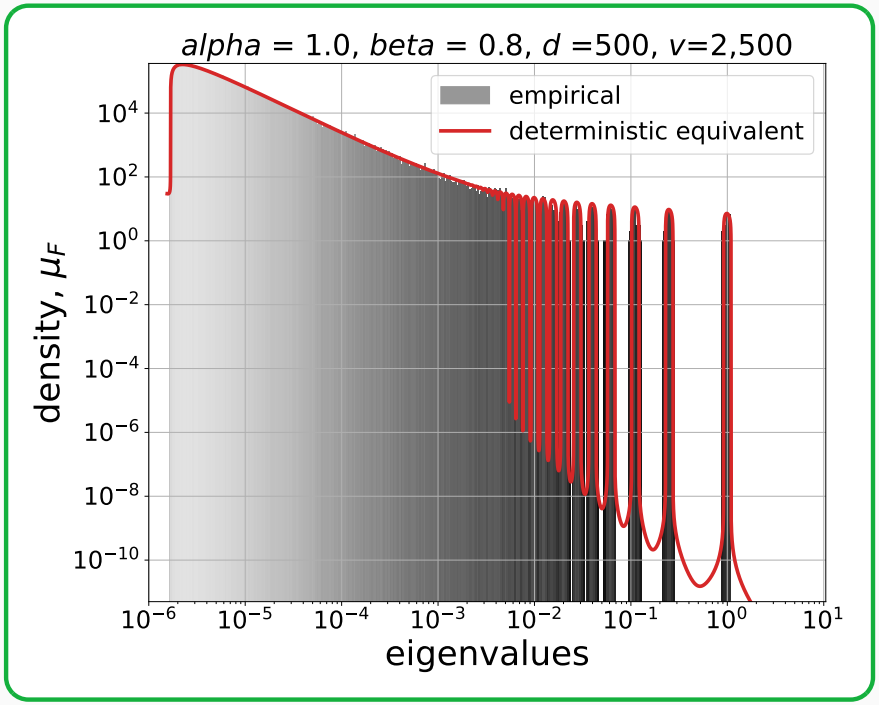


Estimating forcing $\mathcal{F}$ and kernel $\mathcal{K}$

Force: $\mathcal{F}(t) \asymp \hat{\mathcal{F}}_0(\vartheta(t)) + \hat{\mathcal{F}}_{pp}(\vartheta(t)) + \hat{\mathcal{F}}_{ac}(\vartheta(t))$ & **Kernel:** $\mathcal{K}(t) \asymp \hat{\mathcal{K}}_{pp}(\vartheta(t))$

Explicit (w/ simple formulas), and given by $c \times t^{-\tau} d^{-\sigma}$, some $c, \tau, \sigma > 0$

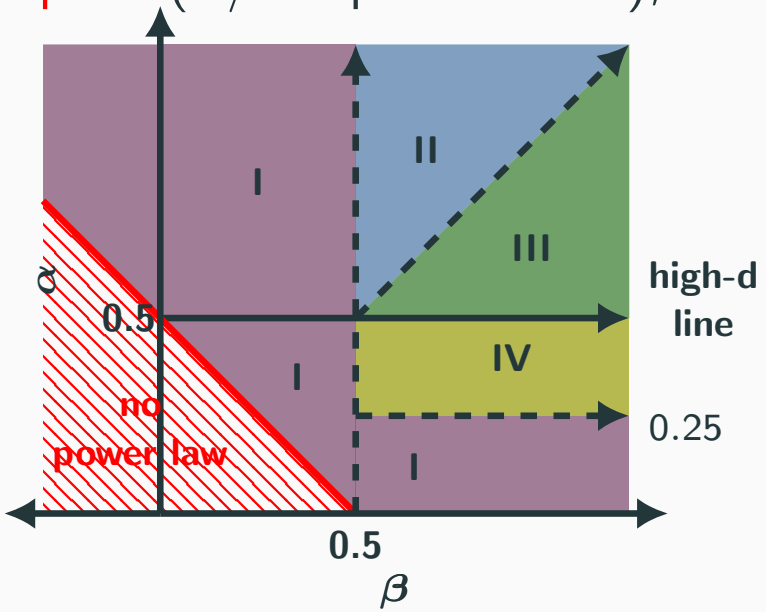
1. $\hat{\mathcal{F}}_0(t) = \hat{\mathcal{F}}_0(0)$ irreducible loss level
2. $\hat{\mathcal{F}}_{pp}(t)$ spikes (pure point part)
3. $\hat{\mathcal{F}}_{ac}(t)$ result of feature misalignment, matrix W causes leading features to be distorted
4. $\hat{\mathcal{K}}_{pp}(t)$ excess risk due to 1 unit of noise, projected into the eigenspaces of the outliers



Compute-Optimal Curves

Force: $\hat{\mathcal{F}}(t) \asymp \hat{\mathcal{F}}_0(t) + \hat{\mathcal{F}}_{pp}(t) + \hat{\mathcal{F}}_{ac}(t)$ & **Kernel:** $\hat{\mathcal{K}}(t) \asymp \hat{\mathcal{K}}_{pp}(t)$

Explicit (w/ simple formulas), and given by $c \times t^{-\tau} d^{-\sigma}$, some $c, \tau, \sigma > 0$



Dominant terms

Phase I:	$\mathcal{P}(t) \asymp \hat{\mathcal{F}}_{pp}(t) + \hat{\mathcal{F}}_0$
Phase II:	$\mathcal{P}(t) \asymp \hat{\mathcal{F}}_{pp}(t) + \hat{\mathcal{F}}_{ac}(t) + \hat{\mathcal{F}}_0$
Phase III:	$\mathcal{P}(t) \asymp \hat{\mathcal{K}}_{pp}(t) + \hat{\mathcal{F}}_{ac}(t) + \hat{\mathcal{F}}_0$
Phase IV:	$\mathcal{P}(t) \asymp \hat{\mathcal{F}}_{pp}(t) + \hat{\mathcal{K}}_{pp}(t) + \hat{\mathcal{F}}_0$

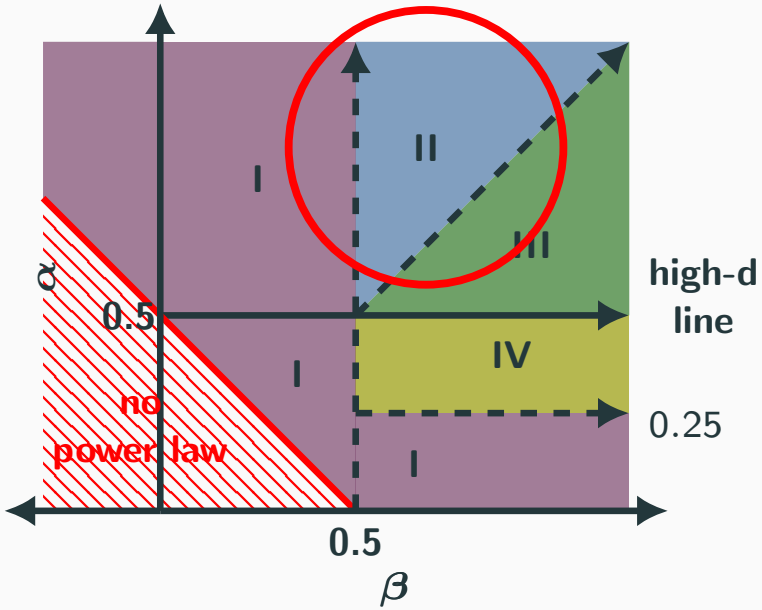
Theorem: Let $\gamma = \gamma_2$ (DANA) and $\gamma_2 + \frac{\gamma_3}{\delta}$ (SGD-M) and let

$$\vartheta(t) \stackrel{\text{def}}{=} \begin{cases} 1 + 2(\gamma_2 + \frac{\gamma_3}{\delta})t, & \text{SGD-M,} \\ 1 + 2\gamma_2 t + \left(\int_0^t \sqrt{\gamma_3(s)}\,\mathrm{d}s\right)^2, & \text{DANA-constant/ DANA-decaying} \end{cases}$$

Then $\mathcal{P}(t) \asymp \hat{\mathcal{F}}(\vartheta(t)) + \gamma\hat{\mathcal{K}}(\vartheta(t))$.

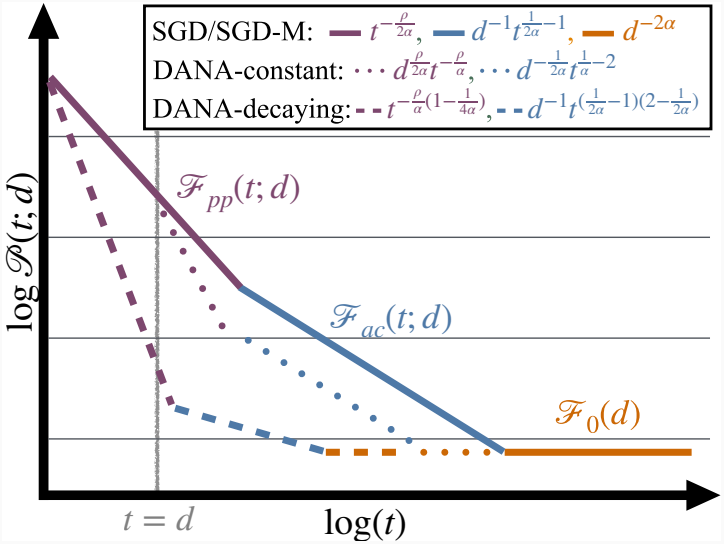
Compute-Optimal Curves

Force: $\hat{\mathcal{F}}(t) \asymp \hat{\mathcal{F}}_0(t) + \hat{\mathcal{F}}_{pp}(t) + \hat{\mathcal{F}}_{ac}(t)$ & **Kernel:** $\hat{\mathcal{K}}(t) \asymp \hat{\mathcal{K}}_{pp}(t)$

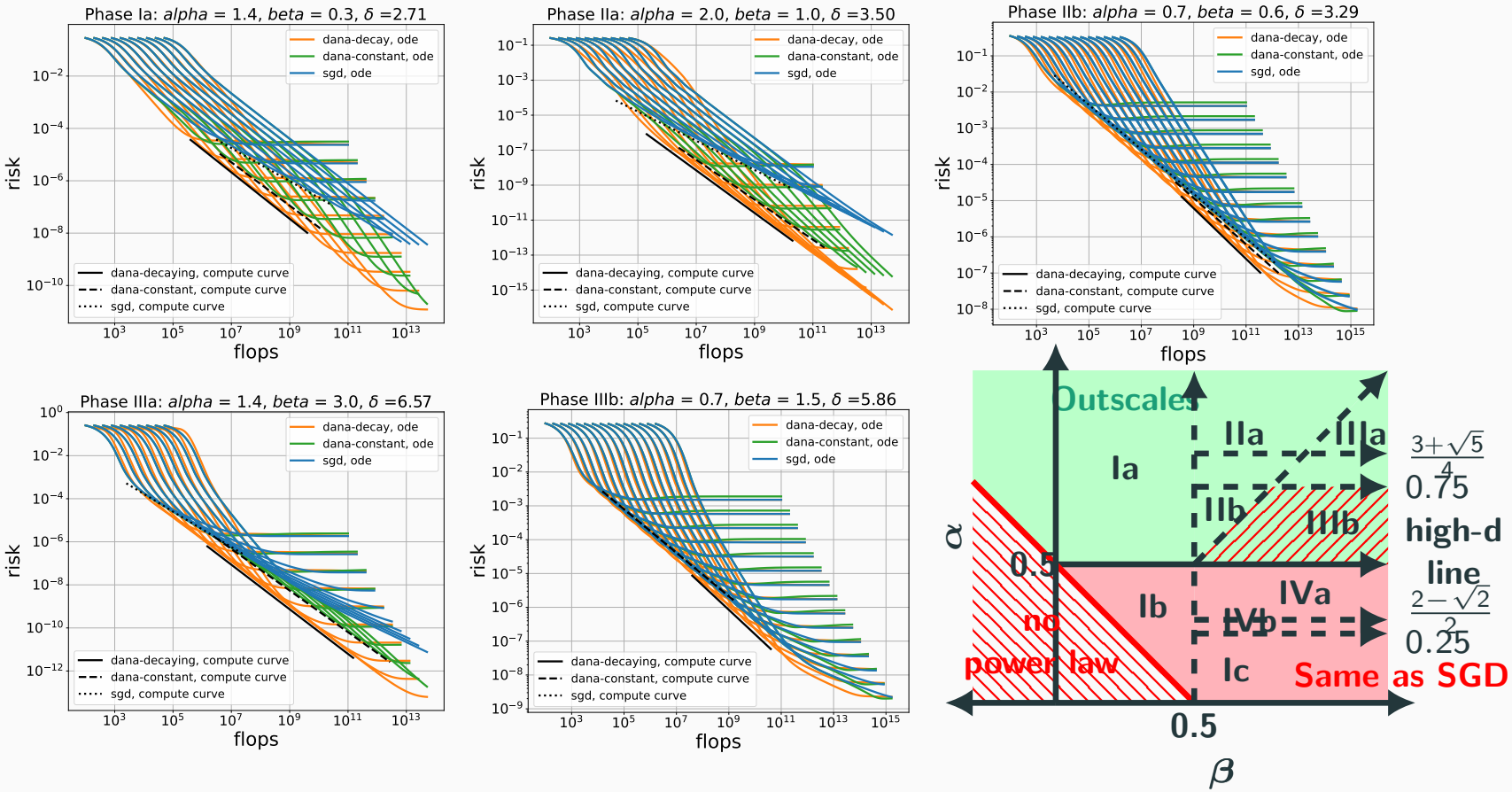


Dominant terms

Phase II: $\mathcal{P}(t) \asymp \hat{\mathcal{F}}_{pp}(t) + \hat{\mathcal{F}}_{ac}(t) + \hat{\mathcal{F}}_0$



Phases



DANA only outscales SGD above the high-dimensional line, $2\alpha > 1$!

Improved Benefit

Components: $\underbrace{\mathcal{F}_0(t),\quad \mathcal{F}_{pp}(t),\quad \mathcal{F}_{ac}(t)}_{\text{full batch}},\quad \underbrace{\mathcal{K}_{pp}(t)}_{\text{alg. noise}}$

Define: $\rho \stackrel{\text{def}}{=} 2\alpha + 2\beta + 1$

Phase II: $(2\beta > 1 \;\&\; \alpha > \beta \;\&\; 2\alpha > 1)$

SGD/SGD-M[1] $t^{-\rho/(2\alpha)} \vee d^{-1}t^{-1+1/(2\alpha)} \vee d^{-2\alpha}$

DANA-decaying, $\kappa_3 > \frac{1}{2\alpha}$ $t^{-(\rho/(2\alpha))(2-\kappa_3)} \vee d^{-1}t^{-(2-\kappa_3)(1-1/(2\alpha)} \vee d^{-2\alpha}$

Phase III: $(2\beta > 1 \;\&\; \alpha < \beta \;\&\; 2\alpha > 1)$

SGD/SGD-M[1] $t^{-2+1/(2\alpha)} \vee d^{-1}t^{-1+1/(2\alpha)} \vee d^{-2\alpha}$

DANA-decaying, $\kappa_3 > \frac{1}{2\alpha}$ $t^{-(2-1/(2\alpha))(2-\kappa_3)} \vee d^{-1}t^{-(2-\kappa_3)(1-1/(2\alpha)} \vee d^{-2\alpha}$

Improved Benefit

Components: $\underbrace{\mathcal{F}_0(t),\quad \mathcal{F}_{pp}(t),\quad \mathcal{F}_{ac}(t)}_{\text{full batch}},\quad \underbrace{\mathcal{K}_{pp}(t)}_{\text{alg. noise}}$

Define: $\rho \stackrel{\text{def}}{=} 2\alpha + 2\beta + 1$

Phase II: $(2\beta > 1 \;\&\; \alpha > \beta \;\&\; 2\alpha > 1)$

SGD/SGD-M[1] $t^{-\rho/(2\alpha)} \vee d^{-1}t^{-1+1/(2\alpha)} \vee d^{-2\alpha}$

DANA-decaying, $\kappa_3 > \frac{1}{2\alpha}$ $t^{-(\rho/(2\alpha))(2-\kappa_3)} \vee d^{-1}t^{-(2-\kappa_3)(1-1/(2\alpha)} \vee d^{-2\alpha}$

Phase III: $(2\beta > 1 \;\&\; \alpha < \beta \;\&\; 2\alpha > 1)$

SGD/SGD-M[1] $t^{-2+1/(2\alpha)} \vee d^{-1}t^{-1+1/(2\alpha)} \vee d^{-2\alpha}$

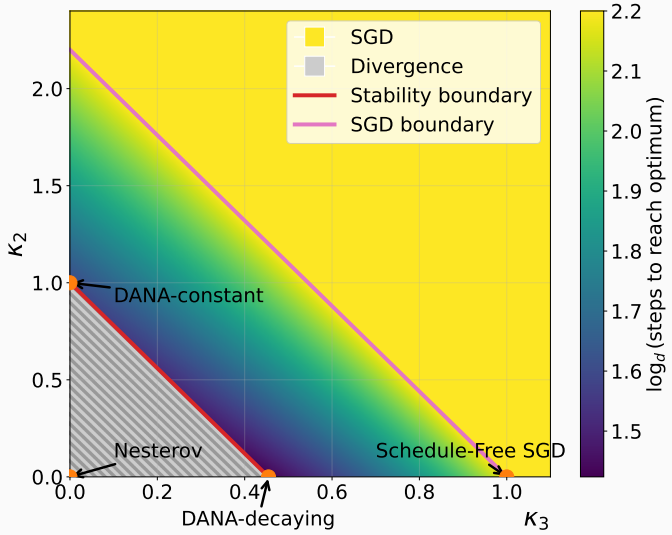
DANA-decaying, $\kappa_3 > \frac{1}{2\alpha}$ $t^{-(2-1/(2\alpha))(2-\kappa_3)} \vee d^{-1}t^{-(2-\kappa_3)(1-1/(2\alpha)} \vee d^{-2\alpha}$

- No random matrix W drop $\mathcal{F}_{ac}$ (Cui et al '21), (Carratino et al '22), (Caponnetto et al, '07), (Yarotsky et al '24), (Varre et al '22), (Lin, et al '24), (Bordelon et al '24), (Dieuleveut et al '16)
- Ridge Regression (w/ W) replace $\mathcal{K}_{pp} \Rightarrow \mathcal{F}_{pp}$ (Defilippis et al '24)

Full DANA Class

$$\text{(DANA)} \qquad y_t = (1 - \delta(1+t)^{-1})y_{t-1} + \nabla \mathscr{P}(\theta_t; x_{t+1}),$$
$$\theta_{t+1} = \theta_t - \gamma_2(t)\nabla \mathscr{P}(\theta_t; x_{t+1}) - \underbrace{c_3 d^{-\kappa_2}(1+t)^{-\kappa_3}}_{\gamma_3} y_t,$$

Plot of $\log_d$(time to reach irreducible loss), $2\alpha > 1$



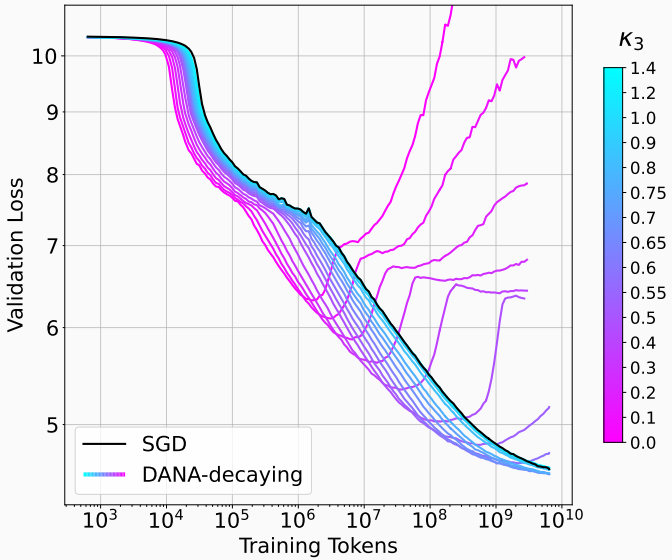
DANA-decaying is **optimal**

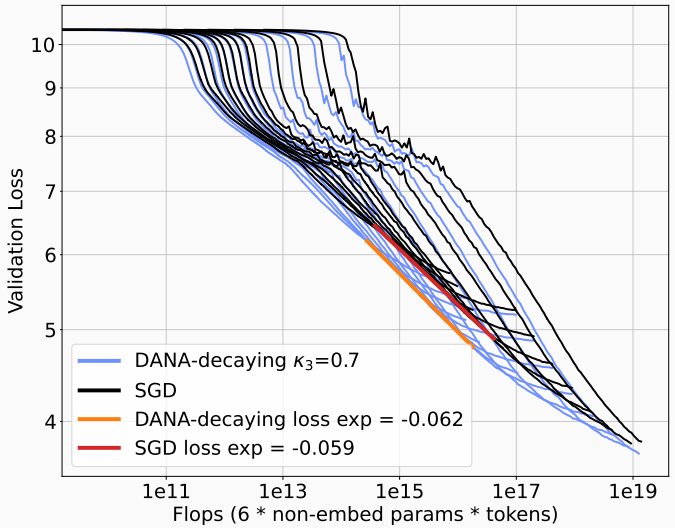
(e.g., fewest iterations to optimum, best loss exponent)

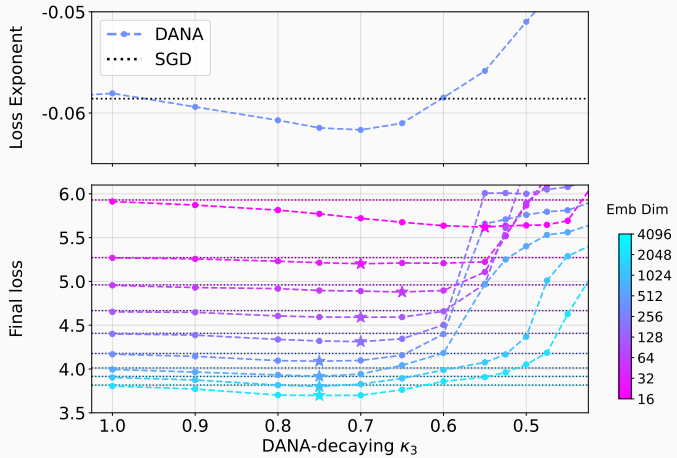
- Other stable scaling rules in the DANA class (above red line)
- DANA stability boundary (red) at $\kappa_2 = -2\alpha\kappa_3 + 1$ and divergence below
- DANA takes same number of iterations as SGD at (pink) line, $\kappa_2 \geq 2\alpha(1-\kappa_3)$
- Iterations to reach irreducible loss,

$$\underbrace{d^{4\alpha^2/(4\alpha-1)}}_{\text{DANA-decay}} \leq \underbrace{d^{\alpha+1/2}}_{\text{DANA-constant}} \leq \underbrace{d^{2\alpha}}_{\text{SGD}}$$

2-Layer LSTM on Language Data







DANA-decaying works on language datasets!

(C4 dataset, C. Raffel, et. al.)

Practical considerations

DANA-decaying

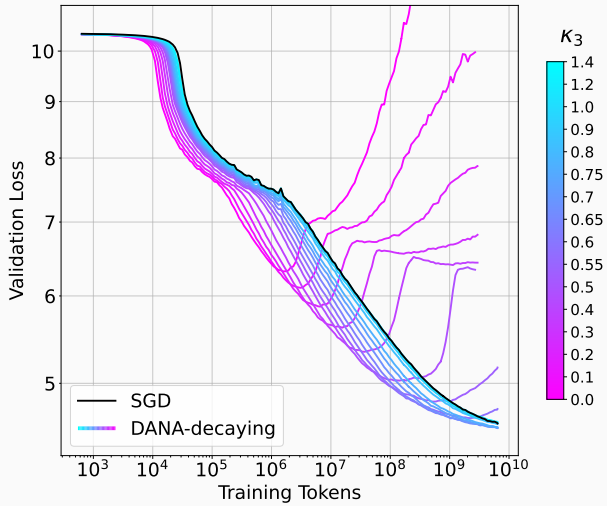
$$y_t = (1 - \delta(1+t)^{-1})y_{t-1} + \nabla\mathscr{R}(\theta_t; x_{t+1}),$$
$$\theta_{t+1} = \theta_t - \gamma_2(t)\nabla\mathscr{R}(\theta_t; x_{t+1}) - \underbrace{c_3(1+t)^{-\kappa_3}}_{\gamma_3} y_t,$$

Practical hyperparameters

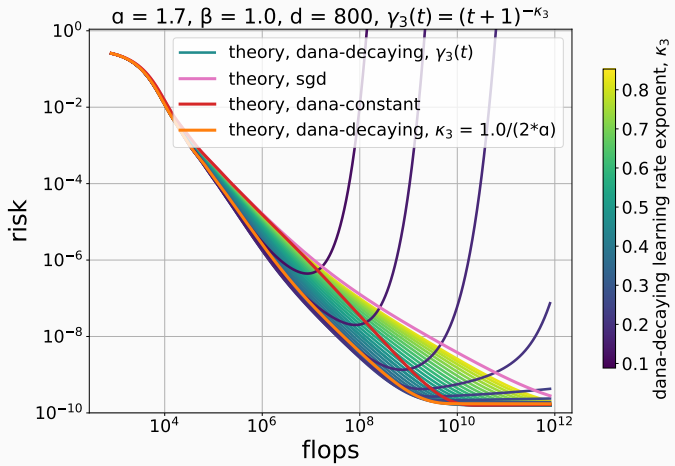
- $\gamma_2 =$ learning rate SGD
- Constant $c_3 = \gamma_2$ (SGD)
- δ large, 8 is good

For κ_3: don't know data exp. α

- For PLRF, $\kappa_3 = 1/(2\alpha)$
- Can sweep across κ_3 and find one that is stable



LSTM with C4 dataset



PLRF

Practical considerations

DANA-decaying

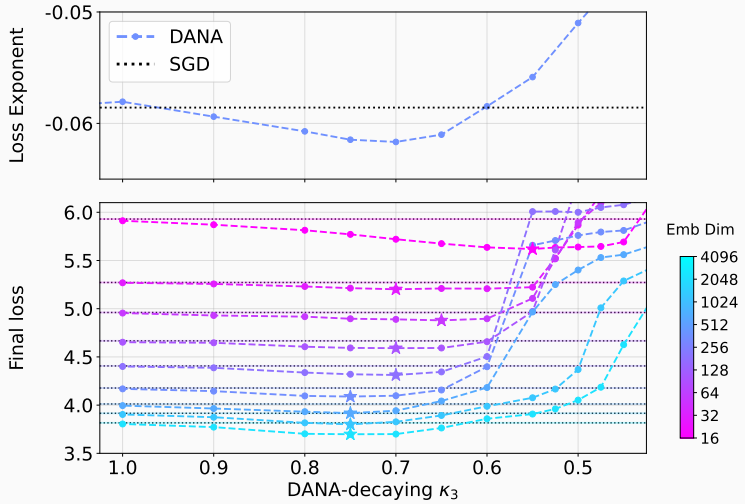
$$y_t = (1 - \delta(1+t)^{-1})y_{t-1} + \nabla\mathscr{R}(\theta_t; x_{t+1}),$$
$$\theta_{t+1} = \theta_t - \gamma_2(t)\nabla\mathscr{R}(\theta_t; x_{t+1}) - \underbrace{c_3(1+t)^{-\kappa_3}}_{\gamma_3} y_t,$$

Practical hyperparameters

- γ_2 = learning rate SGD
- Constant $c_3 = \gamma_2$ (SGD)
- δ large, 8 is good

Transferability of γ_3 across model sizes (d)

- Model size d not well defined for real architectures
- γ_3 does not depend on d



Same κ_3 works across different model sizes!

LSTM with C4 dataset

Open Problems

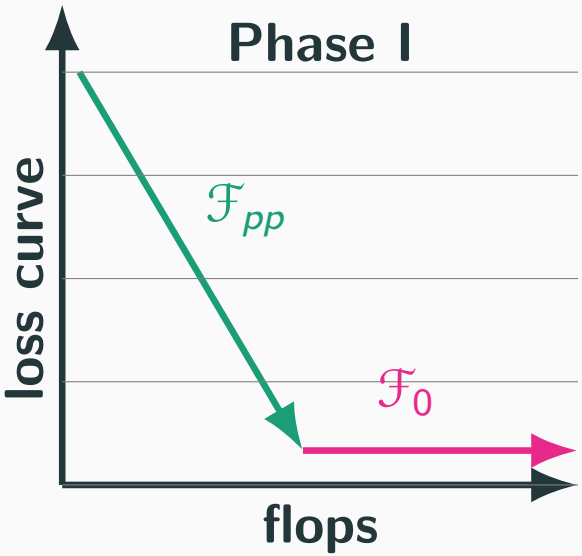
- What are the fundamental algorithm barriers? If you change algorithms, can you change the scaling laws?
- Incorporate new momentum strategies inside more practical algorithms
- Evidence suggests that the (standard) architectures don't change the data-complexity, α. Why does this happen?

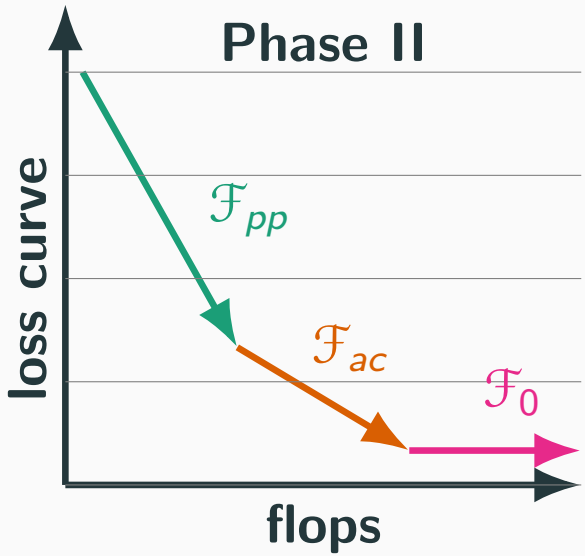
Thank you!

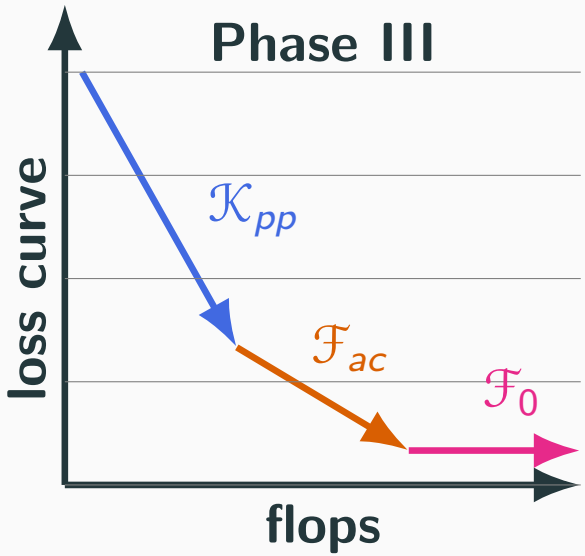
D. Ferbach, K. Everett, G. Gidel, E. Paquette, C. Paquette. *Dimension-adapted Momentum Outscales SGD; arXiv: https://arxiv.org/abs/2505.16098*

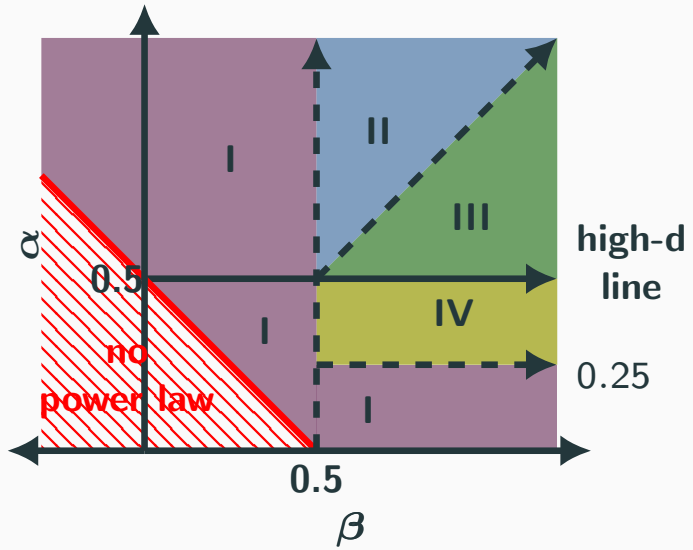
E. Paquette, C. Paquette, L. Xiao, J. Pennington. *4+3 Phases of Compute Optimal Neural Scaling Laws;* NeurIPS 2024
https://openreview.net/forum?id=aVSxwicpAk&noteId=4mjoq2kraU

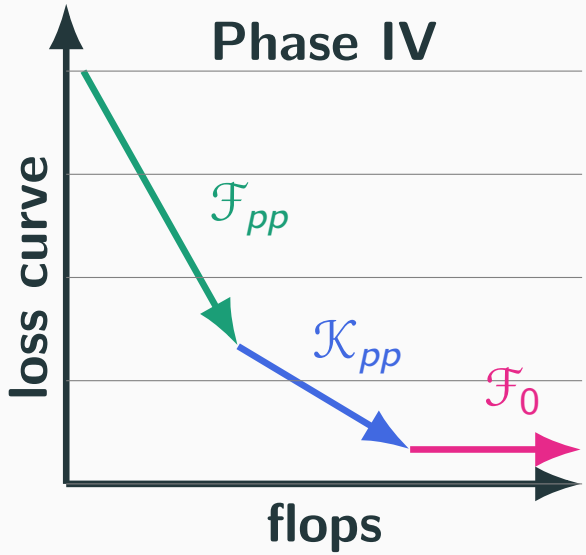
The Phases


Phase I
loss curve
$\mathcal{F}_{pp}$
$\mathcal{F}_0$
flops

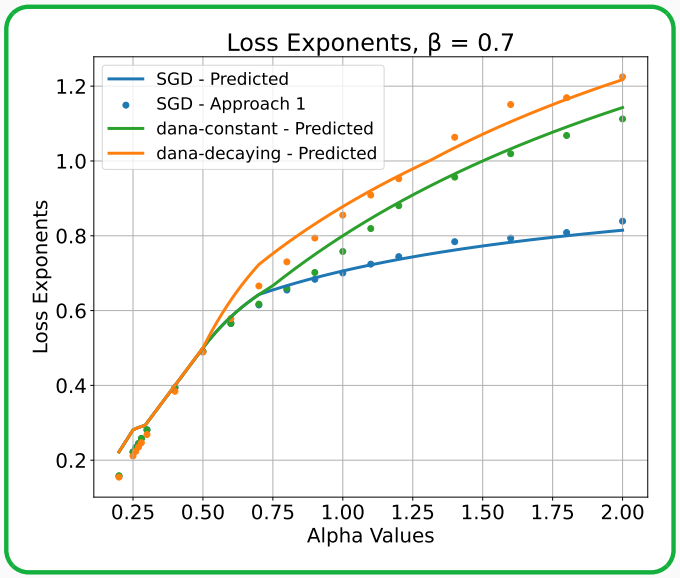

Phase II
loss curve
$\mathcal{F}_{pp}$
$\mathcal{F}_{ac}$
$\mathcal{F}_0$
flops

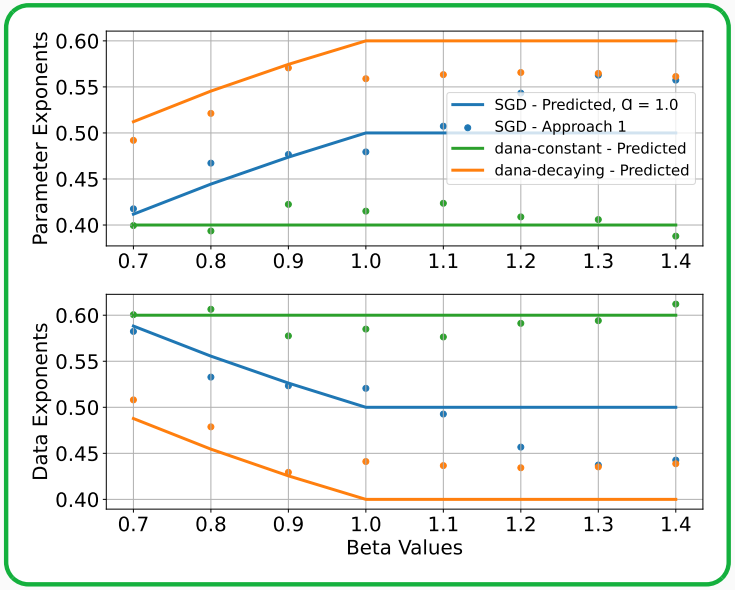

Phase III
loss curve
$\mathcal{K}_{pp}$
$\mathcal{F}_{ac}$
$\mathcal{F}_0$
flops


I
II
III
IV
I
I
α
0.5
no
power law
high-d
line
0.25
0.5
β


Phase IV
loss curve
$\mathcal{F}_{pp}$
$\mathcal{K}_{pp}$
$\mathcal{F}_0$
flops

Other Observations





Good match across compute-optimal exponents!

DANA-decaying uses less data to reach compute-optimality!

Dimension-adapted Momentum

Dimension-adapted Nesterov acceleration (DANA) At each iteration generate new x_{k+1} from distribution,

$$y_t = (1 - \Delta(t))y_{t-1} + \gamma_1(t)\nabla\mathscr{P}(\theta_t; x_{t+1}),$$
$$\theta_{t+1} = \theta_t - \gamma_2(t)\nabla\mathscr{P}(\theta_t; x_{t+1}) - \gamma_3(t)y_t,$$

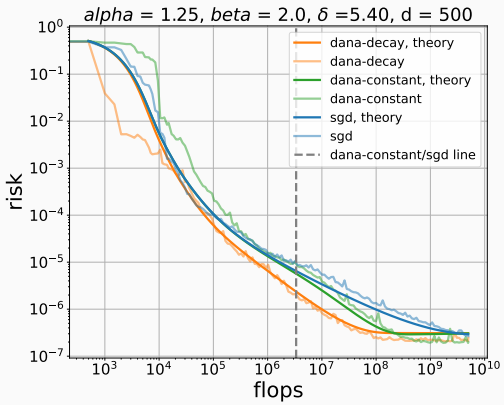
Hyperparameters

$$\gamma_1(t) \equiv 1, \quad \gamma_2(t) = \tilde{\gamma}_2 d^{-\kappa_1}, \quad \gamma_3(t) = \tilde{\gamma}_3 d^{-\kappa_2}(1+t)^{-\kappa_3},$$
$$\Delta(t) = \delta(1+t)^{-1}$$

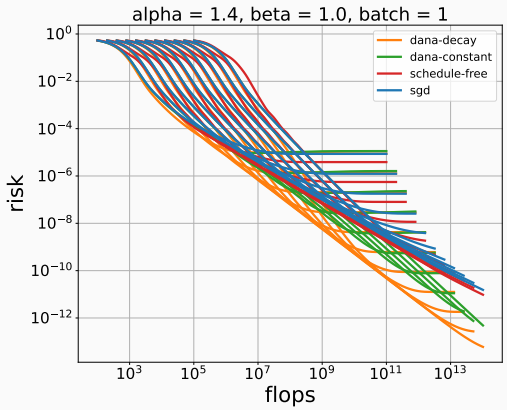
- SGD and classical momentum (SGD-M): $\gamma_2 =$ constant, indep. of d
- Nesterov acceleration; (does not converge)
- Accelerated SGD (Jain, et al. '18, Li et al '23)

Question: What does momentum gain?

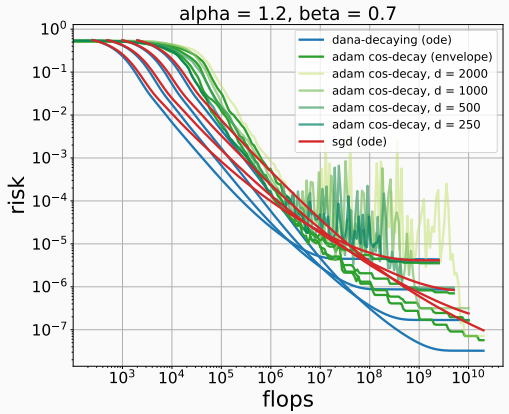
DANA outscales



DANA & SGD



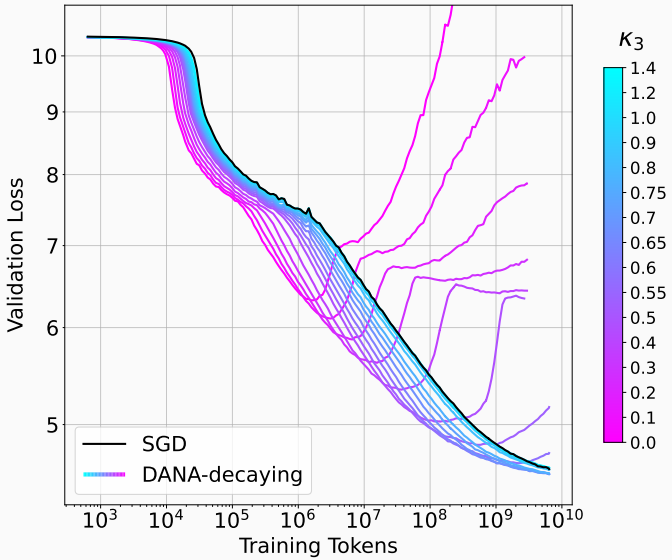
Scaling behavior

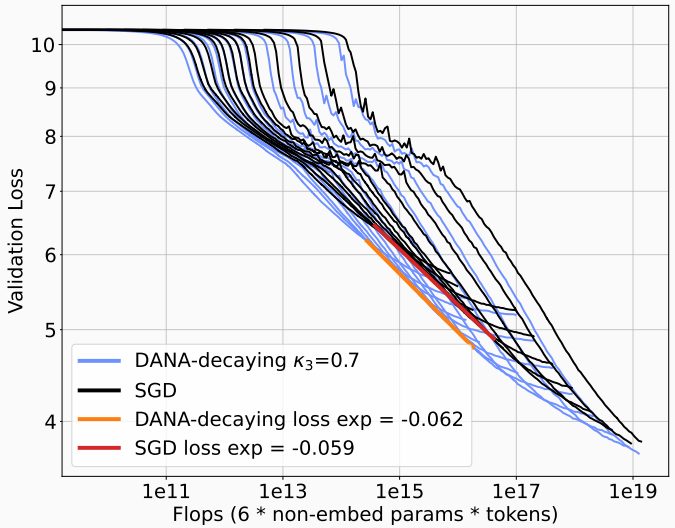


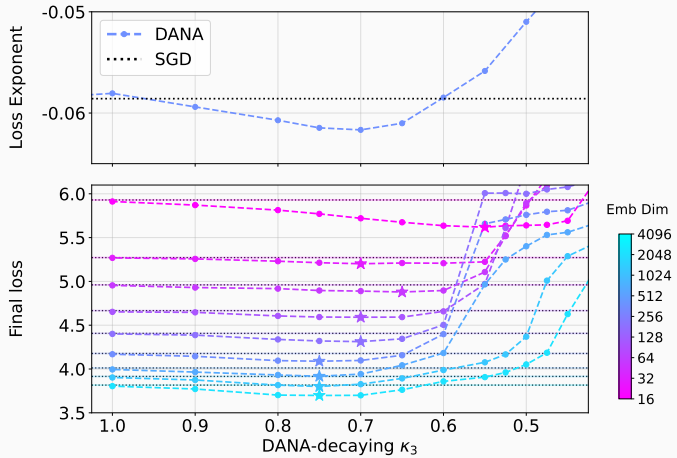
Adam (Kingma, etc) & DANA

Choosing dimension-dependent/data-dependent hyperparameters, DANA can outscale SGD!

2-Layer LSTM on Language Data







DANA-decaying works on language datasets!

(C4 dataset, C. Raffel, et. al.)