Bias Decay Matters : Improving Large Batch Optimization with Connectivity Sharpness

Anonymous authors
Paper under double-blind review

Abstract

As deep learning becomes computationally intensive, the data parallelism is an essential option for the efficient training of high-performance models. Accordingly, the recent studies deal with the methods for increasing batch size in training the model. Many recent studies focused on learning rate, which determines the noise scale of parameter updates (Goyal et al., 2017; You et al., 2017; 2020) and found that a high learning rate is essential for maintaining generalization performance and flatness of the local minimizers (Jastrzebski et al., 2020; Cohen et al., 2021; Lewkowycz et al., 2020). But to fill the performance gap that still exists in the large batch optimization, we study a method to directly control the flatness of local minima. Toward this, we define yet another sharpness measure called *Connectivity sharpness*, a reparameterization invariant, structurally separable sharpness measure. Armed with this measure, we experimentally found the standard *no bias decay heuristic* (Goyal et al., 2017; He et al., 2019b), which recommends the bias parameters and γ and β in BN layers are left unregularized in training, is a crucial reason for performance degradation in large batch optimization. To mitigate this issue, we propose simple bias decay methods including novel adaptive one and found that this simple remedy can fill a large portion of the performance gaps that occur in large batch optimization.

1 Introduction

Recently, deep learning has brought unprecedented success and change in various fields such as computer vision (Radford et al., 2021), natural language understanding (Brown et al., 2020) with the advent of huge models. In order to increase the accessibility and efficiency of such a large model, the importance of data parallelism, which can reduce the learning time by using more computing resources, is being emphasized. In particular, for training deep learning models based on synchronous Stochastic Gradient Descent (SGD) with mini-batch, it is necessary to increase the batch size to fully utilize computational resources, but it is well known that using a large batch degrades the generalization performance of the model.

The basis of recent large batch optimization methods such as Goyal et al. (2017); You et al. (2017; 2020) is to properly increase the learning rate (LR). Tuning learning rate for large batch optimization is quite convincing from three perspectives. First, the LR controls the searching speed in parameter space. Since fewer steps are allowed in large batch optimization, a faster searching for each step is required to match the overall search performance. Second, the LR controls the variance of optimization trajectory. As the batch size increases, the variance of the gradient decreases, which makes it difficult to escape from the sharp minima in large batch optimization (Smith & Le, 2018; Smith et al., 2018; Park et al., 2019; Smith et al., 2020). By increasing the LR, we can artificially control the variance of the optimization trajectory. Third, it is known that models trained with high LR achieve flatter minima than models trained with low LR in batch gradient descent settings (Jastrzebski et al., 2020; Cohen et al., 2021; Lewkowycz et al., 2020).

Since there is still a limit to maintaining performance in a large batch regime only with the existing methods of tuning the learning rate, in this paper, we study a method to directly control the flatness of local minima. For this purpose, we propose yet another sharpness measure called *connectivity sharpness*, inspired by some recent works on pruning at initialization (Lee et al., 2019b; Wang et al., 2020; Lee et al., 2020). Connectivity sharpness is distinguished from the existing sharpness measures in that it is invariant to reparameterization and linearly separable with respect to substructures. The reparameterization invariance is essential because it measures the same sharpness for different networks expressing the same function. At the same time, the substructure separability is vital in identifying which part of the network critically affects the flatness.

To systematically reduce the proposed connectivity sharpness, we analyzed the *no bias decay heuristic* through a step-by-step experiment. Our analysis shows that the scale of bias parameters really matters in large batch optimization with high learning rate and unregularizing bias parameters results poor generalization in large batch optimization. Based on this phenomenon, we propose bias decay (without decaying scale parameter in BN) to properly resolve the dilemma between stability of optimization and generalization. Moreover, we propose a layer-wise bias decay that selectively apply the bias decay for layers whose sharpness of bias is large. We find that this simple remedy can fill a large portion of the performance gaps that occur in large batch optimization.

Our contribution is threefold:

- We propose a novel sharpness measure, *Connectivity sharpness*. We present proposed connectivity sharpness has two properties *reparameterization invariance* and *substructure separability*, which were less considered in previous works on sharpness and generalization.
- By analyzing the substructures connectivity sharpness of networks, we found that sharpness of bias affect the sharpness of entire network. We explain the pitfalls of no bias decay heuristic that results sharper model in terms of connectivity.
- Based on these observations, we propose a novel regularization methods, adaptive β decay, to mitigate the pitfalls of no bias decay heuristics. We show that this method can fill a large portion of the performance gaps in large batch optimization: *improve top-1 accuracy of ResNet50 from 76.56% to 77.24% in ImageNet training with batch size of 32k*.

2 Related works

Large batch optimization Although the large batch training can speed up network training dramatically, it is well-known that the generalization of small batch training is better than that of large batch training. Many researchers delved into the large batch optimization with decent generalization. In order to reduce the generalization gap, Goyal et al. (2017) scaled the learning rate in proportion to the minibatch size and introduces several practical training techniques. However, because the larger the minibatch size, the larger the learning rate, the proposed learning rate scaling in Goyal et al. (2017) could easily diverge in huge minibatch regime. For this reason, You et al. (2017) proposed Layer-wise Adaptive Rate Scaling (LARS) which incorporates the layer-wise gradient norm into the learning rate scaling. Going further, You et al. (2020) extended the LARS strategy to Adam optimizer that gave the birth to LAMB with which successfully train BERT in 76 minutes. Another work (Johnson et al., 2020) also adjusted the learning rate exploiting the variance of gradients. However, the very recent work (Nado et al., 2021) conducted extensive experiments to show that traditional optimizers such as Nesterov could generalize well in large batch regime given ample hyperparameter tuning.

Sharpness/Flatness of minima Keskar et al. (2017) argued that the sharp minima lead to generalization degradation. In similar spirit, He et al. (2019a) showed that local minima are on an asymmetric valley. Among them, they also showed that he local minima biased towards the flat side generalizes better. Under this motivation, Foret et al. (2020) proposed Sharpness-Aware Minimization (SAM) that enhance model generalization by simultaneously minimizing the objective function and loss sharpness. Extending SAM optimizer, Kwon et al. (2021) introduced adaptive sharpness of loss landscape which is invariant to parameter re-scaling, and the authors show that adaptive sharpness is related to generalization much more than sharpness. However, despite several studies on the relationship between sharpness and generalization, Dinh et al. (2017) proved that all minima of neural networks can be arbitrarily sharp depending on appropriate reparameterization, thereby insisting that generalization has nothing to do with sharpness. Owing to this fact, a sharpness measure invariant to reparameterization is indispensable.

Weight decay and generalization Loshchilov & Hutter (2019) suggested that the weight decay should be decoupled from gradient computation, thereby improving the model performance trained

(a) Test accuracy

	0.0	0.005
512	**94.00%**	**94.86%**
8k	89.84%	94.17%

(b) $\log(\mathrm{Tr}(\mathbf{H}(\theta)))$

	0.0	0.005
512	**2.63**	6.28
8k	7.13	**5.99**

(c) $\log(\lambda(\mathbf{H}(\theta)))$

	0.0	0.005
512	**5.50**	8.51
8k	9.30	**8.15**

Table 1: Correlation between sharpness measure and generalization performance of ResNet18 (He et al., 2016) on CIFAR-10 (Krizhevsky, 2009) with the SGD optimizer. In the absence of weight decay (no weight decay for entire network parameter), the sharpness measures tend to lower for better generalization performance. However, in the presence of weight decay (weight decay on entire network parameters), the tendency is reversed.

with adaptive gradient methods such as Adam. As another line of work, Novak et al. (2018) proposed the metric, input-output Jacobian norm of networks, which is shown to be closely related to the model generalizatThe I ion. Based on this observation, Zhang et al. (2019) showed that the weight decay with K-FAC (Martens & Grosse, 2015) regularizes the sensitivity metric, input-output Jabobian norm, which is shown to be closely related to model generalization. Also, the authors find that when the networks with batch normalization (BN) are trained with first-order optimizers, the weight decay play a role in controlling the effective learning rate, thereby bringing regularization effect of gradient noise. Indeed, Neyshabur et al. (2015) showed that the norm-based regularization including the weight decay theoretically reduces the model complexity, which improves the model generalization.

3 Connectivity sharpness

In the first part of this paper, we propose a novel sharpness measure that is the basis of our method in the next section. Toward this, we consider an L-layer multi-layer perceptron (MLP), $f_\theta : \mathbb{R}^{n_0} \to \mathbb{R}^{n_L}$. Here, we denote an input as $x_0 \in \mathbb{R}^{n_0}$, the representation of l-th layer as $x_l \in \mathbb{R}^{n_l}$, and the output of network as $f_\theta(x) = x_L \in \mathbb{R}^{n_L}$. The weight and bias of layer l are $\theta_l \in \mathbb{R}^{n_{l-1} \times n_l}$, $b_l \in \mathbb{R}^{1 \times n_l}$, respectively, and we denote the concatenation of all parameters in a single vector as $\theta \in \mathbb{R}^p$ where $p = \sum_{l=1}^{L}(n_{l-1}+1) \times n_l$. We also define pre-activation of each layer as $h_l \in \mathbb{R}^{n_l}$. With this notation, our MLP satisfies

$$x_l := \sigma(h_l), \quad h_l := x_{l-1}^\top \theta_l + b_l, \quad \forall l = 1, \ldots, L. \tag{1}$$

Here, we place an assumption that the activation function $\sigma(\cdot)$ in each layer is homogeneous for ease of analysis, i.e.,

$$\sigma(a) = \sigma'(a) \odot a, \quad \forall a \in \mathbb{R}. \tag{2}$$

This homogeneous property is satisfied in linear, rectified linear unit (ReLU), and Leaky ReLU. Note that this homogeneous property is widely adopted in practical works including VGG (Simonyan & Zisserman, 2015), ResNet (He et al., 2016). Finally, we solve the following empirical risk minimization (ERM) problem:

$$\min_\theta \mathcal{L}(\theta; \mathcal{D}) := \min_\theta \frac{1}{m} \sum_{i=1}^{m} \ell^{(i)}(\theta) := \min_\theta \frac{1}{m} \sum_{i=1}^{m} \ell\left(\theta; \left(x^i, y^i\right)\right) \tag{3}$$

where $\mathcal{D} = \{(x^i, y^i)\}_{i=1}^m$ is a set of all input-output pairs, and $\ell(\cdot)$ is the loss function of single pair.

3.1 Definition of connectivity sharpness

Sharpness/Flatness of a minimizer is commonly defined in terms of Hessian of objective in equation 3 (Hochreiter & Schmidhuber, 1997; Keskar et al., 2017). Well known sharpness metrics of a minimizer are trace of Hessian, $\mathrm{Tr}(\mathbf{H}(\theta)) := \mathrm{Tr}(\nabla_\theta^2 \mathcal{L})$ (Dinh et al., 2017; Jastrzebski et al., 2021), and maximum eigenvalue of Hessian, $\lambda(\mathbf{H}(\theta))$ (Keskar et al., 2017). While these definitions of sharpness of minima properly identify the curvature of loss function, there are two difficulties when dealing with Hessian : (1) Computation cost of Hessian matrix is prohibitively expensive for state-of-the-art models in computer vision and natural language understanding (Brown et al., 2020; Radford et al., 2021), (2) Hessian is not invariant to the reparameterizations of equivalent network (Dinh et al., 2017).

A well-known problem of parameterization dependent sharpness measure is that they fail in the presence of weight decay. Li et al. (2018) showed that parameterization dependent visualization of loss landscape results in a critical contradiction: a small-batch trained model whose test accuracy is better shows sharper minima than large-batch trained model. We verify that similar results occur in sharpness measures; Table 1b and 1c show $\text{Tr}(\mathbf{H}(\theta))$ and $\lambda(\mathbf{H}(\theta))$ with varying batch size and weight decay. While test accuracy of small-batch trained model consistently better than large-batch trained model, $\text{Tr}(\mathbf{H}(\theta))$ and $\lambda(\mathbf{H}(\theta))$ show inconsistent tendency: with weight decay, large-batch trained model shows better sharpness metrics.

To complement other key sharpness metrics with this issue, we propose yet another sharpness measure, *Connectivity sharpness*, as follows:

$$\mathbf{C}(\theta) := \mathbb{E}_{\mathcal{D}}\left[\left\|\frac{\partial \ell_\theta}{\partial c^\top}\right\|_1\right] = \mathbb{E}_{\mathcal{D}}\left[\left\|\frac{\partial \ell}{\partial \theta^\top} \odot \theta\right\|_1\right] = \mathbb{E}_{\mathcal{D}}\left[\sum_{i=1}^{p}\left|\lim_{\epsilon\to 0}\frac{\ell(\theta+\epsilon\cdot\theta_i)-\ell(\theta)}{\epsilon}\right|\right]. \tag{4}$$

If we compute equation 4 with mini-batch loss $\frac{1}{m}\sum_{i=1}^{m}\ell^{(i)}(\theta)$ instead of single-data loss $\ell^{(i)}(\theta)$, we denote it as $\mathbf{C}_m(\theta)$.

Interpretation of connectivity sharpness To develop a reparameterization invariant sharpness measure, we focus rather on the sensitivity of connection for each parameter. For this, we re-write the equation 3 using auxiliary variable c:

$$\mathcal{L}(c\odot\theta;\mathcal{D}) = \frac{1}{m}\sum_{i=1}^{m}\ell_\theta(c;(x^i,y^i)) := \frac{1}{m}\sum_{i=1}^{m}\ell(c\odot\theta;(x^i,y^i)). \tag{5}$$

This loss function is mainly studied in network pruning literatures (Lee et al., 2019b; Wang et al., 2020; Lee et al., 2020) given that $c \in \{0,1\}^p$. In this case, the *effect of removing* j-th parameter given all connection is activated ($c = \mathbf{1} = (1,\dots,1)$) can be defined as

$$\Delta\mathcal{L}(\theta;\mathcal{D}) = \mathcal{L}(\mathbf{1}\odot\theta;\mathcal{D}) - \mathcal{L}((\mathbf{1}-e_j)\odot\theta;\mathcal{D}) \tag{6}$$

where $e_j \in \mathbb{R}^p$ is one-hot vector with j-th component. Since the computation of effect separately requires prohibitively many computations, they measure *connection sensitivity* (Lee et al., 2019b) as a continuous relaxation of effect for all parameters. This is a gradient of objective function 5 with respect to the connections:

$$\Delta\mathcal{L}(\theta;\mathcal{D}) \approx \nabla_c\mathcal{L}(c\odot\theta)\big|_{c=\mathbf{1}} = \mathbb{E}_{\mathcal{D}}\left[\frac{\partial\ell(c\odot\theta)}{\partial(c\odot\theta)^\top}\frac{\partial(c\odot\theta)^\top}{\partial c}\right]\Bigg|_{c=\mathbf{1}} = \mathbb{E}_{\mathcal{D}}\left[\frac{\partial\ell}{\partial\theta^\top}\odot\theta\right]. \tag{7}$$

Therefore, $\mathbf{C}(\theta)$ can be interpreted as a summation of independent pruning simulation effect of loss (measured by absolute value of $\theta_i \cdot \partial\ell/\partial\theta_i$) for all individual parameters by definition.

It is also worthwhile to note that $\mathbf{C}(\theta)$ is a ℓ_1 norm of gradient rescaled by the parameter coordinate-wisely. Similar rescaling-based measure was proposed in Liang et al. (2019), called Fisher-Rao (FR) metric:

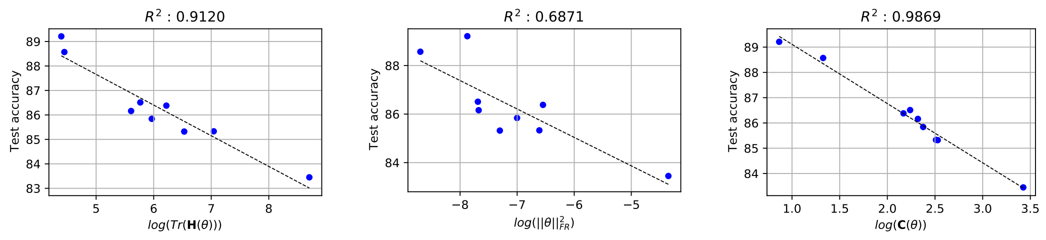
$$\|\theta\|_{\text{FR}}^2 := \theta^\top\mathbf{F}(\theta)\theta = \mathbb{E}_{\mathcal{D}}\left[\theta^\top\frac{\partial\log p_\theta(y|x)}{\partial\theta^\top}\frac{\partial\log p_\theta(y|x)}{\partial\theta}\theta\right] \tag{8}$$

where $\mathbf{F}(\theta)$ is Fisher Information Matrix (FIM)[1] (Amari, 1998) . Liang et al. (2019) shows (i) FR metric is invariant to the network parameterization, and (ii) FR metric is closely related to other generalization metric such as Rademacher complexity and norm-based capacity measures (Neyshabur et al., 2015). One big difference between FR metric and the connectivity sharpness is the existent of bias: FR metric assumes MLP without bias, while we consider biases in equation 1. We will show in Section 3.2 that this seemingly minor difference in assumptions is critical in their correlation with the generalization performance.

In the following subsections, we study the answers to the fundamental questions on proposed connectivity sharpness:

- Is connectivity sharpness a predictive measure of generalization as $\text{Tr}(\mathbf{H}(\theta))$ or $\lambda(\mathbf{H}(\theta))$?
- Is connectivity sharpness invariant to network reparameterization?
- Is there any other helpful property of connectivity sharpness analyzing networks?

[1]Note that the distribution of label y is model's prediction in case of FIM. While this distribution is not identical to the data distribution of ERM, Thomas et al. (2020); Jastrzebski et al. (2021) showed that $\text{Tr}(\nabla_\theta^2\mathcal{L}) \simeq \text{Tr}(\mathbf{F})$ along the optimization trajectory.



(a) Trace of Hessian **(b)** Fisher-Rao Metric **(c)** Connectivity Sharpness

Figure 1: Sharpness-Generalization plots of VGG11 (Simonyan & Zisserman, 2015) trained on CIFAR-10. To measure the correlation between sharpness and generalization, we compute the R^2 score between $\log(\text{sharpness})$ and test accuracy. We found all three sharpness measures have negative correlations with generalization performance. Results on other metrics, datasets, and network architectures are presented in Appendix B.

3.2 Properties of Connectivity Sharpness

Sharpness-Generalization plot To verify connectivity sharpness is a proper measure for sharpness and generalization, we examine the correlation between sharpness metrics ($\text{Tr}(\mathbf{H}(\theta))$, $\|\theta\|_{\text{FR}}$, $\mathbf{C}_m(\theta)$) and generalization performance for networks trained with various hyperparameters. For this, we first train a VGG11 (Simonyan & Zisserman, 2015) on CIFAR-10 (Krizhevsky, 2009) with batch size of 8k for 200 epochs with fixed hyperparameters: weight decay (Loshchilov & Hutter, 2019) as 0.01, the peak learning rate of LAMB optimizer (You et al., 2020) as 0.02 , and no bias decay as in Goyal et al. (2017). These hyperparameters are widely used in large batch optimization. We also warm-up the learning rate for 30% of initial optimization step to stabilize the learning. Then we train networks with eight ablative settings: batch size $\{512, 2048\}$, weight decay $\{0.1, 0.001\}$, peak learning rate $\{0.05, 0.01\}$, and bias decay $\{0.1, 1.0\}$. We plot test accuracy and sharpness measures for networks and compute R^2 scores of correlation between sharpness measure and test accuracy. We find that $\log(\text{sharpness})$ measure shows more clear tendency than naïve sharpness measure. Results are shown in Figure 1.

Since the sharpness measure lead to generalization degradation, $\text{Tr}(\mathbf{H}(\theta))$, $\|\theta\|_{\text{FR}}$, and $\mathbf{C}_m(\theta)$ show negative correlations on generalization performance. We found that $\|\theta\|_{\text{FR}}$ shows less clear relation between test accuracy than $\text{Tr}(\mathbf{H}(\theta))$ and $\mathbf{C}_m(\theta)$. On the other hand, the connectivity sharpness shows clear correlation as much as $\text{Tr}(\mathbf{H}(\theta))$, but much more efficiently without expensive Hessian computation. The results on CIFAR-100 and VGG-19 with Batch Normalization (Ioffe & Szegedy, 2015) are presented in Appendix B.

Reparameterization invariance As explained in Dinh et al. (2017), one possible explanation on the failure of parameter sharpness is its reprameterization dependent property. One can define reparameterization $\rho_{c,l,k}(\cdot) : \mathbb{R}^p \to \mathbb{R}^p$ of MLP formally as follows:

$$\rho_{c,l,k}(\theta) = \begin{cases} c(\theta_{l'})_{ij} & \text{, if } l' = l, j = k \\ (\theta_{l'})_{ij}/c & \text{, if } l' = l+1, i = k \text{ for } \theta, \\ (\theta_{l'})_{ij} & \text{, otherwise} \end{cases} \quad \begin{cases} c(b_{l'})_j & \text{, if } l' = l, j = k \\ (b_{l'})_j & \text{, otherwise} \end{cases} \text{for } b \tag{9}$$

where $c > 0$ is a constant. Note that this reparameterization amplifies c times for incoming weights and bias of j-th component of pre-activation h_l and scales down c for outgoing weights. Therefore, this reparameterization only controls pre-activation $(h_l)_k$ but maintaining $f_\theta(x) = f_{\rho_{c,l,k}(\theta)}(x)$ for any input. We now prove that connectivity sharpness is invariant to $\rho_{c,l,k}(\cdot)$.

Theorem 3.1. *For MLP defined as equation 1, following equation holds for arbitrary $c > 0, l = 1, \dots, L, k = 1, \dots, n_l$:*

$$\mathbf{C}(\theta) = \mathbf{C}(\rho_{c,l,k}(\theta)) \tag{10}$$

Proof. The high level idea of proof is proving connection sensitivity $\partial\ell/\partial\theta \odot \theta$ is invariant to $\rho_{c,l,k}(\cdot)$. We delay the full proof to Appendix C due to the space constraints. □

A simple understanding of reparameterization invariance of connectivity sharpness is following: As mentioned in Section 3.1, connectivity sharpness can be interpreted as a *summation of independent pruning simulation effect for all individual parameters*. Since this effect is measured in terms of the loss function not parameterization, all equivalent networks result in identical summation.

Substructure separability In addition to predictive performance for generalization and reparamerization invariance of connectivity sharpness, we highlight the *substructure separability* of connectivity sharpness. Since the connectivity sharpness is computed as an ℓ_1 norm of connection sensitivity, it is linearly separable to the substructures: that is, the summation of connectivity sharpness of sub-networks is equal to the total connectivity sharpness of entire network:

$$\mathbf{C}((\theta, \theta')) = \mathbf{C}(\theta) + \mathbf{C}(\theta') \tag{11}$$

where (θ, θ') is a concatenation of θ and θ' to a single vector. Since one can decompose the sharpness of entire network as a summation of layer-wise sharpness, this property can be used to determine at which layer of the network sharpness occurs. In Section 4, we apply this property to devise a way to reduce connectivity sharpness and thus improve generalization performance in large batch optimization.

4 Regularizing connectivity sharpness with adaptive β decay

In the previous section, we experimentally examined that connectivity sharpness can be an important factor reflecting generalization performance. Then the question naturally arises: how to reduce this connectivity sharpness to obtain performance improvement. In the rest of the paper, we propose new parameter decaying strategies that can reduce the connectivity sharpness of networks than the 'no bias decay' heuristic, which recommends the bias parameters and γ and β in BN layers are left unregularized in training, in the large batch optimization. Before starting discussion, let us define a normalization layer $\mathrm{N}(\cdot) : \mathbb{R} \to \mathbb{R}$ as

$$\mathrm{N}(x) := \gamma \cdot \frac{x - \mu(x)}{\sigma(x)} + \beta \tag{12}$$

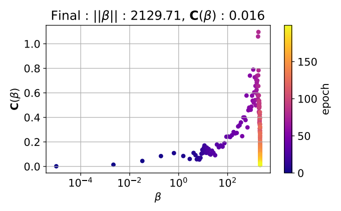
where γ, β is learnable scaling, translation parameters. Note that this normalization layer can be converted to typical weight and bias parameters in MLP by fixing $\mu(x), \sigma(x)$ as accumulated statistics after training with moving averaged value during training.

4.1 β only decay

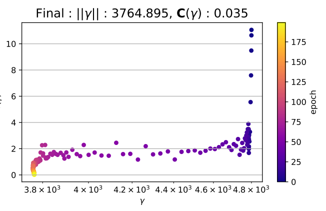
In this subsection, we experimentally find that by simply decaying β only (with γ still unregularized) of the BN layer, the connectivity sharpness of the entire network in a large batch optimization can be drastically reduced and the performance can be greatly improved. Below, we present indirect evidence through a step-by-step experiment on how β only decaying can bring down the connectivity sharpness of the entire network:

- β decay can reduce $\mathbf{C}(\beta)$.
- $\mathbf{C}(\gamma)$ can be reduced when $\mathbf{C}(\beta)$ is reduced
- $\mathbf{C}(\theta)$ can be reduced when $\mathbf{C}(\beta)$ and $\mathbf{C}(\gamma)$ is reduced.

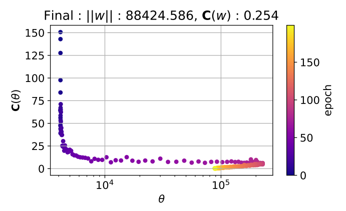
β decay can reduce $\mathbf{C}(\beta)$ Considering the definition of connectivity sharpness in equation 4, we can see that reducing the absolute value of the parameter can contribute to reducing the corresponding connectivity sharpness. To this end, we study how applying decay to two learnable parameters β and γ in the BN layer, along with the convolution parameters to which weight decay is already applied, affects the values of their corresponding $\mathbf{C}(\beta)$ and $\mathbf{C}(\gamma)$. As a typical example, we empirically validate this with ResNet18 (He et al., 2016) trained on CIFAR-10 dataset. By comparing Figure 2 (evolution of parameters and corresponding $\mathbf{C}$ for no-bias decayed network) and Figure 3 (evolution of parameters and corresponding $\mathbf{C}$ for weight decayed network), it can be confirmed that $\mathbf{C}(\beta)$ can be indeed reduced by this simple decay regularizer for the β of BN layer. It can also be confirmed that this regularization effect of weight decay on $\mathbf{C}$ does not clearly appear for γ or



(a) β vs. $\mathbf{C}(\beta)$

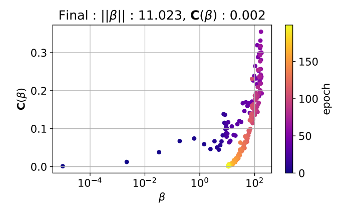


(b) γ vs. $\mathbf{C}(\gamma)$

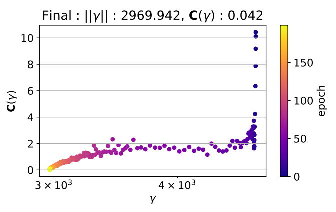


(c) w vs. $\mathbf{C}(w)$

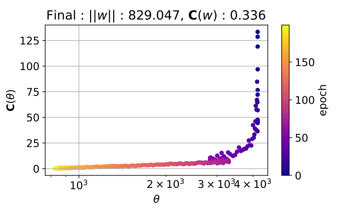
Figure 2: Evolution of parameters and corresponding $\mathbf{C}$ for *no-bias decayed* ResNet18 trained on CIFAR-10 dataset with 8k batch and LAMB optimizer.



(a) β vs. $\mathbf{C}(\beta)$



(b) γ vs. $\mathbf{C}(\gamma)$



(c) w vs. $\mathbf{C}(w)$

Figure 3: Evolution of parameters and corresponding $\mathbf{C}$ for ResNet18 with *weight decay* on corresponding parameter on CIFAR-10 dataset with 8k batch and LAMB optimizer. (a) shows the relationship on β decayed network, (b) shows the relationship on γ-decayed network and (c) shows the relationship on original weight decayed network (apply weight decay on all parameters). Note that in all three cases, the weight decay on other parameters except β, γ is applied following the standard.

other convolution parameters. Here, since no-bias decayed network already applies weight decay for convolution parameters, we strengthen the weight decay coefficient from 0.01 to 1.0 in this case.

If we focus on the trajectory of β and explain it a little more specifically, these bias parameters β of the network are initialized as zero (He et al., 2016) or small values (Lee et al., 2019a) at the start of optimization. This means $\mathbf{C}(\beta)$ is extremely small value at the initialization for all layers. As learning progresses, $\mathbf{C}(\beta)$ grows because the magnitude of β increases from (almost) zero to non-zero value (*progressively sharpening*). This increase in $\mathbf{C}(\beta)$ occurs until the scale of β stabilized. After sufficient training, the scale of β changes a little and $\mathbf{C}(\beta)$ decreases since the gradient scale of β becomes smaller by learning (*progressively flattening*).

$\mathbf{C}(\gamma)$ can be reduced when $\mathbf{C}(\beta)$ is reduced While γ decay does not directly reduce the sharpness of γ in the previous experiments (in Figure 2b and Figure 3b), we show a rough connection that a decrease in $\mathbf{C}(\beta)$ by β decay can lead to a decrease in $\mathbf{C}(\gamma)$. Toward this, we first evaluate the connectivity sharpness of normalization layer. The common idea in normalization layers is *scale invariance* of output and loss:

$$\mathrm{N}(cx) = \mathrm{N}(x) \Rightarrow \ell(\mathrm{N}(cx)) = \ell(\mathrm{N}(x)), \quad \forall c > 0. \tag{13}$$

Note that the definition of x (e.g. activations of single instance for LayerNorm) determines what normalization layer we use. From equation 13, the following holds:

$$x^\top \frac{\partial \ell(\mathrm{N}(x))}{\partial x} = \frac{\partial \ell(\mathrm{N}(cx))}{\partial c} = \frac{\partial \ell(\mathrm{N}(x))}{\partial c} = 0. \tag{14}$$

This property was first mentioned in (Ioffe & Szegedy, 2015) and studied in (Hoffer et al., 2018; Arora et al., 2019). If activation-wise normalization layer (apply equation 12 for each activation) is applied after l-th layer, we obtain

$$0 = (x_l)_i \left(\frac{\partial \ell(\mathrm{N}(x_l))}{\partial x_l}\right)_i = \sigma\left(h_l\right)_i \left(\frac{\partial \ell(\mathrm{N}(x_l))}{\partial x_l}\right)_i \tag{15}$$

$$= (x_{l-1}(\theta_l)_{\cdot i} + (b_l)_i) \left(\frac{\partial \ell(\mathrm{N}(x_l))}{\partial h_l}\right)_i = (\theta_l)_{\cdot i} \left(\frac{\partial \ell(\mathrm{N}(x_l))}{\partial \theta_l}\right)_i + (b_l)_i \left(\frac{\partial \ell(\mathrm{N}(x_l))}{\partial b_l}\right)_i \tag{16}$$

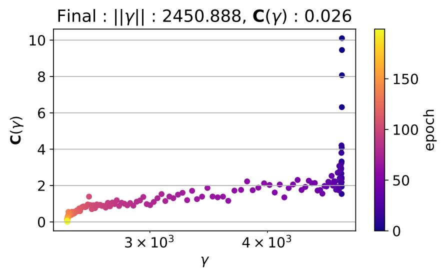


Figure 4: γ vs. $\mathbf{C}(\gamma)$ for β decayed ResNet18 on CIFAR-10 dataset with 8k batch and LAMB optimizer.

Table 2: Effect of β decay on $\mathbf{C}$ of each sub-structure of ResNet18. We use CIFAR-10 dataset with 8k batch and LAMB optimizer.

	No bias decay	β decay (1.0)
Conv	2.74e-1	1.80e-1
γ	3.88e-2	1.87e-2
β	1.81e-2	1.39e-3
FC (w)	3.32e-3	3.56e-3
FC (b)	2.00e-5	1.14e-7
Total	3.34e-1	2.04e-1

Table 3: Pitfalls of no bias decay heuristic. Experiments are on ResNet18 for CIFAR-10/100 datasets and ResNet50 for ImageNet, respectively. In the training of all three datasets, the LAMB optimizer is used. Hyphen (-) is used to indicate the training failure. We found that β decay can effectively improve the generalization performance on all three datasets. On small datasets (CIFAR-10/100), all the other options are better than no bias decay heuristic.

	CIFAR-10 (8k)	CIFAR-100 (8k)	ImageNet (32k)
no bias decay	94.49%	76.77%	76.56%
BN frozen	94.64%	77.59%	-
γ frozen	94.47%	76.99%	76.57%
γ decay (0.01)	94.46%	77.59%	76.62%
β frozen	94.87%	77.81%	-
β decay (1.0)	**94.90%**	**77.99%**	**76.97%**

for all $i = 1, \ldots, n_l$ by the homogeneous property of activation. Therefore, the summation of connection sensitivity of all weights and biases incoming each activation of l-th layer is zero. In practice, stacking normalization layer interleaving with affine transformation is widely used (He et al., 2016; Vaswani et al., 2017). In this case, equation 15 is equivalent to :

$$(\gamma_l)_i \left(\frac{\partial \ell(\mathrm{N}(x_l))}{\partial \gamma_l} \right)_i + (\beta_l)_i \left(\frac{\partial \ell(\mathrm{N}(x_l))}{\partial \beta_l} \right)_i = 0, \quad \forall i, \ldots, n_l \tag{17}$$

Therefore, the connectivity sharpness of γ_l and β_l are equal. While equation 17 holds only for stacking activation-wise normalization layers, we empirically evaluate that β decay regularizes $\mathbf{C}(\gamma)$ in practice. Figure 4 plots γ vs. $\mathbf{C}(\gamma)$ relation for β decayed network (same network considered in Figure 3a). Comparing it to Figure 2b, one can confirm that β decay results scales down γ for 35% (from 3764.895 to 2450.888) and $\mathbf{C}(\gamma)$ for 25% (from 0.035 to 0.026).

$\mathbf{C}(\theta)$ can be reduced when $\mathbf{C}(\beta)$ and $\mathbf{C}(\gamma)$ is reduced. As confirmed in Figure 3a and Figure 4, β decay regularizes $\mathbf{C}(\beta)$ and $\mathbf{C}(\gamma)$. Since connectivity sharpness of BN layer is $\mathbf{C}(\beta) + \mathbf{C}(\gamma)$ by substructure separability in equation 11, β decay regularizes entire BN layer. To extend this regularization effect for entire network, we refer an empirical observation of Tanaka et al. (2020) on gradient structure of MLP that each layer has similar scale. Based on this, we hypothesize that regularization effect of β decay for BN layer might spread to the entire parameters. To empirically verify this, we present connectivity sharpness of sub-structures of two networks (no bias decay, β decay ResNet18 of CIFAR-10 in previous paragraph) in Table 2. From Table 2 we can see that regularization effect of β decay appears across the entire network in terms of connectivity sharpness.

Finally, we make some remarks on the role of β in large batch optimization. According to the Table 3, freezing β or BN layer (β as 0 and γ as 1) possibly hinder the convergence in a large batch regime while manipulating γ has no significant effect. This implies that the bias parameters really matter in large batch optimization, so we can conclude that the *no bias decay* heuristic stabilizes the training with larger β. However, this stabilization sacrifices generalization performance as shown in results of freezing β or BN on CIFAR-10/100. Hence, one can conclude that no bias decay heuristic often results inferior generalization with useless γ, β.

Table 4: Test accuracy of adaptive β decay. We use ResNet18 on CIFAR-10/100 datasets with the LAMB optimizer. First row indicates the batch size. We found that the fixed β decay and adaptive β decay are better than no bias decay heuristic (baselines). Furthermore, the adaptive β decay nearly restores the performances on small batches.

	512	2k	8k	8k w. β decay (1.0)	8k w. adaptive β decay
CIFAR-10	95.12%	94.59%	94.49%	94.90%	**95.01%**
CIFAR-100	78.09%	78.03%	76.77%	77.99%	**78.11%**

Table 5: Test accuracy of of adaptive β decay. We use ResNet50-D (He et al., 2019b) on ImageNet dataset with the LAMB optimizer. We found that fixed β decay is better than no bias decay and adaptive β decay is better than fixed β decay. Interestingly, the adaptive β decay is superior to the performance on small batch (1k).

1k	32k	32k w. β decay (1.0)	32k w. adaptive β decay
77.16%	76.56%	76.97%	**77.24%**

4.2 Adaptive β decay

From Section 4.1, we can hypothesize that the difficulty of large optimization problem is affected by the utilization of bias parameters: (i) In *no bias decay*, the network allows the bias parameters to be completely free and shows decent performance. On the other hand, (ii) in case of *β-frozen/BN frozen*, the network cannot utilize the bias parameters at all, hence the optimization under this constraint becomes hard and it diverges.

In this sense, although the β decay is effective in practice, each layer's utilization of the bias parameters is still not fully considered. Since the connectivity sharpness $\mathbf{C}(\beta)$ is a relative quantity across layers, more accurate metric is required to measure "how much each layer exploits the bias parameters". To this end, motivated by the proportionality of $\mathbf{C}(\beta)$ and $\mathbf{C}(\gamma)$ in equation 17, we devise a simple layer-wise metric r_l that can be used to evaluate the bias utilization of each layer l as $r_l := \mathbf{C}(\beta_l)/\mathbf{C}(\gamma_l)$.

Interpretation of r_l As mentioned in Section 3.1, the connectivity sharpness of each parameter can be interpreted as a sensitivity of loss function to losing the corresponding parameter. Therefore, shrinking β of BN layers with large r_l affects the loss function more severely than layers with small r_l. This means that the layers with large r_l value are less dependent to the input data than those with small r_l because the parameter β has much less correlation with the input data than γ according to the equation 12 (note that γ is multiplied to the input data directly). Hence, if a normalization layer achieves low loss value with high r_l, it can be thought that the optimization proceeds without considering the input data enough, so the parameters might converge to sub-optimal solutions.

Based on this interpretation, we can say that the layers with large r_l tend to use bias for cheating (try to find shortcuts by making the problem easier using bias). Thus we suggest to take weight decay to those layers only since the layers with small r_l already fully exploit the bias while having low connectivity sharpness $\mathbf{C}(\beta_l)$ (and hence have low $\mathbf{C}(\gamma_l)$ as discussed in Section 4.1). To make such selective use of weight decay, we propose the **adaptive** β **decay** that takes weight decay to the layers only with $r_l \geq R$ where $R \in \mathbb{R}$ is a hyperparameter. In the optimization with adaptive β decay, we calculate r_l for each layer l at every optimization step and apply the weight decay to the layers only with $r_l \geq R$. Algorithmic details are presented in Appendix D. As shown in the Table 4 and 5, we verify that the adaptive β decay can effectively leverage the generalization performance on large batch regimes.

5 Conclusion

Large batch optimization is a key technique to fully utilize the huge computational resource. To extend the toolbox of analyzing generalization of large scale models, we proposed a reparameterization invariant sharpness measure, connectivity sharpness. Since connectivity sharpness can be computed with only gradient information, it can be easily adapted to analyze sharpness of large scale models. Based on connectivity sharpness, we found that (i) unregularized bias in normalization layer sharpens model and (ii) this can be fixed with (adaptive) β decay. By using adaptive β decay, we improved top-1 accuracy of ResNet50 from 76.56% to 77.24% in ImageNet training with batch size of 32k.

References

Shun-Ichi Amari. Natural gradient works efficiently in learning. *Neural computation*, 10(2):251–276, 1998.

Sanjeev Arora, Zhiyuan Li, and Kaifeng Lyu. Theoretical analysis of auto rate-tuning by batch normalization. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=rkxQ-nA9FX.

Tom B Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*, 2020.

Jeremy Cohen, Simran Kaur, Yuanzhi Li, J Zico Kolter, and Ameet Talwalkar. Gradient descent on neural networks typically occurs at the edge of stability. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=jh-rTtvkGeM.

Laurent Dinh, Razvan Pascanu, Samy Bengio, and Yoshua Bengio. Sharp minima can generalize for deep nets. In *International Conference on Machine Learning*, pp. 1019–1028. PMLR, 2017.

Pierre Foret, Ariel Kleiner, Hossein Mobahi, and Behnam Neyshabur. Sharpness-aware minimization for efficiently improving generalization. *arXiv preprint arXiv:2010.01412*, 2020.

Priya Goyal, Piotr Dollár, Ross Girshick, Pieter Noordhuis, Lukasz Wesolowski, Aapo Kyrola, Andrew Tulloch, Yangqing Jia, and Kaiming He. Accurate, large minibatch sgd: Training imagenet in 1 hour. *arXiv preprint arXiv:1706.02677*, 2017.

Haowei He, Gao Huang, and Yang Yuan. Asymmetric valleys: Beyond sharp and flat local minima. *Advances in Neural Information Processing Systems*, 32:2553–2564, 2019a.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

Tong He, Zhi Zhang, Hang Zhang, Zhongyue Zhang, Junyuan Xie, and Mu Li. Bag of tricks for image classification with convolutional neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 558–567, 2019b.

Sepp Hochreiter and Jürgen Schmidhuber. Flat minima. *Neural computation*, 9(1):1–42, 1997.

Elad Hoffer, Ron Banner, Itay Golan, and Daniel Soudry. Norm matters: efficient and accurate normalization schemes in deep networks. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, pp. 2164–2174, 2018.

Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pp. 448–456. PMLR, 2015.

Stanislaw Jastrzebski, Maciej Szymczak, Stanislav Fort, Devansh Arpit, Jacek Tabor, Kyunghyun Cho*, and Krzysztof Geras*. The break-even point on optimization trajectories of deep neural networks. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=r1g87C4KwB.

Stanislaw Jastrzebski, Devansh Arpit, Oliver Astrand, Giancarlo B Kerg, Huan Wang, Caiming Xiong, Richard Socher, Kyunghyun Cho, and Krzysztof J Geras. Catastrophic fisher explosion: Early phase fisher matrix impacts generalization. In *International Conference on Machine Learning*, pp. 4772–4784. PMLR, 2021.

Tyler Johnson, Pulkit Agrawal, Haijie Gu, and Carlos Guestrin. Adascale sgd: A user-friendly algorithm for distributed training. In *International Conference on Machine Learning*, pp. 4911–4920. PMLR, 2020.

Nitish Shirish Keskar, Jorge Nocedal, Ping Tak Peter Tang, Dheevatsa Mudigere, and Mikhail Smelyanskiy. On large-batch training for deep learning: Generalization gap and sharp minima. In *5th International Conference on Learning Representations, ICLR 2017*, 2017.

A Krizhevsky. Learning multiple layers of features from tiny images. *Master's thesis, University of Toronto*, 2009.

Jungmin Kwon, Jeongseop Kim, Hyunseo Park, and In Kwon Choi. Asam: Adaptive sharpness-aware minimization for scale-invariant learning of deep neural networks. *arXiv preprint arXiv:2102.11600*, 2021.

Jaehoon Lee, Lechao Xiao, Samuel Schoenholz, Yasaman Bahri, Roman Novak, Jascha Sohl-Dickstein, and Jeffrey Pennington. Wide neural networks of any depth evolve as linear models under gradient descent. *Advances in neural information processing systems*, 32:8572–8583, 2019a.

Namhoon Lee, Thalaiyasingam Ajanthan, and Philip Torr. SNIP: SINGLE-SHOT NETWORK PRUNING BASED ON CONNECTION SENSITIVITY. In *International Conference on Learning Representations*, 2019b. URL https://openreview.net/forum?id=B1VZqjAcYX.

Namhoon Lee, Thalaiyasingam Ajanthan, Stephen Gould, and Philip H. S. Torr. A signal propagation perspective for pruning neural networks at initialization. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=HJeTo2VFwH.

Aitor Lewkowycz, Yasaman Bahri, Ethan Dyer, Jascha Sohl-Dickstein, and Guy Gur-Ari. The large learning rate phase of deep learning: the catapult mechanism. *arXiv preprint arXiv:2003.02218*, 2020.

Hao Li, Zheng Xu, Gavin Taylor, Christoph Studer, and Tom Goldstein. Visualizing the loss landscape of neural nets. *Advances in Neural Information Processing Systems*, 31, 2018.

Tengyuan Liang, Tomaso Poggio, Alexander Rakhlin, and James Stokes. Fisher-rao metric, geometry, and complexity of neural networks. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pp. 888–896. PMLR, 2019.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=Bkg6RiCqY7.

James Martens and Roger Grosse. Optimizing neural networks with kronecker-factored approximate curvature. In *International conference on machine learning*, pp. 2408–2417. PMLR, 2015.

Zachary Nado, Justin M Gilmer, Christopher J Shallue, Rohan Anil, and George E Dahl. A large batch optimizer reality check: Traditional, generic optimizers suffice across batch sizes. *arXiv preprint arXiv:2102.06356*, 2021.

Behnam Neyshabur, Ryota Tomioka, and Nathan Srebro. Norm-based capacity control in neural networks. In *Conference on Learning Theory*, pp. 1376–1401. PMLR, 2015.

Roman Novak, Yasaman Bahri, Daniel A. Abolafia, Jeffrey Pennington, and Jascha Sohl-Dickstein. Sensitivity and generalization in neural networks: an empirical study. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=HJC2SzZCW.

Daniel Park, Jascha Sohl-Dickstein, Quoc Le, and Samuel Smith. The effect of network width on stochastic gradient descent and generalization: an empirical study. In *International Conference on Machine Learning*, pp. 5042–5051. PMLR, 2019.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. *arXiv preprint arXiv:2103.00020*, 2021.

Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *International Conference on Learning Representations*, 2015.

Samuel Smith, Erich Elsen, and Soham De. On the generalization benefit of noise in stochastic gradient descent. In *International Conference on Machine Learning*, pp. 9058–9067. PMLR, 2020.

Samuel L. Smith and Quoc V. Le. A bayesian perspective on generalization and stochastic gradient descent. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=BJij4yg0Z.

Samuel L. Smith, Pieter-Jan Kindermans, and Quoc V. Le. Don't decay the learning rate, increase the batch size. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=B1Yy1BxCZ.

Hidenori Tanaka, Daniel Kunin, Daniel L Yamins, and Surya Ganguli. Pruning neural networks without any data by iteratively conserving synaptic flow. *Advances in Neural Information Processing Systems*, 33, 2020.

Valentin Thomas, Fabian Pedregosa, Bart Merriënboer, Pierre-Antoine Manzagol, Yoshua Bengio, and Nicolas Le Roux. On the interplay between noise and curvature and its effect on optimization and generalization. In *International Conference on Artificial Intelligence and Statistics*, pp. 3503–3513. PMLR, 2020.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pp. 5998–6008, 2017.

Chaoqi Wang, Guodong Zhang, and Roger Grosse. Picking winning tickets before training by preserving gradient flow. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=SkgsACVKPH.

Yang You, Igor Gitman, and Boris Ginsburg. Large batch training of convolutional networks. *arXiv preprint arXiv:1708.03888*, 2017.

Yang You, Jing Li, Sashank Reddi, Jonathan Hseu, Sanjiv Kumar, Srinadh Bhojanapalli, Xiaodan Song, James Demmel, Kurt Keutzer, and Cho-Jui Hsieh. Large batch optimization for deep learning: Training bert in 76 minutes. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=Syx4wnEtvH.

Guodong Zhang, Chaoqi Wang, Bowen Xu, and Roger Grosse. Three mechanisms of weight decay regularization. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=B1lz-3Rct7.

A Experimental Settings

In this appendix, we provide the detailed settings of experiments.

A.1 Learning Rate Scheduling

For the LAMB optimizer, we use the polynomial learning rate scheduling as in Nado et al. (2021) with $p_{warmup} = p_{decay} = 2$ and $\eta_{init} = \eta_{final} = 0.0$. Various η_{peak} are used according to the dataset, network architecture and batch size. We use 30% of initial optimization step for learning rate warm-up. We use 0.9 and 0.999 for the first/second order momentum hyperparameter of LAMB optimizer. For the SGD optimizer, we use the standard cosine scheduling (He et al., 2019b) and linear warm-up with 10 epochs.

A.2 Detailed Configurations

We train the ResNet18 / VGG11 / VGG19-BN networks on CIFAR-10/100 and ResNet50-D (He et al., 2019b) network on ImageNet with LAMB optimizer. We use the peak learning rate $\eta_{peak} = 0.04$ for the batch size 8k on CIFAR-10/100, the same peak learning rate for the batch size 32k on ImageNet. For these base peak learning rates, we use square root learning rate scaling : $\eta_{peak} =$

$0.04 \times \sqrt{\text{batch size}/8\text{k (or 32k)}}$. We use weight decay coefficient 0.01 for non-normalization layers. We use $R = 0.1$ for the criterion of adaptive β decay.

When we train the ResNet18 with SGD optimizer (Table **??**), we use the learning rate $\eta = 0.1$ and momentum 0.9. Other hyperparameters are specified in description of the tables and figures.

Other minor details are as follows :

- For CIFAR-10/100, we apply the standard data augmentation for training : `RandomCrop(32, padding=4)` and `RandomHorizontalFlip()`. We train the network for 200 epochs.
- For ImageNet, we apply the standard data augmentation for training : `RandomResizeCrop(224)` and `RandomHorizontalFlip()`, for evaluation : `Resize(256)` and `CentorCrop(224)`. We train the network for 90 epochs.
- We use the ghost batch normalization layer as in Hoffer et al. (2018) with ghost batch size 32.
- For each dataset, we use the standard data split.

B Additional Sharpness-Generalization experiments

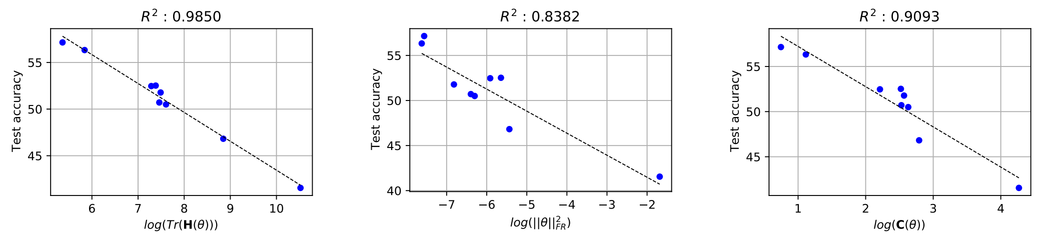


Figure 5: Sharpness-Generalization plot results on VGG11 network trained on CIFAR-100.

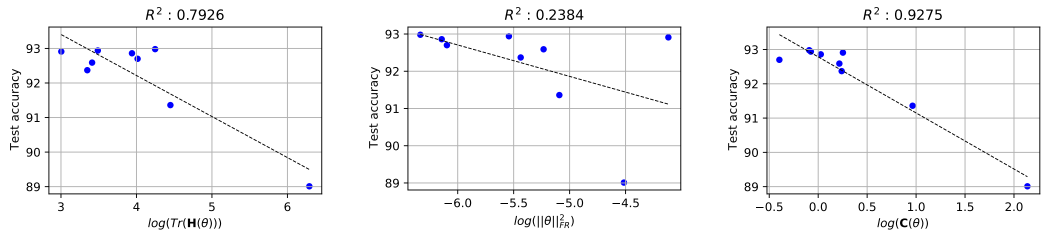


Figure 6: Sharpness-Generalization plot results on VGG19-BN network trained on CIFAR-10.

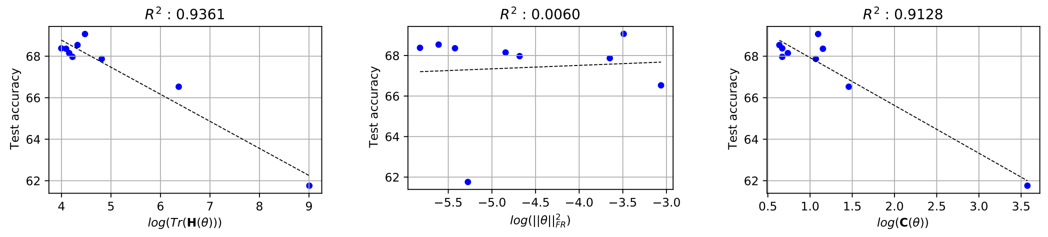


Figure 7: Sharpness-Generalization plot results on VGG19-BN network trained on CIFAR-100.

C Reparameterization invariance of connectivity sharpness proof

Theorem C.1. *For MLP defined as equation 1, following equation holds for arbitrary* $c > 0, l = 1, \dots, L, k = 1, \dots, n_l$*:*

$$\mathbf{C}(\theta) = \mathbf{C}(\rho_{c,l,k}(\theta)) \tag{18}$$

Proof. It is sufficient to show that $\partial \ell / \partial \theta \odot \theta$ is invariant to $\rho_{c,l,k}(\dot{c})$ with arbitrary $c > 0, l = 1, \dots, L, k = 1, \dots, n_L$.

- For incoming weights(θ_l) and biases(b_l), we have $\theta \to c \cdot \theta$ for parameter value, $\partial \ell / \partial \theta \to (1/c) \cdot \partial \ell / \partial \theta$ for its corresponding gradients. Note that inversely scaled outgoing weights affect gradient to incoming edges.
- For outgoing weights(θ_{l+1}), we have $\theta \to (1/c) \cdot \theta$ for parameter value, $\partial \ell / \partial \theta \to c \cdot \partial \ell / \partial \theta$ for its corresponding gradients. Note that scaled incoming weights and bias affect gradient to outcoming edges.
- For the other parameters, both parameter value θ and its gradient $\partial \ell / \partial \theta$ do not change.

□

D Detailed Algorithm of Adaptive β decay

Algorithm 1 Adaptive β decay with LAMB (You et al., 2020)

```
Require: Initialized parameter $\theta^0 \in \mathbb{R}^p$, learning rate scheduling $\{\eta^t\}_{t=1}^T$, momentum hyperparameters $\beta^1, \beta^2 \in (0,1)$ (e.g. $\beta^1 = 0.9, \beta^2 = 0.999$), weight decay coefficient for non-normalization layer $\lambda_w$ (e.g. 0.01), weight decay coefficient for $\lambda_\beta > 0$ (e.g. 1.0), criterion for selection of weight decay $R$ (e.g. 0.1).

  Set $m^0 = 0, v^0 = 0$
  for $t = 1, \ldots, T$ do
      Draw mini-batch $\mathcal{B}$ from $\mathcal{D}$.
      Compute mini-batch gradient as $g^t \leftarrow \frac{1}{|\mathcal{B}|} \sum_{b \in \mathcal{B}} \nabla^\theta \ell_{(b)}(\theta^t)$.
      Update first order momentum as $m^t \leftarrow \beta^1 m^{t-1} + (1 - m^t) g^t$.
      Update second order momentum as $v^t \leftarrow \beta^1 v^{t-1} + (1 - v^t) g^t \odot g^t$.
      Compute the ADAM ratio as $r^t \leftarrow m^t / v^t$.
      if layer $l$ is not normalization layer then
          $\lambda_l^t \leftarrow \lambda_w$
      else
          if parameter is $\beta$ then
              if $r_l > R$ then
                  $\lambda_\beta^t \leftarrow \lambda_\beta$
              else
                  $\lambda_\beta^t \leftarrow 0$
              end if
          else
              $\lambda_l^t \leftarrow 0$
          end if
      end if
      Add weight decay effect to ADAM ratio $u^t \leftarrow r^t + \lambda^t \odot \theta^t$
      Update parameter with layer-wise rescaling
          $$\theta_l^{t+1} \leftarrow \frac{\|\theta^t\|}{\|u^t\|} u^t. \quad (19)$$
  end for
```