

747 A Update Rules for Second-Order Optimizer

748 This section presents the update rules for the optimizers considered in our work, for the sake of
 749 completeness and reproducibility. Let $W_t \in \mathbb{R}^{m \times n}$ denote the weight matrix of a layer at time t ,
 750 and let $G_t \in \mathbb{R}^{m \times n}$ denote its gradient. We use $w_t \in \mathbb{R}^{mn}$ and $g_t \in \mathbb{R}^{mn}$ to represent the flattened
 751 versions of the weights and gradients, respectively. The hyperparameters include the learning rate
 752 (η), momentum coefficients (β), damping factors (ϵ), and inverse exponents (e). Elementwise
 753 multiplication is denoted by \odot , and vector division is applied elementwise.

754 **Adam** [10] adjusts the magnitude of the first momentum by the second momentum. The update rule
 755 is:

$$g_t \leftarrow \nabla_w \mathcal{L}(w_{t-1}) \quad (15)$$

$$m_t \leftarrow \beta_1 m_{t-1} + (1 - \beta_1) g_t \quad (16)$$

$$v_t \leftarrow \beta_2 v_{t-1} + (1 - \beta_2) g_t \odot g_t \quad (17)$$

$$\hat{m}_t \leftarrow \frac{m_t}{1 - \beta_1^t} \quad (18)$$

$$\hat{v}_t \leftarrow \frac{v_t}{1 - \beta_2^t} \quad (19)$$

$$w_t \leftarrow w_{t-1} - \eta \frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \epsilon} \quad (20)$$

756 **Shampoo** [15] preconditions the gradient with a Kronecker-factored preconditioner. The update rule
 757 is:

$$M_t \leftarrow \beta_1 M_{t-1} + (1 - \beta_1) G_t \quad (21)$$

$$L_t \leftarrow \beta_2 L_{t-1} + (1 - \beta_2) G_t G_t^T \quad (22)$$

$$R_t \leftarrow \beta_2 R_{t-1} + (1 - \beta_2) G_t^T G_t \quad (23)$$

$$\hat{L}_t \leftarrow \frac{L_t}{(1 - \beta_2^t)} \quad (24)$$

$$\hat{R}_t \leftarrow \frac{R_t}{(1 - \beta_2^t)} \quad (25)$$

$$W_{t+1} \leftarrow W_t - \eta (\hat{L}_t + \epsilon I)^{-e_L} M_t (\hat{R}_t + \epsilon I)^{-e_R} \quad (26)$$

$$(27)$$

758 **SOAP** [32] runs Adam in a rotated space. The update rule is:

$$M_t \leftarrow \beta_1 M_{t-1} + (1 - \beta_1) G_t \quad (28)$$

$$L_t \leftarrow \beta_2 L_{t-1} + (1 - \beta_2) G_t G_t^T \quad (29)$$

$$R_t \leftarrow \beta_2 R_{t-1} + (1 - \beta_2) G_t^T G_t \quad (30)$$

$$Q_L \leftarrow \text{Eigenvector}(L_t) \quad (31)$$

$$Q_R \leftarrow \text{Eigenvector}(R_t) \quad (32)$$

$$G'_t \leftarrow Q_L^T G_t Q_R \quad (33)$$

$$V_t \leftarrow \beta_2 V_{t-1} + (1 - \beta_2) G'_t \odot G'_t \quad (34)$$

$$W_{t+1} \leftarrow W_t - \eta Q_L \frac{M_t}{\sqrt{V_t} + \epsilon} Q_R^T \quad (35)$$

$$(36)$$

759 **Muon** [19] utilizes Newton-Schulz to compute the matrix sign [7] of the gradient. The update rule is:

$$M_t \leftarrow \beta_1 M_{t-1} + (1 - \beta_1) G_t \quad (37)$$

$$W_{t+1} \leftarrow W_t - \eta \text{Newton-Schulz} \left(\frac{M_t}{\|M_t\|_2 + \epsilon} \right) \quad (38)$$

$$(39)$$

760 **Grafting** [2, 31] is a technique that adjusts the direction of a primary optimizer’s update to match the
 761 step size of a reference optimizer with more stable magnitude. Let ΔW_p denote the update from the
 762 primary optimizer, and ΔW_g the update from the reference (grafted) optimizer. The grafted update is
 763 given by:

$$W_{t+1} \leftarrow W_t - \eta \frac{\|\Delta W_p\|_2}{\|\Delta W_g\|_2 + \epsilon} \Delta W_g \quad (40)$$

764 **Blocking** [31, 32] is a technique that partitions the weight matrix into sub-blocks, and updates are
 765 computed independently for each block.

766 B Derivations of Maximum Update Parameterization

767 B.1 First-Order Optimizers

768 As a warmup, we start with rederiving μ Pfor SGD and Adam under our notation.

769 **SGD.** We plug in $Q = \text{id}$ for SGD:

$$\eta Q(G) |x'\rangle = \eta \frac{d_{\text{in}}}{d_{\text{out}}} |\delta\rangle \underbrace{\langle x|x'\rangle}_{\Theta(1)} = \Theta\left(\eta \frac{d_{\text{in}}}{d_{\text{out}}}\right). \quad (41)$$

770 Therefore we need $\eta = \Theta(\frac{d_{\text{out}}}{d_{\text{in}}})$, recovering results in Yang and Hu [33], Yang et al. [37]. Note
 771 that the use of bra-ket notation reduces the problem of finding the right learning rate into one
 772 straightforward algebraic manipulation.

773 **SignSGD and Adam.** For SignSGD, we have

$$Q(G) = \left(\sqrt{G^{\odot 2}} + \epsilon\right)^{-1} \odot G \quad (42)$$

$$= \left(\sqrt{\delta^{\odot 2} \otimes x^{\odot 2}} + \epsilon\right)^{-1} \odot \delta x^\top \quad (43)$$

$$= \left(\sqrt{\frac{1}{d_{\text{out}}^2} |\delta\rangle^{\odot 2} \otimes |x\rangle^{\odot 2}} + \epsilon\right)^{-1} \odot \frac{d_{\text{in}}}{d_{\text{out}}} |\delta\rangle \langle x| \quad (44)$$

$$= d_{\text{in}} \left(\sqrt{|\delta\rangle^{\odot 2} \otimes |x\rangle^{\odot 2}} + \epsilon'\right)^{-1} \odot |\delta\rangle \langle x|, \quad (45)$$

774 where we reparameterized $\epsilon = \epsilon'/d_{\text{out}}$. For ϵ to be effective without trivializing the preconditioner,
 775 we need $\epsilon' = \Theta(1)$. Yang and Littwin [34] showed that applying a $\Theta(1)$ elementwise scaling
 776 $\left(\sqrt{|\delta\rangle^{\odot 2} \otimes |x\rangle^{\odot 2}} + \epsilon'\right)^{-1}$ to $|\delta\rangle \langle x|$ does not change the scale of its output when applied to $|x'\rangle^3$,
 777 so that $\Theta(Q(G) |x'\rangle) = \Theta(d_{\text{in}} |\delta\rangle \langle x| |x'\rangle) = \Theta(d_{\text{in}})$ and we should set $\eta = \Theta(1/d_{\text{in}})$.

778 The above analysis can be extended straightforwardly to Adam with the same conclusion $\eta =$
 779 $\Theta(1/d_{\text{in}})$ since the accumulation of the gradient over the iterates does not introduce extra scaling
 780 with width.

781 B.2 Second-Order Optimizers

782 With our setup so far, generalizing the derivation to second-order optimizers is straightforward, as one
 783 only needs to additionally keep track of width scaling in the preconditioner. We will omit accumulation
 784 in the preconditioner and the momentum of the gradient, which does not affect the scaling with width
 785 [34].

³The output can be computed via an order-1 `OuterNonLin` instruction in Yang and Littwin [34], which has an $\Theta(1)$ limit.

786 **K-FAC.** In K-FAC, we have

$$\begin{aligned}
Q(G) &= (\delta\delta^\top + \epsilon_B)^{-e_B} \delta x^\top (xx^\top + \epsilon_A)^{-e_A} \\
&= \left(\frac{1}{d_{\text{out}}} |\delta\rangle\langle\delta| + \epsilon_B \right)^{-e_B} |\delta\rangle\langle x| (d_{\text{in}} |x\rangle\langle x| + \epsilon_A)^{-e_A} \\
&= \frac{d_{\text{in}}^{1-e_A}}{d_{\text{out}}^{1-e_B}} (|\delta\rangle\langle\delta| + \epsilon'_B)^{-e_B} |\delta\rangle\langle x| (|x\rangle\langle x| + \epsilon'_A)^{-e_A},
\end{aligned} \tag{46}$$

787 where $\epsilon_B \equiv \epsilon'_B/d_{\text{out}}$ and $\epsilon_A \equiv d_{\text{in}}\epsilon'_A$. As $d \rightarrow \infty$, To ensure ϵ_A and ϵ_B have a consistent effect
788 across widths and that each preconditioner does not trivialize to a scalar multiplication, we require
789 $\epsilon'_B \sim \epsilon'_A \sim 1$. Using the identity

$$(vv^\top + aI)^{-e} = a^{-e} \left(I - \frac{vv^\top}{v^\top v} \right) + (v^\top v + a)^{-e} \frac{vv^\top}{v^\top v} \tag{47}$$

790 we can express the matrix inverses using only outer products and inner products as:

$$\begin{aligned}
Q(G) &= \frac{d_{\text{in}}^{1-e_A}}{d_{\text{out}}^{1-e_B}} \\
&\times \left[\epsilon_B'^{-e_B} \left(I - \frac{|\delta\rangle\langle\delta|}{\langle\delta|\delta\rangle} \right) + (\langle\delta|\delta\rangle + \epsilon'_B)^{-e_B} \frac{|\delta\rangle\langle\delta|}{\langle\delta|\delta\rangle} \right] \\
&\times |\delta\rangle\langle x| \\
&\times \left[\epsilon_A'^{-e_A} \left(I - \frac{|x\rangle\langle x|}{\langle x|x\rangle} \right) + (\langle x|x\rangle + \epsilon'_A)^{-e_A} \frac{|x\rangle\langle x|}{\langle x|x\rangle} \right]
\end{aligned} \tag{48}$$

791 This form makes it clear that 1) K-FAC can be expressed using existing instructions in the Tensor
792 Program, a key result for establishing the existence of an infinite-width limit missing from prior
793 work by Ishikawa and Karakida [18]⁴, and 2) that $Q(G) |x'\rangle = \Theta\left(\frac{d_{\text{in}}^{1-e_A}}{d_{\text{out}}^{1-e_B}}\right) |\delta\rangle$ so that we need
794 $\eta = \Theta\left(\frac{d_{\text{out}}^{1-e_A}}{d_{\text{in}}^{1-e_B}}\right)$. When the batch size is greater than 1, expressing the matrix inverse as outer and
795 inner products is more involved but does not change our conclusion, which we explain in the next
796 section.

797 **Shampoo and Muon.** The analysis for K-FAC can be straightforwardly adapted to Shampoo and
798 Muon. Omitting accumulation and momentum, we have

$$\begin{aligned}
Q_{\text{Shampoo}}(G) &= (\delta x^\top x \delta^\top + \epsilon_B)^{-e_L} \delta x^\top (x \delta^\top \delta x^\top + \epsilon_A)^{-e_R} \\
&= \left(\frac{d_{\text{in}}}{d_{\text{out}}} (\langle x|x\rangle |\delta\rangle\langle\delta| + \epsilon'_B) \right)^{-e_L} \\
&\times \frac{d_{\text{in}}}{d_{\text{out}}} |\delta\rangle\langle x| \left(\frac{d_{\text{in}}}{d_{\text{out}}} (\langle\delta|\delta\rangle |x\rangle\langle x| + \epsilon'_A) \right)^{-e_R} \\
&= \left(\frac{d_{\text{in}}}{d_{\text{out}}} \right)^{1-e_L-e_R} (\langle x|x\rangle |\delta\rangle\langle\delta| + \epsilon'_B)^{-e_L} \\
&\times |\delta\rangle\langle x| (\langle\delta|\delta\rangle |x\rangle\langle x| + \epsilon'_A)^{-e_R}
\end{aligned} \tag{49}$$

799 with $\epsilon'_{A,B} \equiv \frac{d_{\text{out}}}{d_{\text{in}}} \epsilon_{A,B}$. Expressing the matrix inverses as inner and outer products:

$$\begin{aligned}
Q_{\text{Shampoo}}(G) &= \left(\frac{d_{\text{in}}}{d_{\text{out}}} \right)^{1-e_L-e_R} \\
&\times \left[\epsilon_B'^{-e_L} \left(I - \frac{|\delta\rangle\langle\delta|}{\langle\delta|\delta\rangle} \right) + (\langle x|x\rangle \langle\delta|\delta\rangle + \epsilon'_B)^{-e_L} \frac{|\delta\rangle\langle\delta|}{\langle\delta|\delta\rangle} \right] \\
&\times |\delta\rangle\langle x| \\
&\times \left[\epsilon_A'^{-e_R} \left(I - \frac{|x\rangle\langle x|}{\langle x|x\rangle} \right) + (\langle x|x\rangle \langle\delta|\delta\rangle + \epsilon'_A)^{-e_R} \frac{|x\rangle\langle x|}{\langle x|x\rangle} \right].
\end{aligned} \tag{50}$$

⁴The push-through identity used in Ishikawa and Karakida [18] does not apply when considering the accumulation in the preconditioner.

Therefore Shampoo and Muon can also be expressed using existing Tensor Program instructions. Furthermore, we can see that $Q_{\text{Shampoo}}(G) |x'\rangle = \Theta\left(\frac{d_{\text{in}}}{d_{\text{out}}}\right)^{1-e_L-e_R} |\delta\rangle$, implying that we need $\eta = \Theta\left(\frac{d_{\text{out}}}{d_{\text{in}}}\right)^{1-e_L-e_R}$ for proper scaling.

SOAP. SOAP runs the Adam optimizer in the eigenbasis of Shampoo’s preconditioner. Ignoring momentum, we have

$$G' = U^\top G V \quad (51)$$

$$Q(G) = U \left[\left(\sqrt{G'^{\odot 2}} + \epsilon \right)^{-1} \odot G' \right] V^\top \quad (52)$$

where U and V are the eigenbases of the left (L) and right (R) preconditioners from Shampoo. For the single-batch case, these matrices are

$$L = \delta x^\top x \delta^\top + \epsilon_B = \frac{d_{\text{in}}}{d_{\text{out}}} (\langle x|x \rangle |\delta\rangle\langle\delta| + \epsilon'_B) \quad (53)$$

$$R = x \delta^\top \delta x^\top + \epsilon_A = \frac{d_{\text{in}}}{d_{\text{out}}} (\langle \delta|\delta \rangle |x\rangle\langle x| + \epsilon'_A) \quad (54)$$

with $\epsilon'_{A,B} = \frac{d_{\text{out}}}{d_{\text{in}}} \epsilon_{A,B}$.

The matrix L has eigenvalue $\frac{d_{\text{in}}}{d_{\text{out}}} \langle x|x \rangle \langle \delta|\delta \rangle + \epsilon_B$ with eigenvector in the direction of $|\delta\rangle$, and eigenvalue ϵ_B with multiplicity $d_{\text{out}} - 1$ for the orthogonal subspace. Similarly, R has eigenvalue $\frac{d_{\text{in}}}{d_{\text{out}}} \langle \delta|\delta \rangle \langle x|x \rangle + \epsilon_A$ with eigenvector in the direction of $|x\rangle$, and eigenvalue ϵ_A with multiplicity $d_{\text{in}} - 1$ for the orthogonal subspace.

Hence, U is a matrix with the first column $\frac{|\delta\rangle}{\sqrt{d_{\text{out}} \langle \delta|\delta \rangle}}$ and the remaining columns forming an orthonormal basis for the space orthogonal to $|\delta\rangle$. Similarly, V is a matrix with first column $\frac{|x\rangle}{\sqrt{d_{\text{in}} \langle x|x \rangle}}$ and remaining columns forming an orthonormal basis for the space orthogonal to $|x\rangle$.

Transforming the gradient to the eigenbasis:

$$G' = U^\top G V \quad (55)$$

$$= U^\top \left(\frac{d_{\text{in}}}{d_{\text{out}}} |\delta\rangle\langle x| \right) V \quad (56)$$

$$= \frac{|\delta\rangle^\top}{\sqrt{d_{\text{out}} \langle \delta|\delta \rangle}} \left(\frac{d_{\text{in}}}{d_{\text{out}}} |\delta\rangle\langle x| \right) \frac{|x\rangle}{\sqrt{d_{\text{in}} \langle x|x \rangle}} e_1^U e_1^{V^\top} \quad (57)$$

$$= \frac{\sqrt{d_{\text{out}} \langle \delta|\delta \rangle}}{\sqrt{\langle \delta|\delta \rangle}} \left(\frac{d_{\text{in}}}{d_{\text{out}}} |\delta\rangle\langle x| \right) \frac{|x\rangle}{\sqrt{d_{\text{in}} \langle x|x \rangle}} e_1^U e_1^{V^\top} \quad (58)$$

$$= \sqrt{\frac{d_{\text{in}} \langle \delta|\delta \rangle \langle x|x \rangle}{d_{\text{out}}}} e_1^U e_1^{V^\top} \quad (59)$$

where e_1^U and e_1^V are the first standard basis vectors in the dimensions of U and V , respectively.

For ϵ to be effective without trivializing the preconditioner, we need it to be on the same scale as G' , so $\epsilon = \Theta\left(\sqrt{\frac{d_{\text{in}}}{d_{\text{out}}}}\right) = \sqrt{\frac{d_{\text{in}}}{d_{\text{out}}}} \epsilon'$ where $\epsilon' = \Theta(1)$. The Adam-like operation in the eigenbasis then gives:

$$\left(\sqrt{G'^{\odot 2}} + \epsilon \right)^{-1} \odot G' = \frac{1}{1 + \epsilon'} e_1^U e_1^{V^\top} \quad (60)$$

820 Transforming back to the original basis:

$$Q_{\text{SOAP}}(G) = U \left[\left(\sqrt{G'^{\odot 2}} + \epsilon \right)^{-1} \odot G' \right] V^\top \quad (61)$$

$$= \frac{1}{1 + \epsilon'} \frac{|\delta\rangle}{\sqrt{d_{\text{out}} \langle \delta | \delta \rangle}} \frac{d_{\text{in}} \langle x |}{\sqrt{d_{\text{in}} \langle x | x \rangle}} \quad (62)$$

$$= \frac{1}{1 + \epsilon'} \frac{d_{\text{in}} |\delta\rangle \langle x|}{\sqrt{d_{\text{in}} d_{\text{out}} \langle \delta | \delta \rangle \langle x | x \rangle}} \quad (63)$$

821 To determine the proper learning rate, we apply the condition $\eta Q_{\text{SOAP}}(G) |x'\rangle = \Theta(1)$:

$$Q_{\text{SOAP}}(G) |x'\rangle = \frac{1}{1 + \epsilon'} \frac{d_{\text{in}} |\delta\rangle \langle x|x'\rangle}{\sqrt{d_{\text{in}} d_{\text{out}} \langle \delta | \delta \rangle \langle x|x \rangle}} \quad (64)$$

$$= \frac{1}{1 + \epsilon'} \frac{d_{\text{in}} |\delta\rangle \langle x|x'\rangle}{\sqrt{d_{\text{in}} d_{\text{out}} \langle \delta | \delta \rangle \langle x|x \rangle}} \quad (65)$$

822 Since $\langle x|x'\rangle = \Theta(1)$, $\langle \delta|\delta\rangle = \Theta(1)$, and $\langle x|x\rangle = \Theta(1)$, we have:

$$Q_{\text{SOAP}}(G) |x'\rangle = \Theta \left(\frac{d_{\text{in}}}{\sqrt{d_{\text{in}} d_{\text{out}}}} \right) |\delta\rangle = \Theta \left(\sqrt{\frac{d_{\text{in}}}{d_{\text{out}}}} \right) |\delta\rangle \quad (66)$$

823 Therefore, the scaling is $\eta Q_{\text{SOAP}}(G) |x'\rangle = \Theta \left(\eta \sqrt{\frac{d_{\text{in}}}{d_{\text{out}}}} \right)$, implying we need $\eta = \Theta \left(\sqrt{\frac{d_{\text{out}}}{d_{\text{in}}}} \right)$ for
824 SOAP.

825 **Blocking.** Blocking is a technique to reduce the preconditioner cost in second-order optimizers like
826 Shampoo by subdividing each weight matrix into $b_{\text{out}} \times b_{\text{in}}$ blocks and preconditioning each block
827 separately. We derive μP for Shampoo with Blocking but generalizing to other optimizers can be
828 done with a similar analysis. We simply perform the same analysis for Shampoo inside each block
829 and appropriately aggregate their contribution to the feature update. Let $G_{ij} = \delta_i x_j^\top \in \mathbb{R}^{b_{\text{out}} \times b_{\text{in}}}$
830 index the ij -th block of the gradient, where $\delta_i \in \mathbb{R}^{b_{\text{out}}}$ and $x_j \in \mathbb{R}^{b_{\text{in}}}$ are the i and j -th chunk of
831 the vector δ and x . Similarly, let $x'_j \in \mathbb{R}^{b_{\text{in}}}$ be the j -th chunk of the input vector x' . Defining the
832 sub-kets and sub-bras

$$x'_j \equiv |x'_j\rangle, \quad x_j \equiv |x_j\rangle, \quad x_j^\top \equiv b_{\text{in}} \langle x_j|, \quad (67)$$

$$\delta_i \equiv \frac{1}{d_{\text{out}}} |\delta_i\rangle, \quad \delta_i^\top \equiv \frac{b_{\text{out}}}{d_{\text{out}}} \langle \delta_i|. \quad (68)$$

833 The i -th chunk of the update Δh_i is

$$\Delta h_i = \eta \sum_{j=1}^{d_{\text{in}}/b_{\text{in}}} Q(G_{ij}) |x'_j\rangle, \quad (69)$$

834 where

$$\begin{aligned} Q(G_{ij}) &= (\delta_i x_j^\top x_j \delta_i^\top + \epsilon_B)^{-e_L} \delta_i x_j^\top (x_j \delta_i^\top \delta_i x_j^\top + \epsilon_A)^{-e_R} \\ &= \left(\frac{b_{\text{in}} b_{\text{out}}}{d_{\text{out}}^2} (\langle x_j | x_j \rangle |\delta_i\rangle \langle \delta_i| + \epsilon'_B) \right)^{-e_L} \\ &\quad \times \frac{b_{\text{in}}}{d_{\text{out}}} |\delta_i\rangle \langle x_j| \left(\frac{b_{\text{in}} b_{\text{out}}}{d_{\text{out}}^2} (\langle \delta_i | \delta_i \rangle |x_j\rangle \langle x_j| + \epsilon'_A) \right)^{-e_R} \\ &= \left(\frac{b_{\text{in}}}{d_{\text{out}}} \right) \left(\frac{b_{\text{in}} b_{\text{out}}}{d_{\text{out}}^2} \right)^{-(e_L + e_R)} (\langle x_j | x_j \rangle |\delta_i\rangle \langle \delta_i| + \epsilon'_B)^{-e_L} \\ &\quad \times |\delta_i\rangle \langle x_j| (\langle \delta_i | \delta_i \rangle |x_j\rangle \langle x_j| + \epsilon'_A)^{-e_R}, \end{aligned} \quad (70)$$

835 with $\epsilon_{A,B} \equiv \frac{b_{\text{in}} b_{\text{out}}}{d_{\text{out}}^2} \epsilon'_{A,B}$. Therefore,

$$\Delta h_i = \eta \left(\frac{b_{\text{in}} b_{\text{out}}}{d_{\text{out}}^2} \right)^{-(e_L + e_R)} \left(\frac{b_{\text{in}}}{d_{\text{out}}} \right) \sum_{j=1}^{d_{\text{in}}/b_{\text{in}}} \Theta(1) |\delta_i\rangle \quad (71)$$

$$= \eta \left(\frac{b_{\text{in}} b_{\text{out}}}{d_{\text{out}}^2} \right)^{-(e_L + e_R)} \left(\frac{b_{\text{in}}}{d_{\text{out}}} \right) \Theta \left(\frac{d_{\text{in}}}{b_{\text{in}}} \right) |\delta_i\rangle \quad (72)$$

$$= \Theta \left(\eta \left(\frac{d_{\text{in}}}{d_{\text{out}}} \right) \left(\frac{b_{\text{in}} b_{\text{out}}}{d_{\text{out}}^2} \right)^{-(e_L + e_R)} \right), \quad (73)$$

836 where we used the fact that each term in the sum is i.i.d. with a (generally) non-zero mean. Therefore
 837 the learning rate should scale as $\eta = \Theta \left(\left(\frac{d_{\text{out}}}{d_{\text{in}}} \right) \left(\frac{d_{\text{out}}^2}{b_{\text{in}} b_{\text{out}}} \right)^{-(e_L + e_R)} \right)$. Alternatively, let $n_{\text{in}} = d_{\text{in}}/b_{\text{in}}$
 838 and $n_{\text{out}} = d_{\text{out}}/b_{\text{out}}$, we have $\eta = \Theta \left(\left(\frac{d_{\text{out}}}{d_{\text{in}}} \right)^{1-(e_L + e_R)} (n_{\text{in}} n_{\text{out}})^{-(e_L + e_R)} \right)$. When $b_{\text{in}} = d_{\text{in}}$ and
 839 $b_{\text{out}} = d_{\text{out}}$, we recover the regular Shampoo scaling $\eta = \Theta \left(\frac{d_{\text{out}}}{d_{\text{in}}} \right)^{1-e_L-e_R}$. When $e_L + e_R = \frac{1}{2}$
 840 and $b_{\text{in}} = b_{\text{out}} = 1$, degenerating to Adam, we get the correct scaling $\eta = \Theta \left(\frac{1}{d_{\text{in}}} \right)$.

841 **Normalization by Frobenius Norm.** Consider normalizing the preconditioned gradient by its
 842 Frobenius Norm:

$$Q(G) = \frac{Q_1(G)}{\|Q_1(G)\|_F + \epsilon} \quad (74)$$

$$= \frac{Q_1(G)}{\sqrt{\text{Tr}(Q_1(G)^\top Q_1(G))} + \epsilon}. \quad (75)$$

843 Let $Q_1(G) = q \hat{G}_1$ where q is a scalar that absorbs all width-dependent scaling and \hat{G}_1 contains only
 844 bras and kets. Then $Q_1(G)^\top = q \hat{G}_1^\top = q \frac{d_{\text{out}}}{d_{\text{in}}} \hat{G}_1^\dagger$, where X^\dagger is X^\top but with all vectors and their
 845 transposes replaced with kets and bras. Therefore, we get

$$Q(G) = \frac{q_1 \hat{G}_1}{\sqrt{q_1^2 \frac{d_{\text{out}}}{d_{\text{in}}} \text{Tr}(\hat{G}_1^\dagger \hat{G}_1) + \epsilon}} \quad (76)$$

$$= \frac{\hat{G}_1}{\left(\frac{d_{\text{out}}}{d_{\text{in}}} \right)^{1/2} \left(\text{Tr}(\hat{G}_1^\dagger \hat{G}_1)^{1/2} + \frac{\epsilon}{q_1 (d_{\text{out}}/d_{\text{in}})^{1/2}} \right)}. \quad (77)$$

846 We should choose $\epsilon \sim q_1 \sqrt{d_{\text{out}}/d_{\text{in}}}$, and $\eta \sim \sqrt{d_{\text{out}}/d_{\text{in}}}$.

847 **Grafting.** Grafting combines the direction of the update from one optimizer with preconditioner Q_1
 848 with the magnitude of the update from another with preconditioner Q_2 , with the final update given
 849 by:

$$Q(G) = \|Q_2(G)\|_F \frac{Q_1(G)}{\|Q_1(G)\|_F + \epsilon} \quad (78)$$

$$= q_2 \text{Tr}(\hat{G}_2^\dagger \hat{G}_2)^{1/2} \frac{\hat{G}_1}{\text{Tr}(\hat{G}_1^\dagger \hat{G}_1)^{1/2} + \frac{\epsilon}{q_1 (d_{\text{out}}/d_{\text{in}})^{1/2}}}. \quad (79)$$

850 For proper scaling, we need $\epsilon \sim q_1 \sqrt{d_{\text{out}}/d_{\text{in}}}$ and $\eta \sim 1/q_2$, i.e. the learning rate should scale as if
 851 we are training directly with Q_2 .

852 **Depth-scaling.** As explained in the main text, we scale down the output of each residual block
 853 by $1/L$ while ensuring $\Theta(1)$ feature learning inside each block following [6, 9] which showed this
 854 criterion led to well-defined depth-scaling limits for residual networks like the transformer. The only
 855 change to the previous width-scaling derivation is that gradients in the residual blocks now scale as

856 $\frac{1}{Ld_{\text{out}}}$ rather than $\frac{1}{d_{\text{out}}}$. Therefore, we need to update the definition of $|\delta\rangle$ and $\langle\delta|$ to $\delta = \frac{1}{Ld_{\text{out}}} |\delta\rangle$
857 and $\delta^\top = \frac{1}{L} \langle\delta|$. Repeating the above derivations while accounting for these additional L -dependent
858 factors straightforwardly leads to the width-depth joint scaling rules in Table [1](#). We include the
859 derivation for Shampoo here as a concrete example:

$$\begin{aligned}
Q_{\text{Shampoo}}(G) &= (\delta x^\top x \delta^\top + \epsilon_B)^{-e_L} \delta x^\top (x \delta^\top \delta x^\top + \epsilon_A)^{-e_R} \\
&= \left(\frac{d_{\text{in}}}{L^2 d_{\text{out}}} (\langle x|x \rangle |\delta\rangle\langle\delta| + \epsilon'_B) \right)^{-e_L} \\
&\quad \times \frac{d_{\text{in}}}{L d_{\text{out}}} |\delta\rangle\langle x| \left(\frac{d_{\text{in}}}{L^2 d_{\text{out}}} (\langle\delta|\delta\rangle |x\rangle\langle x| + \epsilon'_A) \right)^{-e_R} \\
&= \left(\frac{d_{\text{in}}}{d_{\text{out}}} \right)^{1-e_L-e_R} \left(\frac{1}{L} \right)^{1-2(e_L+e_R)} \\
&\quad \times \underbrace{(\langle x|x \rangle |\delta\rangle\langle\delta| + \epsilon'_B)^{-e_L} |\delta\rangle\langle x| (\langle\delta|\delta\rangle |x\rangle\langle x| + \epsilon'_A)^{-e_R}}_{\Theta(1) \text{ as before}}
\end{aligned} \tag{80}$$

860 with $\epsilon'_{A,B} \equiv \frac{L^2 d_{\text{out}}}{d_{\text{in}}} \epsilon_{A,B}$. Therefore we need $\eta \sim L^{1-2(e_L+e_R)} \left(\frac{d_{\text{out}}}{d_{\text{in}}} \right)^{1-e_L-e_R}$ and $\epsilon \sim \frac{d_{\text{in}}}{L^2 d_{\text{out}}}$.

861 C Extension to Larger Batch Size

862 Throughout the main text we analysed single-sample updates ($B = 1$). All scaling results survive
863 *unchanged* for any *constant* mini-batch size $B = \Theta(1)$ because every object that appears in the
864 update rules becomes only a finite sum of the same single-sample terms.

865 **SGD.** With batch size B the raw update is

$$\Delta W = -\frac{\eta}{B} \sum_{b=1}^B \underbrace{\delta^{(b)} x^{(b)\top}}_{G^{(b)}},$$

866 where each $G^{(b)}$ has entries $\Theta(1/d_{\text{out}})$ (see [§3.1](#)). Averaging over the *finite* set $\{1, \dots, B\}$ do not
867 alter the scaling of the update with width, so the condition $\Delta W x' = \Theta(1)$ is met with the *same*
868 learning-rate scaling $\eta = \Theta(d_{\text{out}}/d_{\text{in}})$ derived for $B = 1$.

869 **Adam.** Adam first rescales the averaged gradient by the element-wise second moment v :

$$\Delta W = -\eta \left(\frac{1}{B} \sum_{b=1}^B G^{(b)} \right) \oslash (\sqrt{v} + \epsilon),$$

870 with $v = \beta_2 v_{\text{old}} + (1 - \beta_2) \frac{1}{B} \sum_b (G^{(b)})^2$. Again since $B = \Theta(1)$, compared to the $B = 1$ update,
871 both the numerator and denominator changes by only a $\Theta(1)$ factor, giving the same width-scaling
872 $\eta = \Theta(1/d_{\text{in}})$.

873 **Shampoo and related preconditioners.** A common operation in second-order optimizers have the
874 form $Q(G) = L^{-e_L} G R^{-e_R}$, where the L and R are left/right Gram matrices of the mini-batch
875 gradient (damped by ϵ_L, ϵ_R as in [Appendix A](#)). Because B is $\Theta(1)$, each matrix is a sum of finitely
876 many rank-1 outer products of $|\delta\rangle$ or $|x\rangle$; hence every eigenvalue of L and R scales identically to
877 the $B = 1$ case, and so are the eigenvalues of L^{-e_L} and R^{-e_R} . Consequently the learning-rate and
878 damping rules in Table [1](#) require no modification.

879 In short, as long as the mini-batch size stays fixed with respect to width, every factor introduced by
880 batching is $\Theta(1)$; therefore the scaling derived for $B = 1$ extends unchanged to all $B = \Theta(1)$.

D The SDE Analysis for Batch Size Scaling

D.1 SDE limits of SGD and Adam

We first briefly review the SDE limit of SGD and Adam studied in [24]:

$$dW = -\eta P^{-1} \left(\nabla L(w) dt + \frac{1}{\sqrt{B}} \Sigma^{1/2} d\mathcal{W}_t \right). \quad (81)$$

where B is the batch size, t denotes the optimization step, P is the preconditioner, and \mathcal{W}_t is the Wiener process. When fixing a total training budget, we are more interested in the SDE where time is measured in number of examples / tokens N , obtained by the change of variable $t = N/B$, $dt = dN/B$, $d\mathcal{W}_t = d\mathcal{W}_N/\sqrt{B}$:

$$dW = -\frac{\eta}{B} P^{-1} \left(\nabla L(w) dN + \Sigma^{1/2} d\mathcal{W}_N \right). \quad (82)$$

It is clear that this SDE, which describes SGD without momentum in the continuous limit, is invariant as we scale B so long as η scales as $\Theta(B)$. Under the assumption that momentum β_1 are close to 1, Malladi et al. [24] showed that scaling η as $\Theta(B)$ while keeping the EMA timescale in momentum $\tau_1 \equiv B/(1 - \beta_1)$ constant preserves the modified SDE that accounts for momentum. Extending this result, Malladi et al. [24] showed that the analogous scaling for Adam requires the additional step of accounting for how the precondition itself scales with B provided the mini-batch gradient variance dominates its mean. Assuming β_2 is close to 1, P approximates the scaled diagonal of the empirical Fisher matrix $P \approx B^{-1/2} \text{diag}(\mathbb{E}[gg^\top])$ with expectation taken over the dataset where g is the flattened per-example gradient. Thus for Adam, we should only scale η as $\Theta(B^{1/2})$ to preserve the SDE.

D.2 Extension to Second-Order Optimizers

Generalizing this argument, when applying a generic preconditioner Q , we must account for how Q scales with B to determine the scaling of the learning rate that preserves the SDE. The batch gradient of the weights in a given layer can be decomposed as:

$$G_B = \bar{G} + \delta G_B, \quad (83)$$

where $\bar{G} = \mathbb{E}[G(x)]$ and $\delta G_B = \Theta(1/\sqrt{B})$ due to the Central Limit Theorem. Following Malladi et al. [24], Choi et al. [8], we assume the noise term dominates the mean, from which we conclude:

$$\bar{G} = \mathbb{E}[G(x)] = \mathbb{E}[G_B(x)] = \Theta(1), \quad \mathbb{E}[G_B G_B^\top] = \Theta(1/B), \quad \mathbb{E}[G_B^\top G_B] = \Theta(1/B) \quad (84)$$

Since the preconditioner is often a function of these quantities, we can infer the scale of the preconditioner. We will write $Q = \Theta(B^q)$ if the preconditioned (batch) gradient $Q(G_B)$ is a factor of $\Theta(B^q)$ larger than G_B itself. In that case, the learning rate should scale as $\Theta(B^{1-q})$ to preserve the SDE. It remains to work out the exponent q for each optimizer. We replace EMA with dataset expectation in the following derivations mirroring the SGD and Adam analysis.

D.3 Scaling Rules for Different Optimizers

Shampoo. We have

$$Q(G) = (\mathbb{E}[G_B G_B^\top] + \epsilon_L)^{-e_L} G (\mathbb{E}[G_B^\top G_B] + \epsilon_R)^{-e_R}. \quad (85)$$

The preconditioner scales as $\Theta(B^{e_L+e_R})$ and we need $\eta = \Theta(B^{1-(e_L+e_R)})$ and $\epsilon_L, \epsilon_R = \Theta(1/B)$.

Muon. We have

$$Q(G) = (\mathbb{E}[G_B] \mathbb{E}[G_B]^\top + \epsilon_L)^{-1/4} G (\mathbb{E}[G_B]^\top \mathbb{E}[G_B] + \epsilon_R)^{-1/4}, \quad (86)$$

The preconditioner scales as $\Theta(1)$ and we need $\eta = \Theta(B)$ and $\epsilon_L, \epsilon_R = \Theta(1)$.

K-FAC. The K-FAC preconditioner is:

$$Q(G) = \left(\frac{1}{B} \sum_{i=1}^B \delta_i \delta_i^\top + \epsilon_B \right)^{-e_B} G \left(\frac{1}{B} \sum_{i=1}^B x_i x_i^\top + \epsilon_A \right)^{-e_A} \quad (87)$$

Assuming noise dominates, the damping parameters should scale as $\epsilon_A, \epsilon_B = \Theta(1/\sqrt{B})$ while $Q = \Theta(B^{(e_A+e_B)/2})$.

Table 2: SDE-based scaling rule for learning rate, damping, and momentum parameters scaling as a function of batch size for different optimizers. $\tau \equiv \frac{B}{(1-\beta)}$ is the half life of the exponential moving average in momentum.

Optimizer	η	ϵ	τ
Adam	\sqrt{B}	$1/\sqrt{B}$	1
Shampoo (e_L, e_R)	$B^{1-(e_L+e_R)}$	$1/B$	1
Muon	B	1	1
Grafted Shampoo $Q_1 \rightarrow Q_2$	η_{Q_2}	$B^{e_L+e_R-1/2}$	1

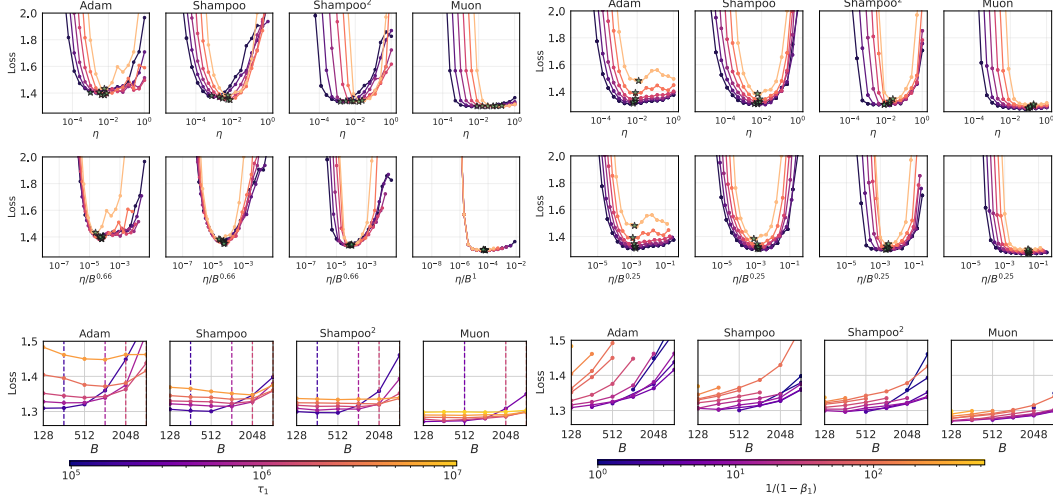


Figure 5: **Batch size scaling.** SDE scaling rules which hold τ rather than β constant led to stable optimal learning rate (bottom left) if β is sufficiently large for all batch sizes. Vertical lines showing values where $\tau/B = 5$ mark the breakdown of the SDE limit for each value of τ and the transition of the optimal value of τ from one to another. (Bottom right) It is the optimal values of β that remains approximately unchanged as we scale B . Fixing the value of β , we find (top right) that the optimal learning rate now scales more slowly with batch size, with an exponent of 0.25 appearing to work well across optimizers.

917 E Blank

918 F Experiment Setup

919 We run our experiments on OpenWebText and the Lichess dataset available on Hugging Face at
 920 <https://huggingface.co/datasets/Lichess/standard-chess-games>. We used character-
 921 level tokenization to reduce the cost of the embedding layers. Not doing so leads total parameter
 922 count being dominated by embedding parameters at small scales, which has been found to lead to
 923 distorted scaling laws for small models [28, 20]. We use the GPT-2 architecture and follow common
 924 practices of removing the bias and replacing LayerNorm with RMSNorm. Learning rates are tuned
 925 on a small 3 layer width 128 model and transferred to larger models. Unless stated otherwise, we use
 926 a batch size of 65536 tokens and linearly warmup the learning rate in the first 10M tokens. Except
 927 for compute-optimal scaling experiments, we train all models for 1B tokens with a linearly decayed
 928 learning rate schedule.

929 We trained all of our models on TPU-V4, supported by the Google TPU Research Cloud program.

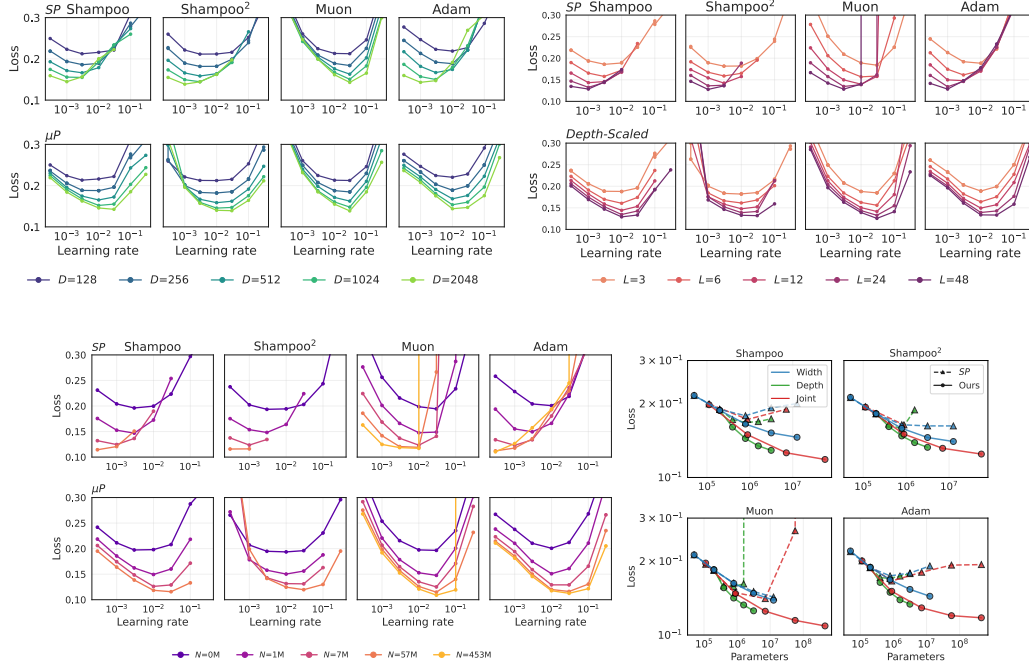


Figure 6: **Learning rate transfer in MLP.** Our proposal scaling rules lead to significantly more stable optimal learning rate for width-only scaling (top left), depth-only scaling (top right), and joint scaling (bottom left) for the residual MLP architecture described by Eq (14). Compared to the transformer, properly scaling the learning rate is more important for effective scaling of MLPs, achieving lower loss compared to SP using a constant learning rate (bottom right).

G MLP Experiments

To better demonstrate the impact of proper learning rate scaling, we perform experiments with the residual MLP architecture described by Eq (14), known to be more sensitive to the learning rate than the transformer architecture [4]. We use the power-law Fourier features regression task [27] as it allows generating unlimited training data to avoid overfitting. All models are trained with 100M examples with a batch size of 4096. Figure 6 summarizes the results, showing our proposed scaling rules are crucial for effectively scaling the MLP models, stabilizing the optimal learning rate across model size and improving final performance as the model increases in size.

H Broader Impact and Limitations

Broader Impact. Our work improves the understanding of how to efficiently scale second-order optimization in deep learning, which has the potential to reduce the cost of training machine learning models and make machine learning research and workflow more accessible.

Limitation. We perceive two main limitations of our work. First, due to limited computational budget, our experiments are relatively small in scale compared to typical training runs in industry where models often exceed billions of parameters. Verifying how well our results generalizes to larger models trained with more compute is an exciting future direction. Second, while we have shown that second-order optimizers like Shampoo and Muon can significantly improve the compute efficiency of training, we do not investigate why they improve the efficiency. Understanding this question holds potential for designing even more efficient future optimizers.