A CONVERGENCE PROOFS

To support our larger point about the irrelevance of this type of result when comparing LAMB and Adam, below we derive a convergence bound for Adam in a similar manner to the LAMB bound in You et al. (2019). Note that all of these bounds are loose upper bounds on the worst case behavior of the algorithms, so there is no reason that comparing them reflects the relative behaviors of optimizers in reality. For example in Equation 3 below, we follow similar operations as the LAMB bound derivation and simply switch a $-$ to a $+$ for algebraic convenience.

We define the following as our optimization objective

$$\min_{x \in \mathbb{R}^d} f(x) := \mathbb{E}_{s \in \mathbb{P}}[\ell(x, s)] + \frac{\lambda}{2}\|x\|^2, \tag{1}$$

with an optimal solution(s) x^*. $x \in \mathbb{R}^d$ are the neural network parameters, ℓ a smooth and possibly nonconvex loss function, $\mathbb{P}$ a data distribution, and λ the regularization strength.

Let T be the number of training steps, h the number of neural network layers, b the batch size, η the learning rate, n the mini-batch size, and $\phi(v) : \mathbb{R}^+ \to \mathbb{R}^+$ a function that is layerwise multiplied by the learning rate in LARS and LAMB updates. Let L be a vector of the layerwise Lipschitz constants for the neural network, and L_{avg} the mean of L. Let s be a training step uniformly sampled from $\{1, 2, ..., T\}$.

We define the stochastic minibatch estimate of the true gradient as $\mathbb{E}[g^{(i)}] = \nabla_i f(x)$ and assume that its variance is bounded by $\mathbb{E}\left[g^{(i)} - \nabla_i f(x)\right]^2 \leq \sigma_i^2$ layerwise for a vector of standard deviations $\sigma := \left[\sigma^{(1)}, \dots \sigma^{(h)}\right]$ and elementwise for $\tilde{\sigma} := \left[\tilde{\sigma}^{(1)}, \dots \tilde{\sigma}^{(h)}\right]$.

The LARS and LAMB update rules are defined as

Algorithm 1 LARS

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1)(g_t + \lambda x_t)$$
$$x_{t+1} = x_t - \eta \frac{m_t}{||m_t||}$$

Algorithm 2 LAMB

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1)\, g_t$$
$$v_t = \beta_2 v_{t-1} + (1 - \beta_2)\, g_t^2$$
$$m_t = \frac{m_t}{(1 - \beta_1^t)}$$
$$v_t = \frac{v_t}{(1 - \beta_2^t)}$$
$$r_t = \frac{m_t}{\sqrt{v_t} + \epsilon} + \lambda x_t$$
$$x_{t+1} = x_t - \eta \frac{r_t}{||r_t||}$$

where m_0, v_0 are initialized to zeros.

Next, let $\eta_t = \eta = \sqrt{\frac{2(f(x_1) - f(x^*))}{\alpha_u^2 \|L\|_1 T}}$ $\forall t \in [T]$, $b = T$, $\alpha_l \leq \phi(v) \leq \alpha_u$ $\forall v > 0$, $\alpha_l, \alpha_u > 0$. Crucially, additionally let $b = T, \beta_1 = 0, \lambda = 0$. Under these conditions You et al. (2019) show the convergence rate for LARS is

$$\left(\mathbb{E}\left[\frac{1}{\sqrt{h}} \sum_{i=1}^{h} \|\nabla_i f(x_s)\|\right]\right)^2 \leq \mathcal{O}\left(\frac{(f(x_1) - f(x^*))L_{avg}}{T} + \frac{\|\sigma\|_1^2}{Th}\right).$$

They also derive the convergence rate of LAMB as

$$\mathbb{E}[\|\nabla f(x_a)\|^2] \leq \mathcal{O}\left(\sqrt{\frac{G^2 d}{h(1 - \beta_2)}} \times \left[\sqrt{\frac{2(f(x_1) - f(x^*))\|L\|_1}{T}} + \frac{\|\tilde{\sigma}\|_1}{\sqrt{T}}\right]\right).$$

Additionally, for $\beta_2 = 0$, the convergence rate of LAMB can be derived as

$$\left(\mathbb{E}\left[\frac{1}{\sqrt{d}} \|\nabla f(x_a)\|_1\right]\right)^2 \leq \mathcal{O}\left(\frac{(f(x_1) - f(x^*))L_{avg}}{T} + \frac{\|\tilde{\sigma}\|_1^2}{Th}\right),$$

Below we derive a similar bound for the $\beta_2 > 0$ case for Adam updates. We note that the $\beta_2 = 0$ case where the bound depends on L_{avg} instead of $||L||_1$ can be very similarly derived for Adam, but is also a very unrealistic condition in practice.

Proof. Under the assumption $\beta_1 = 0, \lambda = 0$, one could write the Adam update rule as follows:

$$x_{t+1}^{(i)} = x_t^{(i)} - \eta_t \sqrt{1-\beta_2^t} \frac{g_t^{(i)}}{\sqrt{v_t^{(i)}}},$$

where $v_t = \beta_2 v_{t-1} + (1-\beta_2) g_t^2$ for all $i \in [h]$.

Since the function f is L-smooth, we have the following:

$$\begin{aligned} f(x_{t+1}) &\leq f(x_t) + \langle \nabla_i f(x_t), x_{t+1}^{(i)} - x_t^{(i)} \rangle + \sum_{i=1}^{h} \frac{L_i}{2} \|x_{t+1}^{(i)} - x_t^{(i)}\|^2 \\ &\leq f(x_t) \underbrace{-\eta_t \sum_{i=1}^{h} \sum_{j=1}^{d_i} [\nabla_i f(x_t)]_j \sqrt{1-\beta_2^t} \frac{g_{t,j}^{(i)}}{\sqrt{v_{t,j}^{(i)}}}}_{T_1} + \sum_{i=1}^{h} \frac{L_i \eta_t^2 d}{2(1-\beta_2)} \end{aligned} \tag{2}$$

Where the last term comes from the fact that $1 - \beta_2^t \leq 1$. We bound term T_1 in the following manner, in line with (You et al., 2019):

$$\begin{aligned} T_1 &= -\eta_t \sum_{i=1}^{h} \sum_{j=1}^{d_i} [\nabla_i f(x_t)]_j \sqrt{1-\beta_2^t} \frac{g_{t,j}^{(i)}}{\sqrt{v_{t,j}^{(i)}}} \\ &\leq -\eta_t \sum_{i=1}^{h} \sum_{j=1}^{d_i} \frac{\sqrt{1-\beta_2}}{G} [\nabla_i f(x_t)]_j g_{t,j}^{(i)} \\ &\quad - \eta_t \sum_{i=1}^{h} \sum_{j=1}^{d_i} \left([\nabla_i f(x_t)]_j \sqrt{1-\beta_2^t} \frac{g_{t,j}^{(i)}}{\sqrt{v_{t,j}^{(i)}}} \right) \mathbb{1}(sign([\nabla_i f(x_t)]_j) \neq sign(g_{t,j}^{(i)})) \end{aligned}$$

Relying on the following inequalities: $\sqrt{v_t} \leq G$ and $1 - \beta_2^t > 1 - \beta_2$.
Taking expectation, we have the following:

$$\begin{aligned} \mathbb{E}[T_1] &\leq -\eta_t \sum_{i=1}^{h} \sum_{j=1}^{d_i} \frac{\sqrt{1-\beta_2}}{G} \mathbb{E}\left[[\nabla_i f(x_t)]_j g_{t,j}^{(i)} \right] \\ &\quad - \eta_t \sum_{i=1}^{h} \sum_{j=1}^{d_i} \frac{\sqrt{1-\beta_2}}{G} \mathbb{E}\left[\left([\nabla_i f(x_t)]_j g_{t,j}^{(i)} \right) \mathbb{1}(sign([\nabla_i f(x_t)]_j) \neq sign(g_{t,j}^{(i)})) \right] \\ \mathbb{E}[T_1] &\leq -\eta_t \sum_{i=1}^{h} \sum_{j=1}^{d_i} \frac{\sqrt{1-\beta_2}}{G} \mathbb{E}\left[[\nabla_i f(x_t)]_j g_{t,j}^{(i)} \right] \\ &\quad + \eta_t \sum_{i=1}^{h} \sum_{j=1}^{d_i} \sqrt{1-\beta_2} \mathbb{E}\left[|[\nabla_i f(x_t)]_j| \, \mathbb{P}(sign([\nabla_i f(x_t)]_j) \neq sign(g_{t,j}^{(i)})) \right] \end{aligned} \tag{3}$$

similarly what is shown in signsgd, we bound the probability by first relaxing the condition, then applying Markov's and then Jensen's inequality:

$$\begin{aligned}
\mathbb{P}(sign([\nabla_i f(x_t)]_j) \neq sign(g_{t,j}^{(i)})) &\leq \mathbb{P}\left(|[\nabla_i f(x_t)]_j - g_{t,j}^{(i)}| \geq |g_{t,j}^{(i)}|\right) \\
&\leq \frac{\mathbb{E}\left[|[\nabla_i f(x_t)]_j - g_{t,j}^{(i)}|\right]}{|[\nabla_i f(x_t)]_j|} \\
&\leq \frac{\sqrt{\mathbb{E}\left[([\nabla_i f(x_t)]_j - g_{t,j}^{(i)})^2\right]}}{|[\nabla_i f(x_t)]_j|} \\
&= \frac{\tilde{\sigma}_{t,i}}{|[\nabla_i f(x_t)]_j|} \\
&\leq \frac{\tilde{\sigma}_i}{\sqrt{n}|[\nabla_i f(x_t)]_j|}
\end{aligned}$$

where the last inequality is from the fact that $\tilde{\sigma}_{t,j}$ is the minibatch variance at time t with batch size n. Substituting this into our derivation of T_1

$$\mathbb{E}[T_1] \leq -\eta_t \frac{\sqrt{1-\beta_2}}{G}||\nabla f(x_t)||^2 + \eta_t\sqrt{1-\beta_2}\sum_{i=1}^{h}\sum_{j=1}^{d_i}\frac{\tilde{\sigma}_i}{\sqrt{n}}$$

and replacing this with our definition of T_1 in Eq. (2) we get

$$\mathbb{E}[f(x_{t+1})] \leq f(x_t) - \eta_t \frac{\sqrt{1-\beta_2}}{G}||\nabla f(x_t)||^2 + \eta_t\sqrt{1-\beta_2}\frac{||\tilde{\sigma}||_1}{\sqrt{n}} + \frac{||L||_1\eta_t^2 d}{2(1-\beta_2)}. \tag{4}$$

We then arrive at the final bound by summing Eq. (4) to step T and cancelling consecutive terms via the telescoping sum, followed by rearranging and then multiplying through by $\frac{G}{T\eta_t\sqrt{1-\beta_2}}$

$$\mathbb{E}[f(x_{T+1})] \leq f(x_1) - \frac{\eta_t\sqrt{1-\beta_2}}{G}\sum_{t=1}^{T}||\nabla f(x_t)||^2 + T\eta_t\sqrt{1-\beta_2}\frac{||\tilde{\sigma}||_1}{\sqrt{n}} + \frac{T||L||_1\eta_t^2 d}{2(1-\beta_2)}.$$

$$\frac{\eta_t\sqrt{1-\beta_2}}{G}\sum_{t=1}^{T}||\nabla f(x_t)||^2 \leq f(x_1) - f(x^*) + T\eta_t\sqrt{1-\beta_2}\frac{||\tilde{\sigma}||_1}{\sqrt{n}} + \frac{T||L||_1\eta_t^2 d}{2(1-\beta_2)}$$

$$\frac{1}{T}\sum_{t=1}^{T}||\nabla f(x_t)||^2 \leq G\left(\frac{f(x_1) - f(x^*)}{T\eta_t\sqrt{1-\beta_2}} + \frac{||\tilde{\sigma}||_1}{\sqrt{n}} + \frac{||L||_1\eta_t d}{2(1-\beta_2)^{\frac{3}{2}}}\right)$$

Taking $\eta_t = \eta = \sqrt{\frac{2(f(x_1)-f(x^*))}{T||L||_1(1-\beta_2)d}}$ and letting $n = T$ as is similarly done in (You et al., 2019), we can recover a bound that, up to some constants, is similar to the bound for LAMB:

$$\begin{aligned}
\mathbb{E}[||\nabla f(x_t)||^2] &\leq \mathcal{O}\left(G\left(\frac{f(x_1) - f(x^*)}{T\sqrt{\frac{2(f(x_1)-f(x^*))}{T||L||_1(1-\beta_2)d}}\sqrt{1-\beta_2}} + \frac{||\tilde{\sigma}||_1}{\sqrt{n}} + \frac{||L||_1\sqrt{\frac{2(f(x_1)-f(x^*))}{T||L||_1(1-\beta_2)d}}d}{2(1-\beta_2)^{\frac{3}{2}}}\right)\right) \\
&= \mathcal{O}\left(G\left(\frac{1}{2}\sqrt{\frac{2(f(x_1) - f(x^*))||L||_1 d}{T}} + \frac{||\tilde{\sigma}||_1}{\sqrt{n}} + \frac{1}{2(1-\beta_2)^2}\sqrt{\frac{2(f(x_1) - f(x^*))||L||_1 d}{T}}\right)\right) \\
&= \mathcal{O}\left(G\left(1 + \frac{1}{(1-\beta_2)^2}\right)\sqrt{\frac{2(f(x_1) - f(x^*))||L||_1 d}{T}} + \frac{G||\tilde{\sigma}||_1}{\sqrt{T}}\right)
\end{aligned}$$

□

B ADDITIONAL EXPERIMENT DETAILS

B.1 RESNET-50 TRAINING BENCHMARK

All experiments were run on Google TPUs (Jouppi et al., 2017). We typically trained on TPUv2-256 or TPUv3-128 in order to accommodate the 32,768 batch size. The ResNet-50 experiments used Jax (Bradbury et al., 2018) using the Flax library, with code released here. The BERT experiments were run using TensorFlow (Abadi et al., 2015) version 1.15. We used the standard train/validation split from the previous literature and MLPerf competition.

For ImageNet, we used the following sequence of TensorFlow functions for pre-processing:[17]

```
tf.image.sample_distorted_bounding_box
tf.image.decode_and_crop_jpeg
tf.image.resize
tf.image.random_flip_left_right
tf.image.convert_image_dtype
```

B.2 BERT PRE-TRAINING

We used the same experimental setup as the official BERT codebase[18] and the standard train/test split from the previous literature. This matches the experimental setup of You et al. (2019). We trained on Google TPUs, using TPUv3-256 or TPUv3-512 for the 32,768 batch size experiments, and TPUv3-1024 for the 65,536 batch size experiments.

We trained the two pretraining objectives on the combined Wikipedia and Books corpus (Zhu et al., 2015) datasets (2.5B and 800M words, respectively). We used sequence lengths of 128 and 512, respectively, for the pretraining tasks. We ran the fine-tuning phase on the SQuaD v1.1 question answering task. In order to match You et al. (2019), we report the F1-score on the dev set as the target metric. We followed the fine-tuning protocol described in the LAMB optimizer setup and did not perform any additional tuning for fine-tuning.

We tuned Adam hyperparameters using quasi-random search (Bousquet et al., 2017) in a simple search space. Hyperparameters included learning rate η, β_1, β_2, the polynomial power for the learning rate warmup p_{warmup}, and weight decay λ. We fixed the ϵ in Adam to 10^{-11} for all BERT experiments. See Appendix E.2 for the search spaces. We selected the best trial using the masked language model accuracy over 10k examples from the training set. The number of training steps for each of the phases, as well as the warmup steps are identical to You et al. (2019) and are listed in Appendix E.2. Each phase of pretraining used completely independent Adam hyperparameters. We found the final hyperparameters within 30 trials of random search for each of the phases, except for the second phase of 65,536 batch size which used 130 trials.

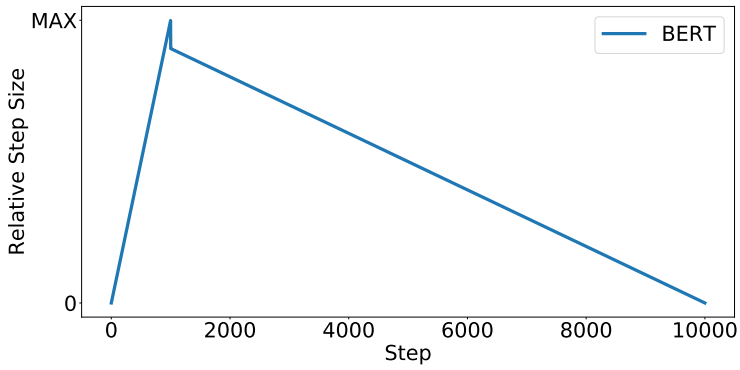

Figure 3: An illustration of the sudden drop in the BERT learning rate schedule in the official codebase.

[17] Full code available at `https://git.io/JtgtE` [18] `https://github.com/google-research/bert`

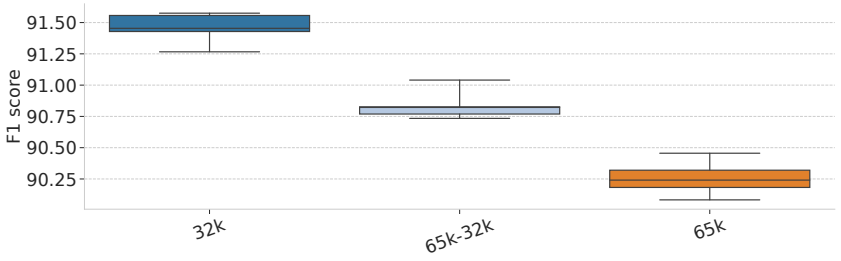

Figure 4: 6 finetuning runs starting from the same pretraining checkpoint to show the stability of our results, at each of the 32,768, mixed 65,536-32,768, and 65,536 batch size settings.

C NESTEROV ABLATIONS

To explore the sensitivity of our best Nesterov momentum configuration (Configuration A), we ablated several elements of the experiment pipeline, one at a time, and tested their impact on performance. Figure 5 shows the results of these experiments. "Base" refers to Nesterov momentum Configuration A (Table 5). "ResNet version" is the same point as "Base" but with ResNet version 1.0 instead of version 1.5. "BN init" is the same point as "Base" but with $\gamma_0 = 1.0$ instead of 0.4138. "Virtual BN" is the same point as "Base" but with a virtual batch size of 256 instead of 64, which is the largest that fits in a single TPUv3 core. "BN & LR tuning" is Configuration B (Table 5), the same point as "Base" but with $p_{decay}, t_{warmup}, \eta_0, \rho, \epsilon$ set to their values in the LARS pipeline. Finally, "L2 variables" is the same point as "Base" but where the L2 regularization is applied to all variables. The only ablation whose median over 50 seeds continues to beat the target 75.9% accuracy (noted by the dotted red line) is "BN & LR tuning", with the rest having between 0.1%-0.3% drops in median accuracy.

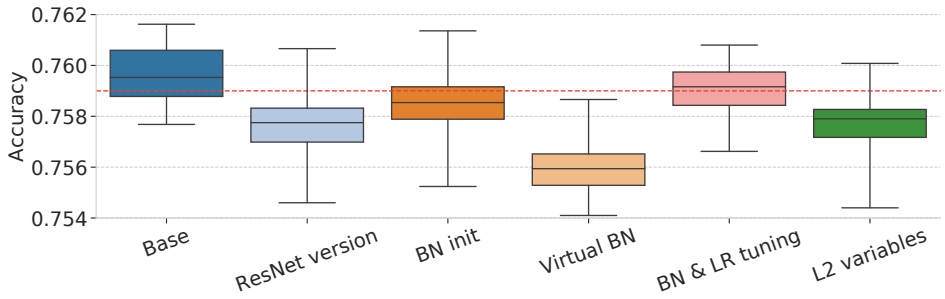

Figure 5: Distributions over 50 training runs for each ablation study around our best Nesterov momentum configuration (Configuration A). The dotted red line is at the target accuracy of 75.9%, and the boxes show the min, max, and quartiles of the distribution of accuracies over the 50 training runs.

D EASE OF HYPERPARAMETER TUNABILITY

Overall, we see a mixed picture of which algorithm is "easier" to tune given our setup. This section analyzes the performance of LARS and Nesterov on as similar of a hyperparamter sweep as we could design, in an attempt to gain insight into how "easy" it is to tune their hyperparameters. The hyperparameter searches in this section contained 10 hyperparamter settings for each of the learning rate (η) and weight decay (λ), on a log scale, resulting in a 2D grid of 100 total hyperparameter settings. The grid is centered around the best points found in the hyperparameter searches of Table 4 (see Table 8 for more details, and Tensorboard links for the best hyperparameter settings). As done in Section 4, we use a simple cosine decay learning rate schedule, so we only have a single learning rate hyperparameter.

Between Figures 6, 7, there is evidence that the strength of LARS is in regularizing, whereas Nesterov is better at minimizing the training objective; this can complicate comparisons on problems where regularization is important such as ImageNet. As seen in Figure 6, for very competitive validation accuracies $> 76.6\%$ LARS has a slightly larger fraction of hyperparameter settings that achieve the target validation accuracy, for accuracies between $[76.2\%, 76.6\%]$ the two algorithms are very close, and for all accuracies $< 76.2\%$ (which include the target 75.9% used in the main text) Nesterov has a larger fraction of hyperparameter settings achieving the goal. However, in Figure 7 we see a reversal when looking at the training accuracy; Nesterov seems to achieve the target training accuracy more frequently than LARS for more competitive targets, giving further evidence that LARS is actually acting more as a regularizer.

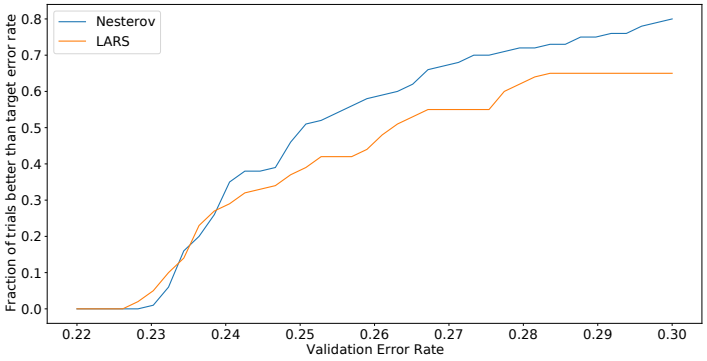

Figure 6: The fraction of hyperparameter settings that achieve better than a target validation error rate on ImageNet using the ResNet-50 setup from Section 4.

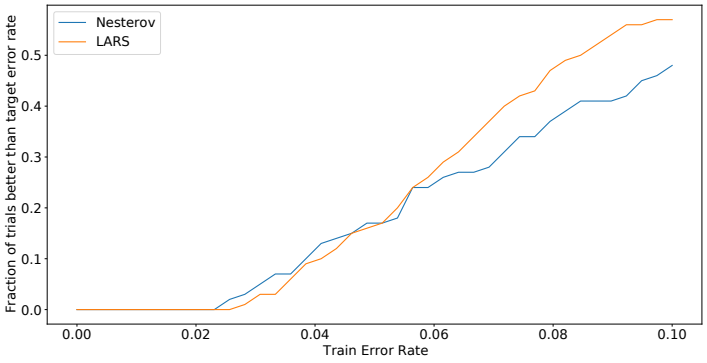

Figure 7: The fraction of hyperparameter settings that achieve better than a target training error rate on ImageNet using the ResNet-50 setup from Section 4.

In Figures 9, 8, we find that LARS has an optimal learning rate that is independent of the value of weight decay, whereas Nesterov seemed to have an optimal learning rate inversely proportional to weight decay. This means that LARS can tune η and then λ independently, which could lead to simpler tuning setups that still result in high-performing settings. This could also mean that for Nesterov the best η values were not contained in our search space for extreme values of λ. However, given that LARS (and LAMB) contain a weight decay term in the denominator of their step size calculations, it could make sense to reparameterize the Nesterov search space to η by $\frac{\eta}{\lambda}$; it is unclear if there should also be reparameterizations applied to LARS tuning as well, and how reparameterizing would affect the fairness of comparisons.

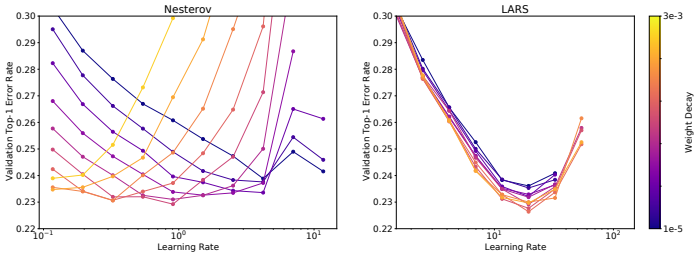

Figure 8: The validation error rate on ImageNet using the ResNet-50 setup from Section 4, bucketed by weight decay value.

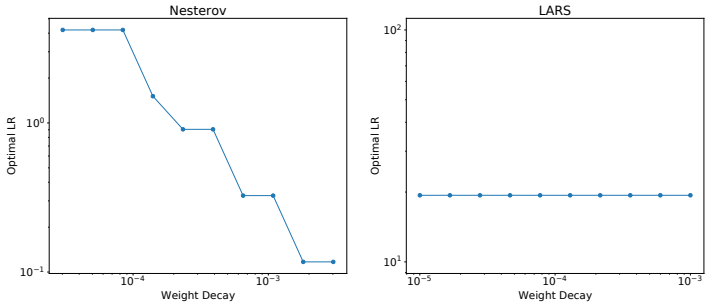

Figure 9: The best performing learning rate for a given weight decay, as measured by the validation error rate on ImageNet using the ResNet-50 setup from Section 4.

Slicing the data a different way, we can look at how η affects performance for a particular value of λ in Figures 10, 11, 12, 13. These figures also show Nesterov can achieve a slightly better training accuracy compared to LARS. However, both algorithms have similar ranges of performance for a given λ, implying neither algorithm is notably more or less sensitive to η when using a fixed λ.

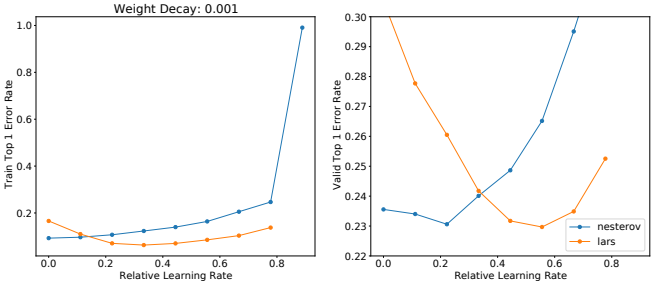

Figure 10: For $\lambda = 10^{-3}$, validation error rate on ImageNet using the ResNet-50 setup from Section 4.

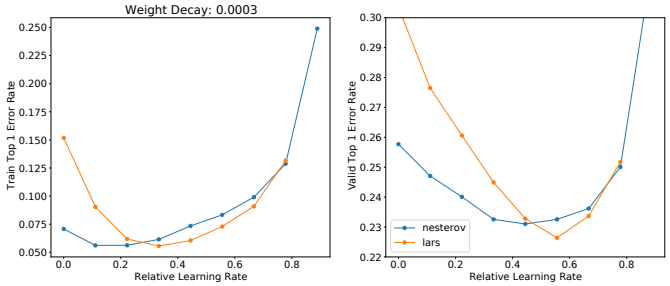

Figure 11: For $\lambda = 3 \times 10^{-4}$, validation error rate on ImageNet using the ResNet-50 setup from Section 4.

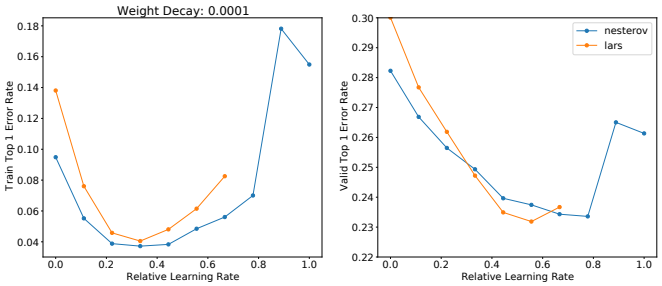

Figure 12: For $\lambda = 10^{-4}$, validation error rate on ImageNet using the ResNet-50 setup from Section 4.

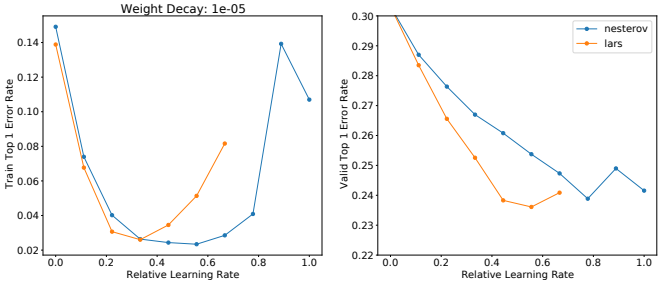

Figure 13: For $\lambda = 10^{-5}$, validation error rate on ImageNet using the ResNet-50 setup from Section 4.

E HYPERPARAMETER TUNING

For each search space, we include all the hyperparameter points and their performance in Tensorboards hosted on `tensorboard.dev`. The exact points generated from each search are available in the "hparams" tab. This data can be downloaded via the Tensorboard DataFrames API as CSV files or pd.DataFrames, for example to be analyzed however any future hyperparameter studies see fit.

E.1 NESTEROV MOMENTUM TRAINING SPEED ON RESNET-50

We considered two configurations of Nesterov hyperparameters: Configuration A, where we tuned a wide set of hyperparameters in the experiment pipeline, and Configuration B, where we reve eeed the less impactful hyperparameters to the same values as the LARS baseline (or in the case of p_{warmup}, a simpler value). We included Configuration B in order to demonstrate the minimal set of changes to the baseline necessary to still reach the target accuracy. The hyperparameter values for these configurations can be found in Table 5.

	Configuration A	Configuration B	LARS
t_{warmup}	638	706	706
p_{warmup}	2.497	2.0	1.0
p_{decay}	1.955	2.0	2.0
ρ	0.94	0.9	0.9
ϵ	4×10^{-6}	10^{-5}	10^{-5}
η_{peak}	7.05	7.05	29.0
η_{final}	6×10^{-6}	6×10^{-6}	10^{-4}
$1 - \mu$	0.02397	0.02397	0.071
λ	5.8×10^{-5}	5.8×10^{-5}	10^{-4}
τ	0.15	0.15	0.10
γ_0	0.4138	0.4138	0.0

Table 5: Nesterov momentum Configurations A and B.

E.2 ADAM ON BERT

The search space used to tune Adam on BERT for all phases of the pipeline can be found in Table 6, which yielded our best Adam results on BERT in Table 7.

Hyperparameter	Range	Scaling
p	$\{1, 2\}$	Discrete
η	$[10^{-5}, 1.0]$	Log
$1 - \beta_1$	$[10^{-2}, 0.5]$	Log
$1 - \beta_2$	$[10^{-2}, 0.5]$	Log
λ	$[10^{-3}, 10]$	Log

Table 6: The search space used to tune Adam on BERT for all phases of the pipeline. λ refers to weight decay and p refers to the polynomial power in the learning rate schedule for both the warmup and decay phases. Tensoboards: 32k Phase 1, 32k Phase 2, 65k Phase 1, 65k Phase 2, Mixed Phase 2.

Batch size	Phase	Seq len	Warmup steps	Train steps	Learning rate	β_1	β_2	λ	p
32,768	1	128	3,125	14,063	5.9415×10^{-4}	0.934271	0.989295	0.31466	1
32,768	2	512	781	1,562	2.8464×10^{-4}	0.963567	0.952647	0.31466	1
65,536	1	128	2,000	7,037	1.3653×10^{-3}	0.952378	0.86471	0.19891	2
32,768	2	512	781	1,562	2.8464×10^{-4}	0.952647	0.963567	0.19891	2
65,536	2	512	390	781	6.1951×10^{-5}	0.65322	0.82451	0.19891	2

Table 7: Best hyperparameters from tuning Adam on BERT-Large pretraining. λ refers to weight decay and p refers to the polynomial power in the learning rate schedule for both the warmup and decay phases. All trials used $\epsilon = 10^{-11}$.

Weights Optimizer	Bias/BN Optimizer	Name	Initial Range	Final Range	Best
Nesterov	Nesterov	η	np.logspace(-.5, .5, 10)	$[0.8, 3]$	1.173
Nesterov	Nesterov	λ	np.logspace(-4, -3, 10)	$[3 \times 10^{-4}, 10^{-3}]$	3.026×10^{-4}
LARS	Heavy-ball momentum	η	np.logspace(0, 2, 10)	$[10, 40]$	14.49
LARS	Heavy-ball momentum	λ	np.logspace(-5, -2, 10)	$[5 \times 10^{-5}, 2 \times 10^{-4}]$	1.708×10^{-4}
LARS	LARS	η	$[1, 30]$	$[10, 30]$	14.18
LARS	LARS	λ	$[10^{-4}, 10^{-1}]$	$[5 \times 10^{-5}, 5 \times 10^{-4}]$	5.278×10^{-5}
Adam ($\epsilon = 10^{-8}$)	Adam ($\epsilon = 10^{-8}$)	η	$[10^{-3}, 1]$	$[4 \times 10^{-3}, 2 \times 10^{-2}]$	0.004596
Adam ($\epsilon = 10^{-8}$)	Adam ($\epsilon = 10^{-8}$)	λ	$[10^{-2}, 4]$	$[2 \times 10^{-1}, 1]$	0.6182
Adam ($\epsilon = 10^{-6}$)	Adam ($\epsilon = 10^{-6}$)	η	np.logspace(-3, 0, 10)	$[3 \times 10^{-3}, 10^{-2}]$	3.332×10^{-3}
Adam ($\epsilon = 10^{-6}$)	Adam ($\epsilon = 10^{-6}$)	λ	np.logspace(-2, 0.5, 6)	$[0.5, 2]$	1.055
LAMB	LAMB	η	np.logspace(-4, 0, 30)	$[4 \times 10^{-3}, 5 \times 10^{-2}]$	0.01134
LAMB	LAMB	λ	np.logspace(-5, -2, 4)	$[1 \times 10^{-2}, 0.1]$	0.02657
LAMB	Adam ($\epsilon = 10^{-8}$)	η	$[10^{-3}, 1]$	$[10^{-2}, 8 \times 10^{-2}]$	0.02569
LAMB	Adam ($\epsilon = 10^{-8}$)	λ	$[10^{-2}, 4]$	$[1, 8]$	2.500
LAMB	Adam ($\epsilon = 10^{-6}$)	η	np.logspace(-3, 0, 10)	$[10^{-2}, 8 \times 10^{-2}]$	0.03378
LAMB	Adam ($\epsilon = 10^{-6}$)	λ	np.logspace(-2, 0.5, 6)	$[1, 8]$	4.197

Table 8: Search spaces used for the 6,000 step, cosine learning rate schedule experiments. All hyperparameters were tuned on a logarithmic scale, except for those which define a discrete sequence of points to evaluate such as "np.logspace". Some Tensorboards: Nesterov, Adam ($\epsilon = 10^{-6}$) Final, Adam ($\epsilon = 10^{-8}$) Initial, Adam ($\epsilon = 10^{-8}$) Final, LAMB/Adam ($\epsilon = 10^{-6}$) Initial, LAMB/Adam ($\epsilon = 10^{-6}$) Final, LAMB/Adam ($\epsilon = 10^{-8}$) Initial, LAMB/Adam ($\epsilon = 10^{-8}$) Final, LAMB/LAMB Final, LARS/LARS Initial, LARS/LARS Final, LARS/Momentum Final.

E.3 Less stringent step budget on ResNet-50

All trials used a cosine decay learning rate schedule and tuned the initial learning rate η and L2 regularization or weight decay parameter[19] λ according to Table 8. We used 50 or more trials to search in the "Initial Range" and then 25 trials to search in the refined "Final Range." Finally, we ran the best point from the latter for 5 random seeds. When LARS or LAMB were used alongside a different optimizer for the batch normalization and ResNet-50 bias parameters, we set $\lambda = 0$ on

[19] As suggested in You et al. (2019), we used L2 regularization for LARS and weight decay for LAMB. For consistency, we used L2 regularization for Nesterov momentum (which is more analogous to LARS) and weight decay for Adam (which is more analogous to LAMB).

	Range	Scaling
η_0	$[10^{-3}, 50.0]$	Log
$\eta_{\text{decay\_factor}}$	$\{10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}\}$	Discrete
$1 - \mu$	$[10^{-3}, 1.0]$	Log
λ	$[10^{-5}, 10^{-1}]$	Log
τ	$[10^{-2}, 2 \times 10^{-1}]$	Linear

Table 9: First search space of the Nesterov tuning journey. The search spaces were mostly by informed guesses by the authors. λ refers to weight decay, which is applied to all variables. Tuned for 251 trials. Trained for 2,815 steps ("72 epochs" as defined by MLPerf epoch calculations). We used a linear learning rate decay schedule that decays for all training steps, starting from η_0 and ending at $\eta_0 \times \eta_{decay\_factor}$. Virtual batch size 128. Tensorboard.

	Range	Scaling
η_0	$[10^{-3}, 50.0]$	Log
$\eta_{\text{decay\_factor}}$	$\{10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}\}$	Discrete
$1 - \mu$	$[10^{-3}, 1.0]$	Log
λ	$[10^{-5}, 10^{-1}]$	Log
τ	$[10^{-2}, 2 \times 10^{-1}]$	Linear

Table 10: Same as Table 9 but trained for 2,658 steps ("68 epochs" as defined by MLPerf epoch calculations) for 50 trials. Tensorboard.

the batch normalization and ResNet-50 bias parameters. When LAMB was used all parameters, the majority of trials diverged during training – it took **67 trials** to get 25 trials that did not NaN during training. Our trial budgets refer to the number of feasible trials, i.e. trials that do not diverge during training.

E.4 Nesterov ResNet50 search space chronology

Below we list the sequence of search spaces we used to arrive at our final values in Table 5. Given that the final results reported in papers are rarely found in a single iteration of experiments, we believe that it is important to document the full journey to arriving at our results.

Note that although we tuned a wide range of hyperparameters to match the LARS result with Nesterov momentum, we later realized that many of these hyperparameters could be reverted to the values from the LARS pipeline (see Table 5). We started tuning with a training budget of 2,815 steps, which is the number of steps in the MLPerf 0.6 submission. We sometimes would decrease this to 2,658 steps to test how decreasing the training budget would affect tuning performance, before eventually moving to the 2,512 steps used to generate the results in the main text.

	Range	Scaling
η_0	$[10^{-1}, 20.0]$	Log
$\eta_{\text{decay\_factor}}$	$\{10^{-5}, 10^{-4}, 10^{-3}\}$	Discrete
t_{decay}	$[2392, 2.658]$	Linear
$1 - \mu$	$[10^{-3}, 1.0]$	Log
λ	$[10^{-5}, 2 \times 10^{-1}]$	Log
τ	$[10^{-2}, 2 \times 10^{-1}]$	Linear

Table 11: λ refers to weight decay, which is now not applied to the bias and batch normalization variables. 50 trials. Trained for 2,658 steps. Linear learning rate decay schedule that decays for t_{decay} steps, starting from η_0 and ending at $\eta_0 \times \eta_{decay\_factor}$. Virtual batch size 128. Tensorboard.

	Range	Scaling
η_{peak}	$[10^{-1}, 32.0]$	Log
$\eta_{\text{decay\_factor}}$	$\{10^{-5}, 10^{-4}, 10^{-3}\}$	Discrete
t_{decay}	$[2392, 2.658]$	Linear
$1 - \mu$	$[10^{-4}, 10^{-1}]$	Log
λ	$[10^{-4}, 10^{-1}]$	Log
τ	$[5 \times 10^{-2}, 0.15]$	Linear

Table 12: λ refers to weight decay, which is not applied to the bias and batch normalization variables. 50 trials. Trained for 2,658 steps. Linear warmup for 500 steps followed by a quadratic decay, which decays until step t_{decay}, and then is constant at the final learning rate $\eta_0 \times \eta_{decay\_factor}$. Virtual batch size 128. We increased the max learning rate based off the larger learning rates used by LARS. **We also ran two additional studies which were the same except with 250 and 977 warmup steps. 250 warmup Tensorboard, 500 warmup Tensorboard, 977 warmup Tensorboard.**

	Range	Scaling
η_{peak}	$[10^{-1}, 32.0]$	Log
$\eta_{\text{decay\_factor}}$	$[3 \times 10^{-5}, 3 \times 10^{-4}]$	Log
t_{decay}	$[2533, 2.815]$	Linear
$1 - \mu$	$[10^{-4}, 10^{-1}]$	Log
λ	$[10^{-4}, 10^{-1}]$	Log
τ	$[5 \times 10^{-2}, 0.15]$	Linear

Table 13: λ refers to weight decay, which is not applied to the bias and batch normalization variables. 50 trials. Trained for 2,815 steps. Linear warmup for 500 steps followed by a quadratic decay, which decays until step t_{decay}, and then is constant at the final learning rate $\eta_0 \times \eta_{decay\_factor}$. Virtual batch size 128. Tensorboard.

	Range	Scaling
η_{peak}	$[10^{-1}, 32.0]$	Log
$\eta_{\text{decay\_factor}}$	$[3 \times 10^{-5}, 3 \times 10^{-4}]$	Log
t_{decay}	$[2533, 2.815]$	Linear
$1 - \mu$	$[5 \times 10^{-3}, 10^{-1}]$	Log
λ	$[10^{-2}, 10^{-1}]$	Log
τ	$[5 \times 10^{-2}, 0.15]$	Linear

Table 14: λ refers to weight decay, which is not applied to the bias and batch normalization variables. 50 trials. Trained for 2,815 steps. Linear warmup for 500 steps followed by a quadratic decay, which decays until step t_{decay}, and then is constant at the final learning rate $\eta_0 \times \eta_{decay\_factor}$. Virtual batch size 128. Tensorboard.

	Range	Scaling
η_{peak}	$[10^{-1}, 32.0]$	Log
$\eta_{\text{decay\_factor}}$	$[3 \times 10^{-5}, 3 \times 10^{-4}]$	Log
t_{decay}	$[2533, 2.815]$	Linear
$1 - \mu$	$[5 \times 10^{-3}, 10^{-1}]$	Log
λ	$[10^{-2}, 10^{-1}]$	Log
τ	$[5 \times 10^{-2}, 0.15]$	Linear

Table 15: The same as Table 14 except with virtual batch size 64. Tensorboard.

	Range	Scaling
η_{peak}	$\{\{10^{\alpha}, 2 \times 10^{\alpha}, ..., 9 \times 10^{\alpha}\}$ $\forall \alpha \in \{-3, ...2\}\} + \{100, \}$	Discrete
$\eta_{\text{decay\_factor}}$	8.144×10^{-5}	–
t_{decay}	2250	–
$1 - \mu$	0.02397	–
λ	0.009992	–
τ	0.07786	–

Table 16: λ refers to weight decay, which is not applied to the bias and batch normalization variables. Trained for 2,815 steps. Virtual batch size 64. Using the best hyperparameters from Table 15, we swept over the peak learning rate in a discrete set of ten values per order of magnitude, **each for three random seeds**, to find the max stable learning rate. Tensorboard.

	Range	Scaling
η_{peak}	4.118	–
$\eta_{\text{decay\_factor}}$	8.144×10^{-5}	–
t_{decay}	2250	–
$1 - \mu$	0.02397	–
λ	$\{\{0.5 \times 10^{\alpha}, 10^{\alpha}, ...\}$ $\forall \alpha \in \{-3, ...0\}\} + \{1.0, \}$	Discrete
τ	0.07786	–

Table 17: λ refers to weight decay, which is not applied to the bias and batch normalization variables. Trained for 2,815 steps. Virtual batch size 64. Using the best hyperparameters from Table 15, we swept over the weight decay in a discrete set of twenty values per order of magnitude, to test how high the regularization has to be in this region of hyperparameter space. Tensorboard.

	Range	Scaling
η_{peak}	4.118	–
$\eta_{\text{decay\_factor}}$	8.144×10^{-5}	–
t_{decay}	2250	–
$1 - \mu$	0.02397	–
λ	0.009992	–
τ	0.07786	–
ρ	$\{0.0, 0.1, 0.3, 0.5, 0.6, 0.7,$ $0.8, 0.9, 0.95, 0.995, 0.999\}$	Discrete
ϵ	$\{10^{-7}, 10^{-6}, 10^{-5}, 10^{-4},$ $10^{-3}, 10^{-2}, 10^{-1}\}$	Discrete

Table 18: λ refers to weight decay, which is not applied to the bias and batch normalization variables. Trained for 2,815 steps. Virtual batch size 64. Using the best hyperparameters from Table 15, we swept over batch normalization hyperparameters. Tensorboard.

	Range	Scaling
η_{peak}	$[2.0, 8.0]$	Log
$\eta_{\text{decay\_factor}}$	$[4 \times 10^{-5}, 1.6 \times 10^{-4}]$	Linear
t_{decay}	$[2100, 2400]$	Linear
$1 - \mu$	$[0.012, 0.04]$	Log
λ	$[7 \times 10^{-3}, 7 \times 10^{-2}]$	Log
τ	$[0.04, 0.1]$	Linear
ρ	$[0.45, 0.55]$	Linear
ϵ	$[5 \times 10^{-6}, 5 \times 10^{-5}]$	Linear

Table 19: λ refers to weight decay, which is not applied to the bias and batch normalization variables. 50 trials. Trained for 2,815 steps. Linear warmup for 500 steps followed by a quadratic decay, which decays until step t_{decay}, and then is constant at the final learning rate $\eta_0 \times \eta_{decay\_factor}$. Virtual batch size 64. Peak learning rate range was consolidated based off the results of Table 16. The weight decay range was consolidated based off the results of Table 17. Tensorboard.

	Range	Scaling
t_{warmup}	$[300, 800]$	Linear
p_{warmup}	$[0.7, 2.0]$	Linear
p_{decay}	1.8	–
η_0	$[0.1, 1.0]$	Log
η_{peak}	$[5.0, 9.0]$	Log
η_{final}	$[10^{-5}, 5 \times 10^{-5}]$	Log
$1 - \mu$	0.02397	–
λ	5×10^{-5}	–
τ	0.15	–
γ_0	$[0.0, 0.6]$	Linear
ρ	0.94	–
ϵ	4×10^{-6}	–

Table 20: Here we switched λ to refer to L2 regularization. We also began training for 2,512 steps, which is the final "64 epochs" used in the Nesterov results reported in the main text. Because of this more stringent step budget, we focused on the learning rate schedule. t_{decay} was set to all remaining steps after the warmup was finished. Tuned for 229 trials. Virtual batch size 64. Tensorboard.

	Range	Scaling
t_{warmup}	638	–
p_{warmup}	$[1.5, 3.0]$	Linear
p_{decay}	$[1.5, 2.5]$	Linear
η_0	0.12	–
η_{peak}	7.05	–
η_{final}	$[10^{-6}, 5 \times 10^{-4}]$	Log
$1 - \mu$	0.02397	–
λ	$[5 \times 10^{-5}, 1 \times 10^{-3}]$	Log
τ	0.15	–
γ_0	$[0.4, 1.0]$	Linear
ρ	0.94	–
ϵ	4×10^{-6}	–

Table 21: Here we began focusing more on the shape of the learning rate schedule, as well as retuning the L2 regularization. λ refers to L2. Several values were picked from the best trial of Table 20. Trained for 2,512 steps steps. Tuned for 15 trials. Virtual batch size 64. Tensorboard.

	Range	Scaling
t_{warmup}	638	–
p_{warmup}	$[1.5, 3.0]$	Linear
p_{decay}	$[1.5, 2.5]$	Linear
η_0	0.12	–
η_{peak}	7.05	–
η_{final}	$[10^{-6}, 5 \times 10^{-4}]$	Log
$1 - \mu$	0.02397	–
λ	$[1 \times 10^{-5}, 1 \times 10^{-4}]$	Log
τ	0.15	–
γ_0	$[0.4, 1.0]$	Linear
ρ	0.94	–
ϵ	4×10^{-6}	–

Table 22: Here we focus in more on tuning the L2 regularization. λ refers to L2. Trained for 2,512 steps steps. Tuned for 37 trials. Virtual batch size 64. Tensorboard.

	Range	Scaling
t_{warmup}	638	–
p_{warmup}	$[1.5, 3.0]$	Linear
p_{decay}	$[1.5, 2.5]$	Linear
η_0	0.12	–
η_{peak}	7.05	–
η_{final}	$[10^{-6}, 5 \times 10^{-4}]$	Log
$1 - \mu$	0.02397	–
λ	$[5 \times 10^{-5}, 6 \times 10^{-5}]$	Linear
τ	0.15	–
γ_0	$[0.4, 1.0]$	Linear
ρ	0.94	–
ϵ	4×10^{-6}	–

Table 23: Again we dial in more on a tighter tuning range for the L2 regularization. λ refers to L2. Trained for 2,512 steps steps. Tuned for 37 trials. Virtual batch size 64. Tensorboard.