Appendices

A Broader impact

This work shows how to adapt Machine Learning (ML) optimization in the presence of a model Exponential Moving Average (EMA). There are a number of benefits to this:

1. Scaling rules democratize the training of ML models: they give ML researchers the ability to replicate the optimization of large scale systems, even if those researchers *do not* have access to i) significant parallel computational resources *or* ii) the technical tooling to do so.
2. Our EMA Scaling Rule lowers compute usage as it removes the necessity for a hyperparameter search over momenta; in the case where our scaling assumptions hold, if we know the value of the optimal momentum ρ_B at some batch size B (for example, the momentum that gives the best transfer performance), then the optimal value at another batch size $\hat{B}$ is exactly the one given by the EMA Scaling Rule $\hat{\rho} = \rho_B^{\kappa}$, for scaling $\kappa = \hat{B}/B$.
3. Our EMA Scaling Rule enables researchers to more quickly iterate through experimental ideas, and opens up access to large-scale training (for example, larger models and larger datasets) for Pseudo-Labeling and Self-Supervised Learning (SSL) techniques.

These points have potential negative consequences:

1. As our EMA Scaling Rule enables researchers to iterate the same experiments more quickly, and perform large-scale training with EMA-based methods, this may encourage a greater number of experiments, or the training of larger models. Either of these possibilities leads to greater energy consumption.
2. As the need to determine momentum hyperparameters has now been removed, researchers who were previously discouraged from attempting to scale these methods due to an *extra* hyperparameter to tune may begin to perform such experiments, leading, once more, to greater energy consumption.

The environmental impact of each of these two points may be significant.

B Limitations

The EMA Scaling Rule provides a recipe for producing training dynamics independent of the batch size used in stochastic optimization. The technology underpinning it will not *always* give the desired behavior, however.

The first issue occurs with the wording present in the EMA Scaling Rule: *[...] and scale other optimizers according to their own scaling rules* (Definition 1.1):

1. This statement requires that the given Stochastic Differential Equation (SDE) approximation we are using for the model optimizer is itself providing well-behaved scaling, that is, that in the *absence* of a model EMA, the model optimization trajectories at the batch sizes B and κB, with optimizer hyperparameters appropriately scaled, are close. In general we know this is not true. First, we know that the SDE approximation for Stochastic Gradient Descent (SGD) breaks at a given κ due to discretization error (Li et al., 2021). Second, we know that if the gradient noise is not sufficiently large, the SDE approximation for Adam does not exist (Malladi et al., 2022), i.e. an SDE motivated scaling rule has no meaning.
2. This statement requires knowledge of how to scale the corresponding model optimizer. We have principled ways to achieve this for SGD (Li et al., 2021), and for the adaptive optimization methods RMSProp and Adam (Malladi et al., 2022). Empirically, a square-root scaling law for LAMB (You et al., 2020) has been observed, however, it has not been derived formally. Problematically, there is no known hyperparameter scaling law or SDE approximation known for LARS (You et al., 2017), which has been used in Bootstrap Your Own Latent (BYOL) (Grill et al., 2020) and many other large-scale training procedures for convolution-based architectures. Despite this, we are able to demonstrate in Appendix H.6 that a combination of the EMA Scaling Rule and progressive scaling can match, or surpass baseline BYOL performance at a batch size of 32,768 using LARS, indicating that although the theoretical guarantees may not hold, there is still practical utility in the tools we provide in this work.

3. It may be the case that the optimal performance attainable by a given model setup exists at a level of discretization/gradient noise where no SDE exists. In this case, SDE-derived scaling rules can never be valid, and no scaling of this dynamics can be achieved with known tools.

The second issue is related to the case when the optimizer scaling rule is valid. In this case, the error for the EMA Scaling Rule at finite learning rate η at large κ can be considerable. In cases where the model EMA plays a role in the overall optimization, the error introduced by the EMA Scaling Rule can break the preservation of model dynamics.

Put another way, an optimizer scaling rule and the EMA Scaling Rule each introduce their own discretization errors. In the case where EMA plays a role in optimization, as soon as the discretization error of *either* the optimizer scaling rule *or* the EMA Scaling Rule is large, the error for the joint optimization procedure is large. This is *at least* as bad as cases that *do not* use a model EMA during the optimization process.

C The scaling toolbox: practical methods for enabling systematic scaling

There are many different components involved in preserving optimization dynamics at different batch sizes. In this appendix we collect into a single place the different concepts and values that we found useful in practice, in an attempt to make the practice of scaling as accessible as possible.

C.1 The continuous time/SDE perspective

Here we discuss the mindset difference required when trying to preserve training dynamics. In ML we typically use stochastic optimization, leading us to think of the optimization in terms of *performing updates*, or *stepping the optimizer*. This notion has become more common in the era of large datasets, where it may be the case that we only see a fraction of the dataset during optimization.

For dynamics preservation under scaling, we suggest that it is simpler to consider the *amount of data* seen by the training process, or alternatively, the amount of *continuous time* in the discretization of SDEs view. The reason is the following. The SDE scaling rule results (Definition 1.1, Li et al. (2019, 2021); Malladi et al. (2022)) follow from showing that different discretizations of the SDE are close to that SDE, providing we appropriately scale hyperparameters (see Section 2.2). Each of these discretizations shares the *total continuous time* $T = \hat{\eta} \times \widehat{N}_{\text{iter}}$[7] of the underlying SDE, but each discretization has a *different* number of iterations $\widehat{N}_{\text{iter}} = N_{\text{iter}}/\kappa$.

This perspective is already adopted, perhaps by accident in some domains. For example, in Computer Vision (CV), it is typical to compare model performance after optimization on ImageNet1k after a *number of epochs*, whilst also specifing a learning rate warmup after a *number of epochs*. This transforms the schedule into the form *wait until the process meets [condition]*, where here *[condition]* is *when the process has seen sufficiently many samples.*

More generally, we can specify any *condition* that is not a property of the discretization procedure itself. Instead, the discretization procedure should be viewed as a numerical approximation method for the SDE we are evolving, and the properties of that discretization process (like *number of steps*) are not *of specific interest* in the world view where we do decouple optimization from the batch size. A specific example of this more general case is present in Section 3.3, where for scaling $\kappa > 2$ we wait until the pre-training Word Error Rate (WER) is sufficiently low.

There may be cases where one is working with a setup that is explicitly defined in terms of quantities related to the discretization process. Indeed, the optimizer hyperparameters are examples of these, and need to be scaled accordingly with κ. The other typical example of this is conditions based on the *number of optimizer steps*, rather than the number of epochs. In this case, these quantities should be scaled to achieve the desired condition in the same amount of time, i.e. as above $\widehat{N}_{\text{iter}} = N_{\text{iter}}/\kappa$, where N_{iter} is the number of iterations specified at the base batch size B. Concretely, if training is specified in a number of steps, then doubling the batch size implies you should train for half the number of steps.

[7] This is in the case of SGD, for RMSProp and Adam one should use $T = \hat{\eta}^2 \times \widehat{N}_{\text{iter}}$ (Malladi et al., 2022).

C.2 Scaling rules for optimization

For ease of reference, we collect all the scaling rules related to batch size modification we are aware of. We begin with the most well-known, the SGD Scaling Rule (Definitions 2.2 and C.1).

Definition C.1 (SGD Scaling Rule)**.** *When running SGD (Definition 2.1) with batch size $\hat{B} = \kappa B$, use a learning rate $\hat{\eta} = \kappa\eta$ (Krizhevsky, 2014; Goyal et al., 2017).*

The SGD Scaling Rule is also known as the Linear Scaling Rule (LSR), although for clarity, this work adopts the naming convention *[Algorithm Name] Scaling Rule*, which means all parameters of those algorithms are appropriately scaled from batch size B to κB.

Next, we give the two scaling rules known for the adapative optimizers RMSProp (Tieleman et al., 2012) and Adam (Kingma & Ba, 2015) in Definition C.2 and Definition C.3 respectively.

Definition C.2 (RMSProp Scaling Rule)**.** *When running RMSProp (Tieleman et al., 2012) with batch size $\hat{B} = \kappa B$, use a learning rate $\hat{\eta} = \sqrt{\kappa}\eta$, beta coefficient $\hat{\beta} = 1 - \kappa \times (1 - \beta)$, and adaptivity parameter $\hat{\epsilon} = \frac{\epsilon}{\sqrt{\kappa}}$ (Malladi et al., 2022).*

Definition C.3 (Adam Scaling Rule)**.** *When running Adam (Kingma & Ba, 2015) with batch size $\hat{B} = \kappa B$, use a learning rate $\hat{\eta} = \sqrt{\kappa}\eta$, beta coefficients $\hat{\beta}_1 = 1 - \kappa \times (1 - \beta_1)$, $\hat{\beta}_2 = 1 - \kappa \times (1 - \beta_2)$, and adaptivity parameter $\hat{\epsilon} = \frac{\epsilon}{\sqrt{\kappa}}$ (Malladi et al., 2022).*

Next, we present a contribution of this work, the EMA Scaling Rule (Definitions 1.1 and C.4), which extends the above scaling rules to allow the presence of a model EMA which is able to contribute to the overall optimization (see Appendices D and E.1 for derivations).

Definition C.4 (EMA Scaling Rule)**.** *When computing the EMA update (Definition 2.3) of a model undergoing stochastic optimization with batch size $\hat{B} = \kappa B$, use a momentum $\hat{\rho} = \rho^\kappa$ and scale other optimizers according to their own scaling rules.*

Concretely, if we are using SGD in the presence of a model EMA, Definitions C.1 and C.4 state that we should take $\hat{\eta} = \kappa\eta$ *and* $\hat{\rho} = \rho^\kappa$ when scaling by $\kappa = \hat{B}/B$.

The final scaling rule is for weight decay, and follows from the scaling logic discussed in Appendix C.1 and Krizhevsky (2014). If we take the weight decay regularization penalty λ defined at batch size B, what should the weight decay $\hat{\lambda}$ be for batch size $\hat{B} = \kappa B$? For simplicity, consider κ updates of optimization of parameters $\boldsymbol{\theta}_t$ in the presence of weight decay only

$$\boldsymbol{\theta}_{t+\kappa} = \boldsymbol{\theta}_{t+\kappa-1} - \eta\,\lambda\,\boldsymbol{\theta}_{t+\kappa-1} = (1 - \eta\,\lambda)\,\boldsymbol{\theta}_{t+\kappa-1} = (1 - \eta\,\lambda)^\kappa\,\boldsymbol{\theta}_t. \tag{11}$$

Therefore, to match the effect of weight decay with a single iteration step, we need to match

$$1 - \hat{\eta}\,\hat{\lambda} = (1 - \eta\,\lambda)^\kappa. \tag{12}$$

Solving for $\hat{\lambda}$ and expanding around $\eta \approx 0$ gives

$$\hat{\lambda} = \frac{1 - (1 - \eta\,\lambda)^\kappa}{\hat{\eta}} \approx \frac{\eta}{\hat{\eta}} \times \kappa\,\lambda + O(\eta). \tag{13}$$

This leads to the Weight Decay Scaling Rule (Definition C.5).

Definition C.5 (Weight Decay Scaling Rule)**.** *When using weight decay with batch size $\hat{B} = \kappa B$, use a penalty term $\hat{\lambda} = (\kappa\hat{\eta}/\eta)\,\lambda$, where $\hat{\eta}$ and η represent the scaled and unscaled learning rates of the corresponding optimizer (Krizhevsky, 2014; Li et al., 2018; Loshchilov & Hutter, 2019).*

The Weight Decay Scaling Rule implies that using *linear* scaling for the learning rate η then the weight decay penalty is automatically scaled, and when using *square-root* scaling for the learning rate η (e.g. in the case of the Adam Scaling Rule (Definition C.3)) then the weight decay penalty should also be scaled with a *square-root* as is proposed in Loshchilov & Hutter (2019).

Finally, we see that if the implementation of weight decay does not have an update scaled by the learning rate, i.e. the update is $\boldsymbol{\theta}_{t+1} = (1 - \lambda)\,\boldsymbol{\theta}_t$, then the scaling rule is optimizer-independent, and becomes linear for small weight decay, i.e. $\hat{\lambda} = \kappa\lambda$, and for arbitrary λ takes the form $\hat{\lambda} = 1 - (1 - \lambda)^\kappa$.

Table 2: Scaled learning rates $\hat{\eta}$ at different batch sizes $\hat{B} = \kappa B$ given reference learning rates η defined at batch size B. The reference values of each column are boldened. Note that this is only valid when there is a notion of *single sample*. In the sequence learning setup (for example, in Section 3.3), the notion of batch size should be appropriately replaced with the *dynamic batch size*, i.e. total sequence length.

	$\hat{\eta} = \kappa\eta$ [SGD]			$\hat{\eta} = \sqrt{\kappa}\eta$ [RMSProp, Adam]		
	$B = 256$		$B = 512$	$B = 256$	$B = 4096$	
Batch size $\hat{B}$	$\eta = 0.1$	$\eta = 0.3$	$\eta = 0.1$	$\eta = 10^{-3}$	$\eta = 4.8$	$\eta = 10^{-3}$
32	0.0125	0.0375	0.00625	0.00035	0.42426	0.00009
64	0.025	0.075	0.0125	0.0005	0.6	0.00013
128	0.05	0.15	0.025	0.00071	0.84853	0.00018
256	**0.1**	**0.3**	0.05	**0.001**	1.2	0.00025
512	0.2	0.6	**0.1**	0.00141	1.69706	0.00035
1024	0.4	1.2	0.2	0.002	2.4	0.0005
2048	0.8	2.4	0.4	0.00283	3.39411	0.00071
4096	1.6	4.8	0.8	0.004	**4.8**	**0.001**
8192	3.2	9.6	1.6	0.00566	6.78823	0.00141
16384	6.4	19.2	3.2	0.008	9.6	0.002
32768	12.8	38.4	6.4	0.01131	13.57645	0.00283
65536	25.6	76.8	12.8	0.016	19.2	0.004

Table 3: Scaled EMA momenta $\hat{\rho} = \rho^{\kappa}$ at different batch sizes $\hat{B} = \kappa B$ given reference momenta ρ defined at batch size B. The reference values of each column are boldened. Again in the sequence learning setup, batch size should be appropriately replaced with a notion of sequence length.

	$B = 256$			$B = 4096$			
Batch size $\hat{B}$	$\rho = 0.9999$	$\rho = 0.999$	$\rho = 0.99$	$\rho = 0.996$	$\rho = 0.992$	$\rho = 0.99$	$\rho = 0.97$
32	0.99999	0.99987	0.99874	0.99997	0.99994	0.99992	0.99976
64	0.99997	0.99975	0.99749	0.99994	0.99987	0.99984	0.99952
128	0.99995	0.9995	0.99499	0.99987	0.99975	0.99969	0.99905
256	**0.9999**	**0.999**	**0.99**	0.99975	0.9995	0.99937	0.9981
512	0.9998	0.998	0.9801	0.9995	0.999	0.99874	0.9962
1024	0.9996	0.99601	0.9606	0.999	0.99799	0.99749	0.99241
2048	0.9992	0.99203	0.92274	0.998	0.99599	0.99499	0.98489
4096	0.9984	0.98412	0.85146	**0.996**	**0.992**	**0.99**	**0.97**
8192	0.9968	0.96849	0.72498	0.99202	0.98406	0.9801	0.9409
16384	0.99362	0.93798	0.5256	0.9841	0.96838	0.9606	0.88529
32768	0.98728	0.8798	0.27625	0.96844	0.93776	0.92274	0.78374
65536	0.97472	0.77405	0.07632	0.93788	0.8794	0.85146	0.61425

C.3 Commonly used values of hyperparameters at different batch sizes

In the literature it is common to give a base learning rate η defined at batch size 256, implicitly using the SGD Scaling Rule, even when using the Adam optimizer. Because the scaling of other optimization hyperparameters was not understood until recently, it is also common to just present these *for the experiment*, e.g. the Adam betas and epsilon, and the EMA momentum, implicitly defined at the scale of the experiment, for example at batch size 4096. One way to deal with this in practice is to define a single reference batch size B at which *all* hyperparameters are defined, and then scale from there. In this case, it is easiest to compute *using linear scaling* the learning rate at the redefined base batch size $\eta = \tilde{\kappa}\,\eta_{\text{orig}}$, where $\tilde{\kappa} = B/B_{\text{orig}}$, and then scale this new reference η as $\hat{\eta} = \kappa\eta$, $\kappa = \hat{B}/B$, along with e.g. the momentum defined at B.

As this process can be slightly frustrating, we have provided tables of typical learning rates in Table 2 and momenta in Table 3.

C.4 Progressive scaling

In Section 3.4 we introduced Progressive Scaling (Definition 3.2) to test our hypothesis that early in the BYOL training procedure, there are dynamics that are challenging to replicate at larger batch

Algorithm 1 Stochastic Gradient Descent with Progressive Scaling

Require: Base learning rate η, base momentum ρ for base batch size B
Require: Initial target model parameters $\boldsymbol{\theta}$ and model EMA parameters $\boldsymbol{\zeta}$
Require: Epochs E and schedule of batch sizes $\mathcal{B} = B_1, B_2, \ldots, B_E$
Require: Loss function $\mathcal{L}$
  for e in $1, 2 \ldots, E$ **do**
    $\hat{B} \leftarrow \mathcal{B}[e]$ ▷ Get current batch size
    $\kappa \leftarrow \hat{B}/B$ ▷ Compute scaling factor
    $\hat{\eta} \leftarrow \kappa\eta$ ▷ Get scaled learning rate
    $\hat{\rho} \leftarrow \rho^{\kappa}$ ▷ Get scaled momentum
    for b in $1, 2 \ldots, \text{floor}(E/\hat{B})$ **do**
      Sample a minibatch of $\hat{B}$ samples $\mathcal{X} = \{\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(\hat{B})}\}$
      $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - (\hat{\eta}/\hat{B}) \sum_{x \in \mathcal{X}} \nabla_{\boldsymbol{\theta}} \mathcal{L}(x; \boldsymbol{\theta}, \boldsymbol{\zeta})$ ▷ SGD Update
      $\boldsymbol{\zeta} \leftarrow \hat{\rho}\, \boldsymbol{\zeta} + (1 - \hat{\rho})\, \boldsymbol{\theta}$ ▷ EMA Update
    end for
  end for

sizes. To remove ambiguity, in Algorithm 1 we provide pseudo-code for how to use Progressive Scaling.

In Algorithm 1, the prefactor of the SGD update could also have been written η/B, although an equivalent use of the base momentum is not possible.

Finally, we outline how to extend Algorithm 1 to more complex setups, like those presented in Section 3.4:

1. Optimizer scaling rules are used appropriately, for example the Adam scaling rule in case of using the Adam optimizer to update parameters $\boldsymbol{\theta}$.
2. Schedules for hyperparameters are computed using the base hyperparameters, and are then modified by application of the scaling law in epoch (outer) loop.
3. Schedules for hyperparameters at the *step* rather than epoch level can be achieved in practice through recomputing the schedule and updating the notion of minibatch index appropriately throughout training.

All of the above techniques are used in Section 3.4. In addition, scheduling batch sizes within epoch is possible, providing one maintains a notion of computation within some fixed continuous time T_{fixed}. We did not investigate this scenario.

D EMA approximation theorems with SDEs

D.1 SGD with model EMA

We will now derive the EMA scaling rule when tracking model parameters and the model is trained using SGD. We employ a strategy similar to Malladi et al. (2022), where we associate to each iterative process a Stochastic Differential Equation (SDE). In order to control the distance between the SDE and the discrete process, we use the tools from Li et al. (2019).

Definition D.1 (Polynomial growth, Definition 1 in (Li et al., 2019))**.** *The set G is the set of continuous functions $\mathbb{R}^d \to \mathbb{R}$ with at most polynomial growth, i.e., for $g \in G$ there exists two scalars $\kappa_1, \kappa_2 > 0$ such that for all $\boldsymbol{x} \in \mathbb{R}^d$, we have $|g(\boldsymbol{x})| \leq \kappa_1(1 + \|\boldsymbol{x}\|^{\kappa_2})$.*

For an integer $\alpha > 0$, G^{α} is the set of functions $\mathbb{R}^d \to \mathbb{R}$ that are α-times continuously differentiable and such that all their derivatives up to order α are in G.

Similarly to Malladi et al. (2022), we use Noisy Gradient Oracle with Scale Parameter (NGOS) to define the update rules on the parameters.

Definition D.2 (Noisy Gradient Oracle with Scale Parameter (NGOS), adaptation of (Malladi et al., 2022))**.** *A NGOS is a tuple $\mathcal{G}_\sigma = (f, \Sigma, \mathcal{Z}_\sigma)$. Given a noise scale parameter $\sigma > 0$, the NGOS $\mathcal{G}_\sigma$*

takes as input the parameters $\boldsymbol{\theta}$ and outputs a random vector $\mathrm{g} = \nabla f(\boldsymbol{\theta}, \boldsymbol{\zeta}) + \sigma\boldsymbol{\epsilon}$ where $\nabla f(\boldsymbol{\theta}, \boldsymbol{\zeta})$ is the gradient of f with respect to $\boldsymbol{\theta}$ at $(\boldsymbol{\theta}, \boldsymbol{\zeta})$, and $\boldsymbol{\epsilon}$ is a random vector drawn from the distribution $\mathcal{Z}_\sigma(\boldsymbol{\theta}, \boldsymbol{\zeta})$ with zero mean and covariance $\boldsymbol{\Sigma}(\boldsymbol{\theta}, \boldsymbol{\zeta})$.

Note that in the above definition, the probability distribution $\mathcal{Z}_\sigma(\boldsymbol{\theta}, \boldsymbol{\zeta})$ is allowed to change with the scale σ, but its first two moments — its mean and its covariance — are fixed with σ. We have the following theorem for model EMA under optimization with SGD:

Theorem D.1 (SDE for SGD + EMA)**.** *Consider the couple $\mathbf{x}_k = (\boldsymbol{\theta}_k, \boldsymbol{\zeta}_k)$ where $\boldsymbol{\theta}_k$ are the iterates of SGD with a NGOS (Definition D.2) and $\boldsymbol{\zeta}_k$ is an EMA of $\boldsymbol{\theta}_k$, defined, starting from $\mathbf{x}_0 = \boldsymbol{x}_0$, by*

$$\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k - \eta \mathbf{g}_k, \quad \textit{with } \mathbf{g}_k = \nabla f(\boldsymbol{\theta}_k, \boldsymbol{\zeta}_k) + \sigma\boldsymbol{\epsilon}_k, \textit{ and } \boldsymbol{\epsilon}_k \sim \mathcal{Z}_\sigma(\boldsymbol{\theta}_k, \boldsymbol{\zeta}_k), \tag{14}$$

$$\boldsymbol{\zeta}_{k+1} = \rho\boldsymbol{\zeta}_k + (1-\rho)\boldsymbol{\theta}_k \quad . \tag{15}$$

Define $\beta_0 = (1-\rho)/\eta$, $\sigma_0 = \sigma\sqrt{\eta}$, and define the SDE for $X_t = (\Theta_t, Z_t)$, starting from $X_0 = \boldsymbol{x}_0$, by

$$d\Theta_t = -\nabla f(\Theta_t, Z_t)dt + \sigma_0 \boldsymbol{\Sigma}(\Theta_t, Z_t)^{\frac{1}{2}} dW_t, \quad \textit{with } W_t \textit{ a Wiener process} \tag{16}$$

$$dZ_t = \beta_0(\Theta_t - Z_t)dt \quad . \tag{17}$$

Assume that f is continuously differentiable, with $f \in G^3$ and $\boldsymbol{\Sigma}^{\frac{1}{2}} \in G^2$ (Definition D.1). Then, for any time horizon $T > 0$ and test function $g \in G^2$, there exists a constant $c > 0$ such that

$$\max_{k=0,\dots,\lfloor T/\eta \rfloor} |\mathbb{E}[g(X_{\eta k})] - \mathbb{E}[g(\mathbf{x}_k)]| \leq c \times \eta \quad . \tag{18}$$

Proof. The proof uses the same tools as in Li et al. (2019). Define $\Delta(\boldsymbol{\theta}, \boldsymbol{\zeta}) = \eta(-\nabla f(\boldsymbol{\theta}, \boldsymbol{\zeta}) + \sigma\boldsymbol{\epsilon}, \beta_0(\boldsymbol{\theta} - \boldsymbol{\zeta}))$ with $\boldsymbol{\epsilon} \sim \mathcal{Z}_\sigma(\boldsymbol{\theta}, \boldsymbol{\zeta})$ the one-step update for the SGD + EMA update, such that $\mathbf{x}_{k+1} = \mathbf{x}_k + \Delta(\mathbf{x}_k)$. We have the first two moments:

$$\mathbb{E}[\Delta(\boldsymbol{\theta}, \boldsymbol{\zeta})] = \eta(-\nabla f(\boldsymbol{\theta}, \boldsymbol{\zeta}), \beta_0(\boldsymbol{\theta} - \boldsymbol{\zeta})) \tag{19}$$

$$\mathbb{V}[\Delta(\boldsymbol{\theta}, \boldsymbol{\zeta})] = \eta\sigma_0^2 \begin{bmatrix} \boldsymbol{\Sigma}(\boldsymbol{\theta}, \boldsymbol{\zeta}) & 0 \\ 0 & 0 \end{bmatrix} \tag{20}$$

and the higher-order moments are $O(\eta^2)$. Similarly, let $\tilde{\Delta}(\boldsymbol{\theta}, \boldsymbol{\zeta})$ be the solution at time η of the SDE defined by Equation (6) starting from $X_0 = (\boldsymbol{\theta}, \boldsymbol{\zeta})$. From Ito's formula, we also obtain

$$\mathbb{E}[\tilde{\Delta}(\boldsymbol{\theta}, \boldsymbol{\zeta})] = \eta(-\nabla f(\boldsymbol{\theta}), \beta_0(\boldsymbol{\theta} - \boldsymbol{\zeta})) \tag{21}$$

$$\mathbb{V}[\tilde{\Delta}(\boldsymbol{\theta}, \boldsymbol{\zeta})] = \eta\sigma_0^2 \begin{bmatrix} \boldsymbol{\Sigma}(\boldsymbol{\theta}, \boldsymbol{\zeta}) & 0 \\ 0 & 0 \end{bmatrix} \tag{22}$$

and the higher-order moments are $O(\eta^2)$. Hence, the moments of the discrete iteration and of the SDE match up to second order. Following the same proof technique as in Li et al. (2019) then leads to the advertized theorem. □

This theorem is a simple adaptation of the results of Li et al. (2019). Intuitively, it is expected that X_t and $\mathbf{x}_k$ are close since $\mathbf{x}_k$ is the Euler-Maruyama discretization of X_t with learning rate η. We then have the corollary.

Corollary D.1.1 (Validity of the EMA Scaling Rule)**.** *Assume that f is continuously differentiable, with $f \in G^3$ and $\boldsymbol{\Sigma}^{\frac{1}{2}} \in G^2$. Let $\boldsymbol{\theta}_k^B, \boldsymbol{\zeta}_k^B$ the iterates of the Equation* (5) *with batch size B and hyperparameters η, ρ. Let $\boldsymbol{\theta}_k^{\kappa B}, \boldsymbol{\zeta}_k^{\kappa B}$ be iterates with batch size κB, learning rate η determined by the SGD Scaling Rule (Definition 2.2) and momentum determined by the EMA Scaling Rule, linear version (Definition 1.1). Then, for any time horizon $T > 0$ and function $g \in G^2$, there exists a constant $d > 0$ such that*

$$\max_{k=0,\dots,\lfloor T/\eta \rfloor} |\mathbb{E}[g(\boldsymbol{\theta}_{\lfloor k/\kappa \rfloor}^{\kappa B}, \boldsymbol{\zeta}_{\lfloor k/\kappa \rfloor}^{\kappa B})] - \mathbb{E}[g(\boldsymbol{\theta}_k, \boldsymbol{\zeta}_k)]| \leq d \times \eta \quad . \tag{23}$$

Proof. The proof is similar to Malladi et al. (2022). Under the scaling rule, both $\mathbf{x}_k = (\boldsymbol{\theta}_k, \boldsymbol{\zeta}_k)$ and $\hat{\mathbf{x}}_{\lfloor k/\kappa \rfloor} = (\boldsymbol{\theta}_{\lfloor k/\kappa \rfloor}^{\kappa B}, \boldsymbol{\zeta}_{\lfloor k/\kappa \rfloor}^{\kappa B})$ have the same limiting SDE. Hence we have from the previous theorem that for all test function g, we can find c, c' such that

$$\max_{k=0,\dots,\lfloor T/\eta \rfloor} |\mathbb{E}[g(X_{\eta k})] - \mathbb{E}[g(\mathbf{x}_k)]| \leq c \times \eta \text{ and } \max_{k=0,\dots,\lfloor T/\eta \rfloor} |\mathbb{E}[g(X_{\eta k})] - \mathbb{E}[g(\hat{\mathbf{x}}_{\lfloor k/\kappa \rfloor})]| \leq c' \times \eta. \tag{24}$$

The triangle inequality then gives

$$\max_{k=0,\dots,\lfloor T/\eta \rfloor} |\mathbb{E}[g(\hat{\mathbf{x}}_{\lfloor k/\kappa \rfloor})] - \mathbb{E}[g(\mathbf{x}_k)]| \leq (c + c') \times \eta. \tag{25}$$

Hence, taking $d = c + c'$ gives the expected result. □

D.2 Adaptive gradient methods with model EMA

We now turn to the case where one uses an adaptive gradient method rather than SGD to train the model. We follow derivations similar to those of Malladi et al. (2022), with an added EMA. Like above, we consider that the loss function f also depends on the EMA tracking parameter $\boldsymbol{\zeta}_k$. We begin with RMSProp with EMA, which iterates:

$$\mathbf{v}_{k+1} = \gamma \mathbf{v}_k + (1-\gamma)\mathbf{g}_k^2, \text{ with } \mathbf{g}_k = \nabla f(\boldsymbol{\theta}_k, \boldsymbol{\zeta}_k) + \sigma \boldsymbol{\epsilon}_k, \text{ and } \boldsymbol{\epsilon}_k \sim \mathcal{Z}_\sigma(\boldsymbol{\theta}_k, \boldsymbol{\zeta}_k), \tag{26}$$

$$\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k - \eta(\sqrt{\mathbf{v}_k} + \varepsilon)^{-1} \times \mathbf{g}_k \tag{27}$$

$$\boldsymbol{\zeta}_{k+1} = \rho \boldsymbol{\zeta}_k + (1-\rho)\boldsymbol{\theta}_k. \tag{28}$$

Like in Malladi et al. (2022), we place ourselves in the high noise regime, in which the term $\mathbf{g}_k^2$ in Equation (26) is approximated by $\mathbf{g}_k^2 \simeq \sigma^2 \text{diag}(\Sigma(\boldsymbol{\theta}_k, \boldsymbol{\zeta}_k))$. We use the same scaling rules, with an additional one for ρ:

$$\gamma_0 = (1-\gamma)/\eta^2, \quad \sigma_0 = \sigma\eta, \quad \varepsilon_0 = \varepsilon\eta, \text{ and } \beta_0 = (1-\rho)/\eta^2, \tag{29}$$

and we let $\mathbf{u}_k = \mathbf{v}_k/\sigma^2$. The equations for RMSProp with EMA then become, using only these new variables and η:

$$\mathbf{u}_{k+1} - \mathbf{u}_k = \eta^2 \gamma_0 (\text{diag}(\Sigma(\boldsymbol{\theta}_k, \boldsymbol{\zeta}_k)) - \mathrm{u}_k), \tag{30}$$

$$\boldsymbol{\theta}_{k+1} - \boldsymbol{\theta}_k = -(\sqrt{\mathbf{u}_k} + \varepsilon_0)^{-1} \left(\eta^2 \nabla f(\boldsymbol{\theta}_k, \boldsymbol{\zeta}_k) + \eta \boldsymbol{\epsilon}_k\right) \tag{31}$$

$$\boldsymbol{\zeta}_{k+1} - \boldsymbol{\zeta}_k = \eta^2 \beta_0 (\boldsymbol{\theta}_k - \boldsymbol{\zeta}_k). \tag{32}$$

This formulation makes it clear that these iterations can be seen as the discretization of the SDE

$$dU_t = \gamma_0 (\text{diag}(\Sigma(\Theta_t, Z_t)) - U_t) dt, \tag{33}$$

$$d\Theta_t = -(\sigma_0 \sqrt{U_t} + \varepsilon_0)^{-1} (\nabla f(\Theta_t, Z_t) dt + \sigma_0 \Sigma(\Theta_t, Z_t)^{1/2} dWt) \tag{34}$$

$$dZ_t = \beta_0 (\Theta_t - Z_t) dt, \tag{35}$$

with step size η^2. Of course, we recover the SDE of Malladi et al. (2022) in the case where $\beta_0 = 0$. A formal proof of closeness between the iterates and the SDE trajectory is out of the scope of the present paper since it would imply redoing much of the theoretical work developed in Malladi et al. (2022). Still, the previous informal analysis hints that for RMSProp, the scaling rule in Equation (29) should be used. In other words, given a certain set of hyperparameters γ, η and ρ, if the batch size goes from B to $\hat{B} = \kappa \times B$, the noise level becomes $\hat{\sigma} = \sigma/\sqrt{\kappa}$, and keeping the quantities in Equation (29) constant means that we should use as new hyperparameters

$$\hat{\gamma} = 1 - (1-\gamma) \times \kappa, \quad \hat{\eta} = \eta \times \sqrt{\kappa}, \text{ and } \hat{\rho} = 1 - (1-\rho) \times \kappa \ .$$

The linear rule $\hat{\rho} = 1 - (1-\rho) \times \kappa$ is at the first order equivalent to the exponential scaling rule $\hat{\rho} = \rho^\kappa$. Hence, even though the limiting SDE differs greatly from that of SGD, and even though the scaling rule regarding the learning rate differs, we recover for the momentum term ρ the exact same scaling rule as for SGD.

We finish the discussion with the case of Adam, which leads once again to the same rule as for SGD. Adam with EMA tracking of the network parameters iterates

$$\mathbf{m}_{k+1} = \beta_1 \mathbf{m}_k + (1-\beta_1)\mathbf{g}_k, \text{ with } \mathbf{g}_k = \nabla f(\boldsymbol{\theta}_k, \boldsymbol{\zeta}_k) + \sigma \boldsymbol{\epsilon}_k, \text{ and } \boldsymbol{\epsilon}_k \sim \mathcal{Z}_\sigma(\boldsymbol{\theta}_k, \boldsymbol{\zeta}_k), \tag{36}$$

$$\mathbf{v}_{k+1} = \beta_2 \mathbf{v}_k + (1-\beta_2)\mathbf{g}_k^2 \tag{37}$$

$$\tilde{\mathbf{m}}_{k+1} = \mathbf{m}_{k+1}/(1-\beta_1^{k+1}) \tag{38}$$

$$\tilde{\mathbf{v}}_{k+1} = \mathbf{v}_{k+1}/(1-\beta_2^{k+1}) \tag{39}$$

$$\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k - \eta(\sqrt{\tilde{\mathbf{v}}_k} + \varepsilon)^{-1} \times \tilde{\mathbf{m}}_{k+1} \tag{40}$$

$$\boldsymbol{\zeta}_{k+1} = \rho \boldsymbol{\zeta}_k + (1-\rho)\boldsymbol{\theta}_k \ . \tag{41}$$

Here, we use the same minor modification of the iterations as in Malladi et al. (2022), where we use $\mathbf{v}_k$ instead of $\mathbf{v}_{k+1}$ in the denominator of the $\boldsymbol{\theta}_k$ update.

We consider the following scaling for the hyperparameters

$$c_1 = (1-\beta_1)/\eta^2, \;\; c_2 = (1-\beta_2)/\eta^2, \;\; \sigma_0 = \sigma\eta, \;\; \varepsilon_0 = \varepsilon\eta, \text{ and } \beta_0 = (1-\rho)/\eta^2, \tag{42}$$

and $\gamma_1(t) = 1 - \exp(-c_1 t)$, $\gamma_2(t) = 1 - \exp(-c_2 t)$, and $\mathbf{u}_k = \mathbf{v}_k/\sigma^2$. The SDE for Adam + EMA is given by

$$dM_t = c_1 \left((\nabla f(\Theta_t, Z_t) - M_t)dt + \sigma_0 \Sigma(\Theta_t, Z_t)^{1/2} dW_t \right) \tag{43}$$

$$dU_t = c_2(\mathrm{diag}(\Sigma(\Theta_t, Z_t)) - U_t)dt \tag{44}$$

$$d\Theta_t = -\frac{\sqrt{\gamma_2(t)}}{\gamma_1(t)} (\sigma_0 \sqrt{U_t} + \varepsilon_0 \sqrt{\gamma_2(t)})^{-1} \times M_t dt \tag{45}$$

$$dZ_t = \beta_0(\Theta_t - Z_t)dt. \tag{46}$$

This is once again the same SDE as in Malladi et al. (2022) with the added EMA term. Like previously, this SDE hints at the fact that the scaling rule in eq. (42) should be used. In other words, given a set of hyperparameters β_1, β_2, η, and ρ, if the batch size goes from B to $\kappa \times B$, then the noise level becomes $\hat{\sigma} = \sigma/\sqrt{\kappa}$ and keeping quantities in eq. (42) constant means that we should use as new hyperparameters

$$\hat{\beta}_1 = 1 - (1-\beta_1) \times \kappa, \;\; \hat{\beta}_2 = 1 - (1-\beta_2) \times \kappa, \;\; \hat{\eta} = \eta \times \sqrt{\kappa}, \text{ and } \hat{\rho} = 1 - (1-\rho) \times \kappa.$$

We once again recover a linear rule for $1-\rho$ which is equivalent to the exponential scaling rule $\hat{\rho} = \rho^\kappa$ in the limit $\rho \to 0$.

E Additional proofs

E.1 Iterations of SGD + EMA

Here we derive a critical component of the EMA Scaling Rule, the matrix equation of Equation (4) from which the EMA Scaling Rule (Definition 1.1) follows.

Theorem E.1 (Iterations of SGD + EMA)**.** *Assuming that gradients change slowly over iterations of SGD (Definition 2.1) and EMA (Definition 2.3):* $\nabla_{\boldsymbol{\theta}} \mathcal{L}(x; \boldsymbol{\theta}_{t+j}, \boldsymbol{\zeta}_{t+j}) \approx \nabla_{\boldsymbol{\theta}} \mathcal{L}(x; \boldsymbol{\theta}_t, \boldsymbol{\zeta}_t) \approx \mathbf{g}$*, for* $j = 1, 2, \ldots, \kappa$ *and representative gradient* $\mathbf{g}$*, iterating over* κ *independent minibatches produces model states*

$$\begin{bmatrix} \boldsymbol{\theta}_{t+\kappa} \\ \boldsymbol{\zeta}_{t+\kappa} \\ \mathbf{g} \end{bmatrix} = \begin{bmatrix} 1 & 0 & -\eta \\ 1-\rho & \rho & 0 \\ 0 & 0 & 1 \end{bmatrix}^\kappa \cdot \begin{bmatrix} \boldsymbol{\theta}_t \\ \boldsymbol{\zeta}_t \\ \mathbf{g} \end{bmatrix} = \begin{bmatrix} \boldsymbol{\theta}_t - \eta\, \kappa\, \mathbf{g} \\ \rho^\kappa\, \boldsymbol{\zeta}_t + (1-\rho^\kappa)\, \boldsymbol{\theta}_t + O\left(\eta \times \beta_\rho\right) \\ \mathbf{g} \end{bmatrix}. \tag{47}$$

Proof. First note that for matrices of the form

$$\boldsymbol{A} = \begin{bmatrix} 1 & 0 & a_{0,2} \\ 1-a_{1,1} & a_{1,1} & 0 \\ 0 & 0 & 1 \end{bmatrix}, \tag{48}$$

their multiplication follows

$$\begin{aligned} \boldsymbol{A}\,\boldsymbol{B} &= \begin{bmatrix} 1 & 0 & a_{0,2} \\ 1-a_{1,1} & a_{1,1} & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & b_{0,2} \\ 1-b_{1,1} & b_{1,1} & 0 \\ 0 & 0 & 1 \end{bmatrix} \\ &= \begin{bmatrix} 1 & 0 & a_{0,2} + b_{0,2} \\ 1 - a_{1,1}\, b_{1,1} & a_{1,1}\, b_{1,1} & (1-a_{1,1})\, b_{0,2} \\ 0 & 0 & 1 \end{bmatrix}, \end{aligned} \tag{49}$$

and

$$\begin{aligned} \boldsymbol{A}\,\boldsymbol{B}\,\boldsymbol{C} &= \begin{bmatrix} 1 & 0 & a_{0,2} + b_{0,2} \\ 1 - a_{1,1}\, b_{1,1} & a_{1,1}\, b_{1,1} & (1-a_{1,1})\, b_{0,2} \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & c_{0,2} \\ 1-c_{1,1} & c_{1,1} & 0 \\ 0 & 0 & 1 \end{bmatrix} \\ &= \begin{bmatrix} 1 & 0 & a_{0,2} + b_{0,2} + c_{0,2} \\ 1 - a_{1,1}\, b_{1,1}\, c_{1,1} & a_{1,1}\, b_{1,1}\, c_{1,1} & (1-a_{1,1})\, b_{0,2} + (1 - a_{1,1}\, b_{1,1})\, c_{0,2} \\ 0 & 0 & 1 \end{bmatrix}. \end{aligned} \tag{50}$$

By induction

$$\boldsymbol{A}^{\kappa} = \begin{bmatrix} 1 & 0 & \kappa \times a_{0,2} \\ 1 - a_{1,1}^{\kappa} & a_{1,1}^{\kappa} & \delta(a_{0,2}, a_{1,1}, \kappa) \\ 0 & 0 & 1 \end{bmatrix}, \tag{51}$$

where

$$\delta(a_{0,2}, a_{1,1}, \kappa) = a_{0,2} \sum_{i=1}^{\kappa-1} \left(1 - a_{1,1}^{i}\right) = a_{0,2} \left(\kappa - \frac{1 - a_{1,1}^{\kappa}}{1 - a_{1,1}}\right), \quad \text{for } a_{1,1} \neq 1. \tag{52}$$

It follows that

$$\begin{bmatrix} 1 & 0 & -\eta \\ 1 - \rho & \rho & 0 \\ 0 & 0 & 1 \end{bmatrix}^{\kappa} = \begin{bmatrix} 1 & 0 & -\kappa\,\eta \\ 1 - \rho^{\kappa} & \rho^{\kappa} & \delta(\eta, \rho, \kappa) \\ 0 & 0 & 1 \end{bmatrix} \tag{53}$$

where the EMA Scaling Rule error

$$\delta(\eta, \rho, \kappa) = (-\eta) \left(\kappa - \frac{1 - \rho^{\kappa}}{1 - \rho}\right) \approx (-\eta) \left(\kappa - \kappa + O(\beta_{\rho})\right) = 0 + O(\eta \times \beta_{\rho}), \tag{54}$$

where $\beta_{\rho} \equiv 1 - \rho$ and the approximation is around $\rho = 1$. □

E.2 Limiting behavior of Polyak-Ruppert averaging

Here we sketch the asymptotic behavior of a target model θ and its EMA ζ. Let us assume that θ converges to the stationary distribution $\lim_{t\to\infty} \theta_t = \theta^*$, $\theta^* \sim p_{\infty}(\theta)$. We are interested in statistical properties of $\zeta^* = \lim_{t\to\infty} \zeta_t$, as this will formalize the notion of how the EMA depends on the a time-horizon defined by its momentum ρ as discussed in Table 1.

As a warm-up, for n independent random variables $\mathrm{x}_1, \ldots, \mathrm{x}_2$, we know that the sample mean $\bar{x} = \frac{1}{n}(x_1, x_2, \ldots, x_n)$ has the statistical properties

$$\mathbb{E}[\bar{x}] = \mu, \qquad \mathrm{Var}[\bar{x}] = \frac{\sigma^2}{n}, \tag{55}$$

where μ and σ are the population mean and variance. This gives us an idea of what to expect. As we will now show, the expectation of ζ^* should have no time-horizon dependence, whereas the variance of ζ^* will depend on its time horizon (i.e. the number of samples it integrates over) which is defined by ρ.

In the case of a weighted sum

$$\bar{x}^{(w)} = \sum_{i=1}^{n} w_i\, x_i, \tag{56}$$

then if the x_i are Independent and Identically Distributed (i.i.d.), then

$$\mathbb{E}[\bar{x}^{(w)}] = \sum_{i=1}^{n} w_i\, \mathbb{E}[x_i] = n\,\bar{w}\,\mu, \qquad \bar{w} = \frac{1}{n} \sum_{i=1}^{n} w_i, \tag{57}$$

and for the variance (Kish, 1965)

$$\mathrm{Var}[\bar{x}^{(w)}] = n \cdot \overline{w^2} \cdot \sigma^2 \qquad \overline{w^2} = \frac{1}{n} \sum_{i=1}^{n} w_i^2, \qquad \sigma^2 = \mathrm{Var}[x_i]. \tag{58}$$

We can verify that we reproduce the well-known result in Equation (55) in the case where all weights are equal to $\frac{1}{n}$ as follows

$$\forall i : w_i = \frac{1}{n} \implies \overline{w^2} = \frac{1}{n} \cdot \sum_{i=1}^{n} \left(\frac{1}{n}\right)^2 = \frac{1}{n^2} \implies \mathrm{Var}[\bar{x}^{(w)}] = n \cdot \frac{1}{n^2} \cdot \sigma^2 = \frac{\sigma^2}{n}. \tag{59}$$

In the case of an exponential moving average we have

$$\zeta_{t+1} = \rho\,\zeta_t + (1-\rho)\,\theta_t = \rho^t\,\zeta_1 + (1-\rho)\sum_{i=0}^{t-1}\rho^i\theta_{t-i}. \tag{60}$$

Let's consider the specific case where we are at iteration k which is sufficiently large that ζ and θ have converged to their stationary distributions. From k, the iterations unfold as

$$\zeta_{t+1} = \rho^{t+1-k}\,\zeta_k + (1-\rho)\sum_{i=0}^{t-k}\rho^i\theta_{t-i}. \tag{61}$$

We rearrange for terms in ζ

$$\zeta_{t+1} - \rho^{t+1-k}\,\zeta_k = (1-\rho)\sum_{i=0}^{t-k}\rho^i\,\theta_{t-i}, \tag{62}$$

and before proceeding to the final result, using $n = t+1-k$, we compute the convenient quantities

$$\bar{\rho} = \frac{1}{n}\sum_{i=0}^{n-1}\rho^i = \frac{1}{n}\times\frac{1-\rho^n}{1-\rho} \tag{63}$$

$$\overline{\rho^2} = \frac{1}{n}\sum_{i=0}^{n-1}\rho^{2i} = \frac{1}{n}\times\frac{1-\rho^{2n}}{1-\rho^2}. \tag{64}$$

Taking expectation of Equation (62) and setting statistics to their stationary values, we have

$$(1-\rho^n)\,\mathbb{E}[\zeta^*] = (1-\rho)\,n\,\bar{\rho}\,\mathbb{E}[\theta^*] = (1-\rho^n)\,\mathbb{E}[\theta^*], \tag{65}$$

where we have used the result in Equation (57). It follows that for $\rho \neq 1$ we have

$$\mathbb{E}[\zeta^*] = \mathbb{E}[\theta^*], \tag{66}$$

independent of ρ. Finally, we can take the variance of Equation (62). First the left hand side

$$\mathrm{Var}\left[\zeta_{t+1} - \rho^n\,\zeta_k\right] = \mathrm{Var}\left[\zeta_{t+1}\right] + \rho^{2n}\ \mathrm{Var}\left[\zeta_k\right] = \left(1+\rho^{2n}\right)\ \mathrm{Var}\left[\zeta^*\right]. \tag{67}$$

Next the right hand side

$$\mathrm{Var}\left[(1-\rho)\sum_{i=0}^{n-1}\rho^i\,\theta_{t-i}\right] = (1-\rho)^2\ \mathrm{Var}\left[\sum_{i=0}^{n-1}\rho^i\,\theta_{t-i}\right] = (1-\rho)^2\cdot\left(\frac{1-\rho^{2n}}{1-\rho^2}\right)\cdot\mathrm{Var}[\theta^*]. \tag{68}$$

Finally, equating left and right hand sizes and rearranging for $\mathrm{Var}[\zeta^*]$ gives

$$\mathrm{Var}\left[\zeta^*\right] = \frac{1-\rho^{2n}}{1+\rho^{2n}}\cdot\frac{1-\rho}{1+\rho}\cdot\mathrm{Var}\left[\theta^*\right] \tag{69}$$

In the limit $t \to \infty$, the momentum-dependent prefactor becomes

$$\lim_{t\to\infty}\left(\frac{1-\rho^{2n}}{1+\rho^{2n}}\cdot\frac{1-\rho}{1+\rho}\right) = \frac{1-\rho}{1+\rho} \implies \lim_{t\to\infty}\mathrm{Var}\left[\zeta^*\right] = \frac{1-\rho}{1+\rho}\cdot\mathrm{Var}\left[\theta^*\right]. \tag{70}$$

Equations (69) and (70) validate our intuition. When $\rho \to 0$, then ζ behaves like θ independent of T, with their variance and expectation matching. When $\rho > 0$, the momentum-dependent prefactor serves as an aggregator over the history when t is sufficiently large compared to k, reducing the variance $\mathrm{Var}[\zeta^*]$ but preserving its expectation. This formalizes the notion of time horizon discussed in Table 1.

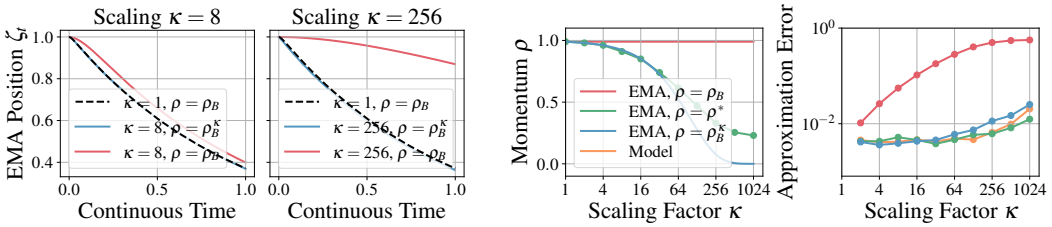
F Additional details and results for Polyak-Ruppert averaging

Additional background Polyak-Ruppert averaging (Definition 3.1) is a simplification of Stochastic Weight Averaging (SWA) (Izmailov et al., 2018) which uses a more complex multi-cycle schedule based weighting of the model parameters. Both Definition 3.1 and SWA present similar favorable properties like wider minima and better generalization (Izmailov et al., 2018). For example, He et al. (2022) observed that a supervised ViT-H/14 overfits on ImageNet1k (Russakovsky et al., 2014) without a model EMA, achieving an accuracy of 80.9%. Equipping a Polyak-Ruppert average ($\rho = 0.9999$) alleviated overfitting and gave a 83.1% accuracy.

Organization In this appendix, we look at additional momenta for one-dimensional noisy parabola, as well as extensions to D-dimensions (Appendix F.1), provide a more detailed view of the results of Section 3.2 (Appendix F.2), and investigate the scenario where the EMA Scaling Rule (Definition 1.1) is applied to batch normalization (Ioffe & Szegedy, 2015) coefficients (Appendix F.3).

F.1 Noisy parabola

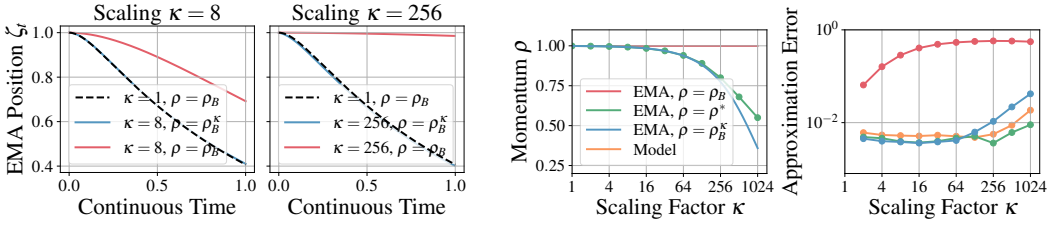
Additional one-dimensional examples First we consider additional one-dimensional examples, investigating the effect of modifying the base momentum ρ_B. We present $\rho_B = 0.99$ in Figure 7, and $\rho_B = 0.999$ in Figure 8. The results for $\rho_B = 0.9999$ are presented in main text in Figure 1.



(a) Trajectory of the model EMA ζ under different scalings κ, with $\rho_B = 0.99$, $\eta_B = 10^{-4}$.

(b) Choices for momentum (left) with corresponding approximation errors (Equation (10)) (right).

Figure 7: (a) We show the effect of scaling by comparing model EMA trajectories of the baseline ($\kappa = 1$, black dashed) to $\kappa = 8$ (left) and $\kappa = 256$ (right), with ($\rho = \rho_B^\kappa$, blue) and without ($\rho = \rho_B$, red) the EMA Scaling Rule. (b, left) The momentum according for different scaling rules and the empirically optimal ρ^* (Equation (10)). (b, right) The approximation error (Equation (10)) of trajectories in (b, left) and the target model (orange). Error for ρ^* is computed using a hold-out to mitigate overfitting.



(a) Trajectory of the model EMA ζ under different scalings κ, with $\rho_B = 0.999$, $\eta_B = 10^{-4}$.

(b) Choices for momentum (left) with corresponding approximation errors (Equation (10)) (right).

Figure 8: (a) We show the effect of scaling by comparing model EMA trajectories of the baseline ($\kappa = 1$, black dashed) to $\kappa = 8$ (left) and $\kappa = 256$ (right), with ($\rho = \rho_B^\kappa$, blue) and without ($\rho = \rho_B$, red) the EMA Scaling Rule. (b, left) The momentum according for different scaling rules and the empirically optimal ρ^* (Equation (10)). (b, right) The approximation error (Equation (10)) of trajectories in (b, left) and the target model (orange). Error for ρ^* is computed using a hold-out to mitigate overfitting.

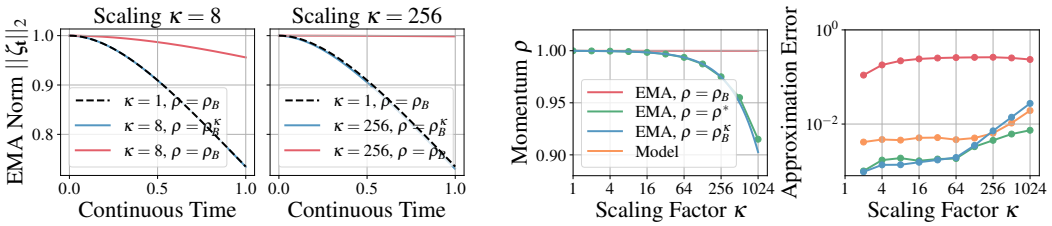
As described by the scaling error term in Equation (54), the approximation error at a given κ is higher for lower momenta ρ. For a large range of scalings κ, the EMA Scaling Rule and the optimal momenta ρ^* are consistent. In summary, we see the synthetic experiments validate the results of Section 3.1 for a range of momenta ρ.

Examples in higher dimensions Our final use of the synthetic *noisy* parabola will consider an extension to D dimensions. Consider the optimization of $\theta \in \mathbb{R}^D$ in a *noisy parabola* at the origin:

$$\mathcal{L}(\theta) = \frac{a}{2}\,\theta^\intercal\theta, \qquad \theta_{k+1} = \theta_k - \eta\, g_k, \qquad g_k = a\,\theta_k + \epsilon_k, \qquad \epsilon_k \sim \mathcal{N}\left(0, \frac{b\,g_k^2 + c}{\kappa}\right), \tag{71}$$

for curvature $a > 0$, scaled additive $b > 0$, and additive $c > 0$ noise coefficients. The scaling factor κ in the covariance denominator implements the reduction in gradient noise as the scaling (i.e., the batch size) increases (Jastrzebski et al., 2017). Let $\theta \in \mathbb{R}^D$ be optimized with SGD (Definition 2.1)

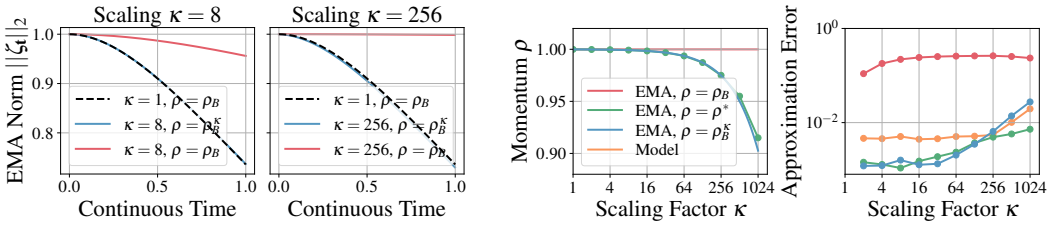
and let there be a Polyak-Ruppert average (Definition 3.1) $\zeta \in \mathbb{R}^D$ with momentum $\rho = 1 - \beta$ for θ. We consider dimensionalities $D = 2$ (Figure 9), $D = 16$ (Figure 10), and $D = 100$ (Figure 11). We observe no significant differences in the EMA scaling behavior as we vary dimensions.



(a) Norm of the model EMA ζ under different scalings κ, with $\rho_B = 0.9999$, $\eta_B = 10^{-4}$, $D = 2$.

(b) Choices for momentum (left) with corresponding approximation errors (Equation (10)) (right).

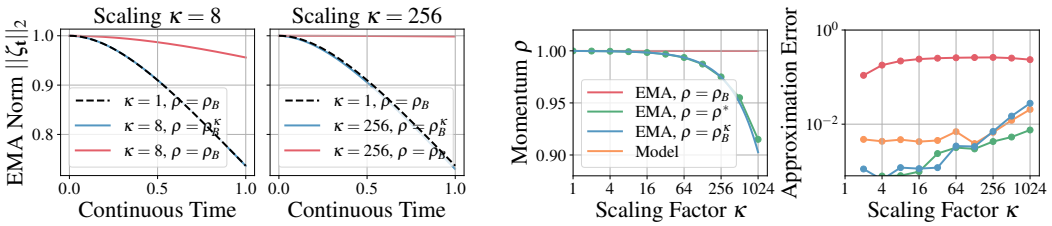
Figure 9: (a) We show the effect of scaling by comparing model EMA trajectories of the baseline ($\kappa = 1$, black dashed) to $\kappa = 8$ (left) and $\kappa = 256$ (right), with ($\rho = \rho_B^\kappa$, blue) and without ($\rho = \rho_B$, red) the EMA Scaling Rule. (b, left) The momentum according for different scaling rules and the empirically optimal ρ^* (Equation (10)). (b, right) The approximation error (Equation (10)) of trajectories in (b, left) and the target model (orange). Error for ρ^* is computed using a hold-out to mitigate overfitting.



(a) Norm of the model EMA ζ under different scalings κ, with $\rho_B = 0.9999$, $\eta_B = 10^{-4}$, $D = 16$.

(b) Choices for momentum (left) with corresponding approximation errors (Equation (10)) (right).

Figure 10: (a) We show the effect of scaling by comparing model EMA trajectories of the baseline ($\kappa = 1$, black dashed) to $\kappa = 8$ (left) and $\kappa = 256$ (right), with ($\rho = \rho_B^\kappa$, blue) and without ($\rho = \rho_B$, red) the EMA Scaling Rule. (b, left) The momentum according for different scaling rules and the empirically optimal ρ^* (Equation (10)). (b, right) The approximation error (Equation (10)) of trajectories in (b, left) and the target model (orange). Error for ρ^* is computed using a hold-out to mitigate overfitting.



(a) Norm of the model EMA ζ under different scalings κ, with $\rho_B = 0.9999$, $\eta_B = 10^{-4}$, $D = 100$.

(b) Choices for momentum (left) with corresponding approximation errors (Equation (10)) (right).

Figure 11: (a) We show the effect of scaling by comparing model EMA trajectories of the baseline ($\kappa = 1$, black dashed) to $\kappa = 8$ (left) and $\kappa = 256$ (right), with ($\rho = \rho_B^\kappa$, blue) and without ($\rho = \rho_B$, red) the EMA Scaling Rule. (b, left) The momentum according for different scaling rules and the empirically optimal ρ^* (Equation (10)). (b, right) The approximation error (Equation (10)) of trajectories in (b, left) and the target model (orange). Error for ρ^* is computed using a hold-out to mitigate overfitting.

Table 4: Supervised ResNet-v2 50 hyperparameters used in Polyak-Ruppert Averaging experiments.

	Supervised ResNet-v2 50
ImageNet1k Test Top-1	$76.27 \pm 0.10\%$
ImageNet1k EMA Test Top-1	$76.55 \pm 0.07\%$
Weight initialization	`kaiming_normal(relu)`
Backbone normalization	BatchNorm
Synchronized BatchNorm over replicas	No
Learning rate schedule	Multi step: $\times 0.1$ at $[30, 60, 80]$ epochs
Learning rate warmup (epochs)	5
Learning rate minimum value	1×10^{-6}
Training duration (epochs)	90
Optimizer	SGD + Momentum
SGD momentum	0.9
Optimizer scaling rule	Linear
Base learning rate	0.4
Base batch size	1024
Base Polyak momentum	0.9999
Weight decay	1×10^{-4}
Weight decay scaling rule	None
Weight decay skip bias	Yes
Numerical precision	`bf16`
Augmentation stack	ImageNet
Label smoothing rate	0.1

F.2 Image Classification

Hyperparameters We present the base hyperparameters for our image experiments in Table 4.

Data For large scale vision evaluation, we use the ImageNet1k dataset (Russakovsky et al., 2014), a widely used dataset containing approximately 1.2 million labeled images, distributed almost uniformly across 1000 different object classes, like animals, plants, and vehicles.

The images in ImageNet1k are are not consistent in resolution. To handle this, they are resized and cropped to a standard size (in our case, 224×224), before further processing. This is part of the standard ImageNet augmentation stack for convolutional networks mentioned in Table 4.

Compute *[This section has been redacted to preserve anonymity during the peer-review process. If this work is accepted, the full details compute used for these experiments, including: the experiments presented, hyperparameter optimization, and the development process, will be provided.]*

Additional results In Figure 12 we present a more detailed view of the results in Section 3.2. First, we see that for all train metrics, model trajectories match, and that a learning rate step schedule after warmup is present. As discussed in Figure 12, a gap in EMA Test Top-1 trajectories begins at scaling $\kappa = 4$, with a more pronounced effect visible at $\kappa = 8$. From Figure 12 it is clear that the (non-EMA) Test Top-1 performance trajectory is no longer matching at these scalings, demonstrating that the problem is not due to a breakdown of the EMA Scaling Rule, but rather, that the model is overfitting at larger batch sizes due to batch normalization (Ioffe & Szegedy, 2015).

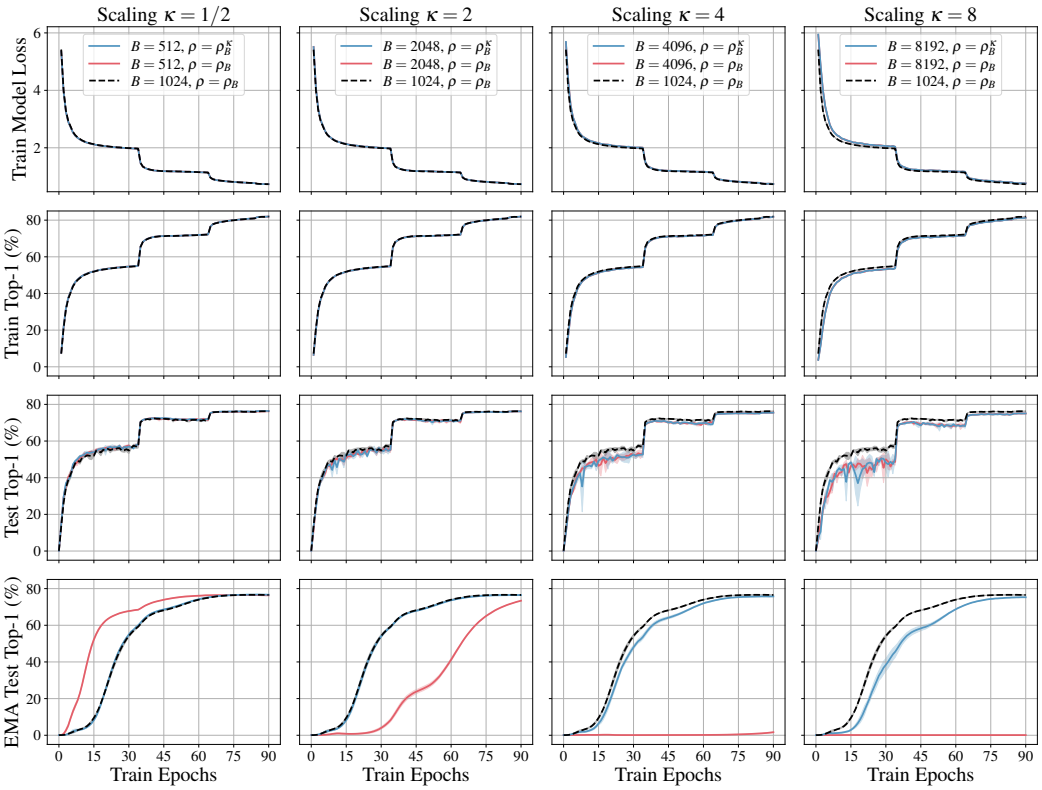


Figure 12: *ResNetv2-50 Polyak-Ruppert averaging on ImageNet1k* for different scalings κ. The baseline model ($\kappa = 1$, black dashed) uses batch size 1024 and momentum $\rho_B = 0.9999$, is scaled down to a batch size of 512 (left), and up to a batch size of 4096 (right) with (blue, $\rho = \rho_B^\kappa$) and without (red, $\rho = \rho_B$) the EMA Scaling Rule (Definition 1.1). Bands indicate the mean and standard deviation across three runs.

F.3 Applying the EMA Scaling Rule to Batch Normalization

In Section 3.2 and Appendix F.2, we investigated a range of scalings κ, *with* and *without* applying the EMA Scaling Rule to the Polyak momentum. In those experiments, we maintained batch normalization (Ioffe & Szegedy, 2015) coefficients of $\rho_{\text{BN}} = 0.9$ throughout[8], i.e. the EMA Scaling Rule is not applied. Yet, the running statistics of Batch Normalization *are* an EMA with values determined by ρ_{BN} and so it is reasonable to suspect we should apply the EMA Scaling Rule to ρ_{BN} also.

In Figure 13 we investigate the effect of applying the EMA Scaling Rule to Batch Normalization coefficients, using $\hat{\rho}_{\text{BN}} = \rho_{\text{BN}}^\kappa$. We observe that the Test Top-1 trajectories *with* the EMA Scaling Rule applied are slightly closer to the reference trajectories for scalings $\kappa \geq 2$ than those trajectories *without* the EMA Scaling Rule. As the effect is not particularly large, at least in this setup, we do pursue further ablating applications of the EMA Scaling Rule to batch normalization coefficients, and always use $\rho_{\text{BN}} = 0.1$ for Batch Normalization, independent of κ.

[8] In many ML frameworks, this value is defined using $\beta_\rho = 1 - \rho$, i.e. the default is 0.1 and corresponds to β_{BN} rather than 0.9 corresponding to ρ_{BN}. We use ρ_{BN} to maintain consistency across this work.

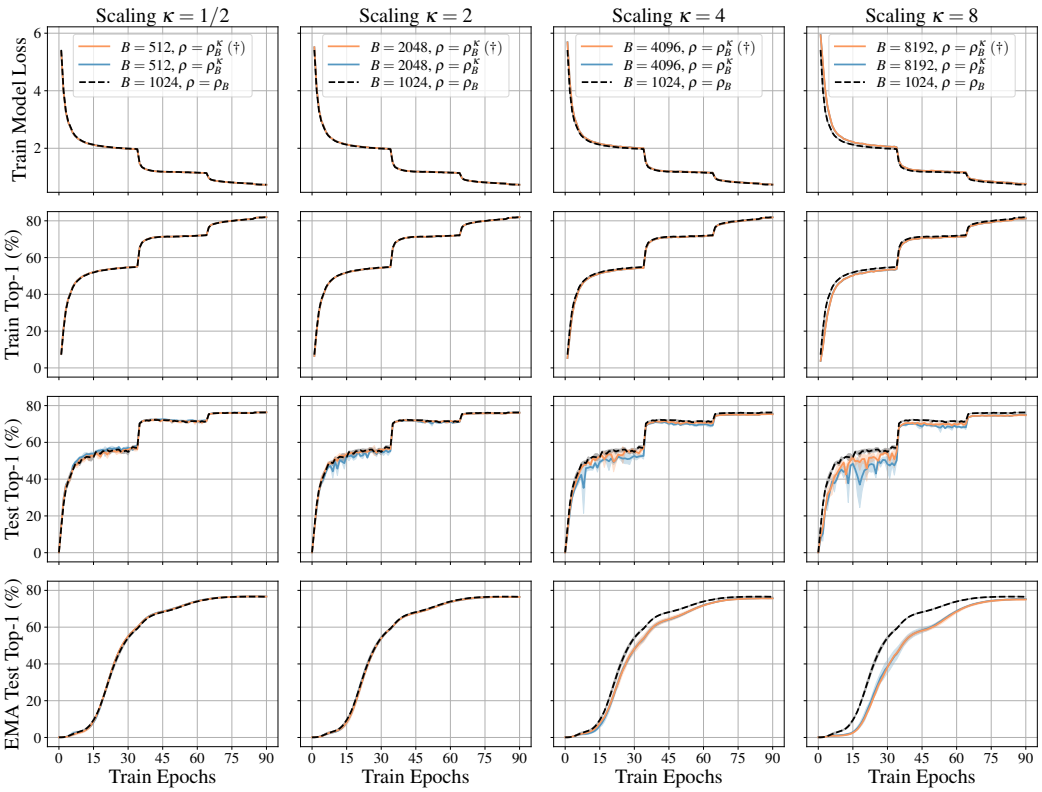


Figure 13: *ResNetv2-50 Polyak-Ruppert averaging on ImageNet1k* for different scalings κ. The baseline model ($\kappa = 1$, black dashed) uses batch size 1024 and momentum $\rho_B = 0.9999$, is scaled down to a batch size of 512 (left), and up to a batch size of 4096 (right) with the EMA Scaling Rule applied to *only* model parameters (blue, $\rho = \rho_B^\kappa$), and model parameters *and* buffers (orange, $\rho = \rho_B^\kappa$ (†)). Bands indicate the mean and standard deviation across three runs.

G Additional details and results for Automatic Speech Recognition (ASR)

In this section we provide additional details for the speech recognition experiments in both the supervised and semi-supervised case.

Data We use the LibriSpeech dataset (Panayotov et al., 2015) which is a dataset of audio-transcription pairs. For supervised Polyak-Ruppert averaging experiments, we use *train-clean-100* as training data, and for semi-supervised pseudo-labeling experiments, we use *train-clean-100* as the labeled and *train-clean-360* and *train-other-500* as the unlabeled data. The standard LibriSpeech validation sets (*dev-clean* and *dev-other*) are used to tune all hyperparameters, as well as to select the best models. Test sets (*test-clean* and *test-other*) are only used for reporting final model performance, measured in WER without an external language model. We maintain the original 16kHz sampling rate, and compute log-mel filterbanks with 80 coefficients for a 25ms sliding window, strided by 10ms, later normalized to zero mean and unit variance for each input sequence.

Acoustic model We employ a vanilla encoder-based only transformer model trained with the Connectionist Temporal Classification (CTC) loss (Graves et al., 2006). We use the training configuration from Likhomanenko et al. (2021a), which has three stages: i) 1D convolutions to perform striding (kernel of 7 with stride of 3); ii) a Transformer encoder with 36 layers, post-LayerNorm, four attention heads, an embedding dimension of 768, an MLP dimension of 3072, a dropout frequency of 0.3, and a layer drop frequency of 0.3; and iii) a linear layer to map to the target vocabulary[9]. To reduce model training time by a factor of approximately $2 - 3\times$, and to reduce memory footprint,

[9]The token set of this vocabulary consists of the 26 English alphabet letters augmented with the apostrophe and a word boundary token.

Table 5: Hyperparameters summary for speech recognition task for supervised (left) and semi-supervised pseudo-labeling (right) training with a vanilla transformer. The $0.3 \rightarrow 0.1$ in the dropout and layer drop rates indicates that a rate of 0.3 is used during pre-training, and a rate of 0.1 is used during pseudo-labeling.

	Supervised	Pseudo-Labeling
Librispeech test-clean / test-other WER	7.8/19.1	4.8/11.5
Optimizer	Adam	Adam
Optimizer scaling rule	Adam	Adam
Base (β_1, β_2)	(0.995, 0.999)	(0.995, 0.999)
Base learning rate	0.0001	0.0001
Base learning rate warmup (steps)	64k	64k
Learning rate schedule	Fixed (no decay)	Fixed (no decay)
Learning rate minimum value	0	0
Base training duration (steps)	400k	500k
Base batch size (dynamic)	$8 \times 290s$	$8 \times 290s$
Base teacher momentum	0.99995	0.9999
Weight decay	None	None
Numerical precision	`bf16`	`bf16`
Augmentation stack	SpecAug	SpecAug
Dropout	0.3	$0.3 \rightarrow 0.1$
Layer drop	0.3	$0.3 \rightarrow 0.1$
Gradient clipping	1	1
Labeled:unlabeled data ratio	N/A	1:3
Base pre-training steps	N/A	20k
Base start of EMA accumulation (steps)	N/A	19k

we use CAPE positional embeddings (Likhomanenko et al., 2021b) instead of relative positional embeddings (Shaw et al., 2018): both models perform similarly.

Training Here we discuss our training procedure for base batch size $B = 8 \times 290s$, which is adapted from Likhomanenko et al. (2021a), and is summarized in Table 5. We use SpecAugment (Park et al., 2019) activated after 5k steps of training: two frequency masks with frequency mask parameter $F = 30$, ten time masks with maximum time-mask ratio $p = 0.1$ and time mask parameter $T = 50$ are used; time warping is not used.

One difference in setup is we use the Adam optimizer, whereas Likhomanenko et al. (2021a) used Adagrad (Duchi et al., 2010). Even though Adagrad can be viewed as a particular limit ($\beta_1 = 0$ and $\beta_2 \rightarrow 1$) of Adam (Kingma & Ba, 2015), we were unable to produce reasonable optimization in practice when applying the Adam Scaling Rule of Malladi et al. (2022) in this limit. As a consequence, we chose to work with the Adam optimizer, where its scaling rule has been shown to work (Malladi et al., 2022), and we take $\beta_1 = 0.995$, $\beta_2 = 0.999$, and $\epsilon = 10^{-8}$. We obtained similar results for $\beta_1 = 0.99$. Finally, we use a linear learning rate warmup (64k steps) after which the learning rate is kept constant until convergence. This performance can be improved further by using a step decay schedule as shown in prior work. We also apply gradient clipping of 1, and do not use weight decay.

Pseudo-Labeling The pseudo-labeling process comprises of two stages: i) The pre-training phase, where we train model on labeled data for 20k steps with model EMA accumulation starting after 19k steps; and ii) the pseudo-labeling phase, where we involve unlabeled data by generating pseudo-labels from the model EMA (teacher) and provide them to the model (student) as if they were ground-truth labels. Pseudo-labels are generated without any dropout applied to the teacher, and no data augmentation is applied for the corresponding inputs. To produce the pseudo-label, we use *hard transcription* (Definition G.1)

Definition G.1 (Hard Transcription)**.** *For a sequence of frames, select the most probable token per frame, removing repetitions* and *the CTC blank token. For example, "h##eelll##ll###oo" is transformed into "hello", where "#" is the CTC blank token.*

These hard transcriptions are then used as transcription for student optimization. We use a 1:3 proportion of labeled to unlabeled data as this was found to be optimal in Likhomanenko et al. (2021a), and we decrease model dropout and layer drop rates to 0.1 after pre-training phase. As we have access to the ground-truth labels on the data being treated as unlabeled, we can track

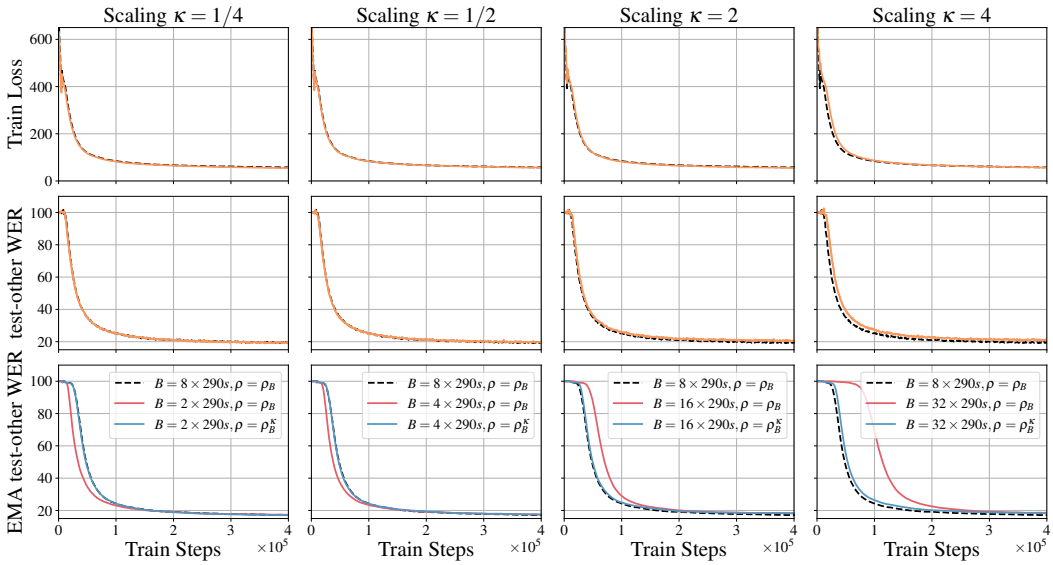


Figure 14: *Transformer Polyak-Ruppert averaging on LibriSpeech (trained on train-clean-100)* with different scalings κ. The baseline ($\kappa = 1$, black dashed) is trained with Adam and momentum $\rho_B = 0.99995$ at a *dynamic batch size* $B = 8 \times 290s$, which corresponds to a single train step on the x-axis. We investigate dynamic batch sizes down to $B = 2 \times 290s$ (left) and up to $B = 32 \times 290s$ (right), with (blue, $\rho = \rho_B^\kappa$), and without (red, $\rho = \rho_B$) the EMA Scaling Rule (model non-EMA is marked by orange). The Adam Scaling Rule (Malladi et al. (2022), Definition C.3) is used throughout. For momentum $\rho_B = 0.9999$ we observe similar trajectories behaviour for all models.

pseudo-label quality by computing pseudo-labels on this data, and compute the WER against their ground-truth. Pseudo-label quality is the primary metric to evaluate progress on unlabeled data, as loss on pseudo-labeled data is unreliable when a teacher model and pseudo-labels are evolving with each time step.

Scaling of batch size Sequential data is typically processed using dynamic batching as it is more computationally efficient than using a fixed number of sequences (Ott et al., 2019). In our work, we use dynamic batching of ~290s audio per GPU. Moreover, for CTC we do not apply any additional sequence normalization. We experimented with fixed batching, but did not observe any significant differences in conclusions compared with the dynamic batching.

We note that dynamic batching is a more challenging setting for achieving systematic scaling, as the number of independent sequences in any given batch may change, and the i.i.d. assumption does not hold at the frame level. Despite these violations of the assumptions of Section 2.2, our results demonstrate that the Adam Scaling Rule (Definition C.3, Malladi et al. (2022)) holds in the case of dynamic batches, as does our EMA Scaling Rule (Definition 1.1).

The base batch size is set to $B = 8 \times 290s$, and in our experiments we scale down to batch size of $B = 2 \times 290s$ and up to batch size of $B = 128 \times 290s$. The number of warmup and pre-training steps, steps before SpecAugment is turn on and model EMA is accumulated are scaled according to Appendix C.1.

Compute *[This section has been redacted to preserve anonymity during the peer-review process. If this work is accepted, the full details compute used for these experiments, including: the experiments presented, hyperparameter optimization, and the development process, will be provided.]*

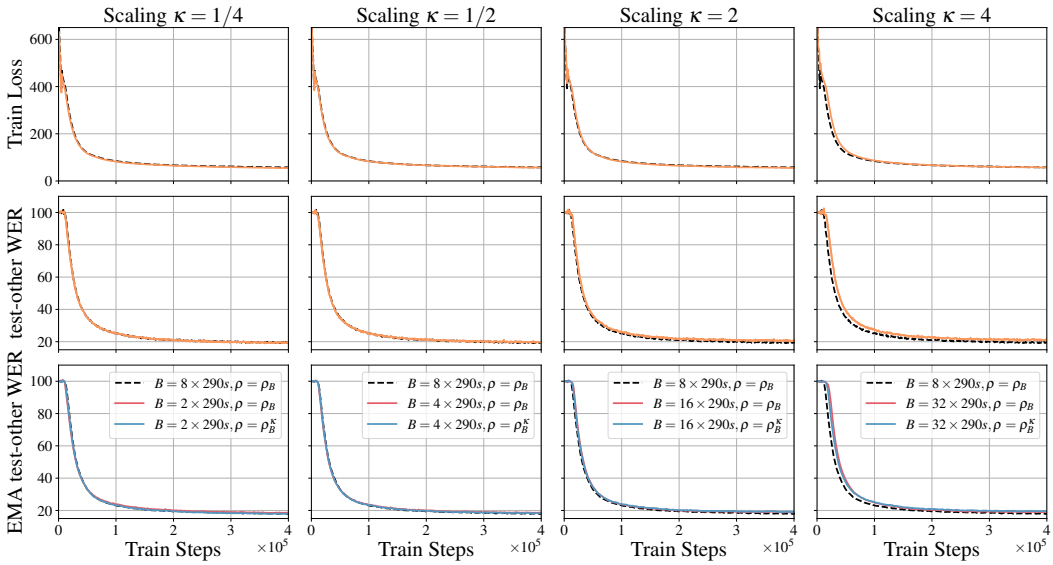


Figure 15: *Transformer Polyak-Ruppert averaging on LibriSpeech (trained on train-clean-100)* with different scalings κ. The baseline ($\kappa = 1$, black dashed) is trained with Adam and momentum $\rho_B = 0.999$ at a *dynamic batch size* $B = 8 \times 290s$, which corresponds to a single train step on the x-axis. We investigate dynamic batch sizes down to $B = 2 \times 290s$ (left) and up to $B = 32 \times 290s$ (right), with (blue, $\rho = \rho_B^\kappa$), and without (red, $\rho = \rho_B$) the EMA Scaling Rule (model non-EMA is marked by orange). The Adam Scaling Rule (Malladi et al. (2022), Definition C.3) is used throughout. If momentum ρ_B is small and accumulation history is short we observe no any significant difference between models which all are matching the reference trajectory despite scaling κ.

G.1 Detailed results

We present detailed comparison between models trained with and without EMA Scaling Rule in Figures 14 and 15 for supervised training and in Figures 16 and 17 for semi-supervised training.

First, we observe that if the Adam Scaling Rule does not hold perfectly[10] (there is a mismatch between trajectories for the model before pseudo-labels are involved) the EMA Scaling Rule also gives discrepancies with the reference trajectory, however they are negligible compared to models trained without EMA Scaling Rule. For the semi-supervised training, to alleviate the difficulties with a breakdown of the Adam Scaling Rule for large κ we postpone the pseudo-labeling process until the model reaches similar WER as the baseline. This allows us to align the initial model conditions for pseudo-labeling. In this scenario we are able to match the reference trajectory up to $\kappa = 8$.

We note that this result reveals that errors for the Adam Scaling Rule *and* the EMA Scaling Rule are contributing, although the way in which they contribute is different, and one can dominate the other. We observe in Figure 16 that if the initial conditions of the models are similar (attained by using the same WER as a condition to begin pseudo-labeling) then the error from the EMA Scaling Rule dominates over that of the Adam Scaling Rule, causing a divergence in training dynamics.

Second, we observe in practice that the EMA Scaling Rule holds for both fixed batching (a sequence length in the batch can vary significantly) and for dynamic batching (when total number of frames in the batch is fixed, though padding still is accounted to the this amount). This shows that EMA Scaling Rule is applicable to sequential data too.

Third, we observe in Figures 15 and 17 that for smaller values of ρ_B, scaling with or without EMA Scaling Rule behave similarly, and reference trajectories match in the supervised and semi-supervised cases. However, if the momentum is too large, the *teacher* moves slowly and is uninformative, whereas if the momentum is too low, the *teacher* and the *student* are effectively be the same model, implying: i) the student will self-predict with high confidence, removing any benefits of dis-

[10] See Malladi et al. (2022) for a discussion on scenarios that lead to a breakdown of the Adam Scaling Rule.

tillation[11]; and ii) training instability or model divergence will happen in the low-resource settings (Likhomanenko et al., 2021a; Higuchi et al., 2022).

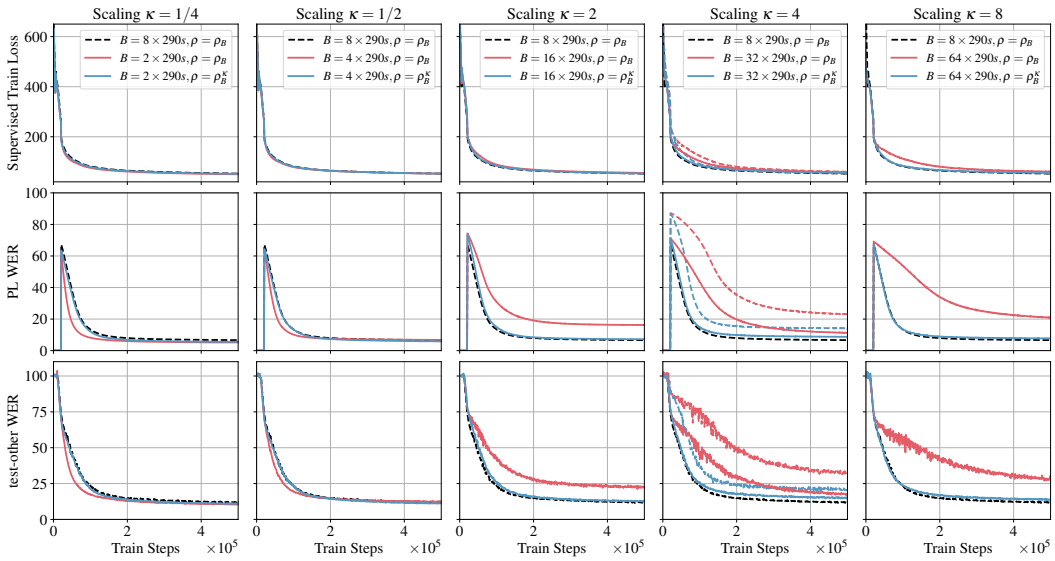


Figure 16: *Transformer pseudo-labeling on LibriSpeech (trained on train-clean-100 as labeled and the rest of LibriSpeech as unlabeled)* with different scalings κ. The baseline ($\kappa = 1$, black dashed) is trained with Adam at a *dynamic batch size* of 8×290 seconds, which corresponds to a single train step on the x-axis. The model EMA (*teacher*) is updated with momentum $\rho_B = 0.9999$. We investigate dynamic batch sizes down to $B = 2 \times 290s$ (left) and up to $B = 64 \times 290s$ (right), with (blue, $\rho = \rho_B^\kappa$) and without (red, $\rho = \rho_B$) the EMA Scaling Rule. The Adam Scaling Rule (Malladi et al. (2022), Definition C.3) is used throughout. For $\kappa \leq 2$, we start pseudo-labeling after $20\text{k}/\kappa$ training steps; while for $\kappa > 2$, we start when pre-training WER matches the baseline WER ($24\text{k}/\kappa$ for $\kappa = 4$ and $29\text{k}/\kappa$ for $\kappa = 8$). For $\kappa = 4$ we experimented with both variants: we start pseudo-labeling after $20\text{k}/\kappa$ (dashed) and when pre-training WER matches the baseline WER (solid, $24\text{k}/\kappa$).

[11]He et al. (2020) alleviated the problem with the proper amount of noise during *student* model training, whilst Xu et al. (2020) used beam-search decoding with a language model.

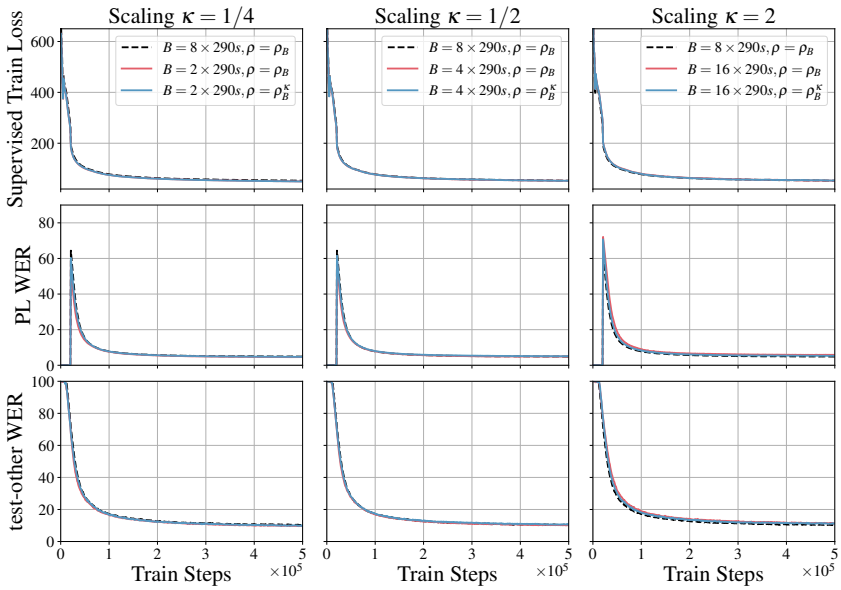


Figure 17: *Transformer pseudo-labeling on LibriSpeech (trained on train-clean-100 as labeled and the rest of LibriSpeech as unlabeled)* with different scalings κ. The baseline ($\kappa = 1$, black dashed) is trained with Adam at a *dynamic batch size* of 8×290 seconds, which corresponds to a single train step on the x-axis. The model EMA (*teacher*) is updated with momentum $\rho_B = 0.999$. We investigate dynamic batch sizes down to $B = 2 \times 290s$ (left) and up to $B = 16 \times 290s$ (right), with (blue, $\rho = \rho_B^\kappa$) and without (red, $\rho = \rho_B$) the EMA Scaling Rule. The Adam Scaling Rule is used throughout. In case of short history accumulation for the momentum (compared to Figure 14) we observe similar to supervised training (Figure 15) no significant different between all models trajectories throughout the training while matching the reference one.

G.2 Scaling to $\kappa = 16$ with Progressive Scaling

Finally, we aim to scale semi-supervised pseudo-labeling further to $\kappa = 16$. In this case we observe that Adam Scaling Rule does not hold in the pre-training phase and there is no model convergence. To overcome this, we apply Progressive Scaling (Definition 3.2). We pre-train models on supervised data with $\kappa = 8$ for 29k of reference steps (model EMA accumulation starts at 28k steps). We then scale to $\kappa = 16$ and begin pseudo-labeling. We see in Figure 18 that Progressive Scaling enables us to scale pseudo-labeling to $\kappa = 16$ with (middle) and without (left) the EMA Scaling Rule. Second, models *with* the EMA Scaling Rule track the baseline much closer than models without the EMA Scaling Rule, although a small gap is present. We further experimented with Progressive Scaling, postponed the transition condition to the $\kappa = 16$ until 75k reference steps. In Figure 18 (right), we see this scaled model tracks the reference trajectory, and so using a combination of the EMA Scaling Rule and Progressive Scaling, we are able to scale pseudo-labeling to $\kappa = 16$, corresponding to a dynamic batch size of $128 \times 290s$.

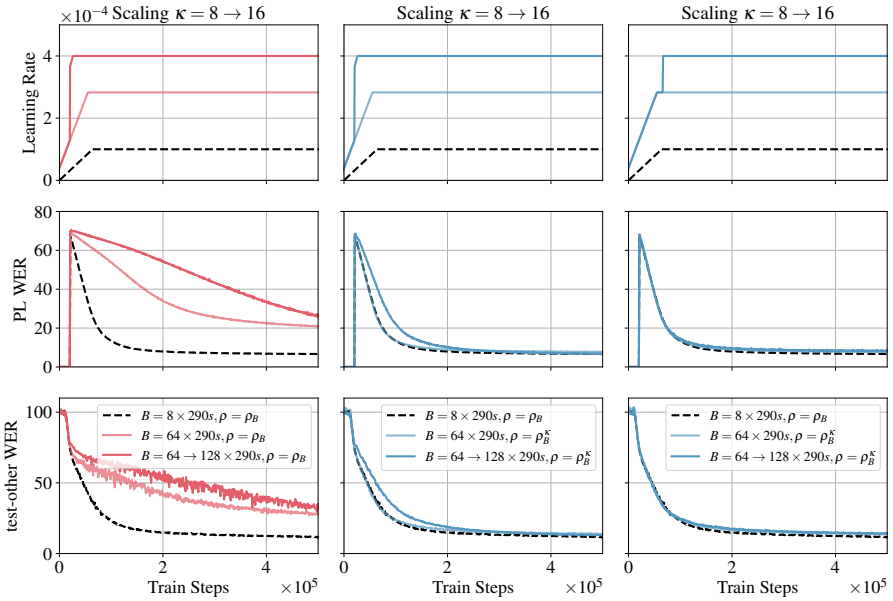


Figure 18: *Transformer pseudo-labeling on LibriSpeech (trained on train-clean-100 as labeled and the rest of LibriSpeech as unlabeled)* with different Progressive Scaling from $\kappa = 8$ to $\kappa = 16$ ($\kappa = 8 \rightarrow 16$). The baseline ($\kappa = 1$, black dashed) is trained with Adam at a *dynamic batch size* of 8×290 seconds, which corresponds to a single train step on the x-axis. The model EMA (*teacher*) is updated with momentum $\rho_B = 0.9999$. The scaling with $\kappa = 8$ is shown with lighter color for reference from Figure 16. We investigate dynamic batch sizes progressively from $B = 64 \times 290s$ to $B = 128 \times 290s$, with (blue, $\rho = \rho_B^\kappa$) and without (red, $\rho = \rho_B$) the EMA Scaling Rule. For reference (top) we show the learning rate schedule with Progressive Scaling. The Adam Scaling Rule (Malladi et al. (2022), Definition C.3) is used throughout. Left and middle correspon to Progressive Scaling with scale from $\kappa = 8$ to $\kappa = 16$ at 29k steps, while right corresponds to 75k steps.

H Additional details and results for self-supervised image representation learning

Organization This appendix is structured into three sections. We first give an overview of our chosen SSL method BYOL (Appendix H.1), our recipe for training BYOL using Vision Transformers (ViTs) (Appendix H.2), ablations of normalization approaches that lead to the development of this recipe (Appendix H.3), and additional results corresponding to longer training duration (Appendix H.4) and further understanding the impact of Progressive Scaling (Appendix H.5).

Second, we demonstrate that the EMA Scaling Rule combined with Progressive Scaling can scale a ResNet-50 BYOL model trained with LARS to batch size 32,768 without performance drop, demonstrating the empirical utility of the tools we provide outside of their theoretical validity (Appendix H.6).

Finally, we show that it is possible to systematically scale DINO (Caron et al., 2021) using a combination of Progressive Scaling and the EMA Scaling Rule, providing a solution for researchers and practitioners wanting to train DINO at scale.

H.1 Components of self-supervised learning

First, a key component of many SSL methods is the *stop-gradient* or StopGrad (Definition H.1).

Definition H.1 (Stop Gradient/StopGrad($\cdot$))**.** *The* stop-gradient *operator StopGrad*($\cdot$) *prevents the flow of gradient information*

$$\frac{df(StopGrad(h(x;\omega));\theta)}{d\omega} \equiv 0 \tag{72}$$

for all parameteric functions h and f and for all parameters θ and ω.

Applying a *stop-gradient* is sometimes called *detaching* in the literature. Now, we introduce the update rule of our representative SSL method BYOL in Definition H.2.

Definition H.2 (BYOL Update)**.** *BYOL learns unsupervised features by minimizing the cosine distance between the predictions of a student backbone $f(\cdot;\theta)$ (typically a ResNet or Vision Transformer), projected through $h(\cdot;\omega)$ (typically a Multi-Layer Perceptron (MLP)), and the predictions of an EMA teacher $f(\cdot;\zeta)$ (Grill et al., 2020). The update for the parameters of BYOL is then*

$$(\theta_{t+1}, \omega_{t+1}) = (\theta_t, \omega_t) - \eta \times \frac{1}{B} \sum_{x \in \mathbb{B}} \nabla_{(\theta,\omega)} \mathcal{L}(x; \theta_t, \omega_t, \zeta_t) \tag{73}$$

$$\zeta_{t+1} = \rho\, \zeta_t + (1-\rho)\, \theta_{t+1} \tag{74}$$

$$\text{with} \quad \mathcal{L}(x; \theta_t, \omega_t, \zeta_t) = \frac{1}{2} \cos\left[h(f(x_1; \theta_t); \omega_t), \mathit{StopGrad}(f(x_2; \zeta_t))\right] + (x_1 \leftrightarrow x_2), \tag{75}$$

where $\cos(\boldsymbol{a}, \boldsymbol{b}) \equiv 1 - \boldsymbol{a} \cdot \boldsymbol{b} / (||\boldsymbol{a}||\,||\boldsymbol{b}||)$ *is the cosine distance, and x_1 and x_2 are two views of a single variate x, often produced by augmentations, and $x_1 \leftrightarrow x_2$ denotes symmetrization over x_1 and x_2.*

As noted in Section 3.4,the BYOL EMA update (Equation (74)) uses θ_{t+1} instead of our analyzed θ_t (Equation (4)). The effect upon the overall EMA update is $O(\eta \times \beta_\rho)$ and so is captured by the EMA Scaling Rule (Definition 1.1).

One more piece of technology typically employed in SSL is a *tracking probe* (Definition H.3) which we will use to evaluate the performance of BYOL on downstream tasks of interest, for example, image classification.

Definition H.3 (Tracking Probe/Linear Probe)**.** *When optimizing model parameters ω_t of an SSL method, simultaneously optimize the parameters ξ of a probe model $r(\cdot;\xi)$ under a downstream objective $\mathcal{L}^{(d)}$. For example, in classification, with data x and samples y*

$$\mathcal{L}^{(d)}(x, y, \theta_t, \xi_t) = -\log P(y | r(\mathit{StopGrad}(h(x; \omega_t)); \xi)) \tag{76}$$

$$\mathcal{L}^{(total)}(x, y; \theta_t, \omega_t, \zeta_t, \xi_t) = \mathcal{L}(x; \theta_t, \omega_t, \zeta_t) + \mathcal{L}^{(d)}(x, y, \omega_t, \xi_t), \tag{77}$$

The is a probe for the teacher, which is typically the better choice due to Polyak-Ruppert averaging effects (see Section 3.2). When the r is a linear model, the tracking probe is called a linear probe.

It is also typical to use a Batch Normalization layer *without* trainable affine terms before this linear layer as in He et al. (2022) to stabilize probe training. In this case, the running statistics can be absorbed into a definition of the linear layer weights and biases, and so this is still a *linear probe*, although we will call this a *pre-bn linear probe* to remove ambiguity.

H.2 A Vision Transformer recipe for BYOL

Hyperparameters We present the base hyperparameters for training BYOL with a ViT-B/16 backbone in Table 6. This recipe was developed by starting from a well-known supervised ViT-B/16 recipe (He et al., 2022) and performing a search over weight decay and learning rate hyperparameter choices. We find that BYOL performs well with heavy weight decay ($\lambda = 0.3$) and a low learning rate ($\eta = 10^{-3}$) at a base batch size $B = 4096$. The AdamW optimizer is used, and so for scaling to other batch sizes $\hat{B} = \kappa B$ we use the Adam Scaling Rule (Definition C.3)[12] We use a pre-bn linear probe as discussed in Appendix H.1. Finally, the performance of BYOL can be further improved by employing multicrop (Caron et al., 2020) by $\approx +2\%$ in absolute test top-1 performance on ImageNet1k compared to without multicrop, however, as this is not our focus, we omit this from the presented recipe.

Compute *[This section has been redacted to preserve anonymity during the peer-review process. If this work is accepted, the full details compute used for these experiments, including: the experiments presented, hyperparameter optimization, and the development process, will be provided.]*

Additional background Achieving large scale SSL training with ViTs to large scale SSL training has been a long standing goal in the community. MoCo-v3 (Chen et al., 2021) enables the use of ViTs with contrastive learning, but achieves this through modificatinos of the ViT training

[12] We note that Adam (Kingma & Ba, 2015) and AdamW (Loshchilov & Hutter, 2019) are equivalent in the limit of zero weight decay, and that the Adam Scaling Rule (Definition C.3) was derived with zero weight decay (Malladi et al., 2022).

Table 6: BYOL ViT-B/16 hyperparameters.

	BYOL ViT-B/16
ImageNet1k Linear Probe Test Top-1	74.47% (Figure 19)
Weight initialization	`trunc_normal(.02)`
Backbone normalization	LayerNorm
Head normalization	BatchNorm
Synchronized BatchNorm over replicas	No
Learning rate schedule	Single Cycle Cosine
Learning rate warmup (epochs)	40
Learning rate minimum value	1×10^{-6}
Training duration (epochs)	480
Optimizer	AdamW
Optimizer scaling rule	Adam
Base (β_1, β_2)	(0.9, 0.95)
Base learning rate	1×10^{-3}
Base batch size	4096
Base teacher momentum	0.99
Weight decay	0.3
Weight decay scaling rule	None
Weight decay skip bias	Yes
Numerical precision	`bf16`
Augmentation stack	BYOL
Stochastic depth	0.1

procedures, including gradient freezing on the image patching layer, and re-introducing Batch Normalization to post-attention MLP layers. Despite these modifications, MoCo-v3 was only trained up to a batch size of 6144, where model performance begins to suffer (Chen et al., 2021). In Figure 6 we demonstrate that combining dynamic batch scaling (Appendix C.4) with the EMA Scaling Rule (Definition 1.1) enables BYOL to be trained using ViTs to batch sizes of 24,576 without any drop in performance compared to the reference batch size of 4096. We emphasize that the piecewise transitions in the schedules are important for preserving training dynamics.

H.3 The role of Batch Normalization and Layer Normalization in BYOL with ViTs

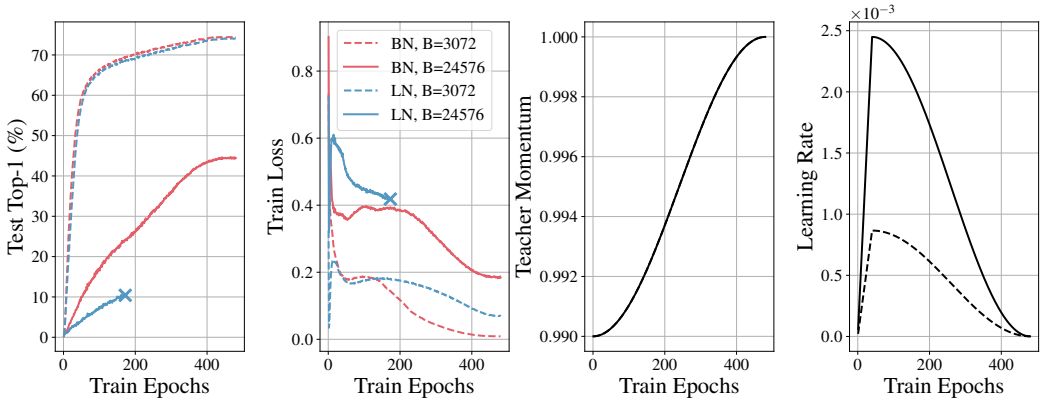


Figure 19: *BYOL ViT-B/16 on ImageNet1k* for different scalings κ. We present runs comparing LayerNorm (blue) to BatchNorm (red) in the projection and prediction heads of BYOL ViT models for batch size 3072 (dashed) and 24,576 (solid) *without the EMA Scaling Rule*. $\kappa = 1$ corresponds to $B = 4096$. In all scenarios the transformer backbone *only* uses LayerNorm. We truncate the training of the large batch size LayerNorm variant to preserve compute (indicated by $\times$).

Here we compare the roles of Batch Normalization (BatchNorm, Ioffe & Szegedy (2015)) and Layer Normalization (LayerNorm, Ba et al. (2016)) in the projection and prediction heads of BYOL (Grill et al., 2020) using ViTs.

It has been observed that BatchNorm plays a critical role in BYOL predictor and projector dynamics (Fetterman & Albrecht, 2020), and using either LayerNorm or *no normalization* significantly decrease in model performance. Subsequently, it was demonstrated (Richemond et al., 2020) that competitive BYOL performance could be achieved through a combination of Group Normalization (GroupNorm, Wu & He (2018)) and Weight Standardization (Qiao et al., 2019). Additionally, Richemond et al. (2020) showed that if BatchNorm is used in the backbone, one can use LayerNorm or *no normalization* in the predictor and projector without any performance drop.

In this work, we we show it is possible to train BYOL ViT using *only LayerNorm* across the backbone, projector and predictor (see Figure 19), decoupling BYOL's reliance on batch statistics, a desirable trait for a representation learning algorithm (Brock et al., 2021). At batch size 3072, using LayerNorm in the predictor and projector achieves competitive performance (74.10%), performing slightly worse than using BatchNorm (74.47%). At the larger batch size of 24,576, runs perform significantly worse as the EMA Scaling Rule was not applied.

H.4 Longer training duration with incremental Progressive Scaling

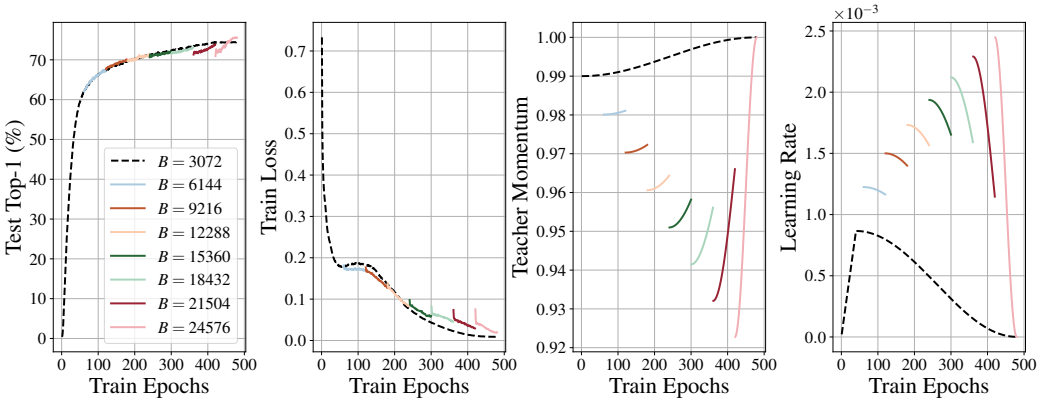


Figure 20: *BYOL ViT-B/16 on ImageNet1k* for different scalings κ. The baseline model ($\kappa = 0.75$, black dashed) uses batch size 3072 and teacher momentum $\rho_B = 0.99$. We increment the batch size by 3072 every 60 epochs to a final batch size of 24,576 using Progressive Scaling (Definition 3.2).

Here we use the same base hyperparameters as Table 6, except that we train for 480 instead of 300 epochs. To mitigate the student impulse phenomena discussed in Section 3.4, in Figure 20 we investigate increasing the batch size every 60 epochs using Progressive Scaling (Definition 3.2). We observe that this more gradual procedure enables closer tracking of the baseline train loss trajectory. Additionally, this procedure results in a scaled linear probe performance that outperforms the baseline (75.64% compared to the baseline performance of 74.47%). The same procedure can be applied to the LayerNorm variant discussed in Appendix H.3, which produces a similar result (75.09% compared to the baseline performance of 74.10%).

H.5 Building intuition around Progressive Scaling and momentum sensitivity

Our final BYOL ViT results are to help build intuition around Progressive Scaling (Definition 3.2), as well as when the EMA Scaling Rule is most important. In Figure 21 we explore transitioning from the baseline batch size 4096 model to batch size 24,576 in a *single transition* after 60 epochs. After this transition, we continue training for 240 epochs for a range of momenta: $\rho \in \{0.8, 0.9, 0.95, 0.97, 0.9867, 0.994, 0.999\}$ *without* the EMA Scaling Rule.

We observe that after the transition, any $0.9 \leq \rho \leq 0.994$ produces a linear probe performance that matches or outperforms the baseline at the end of training. This indicates that after the initial training period, BYOL becomes less sensitive to the choice of teacher momentum. Note that without the initial 60 epochs of training with batch size 4096, *all models*, including those employing the EMA Scaling Rule diverge (see $B = 24,576$ in Figure 6).

We present an illustration for why this might happen in Figure 22. First, we see that using the EMA Scaling Rule *always* keeps the model within the acceptable momentum region. We also wee that

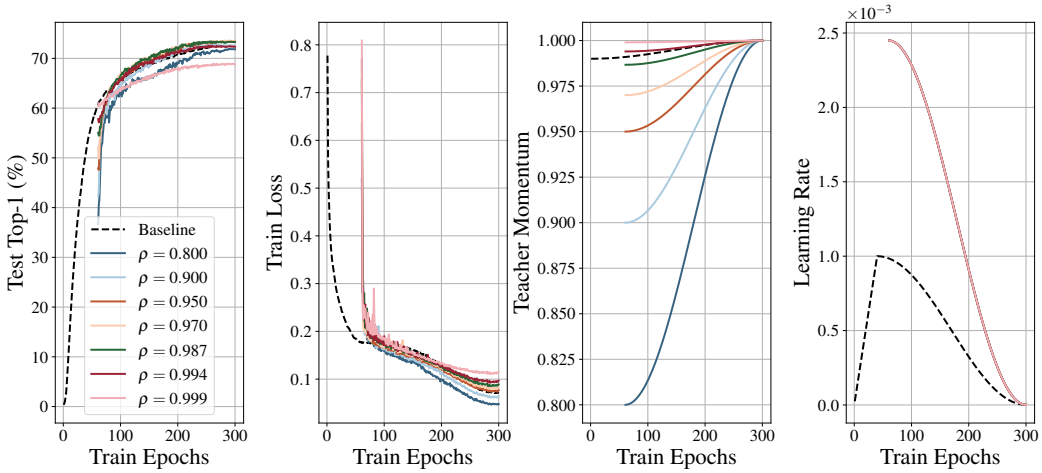


Figure 21: *BYOL ViT-B/16 on ImageNet1k* for different momenta ρ. The baseline model ($\rho = 0.99$, black dashed) uses batch size 4096. At the 60th epoch we apply Progressive Scaling (Definition 3.2) and transition to batch size 24576. We train for a further 240 epochs without EMA scaling for a range of momenta: $\rho \in \{0.9, 0.95, 0.97, 0.9867, 0.994\}$.

not using the EMA Scaling Rule can keep the model within the acceptable momentum region for a range of batch sizes, depending on how large wide in momenta the acceptable region is at the base batch size. Finally, we see that the momentum value matters much more at low values of momenta (the acceptable momentum region shrinks), whereas at large momenta, this region of acceptability widens.

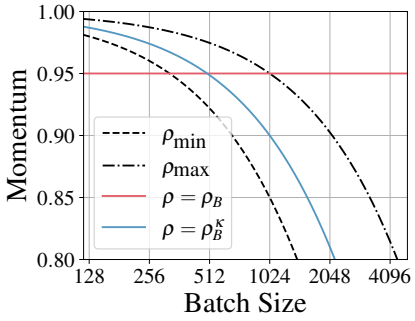


Figure 22: *A hypothetical scenario where there is an upper and lower limit for momenta qualitatively leading to the same result.*. We assume at base batch size $B = 1024$ there is an upper ($\rho_{\max}$, black dashdot) and lower ($\rho_{\min}$, black dashed) limit for *valid momenta*. We show what happens if we start with $\rho_B = 0.95$ at a batch size of 4096, and scale with ($\rho = \rho_B^\kappa$, blue) and without ($\rho = \rho_B$, red) the EMA Scaling Rule.

H.6 Scaling a ResNet-50 BYOL using LARS and Progressive Scaling

Here we investigate whether Progressive Scaling and the EMA Scaling Rule can be used in practice where there is no known optimizer SDE approximation. We use the default 300 epoch configuration for BYOL (Grill et al., 2020) in Figure 23. We see that although trajectories during training do not match, we are able to match or surpass the linear probe performance of the BYOL baseline at the larger batch size if 32,768. *This indicates that the contributions of our work have practical utility beyond the theoretical constraints.*

Compute *[This section has been redacted to preserve anonymity during the peer-review process. If this work is accepted, the full details compute used for these experiments, including: the experiments presented, hyperparameter optimization, and the development process, will be provided.]*

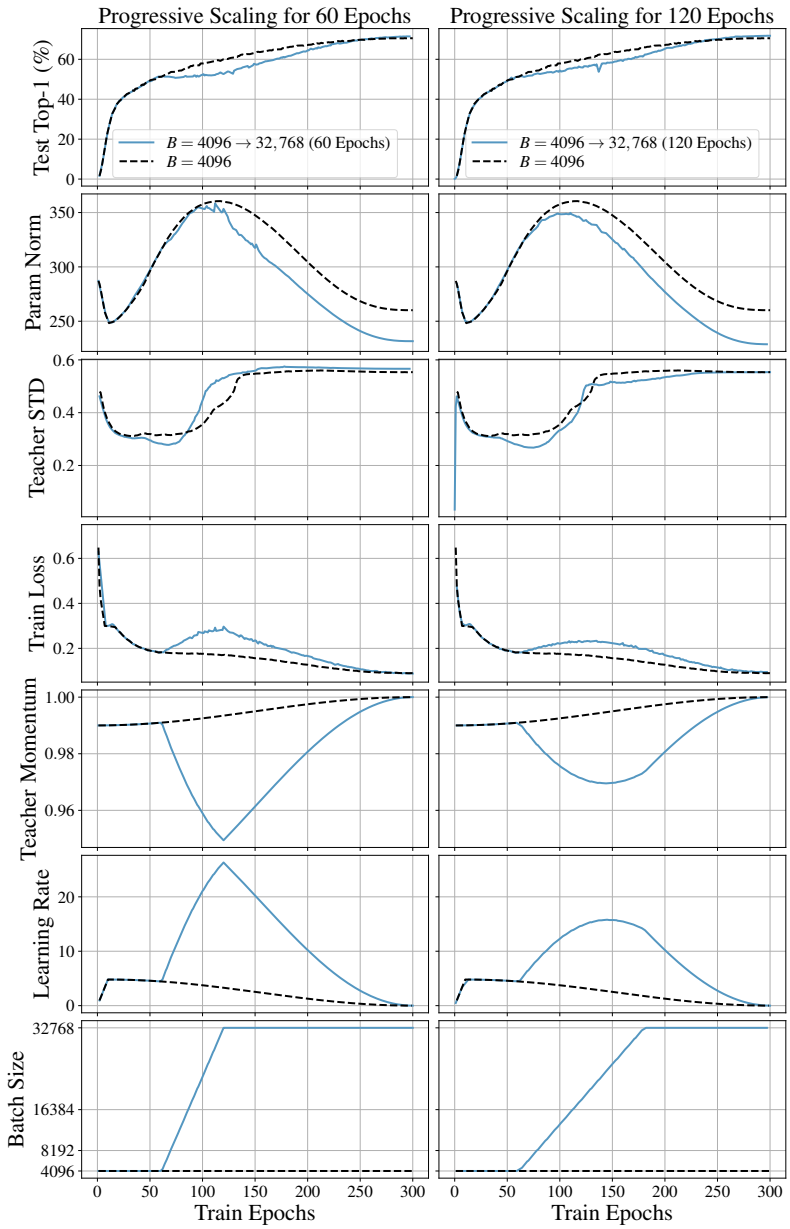


Figure 23: *ResNet50 BYOL on ImageNet1k using LARS* for different configurations of progressive scaling. The baseline (black dashed) uses batch size 4096 and momentum $\rho_B = 0.99$. We consider progressive scaling (blue) smoothly from epoch 60 for 60 epochs (left) and 120 epochs (right) up until batch size 32,768, scaling the learning rate linearly, and applying the EMA Scaling Rule.

H.7 Preventing collapse phenomena in DINO at scale

Until now, our representatives SSL method has been BYOL for reasons discussed in Section 3.4. Here, we will turn our attention to DIstillation with NO labels (DINO) (Caron et al., 2021), which has the update rule presented in Definition H.4.

Definition H.4 (DINO Update)**.** *DINO learns unsupervised features by matching predictions over emergent pseudo-labels of a student backbone and head $f(\cdot\,;\theta)$ to those of an EMA teacher $f(\cdot\,;\zeta)$ through a cross-entropy guided distillation procedure. DINO has a additional centering procedure, which is a form of batch normalization with momentum $\rho_c = 0.9$ which we do not scale using the*

Table 7: DINO ViT-B/16 Training hyperparameters.

	DINO ViT-B/16
CIFAR10 Linear Probe Top-1 ($\rho_B = 0.996$)	85.38%
CIFAR10 Linear Probe Top-1 ($\rho_B = 0.992$)	86.96%
Weight initialization	`trunc_normal(.02)`
Normalization	Layer Norm
Learning rate schedule	Single Cycle Cosine
Learning rate warmup (epochs)	50
Learning rate minimum value	1×10^{-6}
Training duration (epochs)	280
Optimizer	AdamW
Optimizer scaling rule	Adam
Base (β_1, β_2)	(0.9, 0.95)
Base learning rate	3×10^{-4}
Base batch size (B)	1024
Base teacher momentum (ρ_B)	0.992 or 0.996
Base weight decay	0.04
Weight decay scaling rule	Linear
Weight decay skip bias	Yes
Center Momentum	0.9
Center Momentum Scaling Rule	None
Precision	`bf16`
Augmentation stack	DINO Multi-crop

EMA Scaling Rule. The update for the parameters of DINO is

$$\theta_{t+1} = \theta_t - \eta \times \frac{1}{B} \sum_{x \in \mathbb{B}} \nabla_\theta \mathcal{L}(x; \theta_t, \zeta_t, \mathbf{c}_t) \tag{78}$$

$$\zeta_{t+1} = \rho\, \zeta_t + (1 - \rho)\, \theta_{t+1} \tag{79}$$

$$\mathbf{c}_{t+1} = \rho_c\, \mathbf{c}_t + (1 - \rho_c)\, \mathbb{E}_{x'} \zeta(x') \tag{80}$$

$$\textit{with} \;\; \mathcal{L}(x; \theta_t, \zeta_t, \mathbf{c}_t) = H\big(f(x_1, \theta_t), f(x_2, \zeta_t) - \mathbf{c}_t\big) + (x_1 \leftrightarrow x_2), \tag{81}$$

where $H(\boldsymbol{a}, \boldsymbol{b}) \equiv -\sum_{m=1}^{M} p_m(\boldsymbol{a}) \log p_m(\boldsymbol{b})$ is the cross-entropy between categorical distributions over M (emergent pseudo-)classes given logits $\boldsymbol{a}, \boldsymbol{b} \in \mathbb{R}^M$, x_1 and x_2 are two views of a single variate x, often produced by augmentations, and $x_1 \leftrightarrow x_2$ denotes symmetrization over x_1 and x_2.

In practice, DINO employs multi-crop (Caron et al., 2021). We omit this detail for clarity of presentation, although we *do* use multi-crop in the experiments that follow.

Our interest DINO is due to the difficulty in its optimization[13], and in particular, preventing collapse phenomena in DINO at batch sizes above 1024, which is an open research problem. In this section, we will show that a combination of the EMA Scaling Rule (Definition 1.1) and Progressive Scaling (Definition 3.2) enable training of DINO beyond batch size 1024 without sacrificing performance.

Hyperparameters Base hyperparameters are presented in Table 7.

Compute *[This section has been redacted to preserve anonymity during the peer-review process. If this work is accepted, the full details compute used for these experiments, including: the experiments presented, hyperparameter optimization, and the development process, will be provided.]*

Results In Figures 24 and 25 we show the results obtained training DINO on CIFAR-10 with $\rho_B = 0.996$ and $\rho_B = 0.992$ respectively at the reference batch size of 1024. We employ smooth Progressive Scaling (Definition 3.2) between epochs 120 and 180.

At batch size 2048, the training loss matches the reference *only* when the EMA Scaling Rule is applied, whereas the run *without* the scaling rule diverges from the reference. The impact of this

[13]For an example, see `https://github.com/facebookresearch/dino/issues/43#issuecomment-881453515`.

divergence is emphasized as we consider the larger batch size of 4096. Here. there is also a gap *with* the EMA Scaling Rule, however is approximately three times smaller than the gap *without* the EMA Scaling Rule.

Additionally, we observe that using $\rho_B = 0.992$ yields higher Top-1 accuracy over $\rho_B = 0.996$, and in our experiments, using the EMA Scaling Rule *always* performs better in terms of linear probe performance than not using the scaling rule.

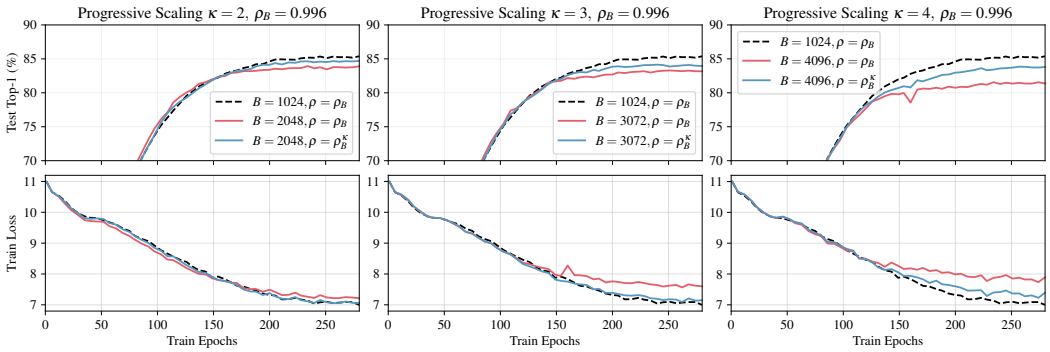


Figure 24: *DINO ViT-B/16 on CIFAR-10* for different scalings κ and base teacher momentum $\rho_B = 0.996$. The baseline model ($\kappa = 1$, black dashed) uses batch size 1024 and center momentum $\rho_c = 0.9$, and is scaled up from batch size 2048 (left) to 4096 (right) with (blue, $\rho = \rho_B^\kappa$) and without (red, $\rho = \rho_B$) the EMA Scaling Rule. Between epochs 100 and 180 we scale the batch size using progressive scaling (Definition 3.2).

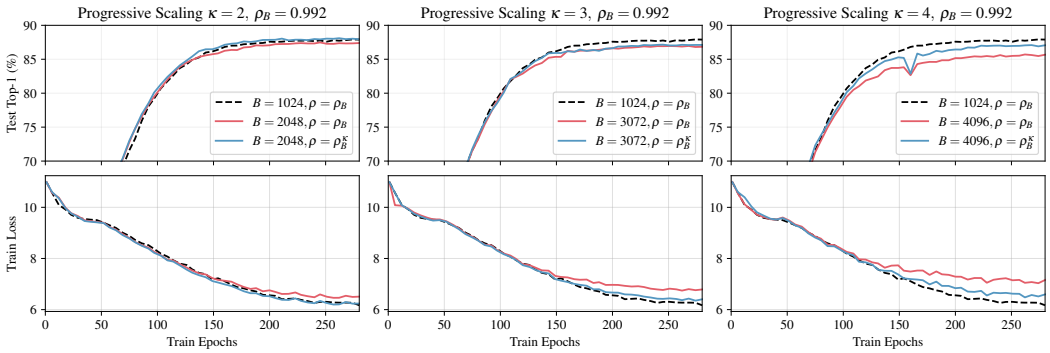


Figure 25: *DINO ViT-B/16 on CIFAR-10* for different scalings κ and base teacher momentum $\rho_B = 0.992$. The baseline model ($\kappa = 1$, black dashed) uses batch size 1024 and center momentum $\rho_c = 0.9$, and is scaled up from batch size 2048 (left) to 4096 (right) with (blue, $\rho = \rho_B^\kappa$) and without (red, $\rho = \rho_B$) the EMA Scaling Rule. Between epochs 100 and 180 we scale the batch size using progressive scaling (Definition 3.2).

In Figure 26 we show how the hyperparameters ρ, B and learning rate change with the progressive scaling in Definition 3.2.

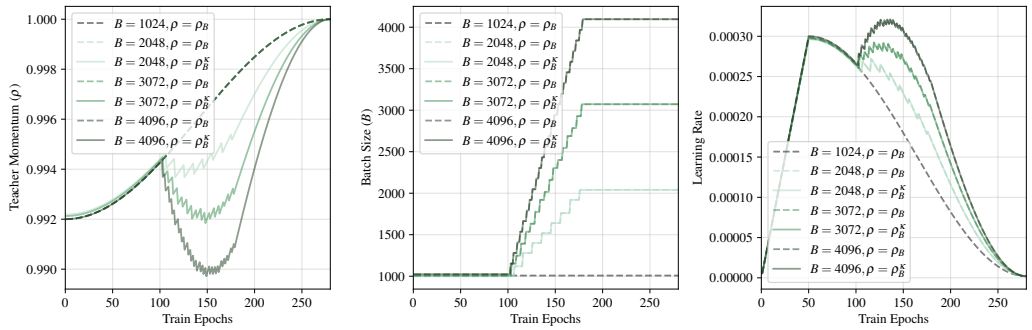


Figure 26: *DINO ViT-B/16 on CIFAR-10* for different scalings κ and base teacher momentum $\rho_B = 0.992$. We show how the hyperparameters ρ, B and learning rate change with the Progressive Scaling in Definition 3.2. These hyperparameters correspond to the training runs in Figure 25. Those for Figure 24 are identical, with the exception of ρ that starts at 0.996 instead of 0.992.

We also attempted to use a sharp batch size transition (Figures 27 and 28), which leads to the collapse pheonomena observed in prior work. This collapse happens with and without the EMA Scaling Rule. We suspect this is due to dynamics specific to DINO's early phase that are even more challenging to replicate under discretization than those of BYOL.

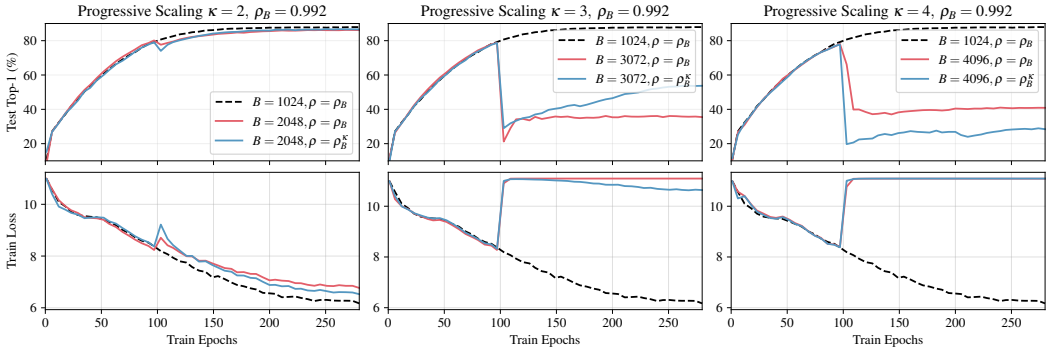


Figure 27: *DINO ViT-B/16 on CIFAR-10* for different scalings κ and base teacher momentum $\rho_B = 0.992$. The baseline model ($\kappa = 1$, black dashed) uses batch size 1024 and center momentum $\rho_c = 0.9$, and is scaled up from batch size 2048 (left) to 4096 (right) with (blue, $\rho = \rho_B^\kappa$) and without (red, $\rho = \rho_B$) the EMA Scaling Rule. Progressive Scaling is employed with a sharp transition at epoch 100, leading to a collapse phenomenon.

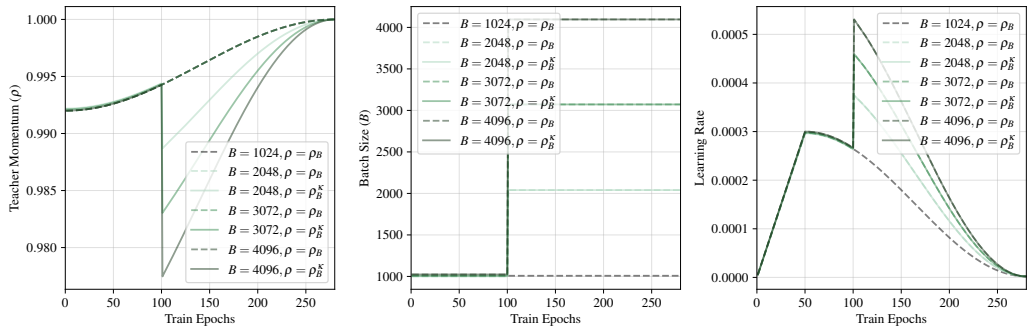


Figure 28: *DINO ViT-B/16 on CIFAR-10* with $\rho_B = 0.992$ and a sharp transition in batch size at epoch 100. We show how the hyperparameters ρ, B and learning rate change with sudden scaling. These hyperparameters correspond to the training runs in Figure 27.

Our results in this section show it is possible to scale DINO to large batch sizes *without* sacrificing performance by using *both* the EMA Scaling Rule and Progressive Scaling, providing the batch size schedule of Progressive Scaling is not sudden. This resolves an open problem in SSL research.