Low-Pass Filtering SGD for Recovering Flat Optima in the Deep Learning Optimization Landscape

Devansh Bisla
db3484@nyu.edu

Jing Wang
jw5665@nyu.edu

Anna Choromanska
ac5455@nyu.edu

Abstract

In this paper, we study the sharpness of a deep learning (DL) loss landscape around local minima in order to reveal systematic mechanisms underlying the generalization abilities of DL models. Our analysis is performed across varying network and optimizer hyper-parameters, and involves a rich family of different sharpness measures. We compare these measures and show that the low-pass filter based measure exhibits the highest correlation with the generalization abilities of DL models, has high robustness to both data and label noise, and furthermore can track the double descent behavior for neural networks. We next derive the optimization algorithm, relying on the low-pass filter (LPF), that actively searches the flat regions in the DL optimization landscape using SGD-like procedure. The update of the proposed algorithm, that we call LPF-SGD, is determined by the gradient of the convolution of the filter kernel with the loss function and can be efficiently computed using MC sampling. We empirically show that our algorithm achieves superior generalization performance compared to the common DL training strategies. On the theoretical front we prove that LPF-SGD converges to a better optimal point with smaller generalization error than SGD.

1 INTRODUCTION

Training a deep network requires finding network parameters that minimize the loss function that is defined as the sum of discrepancies between target data labels and their estimates obtained by the network. This sum is computed over the entire training data set. Due to the multi-layer structure of the network and non-linear nature of the network activation functions, DL loss function is a non-convex function of network parameters. Increasing the number of training data points complicates this function since it increases the number of its summands. Increasing the number of parameters (by adding more layers or expanding the existing ones) increases the complexity of each summand and results in the growth, which can be exponential (Anandkumar et al., 2015), of the number of critical points of the loss function. A typical use case for DL involves massive (i.e., high-dimensional and/or large) data sets and very large networks (i.e., with billions of parameters), which results in an optimization problem that is heavily non-convex and difficult to analyze.

In order to design efficient optimization algorithms for DL we need to understand which regions of the DL loss landscape lead to good generalization and how to reach them. Existing work (Feng and Tu, 2020; Chaudhari et al., 2017; Hochreiter and Schmidhuber, 1997; Jastrzebski et al., 2018; Wen et al., 2018; Keskar et al., 2017; Sagun et al., 2018; Simsekli et al., 2019; Jiang et al., 2020; Dziugaite and Roy, 2017; Jiang et al., 2017; Izmailov et al., 2018; Neyshabur et al., 2017) shows that properly regularized SGD recovers solutions that generalize well and provides an evidence that these solutions lie in wide valleys of the landscape. In the prior studies (Foret et al., 2021; Chaudhari et al., 2017), it was demonstrated that the spectrum of the Hessian at a well-generalizing local minimum of the loss function is often almost flat, i.e., the majority of eigenvalues are close to zero. An intuitive illustration of why flatness potentially leads to better generalization is presented in Figure 1. However, existing studies focus on simplified DL frameworks, small data sets, and/or limited optimization settings and thus are inconclusive. There exists no *comprehensive study* confirming that good generalization abilities of the DL model correlate with the properties of the loss landscape around recovered solutions and some research argues against the existence of this relationship (Dinh et al., 2017). The lack of more comprehensive approaches has kept the

Proceedings of the 25[th] International Conference on Artificial Intelligence and Statistics (AISTATS) 2022, Valencia, Spain. PMLR: Volume 151.

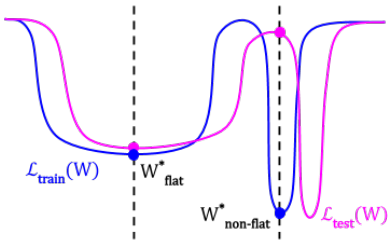

Figure 1: Consider the train and test loss ($\mathcal{L}_{\text{train}}(W)$ and $\mathcal{L}_{\text{test}}(W)$ resp, where W are model parameters), which have similar shape but are shifted with respect to each other. The local minimum that lies in the flat region (W^*_{flat}) admits a similar value of the train and test loss (generalizes well), despite the shift. The local minimum that lies in the narrow region of the landscape ($W^*_{\text{non-flat}}$) admits a significantly larger value of the test loss compared to the train loss (generalizes very poorly), due to the shift between them.

field from moving away from purely generic DL training methodologies, which are not equipped with any mechanisms allowing them to take advantage of the properties of the non-convex loss landscape underlying the DL optimization problem.

In this paper we first empirically confirm that the sharpness of a DL loss landscape around local minima indeed correlates with the generalization abilities of a DL model. In our work we consider network architectures with and without skip connections and batch norm, and with varying widths, and DL optimizer with varying settings of the hyper-parameters, such as the batch size, learning rate, momentum coefficient, and weight decay regularization. To the best of our knowledge, our study is the most comprehensive research study to date of the connection between DL landscape geometry and the generalization abilities of DL models. We then perform comparative studies of different existing sharpness measures in order to identify the most promising one – the LPF based measure – that best promotes good performance of the model and that can be efficiently computed. The identified measure is robust to the presence of the data and label noise and furthermore mirrors the double descent behavior of the test accuracy (Nakkiran et al., 2020; Belkin et al., 2019). We incorporate the LPF based sharpness measure to SGD in order to guide the optimization process towards the well-generalizing regions of the loss landscape, and we show that the resulting algorithm is more accurate than the common baselines, including recently proposed state-of-the-art method SAM (Foret et al., 2021). Finally, we provide convergence guarantee showing that LPF-SGD recovers a better optimal point with smaller generalization error than SGD.

What is not new in this paper? The evidence that well-generalizing solutions lie in wide valleys of the DL loss landscape is not new as well as investigated flatness measures.

What is new in this paper? The contributions of our work compared to prior works include: a) the comprehensiveness of the study, b) the direct comparison of several flatness measures and their levels of correlation to the generalization abilities of the model, c) the novel algorithm superior to SGD and prior flatness-aware techniques, and d) its theoretical and empirical analysis.

This paper is organized as follows: Section 2 reviews the literature, Section 3 studies the sharpness and generalization properties of the local minima of the DL loss function, Section 4 introduces the LPF-SGD algorithm that is analyzed in Section 5, and Section 6 shows experiments. Finally, Section 7 concludes the paper. Supplement contains additional material.

2 RELATED WORK

The loss function in DL is typically optimized using standard or variant forms of SGD (Bottou, 1998), which iteratively updates parameters by taking steps in the anti-gradient direction of the loss function computed for randomly sampled data mini-batches. Various optimization approaches, including adaptive learning rates (Duchi et al., 2011; Kingma and Ba, 2014; Zeiler, 2012; Luo et al., 2019; Shazeer and Stern, 2018; Ginsburg et al., 2019), momentum-based strategies (Qian, 1999; Nesterov, 2005; Tieleman and Hinton, 2012; Hardt et al., 2016), approximate steepest descent methods (Neyshabur et al., 2015), continuation methods (Allgower and Georg, 2012; Mobahi and Fisher III, 2015; Gulcehre et al., 2016), alternatives to standard backpropagation algorithm based on alternating minimization (Carreira-Perpiñán and Wang, 2014; Zhang and Brand, 2017; Zhang and Kleijn, 2017; Askari et al., 2018; Zeng et al., 2018; Lau et al., 2018; Gotmare et al., 2018; Choromanska et al., 2019; Taylor et al., 2016; Zhang et al., 2016b) or TargetProp (LeCun, 1986, 1987; LeCun et al., 1988; Lee et al., 2015; Bartunov et al., 2018), as well as architectural modifications (Srivastava et al., 2014; Ioffe and Szegedy, 2015; Cooijmans et al., 2016; Salimans and Kingma, 2016; Desjardins et al., 2015; Cowen et al., 2019), were proposed to accelerate the training process and improve the generalization performance of the DL models. However there exists no strong evidence that these techniques indeed improve the training of DL models. Moreover, it was shown that well-tuned vanilla SGD often can achieve superior performance compared to mentioned modified techniques (Wilson et al., 2017). In addition, some prior work (Goodfellow and Vinyals, 2015; Sutskever et al., 2013) emphasizes the sensitivity of aforementioned optimizers to initial conditions and learning rates. Finally, theoretical approaches to DL optimization have focused

on the convergence properties of vanilla SGD and consider only shallow models (Brutzkus and Globerson, 2017; Li and Yuan, 2017; Zhong et al., 2017; Tian, 2017; Brutzkus et al., 2018; Li et al., 2018) or deep, but linear ones (Arora et al., 2019). These results do not hold for DL models used in practice.

The shape of the DL loss function is poorly understood. Consequently, the aforementioned generic, rather than landscape-adaptive, optimization strategies are commonly chosen to train DL models, despite their potentially poor fit to the underlying non-convex DL optimization problem. Most of what we know about the DL loss landscape is either based on unrealistic assumptions and/or holds only for shallow (two-layer) networks. The existing literature emphasizes i) the proliferation of saddle points (Dauphin et al., 2014; Baldi and Hornik, 1989; Saxe et al., 2014) (including degenerate or hard to escape ones (Ge et al., 2015; Anandkumar and Ge, 2016; Dauphin et al., 2014)), ii) the equivalency of some local minima (Chaudhari and Soatto, 2015; Haeffele and Vidal, 2015; Janzamin et al., 2015; Kawaguchi, 2016; Soudry and Carmon, 2016; Freeman and Bruna, 2017; Nguyen and Hein, 2017; Vidal et al., 2017; Haeffele and Vidal, 2015; Du and Lee, 2018; Safran and Shamir, 2016, 2017; Hardt and Ma, 2017; Ge et al., 2017; Yun et al., 2018; Draxler et al., 2018; Freeman and Bruna, 2017; Trager et al., 2020; Choromanska et al., 2015a,b), iii) the existence of a large number of isolated minima and few dense regions with lots of minima close to each other (Baldassi et al., 2015, 2016a,b) (shown for shallow networks), and iv) the existence of global and local minima yielding models with different generalization performance (Keskar et al., 2017). Furthermore, it was demonstrated that becoming stuck in poor minima is a major problem only for smaller networks (Goodfellow and Vinyals, 2015).

There exists only a few approaches that aim at adapting the DL optimization strategy to the properties of the loss landscape and encouraging the recovery of flat optima. They can be summarized as i) regularization methods that regularize gradient descent strategy with sharpness measures such as the Minimum Description Length (Hochreiter and Schmidhuber, 1997), local entropy (Chaudhari et al., 2017), or variants of ϵ-sharpness (Foret et al., 2021; Keskar et al., 2017), ii) surrogate methods that evolve the objective function according to the diffusion equation (Mobahi, 2016), iii) averaging strategies that average model weights across training epochs (Izmailov et al., 2018; Cha et al., 2021), and iv) smoothing strategies that smoothens the loss landscape by introducing noise in the model weights and average model parameters across multiple workers run in parallel (Wen et al., 2018; Haruki et al., 2019; Lin et al., 2020) (such methods focus on distributed training of DL models with an extremely large batch size - a setting where SGD and its variants struggle).

3 INTERPLAY BETWEEN SHARPNESS AND GENERALIZATION

In this section we study the correlation between network generalization and the sharpness of the local minima of the DL loss function. Without loss of generality we consider balanced networks (Neyshabur et al., 2015), where the norms of incoming weights to different units are roughly the same (for details see Supplement, Section 8). This assumption is necessary since the properties of the loss landscape of two equivalent (realizing the same function) imbalanced networks can be radically different (e.g., their landscapes can be stretched along different directions), which would lead to inconsistencies in the analysis. The importance of balancing the network is overlooked in existing studies.

All codes for experiments presented in this section as well as Section 6 are available at https://github.com/devansh20la/LPF-SGD.

We consider a broad family of different measures of sharpness: ϵ-sharpness (Keskar et al., 2017), PAC-Bayes measure (Jiang et al., 2020) ($\mu_{PAC-Bayes}$), Fisher Rao Norm (Liang et al., 2019) (FRN), gradient of the local entropy (Chaudhari et al., 2017) (μ_{LE}), Shannon entropy (Pereyra et al., 2017) ($\mu_{entropy}$), Hessian-based measures (Maddox et al., 2020; MacKay, 1992) (the Frobenius norm of the Hessian ($||H||_F$), trace of the Hessian (Trace (H), the largest eigenvalue of the Hessian ($\lambda_{max}(H)$), and the effective dimensionality of the Hessian), and the low-pass filter based measure (LPF). Decreasing the value of any of these measures leads to flatter optima. The definitions of these measures as well as the algorithms numerically computing them are provided in the Supplement (Section 9). Similarly, the Supplement (Section 10) contains the analysis of the sensitivity of these measures to the changes in the curvature of the synthetically generated landscapes (i.e., landscapes, where the curvature of the loss function is known). Below we only provide the definition of the LPF measure as this one constitutes the foundation of our algorithm.

Definition 1 (LPF). *Let $K \sim \mathcal{N}(0, \sigma^2 I)$ be a kernel of a Gaussian filter. LPF based sharpness measure at solution θ^* is defined as the convolution of the loss function with the Gaussian filter computed at θ^*:*

$$(L \circledast K)(\theta^*) = \int L(\theta^* - \tau) K(\tau) d\tau. \tag{3.1}$$

Next, we aim at revealing whether there exists any correlation between sharpness of local minima and the performance of the deep networks and, if so, which

Hyper-parameter	Settings
Momentum	[0.0, 0.5, 0.9]
Width	[32, 48, 64]
Weight decay	[0.0, $1e^{-4}$, $5e^{-4}$]
Learning rate	[$5e^{-3}$, $7.5e^{-3}$, $1e^{-2}$]
Batch size	[32, 128, 512]
Skip connection	[False, True]
Batch norm	[False, True]

Table 1: Choices of hyper-parameters.

flatness measure exhibits the highest correlation with the model generalization.

3.1 Sharpness versus generalization

We train 2916 ResNet18 (He et al., 2016) models on publicly available CIFAR-10 (Krizhevsky et al., 2014a) data set by varying network architecture (width, skip connections (skip), and batch norm (bn)), optimizer hyper-parameters (momentum coefficient(mom), learning rate (lr), batch size (batch)), and regularization (weight decay (wd)) according to the Table 1. For each set of hyper-parameters, we train three models with different seeds. We follow (Dziugaite et al., 2020; Jiang et al., 2020) in terms of stopping criteria, as we focus our analysis on practical models, and train each model until the cross entropy loss reached the value of ≈ 0.01, which corresponds to $\approx 99\%$ training accuracy. Any model that has not reached this threshold was discarded from further analysis. Note that the other stopping criteria, for example these based on #epochs/#iterations, lead to under/over trained models as some models train faster than others. More details of training can be found in the Supplement (Section 11.1). At convergence, when the model approaches the final weights, we compute the Kendall rank correlation coefficient between generalization gap and sharpness measures (averaged over the seeds). Table 2 captures the results along with 95% confidence interval. Note that Table 2 is extremely compute intensive (≈9000 GPU hours), which makes it prohibitive to obtain similar results across multiple data sets and models. In Figure 2 we show a scatter plot of normalized LPF based measure (with the highest overall correlation) and normalized LE based measure (with the lowest overall correlation) with the corresponding generalization gap. Note that our analysis includes models with wide-ranging generalization gap.

Table 2 and Figure 2 clearly reveal that there exist sharpness measures for which one can observe the correlation between the generalization gap and the sharpness of the local minima (Kendall rank correlation coefficient above 0.35 is considered to indicate strong correlation (Botsch, 2011)). We observe that LPF based measure outperforms other sharpness measures in terms of tracking the generalization behavior of the network when we vary weight decay, batch size

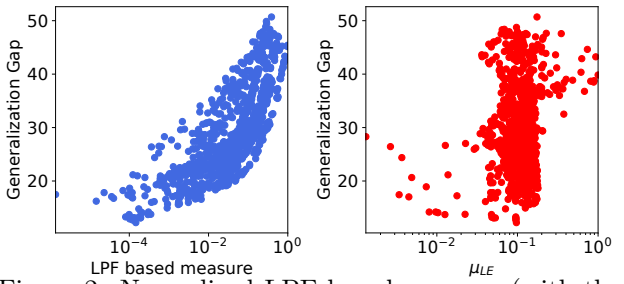

Figure 2: Normalized LPF based measure (with the highest overall correlation) (left) and normalized LE based measure (with the lowest overall correlation) (right) versus generalization gap.

and skip connections. It also shows strong positive correlation for both momentum and learning rate. It is however negatively correlated with generalization gap when varying the width of the network and batch normalization. Note that when varying batch normalization we observe only a weak correlation of all flatness measures with the generalization gap. In terms of varying the width, we observe in certain cases, including LPF based measure, strong negative correlation. This can be intuitively explained by the fact that network with higher capacity (higher width) and sharp minima can still outperform smaller network, even if its landscape is more flat. Finally, note that the table reporting the Kendall rank correlation coefficient between hyper-parameters (columns) and various sharpness measures (rows) (see Table 11) is deferred to the Supplement (Section 11.2). This table provides guidelines on how to design or modify deep architectures to populate the DL loss landscape with flat minima - this research direction lies beyond the scope of this work and will be investigated in the future.

Next we will investigate how robust different flatness measures are to the data and label noise, and whether they track the double descent phenomenon.

3.2 Sharpness versus generalization under data and label noise

Large neural networks can easily achieve zero training error (Zhang et al., 2016a) in the presence of data or label noise while generalizing very poorly. In this section, we evaluate the capability of sharpness measures to explain generalization performance of deep networks in presence of such noise. For experiments with noisy data, we corrupt each input image from publicly available CIFAR-10 data set with Gaussian noise with 0 mean and σI standard deviation and vary σ in the range $[0, 4]$. For experiments with noisy labels, we flip the true label of each input sample with probability α and vary α in the range $[0, 1]$. The flipped label is chosen uniformly at random from the set of all classes, excluding the true class. We train 10 ResNet18 models with different levels of label noise and 20 ResNet18

Measure	mom	width	wd	lr	batch	skip	bn	Overall
$\lambda_{\max}(H)$	$0.882_{\pm 0.03}$	$0.070_{\pm 0.07}$	$0.299_{\pm 0.07}$	$0.598_{\pm 0.06}$	$0.975_{\pm 0.01}$	$-0.148_{\pm 0.09}$	$-0.089_{\pm 0.09}$	$0.415_{\pm 0.00}$
$\|H\|_F$	$0.925_{\pm 0.02}$	$0.004_{\pm 0.07}$	$0.304_{\pm 0.07}$	$0.763_{\pm 0.05}$	$0.992_{\pm 0.01}$	$0.077_{\pm 0.09}$	$-0.094_{\pm 0.09}$	$0.481_{\pm 0.04}$
Trace (H)	$0.938_{\pm 0.02}$	$0.158_{\pm 0.08}$	$0.328_{\pm 0.07}$	$0.665_{\pm 0.05}$	$0.981_{\pm 0.01}$	$0.229_{\pm 0.09}$	$-0.081_{\pm 0.09}$	$0.538_{\pm 0.00}$
d_{eff}	$0.352_{\pm 0.07}$	$0.080_{\pm 0.07}$	$0.188_{\pm 0.07}$	$0.145_{\pm 0.07}$	$0.342_{\pm 0.07}$	$0.195_{\pm 0.09}$	$\mathbf{0.043_{\pm 0.09}}$	$0.326_{\pm 0.00}$
ϵ-sharpness	$0.776_{\pm 0.04}$	$-0.130_{\pm 0.07}$	$0.304_{\pm 0.07}$	$0.684_{\pm 0.05}$	$0.961_{\pm 0.02}$	$-0.239_{\pm 0.09}$	$-0.098_{\pm 0.09}$	$0.383_{\pm 0.02}$
$\mu_{PAC-Bayes}$	$\mathbf{0.994_{\pm 0.01}}$	$-0.836_{\pm 0.04}$	$0.408_{\pm 0.07}$	$\mathbf{0.900_{\pm 0.03}}$	$0.994_{\pm 0.01}$	$0.389_{\pm 0.08}$	$-0.094_{\pm 0.09}$	$0.466_{\pm 0.03}$
FRN	$0.812_{\pm 0.04}$	$\mathbf{0.238_{\pm 0.07}}$	$0.161_{\pm 0.07}$	$0.495_{\pm 0.06}$	$0.849_{\pm 0.04}$	$-0.073_{\pm 0.09}$	$-0.098_{\pm 0.09}$	$0.270_{\pm 0.02}$
$\mu_{entropy}$	$0.718_{\pm 0.05}$	$0.182_{\pm 0.07}$	$-0.072_{\pm 0.07}$	$0.327_{\pm 0.07}$	$0.716_{\pm 0.05}$	$-0.326_{\pm 0.08}$	$-0.123_{\pm 0.09}$	$0.296_{\pm 0.00}$
μ_{LE}	$0.160_{\pm 0.08}$	$-0.817_{\pm 0.04}$	$0.138_{\pm 0.07}$	$0.106_{\pm 0.07}$	$0.115_{\pm 0.07}$	$-0.050_{\pm 0.09}$	$-0.076_{\pm 0.09}$	$-0.064_{\pm 0.04}$
LPF	$0.990_{\pm 0.01}$	$-0.762_{\pm 0.05}$	$\mathbf{0.428_{\pm 0.07}}$	$0.867_{\pm 0.04}$	$\mathbf{0.996_{\pm 0.01}}$	$\mathbf{0.655_{\pm 0.07}}$	$-0.085_{\pm 0.09}$	$\mathbf{0.606_{\pm 0.03}}$

Table 2: Kendall rank correlation coefficient between generalization gap and sharpness measure (rows) averaged over set of models trained by varying only a single hyper-parameter (cols), along with 95% confidence interval. Larger value indicates better correlation.

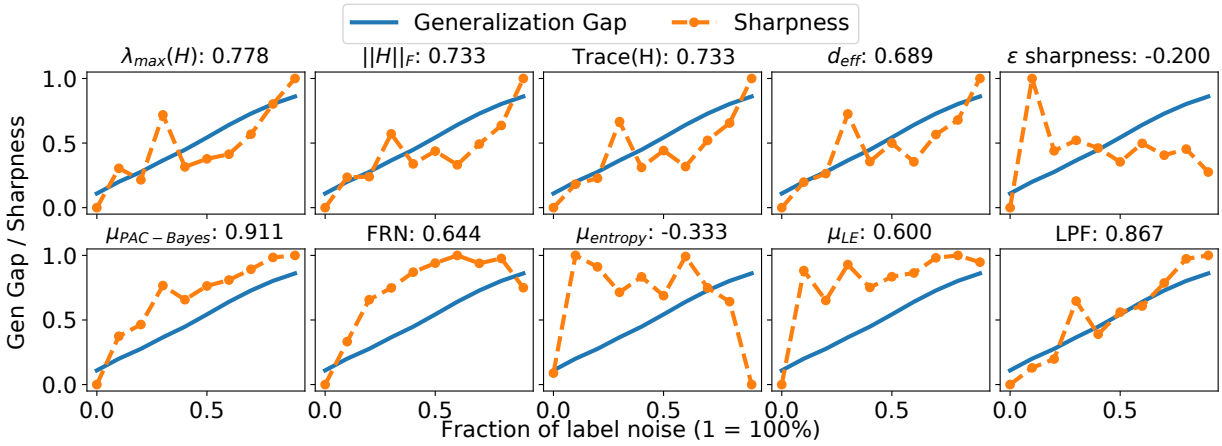

Figure 3: Normalized sharpness measures and generalization gap for varying levels of label noise. Kendall rank correlation coefficient between generalization gap and sharpness is provided in the titles above the figures.

models with different levels of data noise. Each experiment was repeated 5 times using different seeds. The experimental details are moved to the Supplement (Section 11.3).

In Figure 3 we plot the values of normalized sharpness measures and generalization gap for varying level of label noise (experiments for data noise are in Figure 6, Section 11.3 in the Supplement). We also report the Kendall rank correlation coefficient between sharpness measures and generalization gap (averaged over the seeds). We observe that LPF based measure is robust to both the data and label noise (it is the second best measure) and aligns well with the generalization behavior of the network. Finally, Section 11.4 in the Supplement shows that LPF-SGD based measure tracks the double descent phenomenon most closely from among all the considered sharpness measures.

4 LPF-SGD

Based on the results from the previous section, we identify LPF based sharpness measure as the most suitable one to incorporate into the DL optimization strategy to actively search for the flat regions in the DL loss landscape. Guiding the optimization process towards the well-generalizing flat regions of the DL loss landscape using LPF based measure can be done by solving the following optimization problem

$$\min_{\theta}(L \circledast K)(\theta) = \min_{\theta} \int L(\theta - \tau)K(\tau)d\tau, \qquad (4.1)$$

where K is a Gaussian kernel and $L(\theta)$ is the training loss function. We solve the problem given in Equation 4.1 using SGD. The gradient of the convolution between the loss function and the Gaussian kernel is

$$\nabla_\theta(L \circledast K)(\theta) = \nabla_\theta \int_{-\infty}^{\infty} L(\theta - \tau)K(\tau)d\tau \qquad (4.2)$$

$$\approx \frac{1}{M}\sum_{i=0}^{M} \nabla_\theta(L(\theta - \tau_i)), \qquad (4.3)$$

where τ_i comes from the same distribution as entries in the filter kernel K, and the integral is approximated using the MC method. Thus, to recap, we smooth the loss surface by performing a convolution between the loss function and the Gaussian low pass filter. The multiplicative term in Equation 4.1 and Equation 4.2 is the Gaussian probability density function. The action of the convolution is realized by sampling from the filter kernel and accumulating results (Equation 4.3). The resulting algorithm, that we call LPF-SGD, is in

fact extremely simple and its pseudo-code is provided in Algorithm 4.1 (for the ease of notation, $\mathcal{N}(0, \gamma\Sigma)$ in the algorithm stands for a sample from a Gaussian distribution). The balancing of model weights in our algorithm is incorporated into the construction of the LPF kernel K (thus there is no need to explicitly balance the network). To be more specific, $K \sim \mathcal{N}(0, \gamma\Sigma)$ and we set the matrix Σ to be proportional to the norm of the parameters in each filter of the network, i.e., we set $\Sigma = diag(||\theta_1^t||, ||\theta_2^t|| \cdots ||\theta_k^t||)$, where θ_k^t is the weight matrix of the k^{th} filter at iteration t and γ is the LPF radius.

Remark 1. *For imbalanced networks, the norm of the weights in one filter can be much larger than the norm of the weights in another filter (Figure 4, Supplement). In such cases isotropic covariance would lead to large perturbations in one filter while only minor in the other one. The parameter-dependent strategy (anisotropic, as in LPF-SGD) is equivalent to using isotropic covariance on a balanced network. We compare isotropic vs anistropic co-variance matrix in section 13 of the Supplement and show that anisotropic approach is superior (Table 21) in terms of final performance.*

Finally, we increase γ during network training to progressively increase the area of the loss surface that is explored at each gradient update. This is done according to the following rule:

$$\gamma_t = \gamma_0(\alpha/2(-\cos(t\pi/T) + 1) + 1), \tag{4.4}$$

where T is total number of iterations (epochs * gradient updates per epoch) and α is set such that $\gamma_T = (\alpha + 1) * \gamma_0$. This policy can be justified by the fact that the more you progress with the training, the more you care to recover flat regions in the loss landscape.

Algorithm 4.1 LPF-SGD

Inputs: θ^t: weights
Hyperpar: γ: filter radius, M: # MC iterations
while not converged **do**
  Sample data batch $B = (x_1, y_1) \cdots (x_b, y_b)$
  Split batch into M splits $B = \{B_1 \cdots B_M\}$
  $g \leftarrow 0$
  for i=1 to M **do**
    $\Sigma = diag(||\theta_i^t||_{i=1}^k)$
    $g = g + \frac{1}{M}\nabla_\theta L(B_i, \theta^t + \mathcal{N}(0, \gamma\Sigma))$
  end for
  $\theta^{t+1} = \theta^t - \eta * g$ // Update weights
end while

5 THEORY

In this section we theoretically analyze LPF-SGD and show that it converges to the optimal point with smaller generalization gap than SGD. All proofs are deferred to the Supplement (Section 15). Existing work (Bousquet and Elisseeff, 2001, 2002; Hardt et al., 2016; Bassily et al., 2020; Ramezani-Kebrya et al., 2018) shows that the generalization error is highly correlated with the Lipschitz continuous and smoothing properties of the objective function. Inspired by the analysis in (Lakshmanan and Pucci De Farias, 2008; Duchi et al., 2012), we first formally confirm that indeed Gaussian LPF leads to a smoother objective function and then show that SGD run on this smoother function recovers solution with smaller generalization error than in case of the original objective. We use classical approach to analyze the generalization performance and consider standard definition of generalization error that measures how far the empirical loss is from the true loss (for LPF-SGD, the true loss is the smoothed original loss).

Below we analyze the case when $\Sigma = \sigma^2 I$ and defer the analysis for non-scalar Σ (i.e., $\Sigma = \gamma diag(||\theta_1||, ||\theta_2|| \cdots ||\theta_k||)$) to the Supplement (Section 15). For the purpose of our analysis we assume Σ is kept fixed during the duration of the algorithm.

5.1 Properties of Gaussian LPF

Let $\mathcal{S} = \{\xi_1, \cdots, \xi_m\}$ denote the set of m samples drawn i.i.d. from an unknown distribution $\mathcal{D}$. Let $l_o(\theta; \xi)$ denote the loss of the model parametrized by θ for a specific example ξ. Assume $K \sim \mathcal{N}(0, \sigma^2 I)$. Denote the distribution $\mathcal{N}(0, \sigma^2 I)$ as μ. Define the convolution of $l_o(\theta; \xi)$ with the Gaussian kernel K as

$$\begin{aligned} l_\mu(\theta; \xi) = (l_o(\cdot; \xi) \circledast K)(\theta) &= \int_{\mathbb{R}^d} l_o(\theta - \tau; \xi)\mu(\tau)d\tau \\ &= \mathbb{E}_{Z\sim\mu}[l_o(\theta + Z; \xi)] \end{aligned} \tag{5.1}$$

where Z is a random variable satisfying distribution μ. The loss function smoothed by the Gaussian LPF, that we denote as l_μ, satisfies the following theorem.

Theorem 1. *Let μ be the $\mathcal{N}(0, \sigma^2 I_{d\times d})$ distribution. Assume the differentiable loss function $l_o(\theta; \xi) : \mathbb{R}^d \to \mathbb{R}$ is α-Lipschitz continuous and β-smooth with respect to l_2-norm. The smoothed loss function $l_\mu(\theta; \xi)$ is defined as (5.1). Then the following properties hold:*

i) l_μ is α-Lipschitz continuous.

ii) l_μ is continuously differentiable; moreover, its gradient is $\min\{\frac{\alpha}{\sigma}, \beta\}$-Lipschitz continuous, i.e., f_μ is $\min\{\frac{\alpha}{\sigma}, \beta\}$-smooth.

iii) If l_o is convex, $l_o(\theta; \xi) \le l_\mu(\theta; \xi) \le l_o(\theta; \xi) + \alpha\sigma\sqrt{d}$.

In addition, for each bound i)-iii), there exists a function l_o such that the bound cannot be improved by more than a constant factor.

Theorem 1 confirms that indeed l_μ is smoother than the original objective l_o. At the same time, if $\frac{\alpha}{\sigma} < \beta$, increasing σ leads to an increasingly smoother objective function, which is consistent with our intuition.

5.2 Generalization error and stability

In this section, we consider the generalization error of LPF-SGD algorithm and confront it with the generalization error of SGD. We first define the true loss as $L^{true}(\theta) := \mathbb{E}_{\xi\sim D} l(\theta;\xi)$, where l is an arbitrary loss function (i.e., l_o for SGD case and l_μ for LPF-SGD case). Note that the smoothed loss l_μ is by construction an estimator of the original loss l_o with bounded bias and variance, especially in a reasonable range of σ, thus the comparison captures relevant insights regarding LPF-SGD properties compared to SGD. Since the distribution $\mathcal{D}$ is unknown, we replace the true loss by the empirical loss given as $L^{\mathcal{S}}(\theta) := \frac{1}{m}\sum_{i=1}^{m} l(\theta;\xi)$.

We assume $\theta = A(\mathcal{S})$ for a potentially randomized algorithm A. The generalization error is given as:

$$\epsilon_g := \mathbb{E}_{\mathcal{S},A}[L^{true}(A(\mathcal{S})) - L^{\mathcal{S}}(A(\mathcal{S}))]. \tag{5.2}$$

In order to bound ϵ_g, we consider the following bound.

Definition 2 (ϵ_s-uniform stability (Hardt et al., 2016)). *Let $\mathcal{S}$ and $\mathcal{S}'$ denote two data sets from input data distribution $\mathcal{D}$ such that $\mathcal{S}$ and $\mathcal{S}'$ differ in at most one example. Algorithm A is ϵ_s-uniformly stable if and only if for all data sets $\mathcal{S}$ and $\mathcal{S}'$ we have*

$$\sup_{\xi} \mathbb{E}[l(A(\mathcal{S});\xi) - l(A(\mathcal{S}');\xi)] \leq \epsilon_s. \tag{5.3}$$

Next theorem implies that the generalization error could be bounded using the uniform stability.

Theorem 2 (Hardt et al. 2016). *If A is an ϵ_s-uniformly stable algorithm, then the generalization error (the gap between the true risk and the empirical risk) of A is upper-bounded by the stability factor ϵ_s:*

$$\epsilon_g := \mathbb{E}_{\mathcal{S},A}[L^{true}(A(\mathcal{S})) - L^{\mathcal{S}}(A(\mathcal{S}))] \leq \epsilon_s \tag{5.4}$$

Theorems 1, 2, and uniform stability bound for SGD (Theorem 4 in the Supplement) can be combined to analyze LPF-SGD. Intuitively, Theorem 1 explains how the Lipschitz continuous and smoothness properties of the objective function change after the function is smoothed with Gaussian LPF. This change can be propagated through Theorems 2. The details of the analysis are deferred to the Supplement (Section 15) and below we show the final result, Theorem 3. A justification of the theoretical approach we took when deriving it is also in the Supplement (Section 15.2).

Theorem 3 (Generalization error (GE) bound of LPF-SGD). *Assume that $l_o(\theta;\xi) \in [0,1]$ is a α-Lipschitz and β-smooth loss function for every example ξ. Suppose that we run SGD and LPF-SGD for T steps with non-increasing learning rate $\eta_t \leq c/t$. Denote the GE bound of SGD and LPF-SGD as $\hat{\epsilon}_g^o$ and $\hat{\epsilon}_g^\mu$, respectively. Then their ratio is*

$$\rho = \frac{\hat{\epsilon}_g^\mu}{\hat{\epsilon}_g^o} = \frac{1-p}{1-\hat{p}}\left(\frac{2c\alpha}{T}\right)^{\hat{p}-p} = O\left(\frac{1}{T^{\hat{p}-p}}\right), \tag{5.5}$$

where $p = \frac{1}{\beta c+1}$, $\hat{p} = \frac{1}{\min\{\frac{\alpha}{\sigma},\beta\}c+1}$. Finally, the following two properties hold:

Data set	Model	mSGD	E-SGD	ASO	SAM	LPF-SGD
CIFAR 10	18	$11.5_{\pm 0.3}$	$10.9_{\pm 0.3}$	$10.8_{\pm 0.5}$	$10.0_{\pm 0.1}$	$\mathbf{9.0_{\pm 0.2}}$
	50	$10.2_{\pm 0.4}$	$10.4_{\pm 0.1}$	$10.0_{\pm 0.2}$	$8.8_{\pm 0.3}$	$\mathbf{8.6_{\pm 0.1}}$
	101	$9.5_{\pm 0.4}$	$9.9_{\pm 0.3}$	$9.3_{\pm 0.2}$	$\mathbf{8.3_{\pm 0.3}}$	$8.7_{\pm 0.1}$
CIFAR 100	18	$38.3_{\pm 0.3}$	$38.2_{\pm 0.2}$	$37.3_{\pm 0.4}$	$36.2_{\pm 0.2}$	$\mathbf{30.0_{\pm 0.2}}$
	50	$35.6_{\pm 1.0}$	$34.5_{\pm 0.3}$	$34.3_{\pm 0.8}$	$33.1_{\pm 0.9}$	$\mathbf{30.6_{\pm 0.4}}$
	101	$32.7_{\pm 0.5}$	$33.5_{\pm 0.5}$	$31.9_{\pm 0.2}$	$30.7_{\pm 0.4}$	$\mathbf{29.9_{\pm 0.4}}$
tImgNet	18	$64.1_{\pm 0.1}$	$64.7_{\pm 0.1}$	$63.7_{\pm 0.2}$	$63.1_{\pm 0.3}$	$\mathbf{59.1_{\pm 0.2}}$
ImgNet	18	$36.5_{\pm 0.1}$	$40.9_{\pm 0.1}$	$38.0_{\pm 0.2}$	$35.9_{\pm 0.1}$	$\mathbf{35.4_{\pm 0.1}}$

Table 3: Mean validation error for SGD, E-SGD, ASO, SAM and LPF-SGD. We use ResNet-18, -50, and -101 models and CIFAR-10, CIFAR-100, TinyImageNet (tImgNet), and ImageNet (ImgNet) data sets. The models were trained with no data augmentations.

i) If $\sigma > \frac{\alpha}{\beta}$ and $T \gg 2c\alpha^2\left(\frac{1-p}{1-\hat{p}}\right)^{\frac{1}{\hat{p}-p}}$, $\rho \ll 1$.

ii) If $\sigma > \frac{\alpha}{\beta}$ and $T > 2c\alpha^2 e^{-p}$, increasing σ leads to a smaller ρ.

By point i) in Theorem 3, when number of iterations is large enough, GE bound of LPF-SGD is much smaller than that of SGD, which implies LPF-SGD converges to a better optimal point with lower generalization error than SGD. Moreover, point ii) in Theorem 3 indicates that increasing σ leads to a smaller generalization error.

6 EXPERIMENTS

In this section we compare the performance of our method with momentum SGD-based optimization strategy as well as sharpness-aware DL optimizers that encourage the recovery of flat optima. Additional training details are provided in Section 12.1 (Supplement).

Image Classification Here we compare LPF-SGD with momentum SGD (mSGD) (Saad, 1998; Polyak, 1964) as well as sharpness-aware DL optimizers, Entropy-SGD (E-SGD) (Chaudhari et al., 2017), Adaptive SmoothOut with denoising (Wen et al., 2018) (ASO), SAM, and newly introduced adaptive variant of SAM, ASAM (Kwon et al., 2021). Differently than in SAM paper, in all the experiments in our work we trained our models on a single GPU worker without additional label smoothing and gradient clipping. This is done in order to obtain a clear understanding how the methods themselves perform without additional regularizations. We utilized publicly available code repositories for both model architectures and the methods that we compare LPF-SGD with. For each pair of model and data set, we use standard mSGD settings of batch size, weight decay, momentum coefficient, learning rate, learning rate schedule, and total number of epochs across all optimizers. The details of hyperparameter selection as well as exemplary convergence curves are shown in Section 12.2 (Supplement).

Model	Aug	CIFAR-10					CIFAR-100				
		mSGD	E-SGD	ASO	SAM	LPF-SGD	mSGD	E-SGD	ASO	SAM	LPF-SGD
WRN16-8	B	$4.2_{\pm 0.2}$	$4.5_{\pm <0.1}$	$4.2_{\pm 0.1}$	$3.8_{\pm 0.1}$	$\mathbf{3.7_{\pm <0.1}}$	$20.6_{\pm 0.2}$	$20.7_{\pm 0.1}$	$20.2_{\pm 0.2}$	$19.4_{\pm 0.2}$	$\mathbf{18.9_{\pm 0.1}}$
	B+C	$3.9_{\pm 0.1}$	$3.8_{\pm 0.1}$	$3.6_{\pm 0.1}$	$3.3_{\pm 0.1}$	$\mathbf{3.2_{\pm 0.1}}$	$20.0_{\pm 0.1}$	$20.1_{\pm 0.2}$	$19.7_{\pm 0.1}$	$18.9_{\pm 0.1}$	$\mathbf{18.3_{\pm 0.1}}$
	B+A+C	$3.3_{\pm 0.1}$	$3.6_{\pm 0.1}$	$3.2_{\pm 0.1}$	$\mathbf{2.9_{\pm 0.1}}$	$3.1_{\pm 0.1}$	$19.3_{\pm 0.2}$	$19.4_{\pm 0.1}$	$18.9_{\pm 0.1}$	$18.1_{\pm 0.1}$	$\mathbf{17.6_{\pm 0.1}}$
WRN28-10	B	$4.0_{\pm 0.1}$	$4.0_{\pm 0.0}$	$3.9_{\pm 0.1}$	$\mathbf{3.2_{\pm 0.1}}$	$3.5_{\pm 0.1}$	$19.1_{\pm 0.2}$	$19.8_{\pm 0.2}$	$18.8_{\pm 0.1}$	$\mathbf{17.4_{\pm <0.1}}$	$\mathbf{17.4_{\pm 0.1}}$
	B+C	$3.1_{\pm <0.1}$	$3.3_{\pm 0.1}$	$3.3_{\pm 0.1}$	$\mathbf{2.7_{\pm 0.1}}$	$\mathbf{2.7_{\pm <0.1}}$	$18.3_{\pm 0.1}$	$18.8_{\pm 0.2}$	$18.0_{\pm 0.3}$	$17.1_{\pm 0.1}$	$\mathbf{16.9_{\pm 0.2}}$
	B+A+C	$2.6_{\pm 0.1}$	$3.0_{\pm 0.1}$	$2.7_{\pm 0.1}$	$\mathbf{2.3_{\pm 0.1}}$	$2.5_{\pm 0.1}$	$17.3_{\pm 0.2}$	$17.6_{\pm 0.1}$	$16.9_{\pm 0.4}$	$\mathbf{15.9_{\pm 0.1}}$	$\mathbf{15.9_{\pm 0.1}}$
Shake Shake (26 2x96d)	B	$2.9_{\pm 0.1}$	$3.3_{\pm 0.1}$	$2.8_{\pm 0.1}$	$2.6_{\pm 0.1}$	$\mathbf{2.5_{\pm <0.1}}$	$17.1_{\pm 0.1}$	$17.5_{\pm 0.3}$	$17.5_{\pm 0.1}$	$16.9_{\pm 0.1}$	$\mathbf{16.6_{\pm 0.3}}$
	B+C	$2.5_{\pm <0.1}$	$3.1_{\pm 0.0}$	$2.4_{\pm 0.0}$	$\mathbf{2.2_{\pm <0.1}}$	$\mathbf{2.2_{\pm <0.1}}$	$17.0_{\pm 0.4}$	$17.3_{\pm 0.1}$	$16.8_{\pm 0.2}$	$16.6_{\pm 0.2}$	$\mathbf{16.1_{\pm 0.2}}$
	B+A+C	$2.1_{\pm 0.1}$	$3.1_{\pm 0.1}$	$2.1_{\pm 0.1}$	$2.0_{\pm 0.1}$	$\mathbf{2.0_{\pm 0.1}}$	$15.7_{\pm 0.1}$	$16.3_{\pm 0.1}$	$15.6_{\pm 0.0}$	$15.3_{\pm 0.2}$	$\mathbf{15.0_{\pm <0.1}}$
PyNet110 ($\alpha = 270$)	B	$3.7_{\pm 0.1}$	$3.9_{\pm 0.1}$	$3.7_{\pm 0.0}$	$\mathbf{3.1_{\pm <0.1}}$	$3.4_{\pm 0.1}$	$18.4_{\pm 0.3}$	$19.1_{\pm 0.1}$	$18.5_{\pm 0.3}$	$18.3_{\pm 0.2}$	$\mathbf{18.0_{\pm 0.1}}$
	B+C	$2.9_{\pm 0.1}$	$2.8_{\pm 0.1}$	$2.9_{\pm 0.1}$	$\mathbf{2.5_{\pm 0.2}}$	$\mathbf{2.5_{\pm <0.1}}$	$17.7_{\pm 0.1}$	$18.1_{\pm 0.2}$	$17.7_{\pm 0.4}$	$17.4_{\pm 0.3}$	$\mathbf{17.0_{\pm 0.1}}$
	B+A+C	$2.2_{\pm 0.0}$	$2.3_{\pm 0.1}$	$2.3_{\pm 0.0}$	$\mathbf{2.0_{\pm <0.1}}$	$2.1_{\pm 0.0}$	$16.9_{\pm 0.0}$	$15.9_{\pm <0.1}$	$16.0_{\pm 0.1}$	$15.8_{\pm 0.1}$	$\mathbf{15.6_{\pm 0.1}}$
PyNet272 ($\alpha = 200$)	B	$3.3_{\pm 0.1}$	$3.6_{\pm 0.0}$	$3.3_{\pm 0.1}$	$\mathbf{2.9_{\pm 0.1}}$	$3.3_{\pm 0.1}$	$17.6_{\pm 0.2}$	$18.5_{\pm 0.1}$	$17.4_{\pm 0.1}$	$17.0_{\pm 0.2}$	$\mathbf{16.7_{\pm <0.1}}$
	B+C	$2.6_{\pm 0.0}$	$2.7_{\pm 0.1}$	$2.7_{\pm 0.1}$	$2.4_{\pm 0.0}$	$\mathbf{2.3_{\pm 0.1}}$	$16.9_{\pm <0.1}$	$17.5_{\pm 0.3}$	$16.9_{\pm 0.1}$	$16.2_{\pm 0.2}$	$\mathbf{15.9_{\pm <0.1}}$
	B+A+C	$2.1_{\pm 0.2}$	$2.2_{\pm 0.1}$	$2.1_{\pm 0.1}$	$\mathbf{2.0_{\pm <0.1}}$	$\mathbf{2.0_{\pm <0.1}}$	$14.9_{\pm <0.1}$	$15.0_{\pm 0.1}$	$14.9_{\pm 0.3}$	$\mathbf{14.7_{\pm 0.2}}$	$\mathbf{14.7_{\pm 0.1}}$

Table 5: Mean validation error with 95% confidence interval. B refers to basic data augmentation, C refers to Cutout augmentation, and A refers to AutoAugmentation.

In the first set of our experiments we utilized image classification models based on ResNets (He et al., 2016). They are prone to over-fitting when trained without any data augmentation. The sharpness based optimizers prevent overfitting by seeking flat minima. Therefore, we first compare the performance of optimizers on ResNets without performing data augmentation. We use ResNet18, 50, and 101 models and open sourced CIFAR-10 and CIFAR-100 (Krizhevsky et al., 2014b), TinyImageNet (Deng et al., 2009), and ImageNet (Deng et al., 2009) data sets. We run each experiment for 5 seeds. We report the mean validation error along with the 95% confidence interval (Table 3). Table 3, as well as Figures 8 and 9 in the Supplement show that LPF-SGD compares favorably to all other methods in terms of the generalization performance. We also evaluate various sharpness measures for models trained with mSGD, SAM and LPF-SGD (Table 14 in the Supplement) and show that LPF-SGD achieves the lowest value of the LPF sharpness based measure leading to the lowest validation error.

Next, we evaluate the performance of LPF-SGD on standard ImageNet training available in PyTorch (Paszke et al., 2017), where we train a ResNet18 model with basic data augmentation. Table 4 shows the validation error rate for various optimizers, among which our technique achieves superior performance.

Opt	mSGD	ASO	SAM	LPF-SGD
Val Error	30.24	$29.54_{\pm 0.10}$	$29.49_{\pm 0.10}$	$\mathbf{29.45_{\pm 0.02}}$

Table 4: Mean validation error rate with 95% confidence interval on ImageNet data set trained with ResNet18 model and basic augmentation.

Finally, we evaluate the performance of LPF-SGD on state-of-the-art architectures. We trained WRN16-8, WRN28-10 (Zagoruyko and Komodakis, 2016), ShakeShake (26 2x96d) (Gastaldi, 2017), PyramidNet-110(α= 270), and PyramidNet-272(α=200) (Dongyoon et al., 2017) on open sourced CIFAR-10 and CIFAR-100 data sets. We also utilized three progressively increasing data augmentation schemes: i) basic (random cropping and horizontal flipping), ii) basic + cutout (DeVries and Taylor, 2017) and iii) basic + auto-augmentation (Cubuk et al., 2019) + cutout (DeVries and Taylor, 2017). We run 5 seeds for WRN16-8 and WRN28-10 and 2 seeds for ShakeShake and PyramidNet models. We report the mean validation error with a 95% confidence interval on the validation set (Table 5). Table 5 shows that LPF-SGD either outperforms or show comparable performance to SAM, and at the same time is always superior to E-SGD and ASO. As in case of ResNets, LPF-SGD clearly wins with mSGD that does not explicitly encourage the recovery of flat optima in the DL optimization landscape.

Model	Aug	CIFAR-100	
		ASAM	LPF-SGD
WRN 16-8	B	$19.2_{\pm 0.0}$	$\mathbf{18.9_{\pm 0.1}}$
	B+C	$\mathbf{18.3_{\pm 0.1}}$	$\mathbf{18.3_{\pm 0.1}}$
	B+A+C	$\mathbf{17.5_{\pm 0.2}}$	$17.6_{\pm 0.1}$
WRN 28-10	B	$17.6_{\pm 0.2}$	$\mathbf{17.4_{\pm 0.1}}$
	B	$\mathbf{16.7_{\pm 0.1}}$	$16.9_{\pm 0.2}$
	B+A+C	$16.1_{\pm 0.1}$	$\mathbf{15.9_{\pm 0.1}}$
ShakeShake (26 2x96d)	B	$17.0_{\pm 0.1}$	$\mathbf{16.6_{\pm 0.3}}$
	B+C	$16.7_{\pm 0.2}$	$\mathbf{16.1_{\pm 0.2}}$
	B+A+C	$15.2_{\pm 0.1}$	$\mathbf{15.0_{\pm <0.1}}$

Table 6: Validation error rate with 95% confidence interval for ASAM and LPF-SGD.

We also present experiments capturing comparison of LPF-SGD to a newly introduced ASAM (Kwon et al., 2021) method. For this experiment we train WRN16-8, WRN 28-10, and ShakeShake models on CIFAR-100 data set. LPF-SGD is found to be superior to ASAM as captured in Table 6.

Opt \ Model	WRN16-8	WRN28-10	PyNet110
mSGD	0.100 s	0.292 s	0.470 s
SAM	0.202 s	0.582 s	0.924 s
LPF-SGD (M=2)	0.115 s	0.319 s	0.566 s
LPF-SGD (M=4)	0.138 s	0.380 s	0.679 s
LPF-SGD (M=8)	0.196 s	0.500 s	0.904 s

Table 7: Computational time for a single iteration of mSGD, SAM, and LPF-SGD for various settings of M.

In Table 7, we also show the computational cost for a single iteration of mSGD, SAM, and LPF-SGD for multiple values of M, computed on a NVIDIA GTX1080 GPU. For each LPF-SGD update, we split the batch of input data into M mini-batches, compute MC iterations on each mini-batch, and accumulate the gradients before making a final weight update. Therefore, the number of operations (multiplications + additions) between mSGD and LPF-SGD are preserved. LPF-SGD incurs additional cost, in comparison to SGD, due to operating on smaller batches internally (by a factor of M) and the computation of Σ matrix. SAM, on the other hand, relies on a nested optimization scheme, which is overall slower compared to LPF-SGD, which is why LPF-SGD consistently outperforms SAM timewise for all explored settings of M. Note that the computational time can be further improved as diagonal blocks of the Sigma matrix are independent and can be computed by parallel CPU threads.

Machine Translation Here we train a transformer model based on Vaswani et al. (2017) to perform German to English translation on WMT2014 data set (Bojar et al., 2014). The model was trained using Adam (Kingma and Ba, 2014). We compare the performance of LPF-Adam (our method) with Adam as well as Entropy-Adam (E-Adam), ASO-Adam, and SAM-Adam. Section 12.3 (Supplement) contains training details. In Table 8, we show BLEU score for both validation and test data sets and reveal that in the context of machine translation recovering flat optima with our proposed optimizer leads to the highest scores.

Optimizer	Validation	Test
Adam	$21.76_{\pm 0.26}$	$20.97_{\pm 0.43}$
E-Adam	$19.91_{\pm 0.03}$	$18.84_{\pm 0.18}$
ASO-Adam	$21.91_{\pm 0.24}$	$20.77_{\pm 0.21}$
SAM-Adam	$22.06_{\pm 0.04}$	$20.92_{\pm 0.15}$
LPF-Adam	$\mathbf{22.10_{0.15}}$	$\mathbf{21.14_{0.23}}$

Table 8: BLEU scores with 95% confidence interval for German to English translation on WMT2014 data set.

Finally, the ablation studies aiming at understanding the impact of various hyper-parameters of the LPF-SGD optimizer on its performance, and the studies showing that prolonging mSGD training does not help mSGD to match LPF-SGD, as well as the studies confirming that LPF-SGD shows more consistent robustness to adversarial attacks compared to other methods are deferred to Section 13 and 14 (Supplement).

7 CONCLUSIONS

In this paper we show comprehensive empirical study investigating the connection between the flatness of the optima of the DL loss landscape and the generalization properties of DL models. We derive an algorithm, LPF-SGD, for training DL models based on the sharpness measure that best correlates with model generalization abilities. LPF-SGD compares favorably to common training strategies. Regarding societal impact of our work and extensions, new landscape-driven DL optimization tools, such as LPF-SGD, will have a strong impact on a wide range of DL applications, where better solvers translate to more efficient utilization of computational resources, i.e., new optimization strategies will maximize the performance of DL architectures. The outcomes of such research works can be leveraged by public and private entities to shift to significantly more powerful computational learning platforms.

Acknowledgements

The authors would like to acknowledge the support of the NSF Award #2041872 in sponsoring this research.

References

E. L. Allgower and K. Georg. *Numerical continuation methods: an introduction*, volume 13. Springer, 2012.

A. Anandkumar and R. Ge. Efficient approaches for escaping higher order saddle points in non-convex optimization. *arXiv:1602.05908*, 2016.

A. Anandkumar, K. Chaudhuri, P. Liang, S. Oh, and U. N. Niranjan. Non-convex optimization, workshop at nips, 2015.

S. Arora, N. Cohen, N. Golowich, and W. Hu. A convergence analysis of gradient descent for deep linear neural networks. In *ICLR*, 2019.

A. Askari, G. Negiar, R. Sambharya, and L. El Ghaoui. Lifted neural networks. arXiv:1805.01532, 2018.

C. Baldassi, A. Ingrosso, C. Lucibello, L. Saglietti, and R. Zecchina. Subdominant dense clusters allow for simple learning and high computational performance in neural networks with discrete synapses. *Physical review letters*, 115(12):128101, 2015.

C. Baldassi, C. Borgs, J. T. Chayes, A. Ingrosso, C. Lucibello, L. Saglietti, and R. Zecchina. Unreasonable

effectiveness of learning neural networks: From accessible states and robust ensembles to basic algorithmic schemes. In *PNAS*, 2016a.

C. Baldassi, F. Gerace, C. Lucibello, L. Saglietti, and R. Zecchina. Learning may need only a few bits of synaptic precision. *Physical Review E*, 93(5):052313, 2016b.

P. Baldi and K. Hornik. Neural networks and principal component analysis: Learning from examples without local minima. *Neural Networks*, 2:53–58, 1989.

S. Bartunov, A. Santoro, B. Richards, L. Marris, G. E. Hinton, and T. Lillicrap. Assessing the scalability of biologically-motivated deep learning algorithms and architectures. In *NeurIPS*, 2018.

R. Bassily, V. Feldman, C. Guzmán, and K. Talwar. Stability of stochastic gradient descent on nonsmooth convex losses. *arXiv:2006.06914*, 2020.

M. Belkin, D. Hsu, S. Ma, and S. Mandal. Reconciling modern machine-learning practice and the classical bias–variance trade-off. *Proceedings of the National Academy of Sciences*, 116(32):15849–15854, 2019.

O. Bojar, C. Buck, C. Federmann, B. Haddow, P. Koehn, J. Leveling, C. Monz, P. Pecina, M. Post, H. Saint-Amand, R. Soricut, L. Specia, and A. Tamchyna. Findings of the 2014 workshop on statistical machine translation. In *Proceedings of the Ninth Workshop on Statistical Machine Translation*, pages 12–58, Baltimore, Maryland, USA, June 2014. Association for Computational Linguistics. doi: 10.3115/v1/W14-3302.

R. E. Botsch. Chapter 12. Significance and Measures of Association, 2011.

L. Bottou. Online algorithms and stochastic approximations. In *Online Learning and Neural Networks*. Cambridge University Press, 1998.

O. Bousquet and A. Elisseeff. Algorithmic stability and generalization performance. *Advances in Neural Information Processing Systems*, pages 196–202, 2001.

O. Bousquet and A. Elisseeff. Stability and generalization. *JMLR*, 2(Mar):499–526, 2002.

A. Brutzkus and A. Globerson. Globally optimal gradient descent for a ConvNet with Gaussian inputs. In *ICML*, 2017.

A. Brutzkus, A. Globerson, E. Malach, and S. Shalev-Shwartz. SGD learns over-parameterized networks that provably generalize on linearly separable data. In *ICLR*, 2018.

M. Carreira-Perpiñán and W. Wang. Distributed optimization of deeply nested systems. In *AISTATS*, 2014.

J. Cha, S. Chun, K. Lee, H.-C. Cho, S. Park, Y. Lee, and S. Park. SWAD: Domain generalization by seeking flat minima. In *NeurIPS*, 2021.

P. Chaudhari and S. Soatto. On the energy landscape of deep networks. *CoRR*, abs/1511.06485, 2015.

P. Chaudhari, A. Choromanska, S. Soatto, Y. LeCun, C. Baldassi, C. Borgs, J. Chayes, L. Sagun, and R. Zecchina. Entropy-sgd: Biasing gradient descent into wide valleys. In *ICLR*, 2017.

A. Choromanska, M. Henaff, M. Mathieu, G. Ben Arous, and Y. LeCun. The loss surfaces of multilayer networks. In *AISTATS*, 2015a.

A. Choromanska, Y. LeCun, and G. Ben Arous. Open problem: The landscape of the loss surfaces of multilayer networks. In *COLT*, 2015b.

A. Choromanska, B. Cowen, S. Kumaravel, R. Luss, M. Rigotti, I. Rish, P. Diachille, V. Gurev, B. Kingsbury, R. Tejwani, and D. Bouneffouf. Beyond backprop: Online alternating minimization with auxiliary variables. In *ICML*, 2019.

T. Cooijmans, N. Ballas, C. Laurent, and A. Courville. Recurrent batch normalization. *arXiv:1603.09025*, 2016.

B. Cowen, A. Nandini Saridena, and A. Choromanska. Lsalsa: accelerated source separation via learned sparse coding. *Machine Learning (also appeared at ECML-PKDD)*, 108:1307–1327, 2019. doi: 10.1007/s10994-019-05812-3.

E. D. Cubuk, B. Zoph, D. Mane, V. Vasudevan, and Q. V. Le. Autoaugment: Learning augmentation strategies from data. In *CVPR*, 2019.

Y. N. Dauphin, R. Pascanu, C. Gulcehre, K. Cho, S. Ganguli, and Y. Bengio. Identifying and attacking the saddle point problem in high-dimensional non-convex optimization. In *NIPS*, 2014.

J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR*, 2009.

G. Desjardins, K. Simonyan, R. Pascanu, and K. Kavukcuoglu. Natural neural networks. In *NIPS*. 2015.

T. DeVries and G. W. Taylor. Improved regularization of convolutional neural networks with cutout. *arXiv:1708.04552*, 2017.

L. Dinh, R. Pascanu, S. Bengio, and Y. Bengio. Sharp minima can generalize for deep nets. In *ICML*, 2017.

H. Dongyoon, K. Jiwhan, and K. Junmo. Deep pyramidal residual networks. In *CVPR*, 2017.

F. Draxler, K. Veschgini, M. Salmhofer, and F. Hamprecht. Essentially no barriers in neural network energy landscape. In *ICML*, 2018.

S. Du and J. Lee. On the power of over-parametrization in neural networks with quadratic activation. In *ICML*, 2018.

J. Duchi, E. Hazan, and Y. Singer. Adaptive subgradient methods for online learning and stochastic optimization. *JMLR*, 12:2121–2159, 2011.

J. C. Duchi, P. L. Bartlett, and M. J. Wainwright. Randomized smoothing for stochastic optimization. *SIAM Journal on Optimization*, 22(2):674–701, 2012.

G. K. Dziugaite and D. M. Roy. Computing nonvacuous generalization bounds for deep (stochastic) neural networks with many more parameters than training data. In *UAI*, 2017.

G. K. Dziugaite, A. Drouin, B. Neal, N. Rajkumar, E. Caballero, L. Wang, I. Mitliagkas, and D. M Roy. In search of robust measures of generalization. *arXiv preprint arXiv:2010.11924*, 2020.

Y. Feng and Y. Tu. How neural networks find generalizable solutions: Self-tuned annealing in deep learning. *CoRR*, abs/2001.01678, 2020.

P. Foret, A. Kleiner, H. Mobahi, and B. Neyshabur. Sharpness-aware minimization for efficiently improving generalization. In *ICLR*, 2021.

C. D. Freeman and J. Bruna. Topology and geometry of half-rectified network optimization. In *ICLR*, 2017.

X. Gastaldi. Shake-shake regularization. *arXiv:1705.07485*, 2017.

R. Ge, F. Huang, C. Jin, and Y. Yuan. Escaping from saddle points — online stochastic gradient for tensor decomposition. In *COLT*, 2015.

R. Ge, C. Jin, and Y. Zheng. No spurious local minima in nonconvex low rank problems: A unified geometric analysis. In *ICML*, 2017.

B. Ghorbani, S. Krishnan, and Y. Xiao. An investigation into neural net optimization via hessian eigenvalue density. *arXiv:1901.10159*, 2019.

B. Ginsburg, P. Castonguay, O. Hrinchuk, O. Kuchaiev, R. Leary, V. Lavrukhin, J. Li, H. Nguyen, Y. Zhang, and J. M. Cohen. Training deep networks with stochastic gradient normalized by layerwise adaptive second moments. *CoRR*, abs/1905.11286, 2019.

I. J. Goodfellow and O. Vinyals. Qualitatively characterizing neural network optimization problems. In *ICLR*, 2015.

I. J. Goodfellow, J. Shlens, and C. Szegedy. Explaining and harnessing adversarial examples. In *ICLR*, 2015.

A. Gotmare, T. Valentin, J. Brea, and M. Jaggi. Decoupling backpropagation using constrained optimization methods. *Proc. of ICML 2018 Workshop on Credit Assignment in Deep Learning and Deep Reinforcement Learning*, 2018.

C. Gulcehre, M. Moczulski, F. Visin, and Y. Bengio. Mollifying networks. *CoRR*, abs/1608.04980, 2016.

B. Haeffele and R. Vidal. Global optimality in tensor factorization, deep learning, and beyond. *CoRR*, abs/1506.07540, 2015.

M. Hardt and T. Ma. Identity matters in deep learning. In *ICLR*, 2017.

M. Hardt, B. Recht, and Y. Singer. Train faster, generalize better: Stability of stochastic gradient descent. In *ICML*, 2016.

K. Haruki, T. Suzuki, Y. Hamakawa, T. Toda, R. Sakai, M. Ozawa, and M. Kimura. Gradient noise convolution (gnc): Smoothing loss function for distributed large-batch sgd. *arXiv:1906.10822*, 2019.

K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *CVPR*, 2016.

S. Hochreiter and J. Schmidhuber. Flat minima. *Neural Computation*, 9(1):1–42, 1997.

M. S. Inci, T. Eisenbarth, and B. Sunar. Deepcloak: Adversarial crafting as a defensive measure to cloak processes. *arXiv:1808.01352*, 2018.

S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *CoRR*, abs/1502.03167, 2015.

P. Izmailov, D. Podoprikhin, T. Garipov, D. Vetrov, and A. G. Wilson. Averaging weights leads to wider optima and better generalization. In *UAI*, 2018.

M. Janzamin, H. Sedghi, and A. Anandkumar. Beating the Perils of Non-Convexity: Guaranteed Training of Neural Networks using Tensor Methods. *arXiv:1506.08473*, 2015.

S. Jastrzebski, Z. Kenton, D. Arpit, N. Ballas, A. Fischer, Y. Bengio, and A. Storkey. Finding flatter minima with sgd. In *ICLR Workshop Track*, 2018.

L. Jiang, Z. Zhou, T. Leung, L.-J. Li, and F.-F. Li. Mentornet: Regularizing very deep neural networks on corrupted labels. *CoRR*, abs/1712.05055, 2017.

Y. Jiang, B. Neyshabur, H. Mobahi, D. Krishnan, and S. Bengio. Fantastic generalization measures and where to find them. In *ICLR*, 2020.

K. Kawaguchi. Deep learning without poor local minima. In *NIPS*. 2016.

N. S. Keskar, D. Mudigere, J. Nocedal, M. Smelyanskiy, and P. T. P. Tang. On large-batch training for deep learning: Generalization gap and sharp minima. In *ICLR*, 2017.

D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014.

A. Krizhevsky, V. Nair, and G. Hinton. The cifar dataset, 2014a.

A. Krizhevsky, V. Nair, and G. Hinton. The cifar dataset. 2014b.

J. Kwon, J. Kim, H. Park, and I. K. Choi. Asam: Adaptive sharpness-aware minimization for scale-invariant learning of deep neural networks. In *ICML*, 2021.

H. Lakshmanan and D. Pucci De Farias. Decentralized resource allocation in dynamic networks of agents. *SIAM Journal on Optimization*, 19(2):911–940, 2008.

T. T.-K. Lau, J. Zeng, B. Wu, and Y. Yao. A proximal block coordinate descent algorithm for deep neural network training. *arXiv:1803.09082*, 2018.

Y. LeCun. Learning process in an asymmetric threshold network. In *Disordered systems and biological organization*, pages 233–240. Springer, 1986.

Y. LeCun. *Modèles connexionnistes de l'apprentissage*. PhD thesis, PhD thesis, These de Doctorat, Universite Paris 6, 1987.

Y. LeCun, D. Touresky, G. Hinton, and T. Sejnowski. A theoretical framework for back-propagation. In *Proceedings of the 1988 connectionist models summer school*, pages 21–28. CMU, Pittsburgh, Pa: Morgan Kaufmann, 1988.

Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. Backpropagation applied to handwritten zip code recognition. *Neural Comput.*, 1(4):541–551, 1989a.

Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1(4):541–551, 1989b.

D.-H. Lee, S. Zhang, A. Fischer, and Y. Bengio. Difference target propagation. In *ECML-PKDD*, 2015.

Y. Li and Y. Yuan. Convergence analysis of two-layer neural networks with relu activation. In *NIPS*. 2017.

Y. Li, T. Ma, and H. Zhang. Algorithmic regularization in over-parameterized matrix sensing and neural networks with quadratic activations. In *COLT*, 2018.

T. Liang, T. Poggio, A. Rakhlin, and J. Stokes. Fisher-rao metric, geometry, and complexity of neural networks. In *AISTATS*, 2019.

T. Lin, L. Kong, S. Stich, and M. Jaggi. Extrapolation for large-batch training in deep learning. In *International Conference on Machine Learning*, pages 6094–6104. PMLR, 2020.

L. Luo, Y. Xiong, Y. Liu, and X. Sun. Adaptive gradient methods with dynamic bound of learning rate. In *ICLR*, 2019.

D. J. C. MacKay. Bayesian model comparison and backprop nets. In *NeurIPS*, 1992.

W. J. Maddox, G. Benton, and A. G. Wilson. Rethinking parameter counting in deep models: Effective dimensionality revisited. *arXiv:2003.02139*, 2020.

A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu. Towards deep learning models resistant to adversarial attacks. *arXiv preprint arXiv:1706.06083*, 2017.

H. Mobahi. Training recurrent neural networks by diffusion. *CoRR*, abs/1601.04114, 2016.

H. Mobahi and J. Fisher III. On the link between Gaussian homotopy continuation and convex envelopes. In *Workshop on Energy Minimization Methods in CVPR*, pages 43–56. Springer, 2015.

S.-M. Moosavi-Dezfooli, A. Fawzi, and P. Frossard. Deepfool: a simple and accurate method to fool deep neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2574–2582, 2016.

P. Nakkiran, G. Kaplun, Y. Bansal, T. Yang, B. Barak, and I. Sutskever. Deep double descent: Where bigger models and more data hurt. In *ICLR*, 2020.

Y. Nesterov. Smooth minimization of non-smooth functions. *Mathematical Programming*, 103:127–152, 2005.

B. Neyshabur, R. R. Salakhutdinov, and N. Srebro. Path-sgd: Path-normalized optimization in deep neural networks. In *NIPS*. 2015.

B. Neyshabur, S. Bhojanapalli, D. Mcallester, and N. Srebro. Exploring generalization in deep learning. In *NIPS*. 2017.

Q. N. Nguyen and M. Hein. The loss surface of deep and wide neural networks. In *ICML*, 2017.

A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer. Automatic differentiation in pytorch. 2017.

G. Pereyra, G. Tucker, J. Chorowski, Ł. Kaiser, and G. Hinton. Regularizing neural networks by penalizing confident output distributions. *arXiv:1701.06548*, 2017.

F. Pittorino, C. Lucibello, C. Feinauer, G. Perugini, C. Baldassi, E. Demyanenko, and R. Zecchina. Entropic gradient descent algorithms and wide flat minima. In *ICLR*, 2021.

B.T. Polyak. Some methods of speeding up the convergence of iteration methods. *USSR Computational Mathematics and Mathematical Physics*, 4(5):1–17, 1964. ISSN 0041-5553. doi: https://doi.org/10.1016/0041-5553(64)90137-5.

N. Qian. On the momentum term in gradient descent learning algorithms. *Neural Networks*, 12(1):145–151, 1999.

A. Ramezani-Kebrya, A. Khisti, and B. Liang. On the stability and convergence of stochastic gradient descent with momentum. *CoRR*, abs/1809.04564, 2018.

J. Rauber, W. Brendel, and M. Bethge. Foolbox: A python toolbox to benchmark the robustness of machine learning models. In *Reliable Machine Learning in the Wild Workshop, 34th International Conference on Machine Learning*, 2017.

O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, et al. Imagenet large scale visual recognition challenge. *International journal of computer vision*, 115(3):211–252, 2015.

D. Saad. Online algorithms and stochastic approximations. *Online Learning*, 5:6–3, 1998.

I. Safran and O. Shamir. On the quality of the initial basin in overspecified neural networks. In *ICML*, 2016.

I. Safran and O. Shamir. Spurious local minima are common in two-layer relu neural networks. *CoRR*, abs/1712.08968, 2017.

L. Sagun, U. Evci, V. Ugur Güney, Y. Dauphin, and L. Bottou. Empirical analysis of the hessian of overparametrized neural networks. In *ICLR Workshop*, 2018.

T. Salimans and D. Kingma. Weight normalization: A simple reparameterization to accelerate training of deep neural networks. *arXiv:1602.07868*, 2016.

A. M. Saxe, J. L. McClelland, and S. Ganguli. Exact solutions to the nonlinear dynamics of learning in deep linear neural networks. In *ICLR*. 2014.

N. Shazeer and M. Stern. Adafactor: Adaptive learning rates with sublinear memory cost. *CoRR*, abs/1804.04235, 2018.

U. Simsekli, L. Sagun, and M. Gürbüzbalaban. A tail-index analysis of stochastic gradient noise in deep neural networks. *CoRR*, abs/1901.06053, 2019.

J. Sokolic, R. Giryes, G. Sapiro, and M. Rodrigues. Generalization error of invariant classifiers. In *AISTATS*, 2017.

D. Soudry and Y. Carmon. No bad local minima: Data independent training error guarantees for multilayer neural networks. *CoRR*, abs/1605.08361, 2016.

N. Srivastava, G. E. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *JMLR*, 2014.

I. Sutskever, J. Martens, G. Dahl, and G. Hinton. On the importance of initialization and momentum in deep learning. In *ICML*, 2013.

G. Taylor, R. Burmeister, Z. Xu, B. Singh, A. Patel, and T. Goldstein. Training neural networks without gradients: A scalable admm approach. In *ICML*, 2016.

Y. Tian. An analytical formula of population gradient for two-layered relu network and its applications in convergence and critical point analysis. In *ICML*, 2017.

T. Tieleman and G. Hinton. Lecture 6.5: RmsProp: Divide the Gradient by a Running Average of Its Recent Magnitude. *Coursera: Neural networks for machine learning*, 4(2):26–31, 2012.

M. Trager, K. Kohn, and J. Bruna. Pure and spurious critical points: a geometric study of linear networks. In *ICLR*, 2020.

A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. Attention is all you need. In *NeurIPS*, 2017.

R. Vidal, J Bruna, R. Giryes, and S. Soatto. Mathematics of deep learning. *CoRR*, abs/1712.04741, 2017.

W. Wen, Y. Wang, F. Yan, C. Xu, Y. Chen, and H. Li. Smoothout: Smoothing out sharp minima for generalization in large-batch deep learning. *CoRR*, abs/1805.07898, 2018.

A. C. Wilson, R. Roelofs, M. Stern, N. Srebro, and B. Recht. The marginal value of adaptive gradient methods in machine learning. In *NeurIPS*, 2017.

C. Yun, S. Sra, and A. Jadbabaie. Global optimality conditions for deep neural networks. In *ICLR*, 2018.

S. Zagoruyko and N. Komodakis. Wide residual networks. *arXiv:1605.07146*, 2016.

M. D. Zeiler. ADADELTA: an adaptive learning rate method. *CoRR*, abs/1212.5701, 2012.

J. Zeng, T. T.-K. Lau, S. Lin, and Y. Yao. Global convergence in deep learning with variable splitting via the kurdyka-lojasiewicz property. *arXiv:1803.00225*, 2018.

C. Zhang, S. Bengio, M. Hardt, B. Recht, and O. Vinyals. Understanding deep learning requires rethinking generalization. *CoRR*, abs/1611.03530, 2016a.

G. Zhang and W. B. Kleijn. Training deep neural networks via optimization over graphs. arXiv:1702.03380, 2017.

Z. Zhang and M. Brand. Convergent block coordinate descent for training Tikhonov regularized deep neural networks. In *NeurIPS*, 2017.

Z. Zhang, Y. Chen, and V. Saligrama. Efficient training of very deep neural networks for supervised hashing. In *CVPR*, 2016b.

K. Zhong, Z. Song, P. Jain, P. L. Bartlett, and I. S. Dhillon. Recovery guarantees for one-hidden-layer neural networks. In *ICML*, 2017.

Supplementary Material: Low-Pass Filtering SGD for Recovering Flat Optima in the Deep Learning Optimization Landscape

8 NETWORK BALANCING

We consider balanced networks, i.e., networks where norms of weights in each layer are roughly the same. In this section, we present a normalization scheme utilized in Section 3 to balance the network. Let x be the input to the network, W_i be the weight matrix of the i^{th} layer, $\hat{B}_i$ denote bias matrix and $\sigma(\ )$ denote the relu nonlinearity. The output of a network with three layers; convolution, batch normalization and relu can be written as

$$f(x) = \sigma\Big(\frac{(W_1X) - E[W_1X]}{\sqrt{Var(W_1X)}}W_2 + B_2\Big). \tag{8.1}$$

Let D_i denote a diagonal normalization matrix associated with the i^{th} layer. The diagonal elements of the matrix are defined as $D_i[j,j] = \frac{1}{||W_i^j||_F + ||B_i^j||_F}$, where W_i^j is the weight matrix of j^{th} filter in the i^{th} layer. We normalize the parameters of the network as,

$$f(x) = \sigma\Big(\frac{(D_1W_1X) - E[D_1W_1X]}{\sqrt{Var(D_1W_1X)}}D_2(W_2 + B_2)D_2^{-1}\Big) \tag{8.2}$$

Note that $\hat{W}_i = D_iW_i(D_i)^{-1} = W_i$. Since $\sigma(\lambda x) = \lambda\sigma(x)$ for $\lambda \geq 0$, we can rewrite the above equation as

$$f(x) = \sigma\Big(\frac{(D_1W_1X) - E[D_1W_1X]}{\sqrt{Var(D_1W_1X)}}D_2(W_2 + B_2)\Big)D_2^{-1}. \tag{8.3}$$

We keep the multiplication with the matrix D_2^{-1} as a constant parameter in the network but it can also be combined with the parameters of the next layer. We normalize the parameters of each layer as we move from the first layer to the last layer of the network. Figure 4 shows filter wise parameter norm (D^{-1}) of LeNet and ResNet18 models trained on MNIST and CIFAR-10 data sets respectively. In Table 9, we show the mean training cross entropy loss before and after normalization.

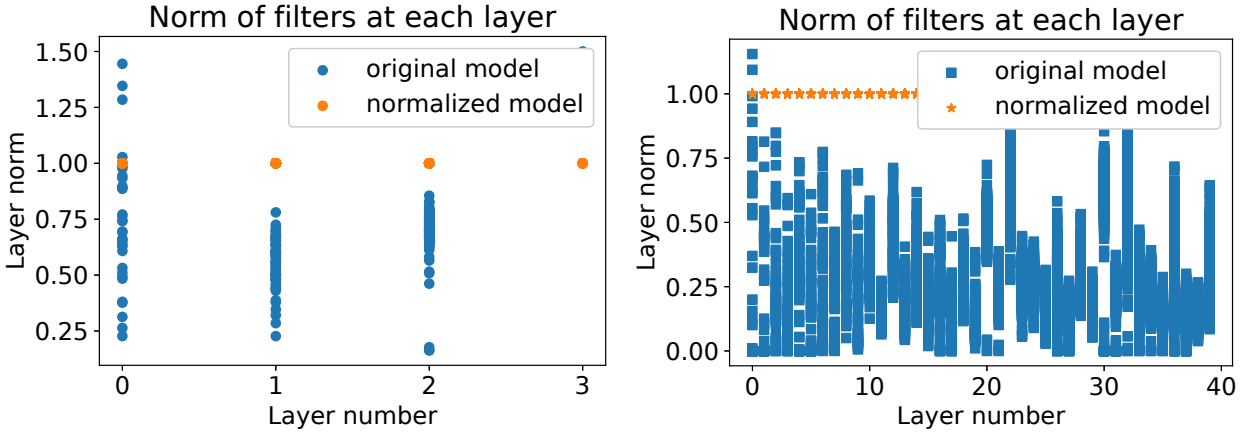

Figure 4: Norm of the filter in each layer of (**left**) LeNet and (**right**) ResNet18 networks trained on MNIST (**left**) and CIFAR-10 (**right**) data set before and after normalization.

Model	Loss before normalization	Loss after normalization
LeNet	0.0006375186720272088	0.0006375548382939456
ResNet18	0.0014627227121118522	0.0014617550651283215

Table 9: Validation loss for Lenet (MNIST) and Resnet18 (CIFAR-10) before and after normalization.

9 SHARPNESS MEASURES

In this section, we define various sharpness measures and the corresponding algorithms utilized in section 3 of the main paper to study the interplay between sharpness and generalization gap.

Let m denote the number of samples in the training data set. Let θ denote the set of network parameters. $L(\theta)$ is the loss function, whose gradient is denoted as $\nabla_\theta L(\theta)$ and the Hessian matrix is denoted as $H = \nabla^2_\theta L(\theta)$. Also, $\lambda_i(H)$ is the i^{th} eigenvalue of the Hessian.

9.1 PAC-Bayes measure

Definition 3 (PAC-Bayes measure (Jiang et al., 2020))**.** *For any $\delta > 0$ and a prior and posterior distributions over the parameters given respectively as $P = \mathcal{N}(\theta^0, \sigma^2 I)$ and $Q = \mathcal{N}(\theta^*, \sigma^2 I)$, with probability $1 - \delta$ the following bound holds:*

$$\mathbb{E}_{\theta\sim Q}[L^{true}(\theta)] \leq \mathbb{E}_{\theta\sim Q}[L(\theta)] + \sqrt{\frac{\frac{||\theta^* - \theta^0||_2^2}{2\sigma^2} + ln(\frac{m}{\delta})}{2(m-1)}}, \tag{9.1}$$

where m is the size of the training data, L^{true} is the true risk under unknown data distribution, and θ^0 are the initial network parameters. The PAC-Bayes sharpness measure is then defined as:

$$\mu_{PAC-Bayes} = \frac{||\theta^* - \theta^0||_2^2}{4\sigma^2} + 0.5 \ln\left(\frac{m}{\delta}\right), \tag{9.2}$$

where σ is chosen such that $\mathbb{E}_{\theta\sim Q}[L(\theta)] - L(\theta^*) \leq 0.1$.

Algorithm 9.1 $\mu_{PAC-Bayes}$

Input: θ^*: final weights, M: MC iterations, ψ: tolerance
Output: σ

```
σ_min = FLOAT_EPSILON_MIN
σ_max = FLOAT_EPSILON_MAX
while TRUE do
  σ = (σ_min + σ_max))/2
  l̂ = 0
  for j = 1 to M do
    θ = θ* + N(0, σ²I)
    l̂ += L̂(θ)
  end for
  l̂ = l̂/M
  d = l̂ − L(θ*)
  if ε − ψ ≤ d ≤ ε + ψ then
    return σ
  end if
  if d < ε − ψ then
    σ_min = σ
  else if d > ε + ψ then
    σ_max = σ
  end if
end while
```

9.2 ϵ-sharpness

Definition 4 (ϵ-sharpness (Keskar et al., 2017)). *ϵ-flatness captures the maximal range of the perturbations of the parameters that do not increase the value of the loss function by more than ϵ and is defined as the maximal value of the radius R of the Euclidean ball $B_2(\theta^*, R)$ centered at the local minimum (θ^*) of the loss function such that $\forall_{\theta \in B_2(\theta^*, R)} \mathcal{L}(\theta) - \mathcal{L}(\theta^*) < \epsilon$. ϵ-sharpness is the inverse of ϵ-flatness.*

Algorithm 9.2 ϵ - sharpness

Input: θ^*: final weights, ψ: tolerance, ϵ: target deviation in loss
Output: ϵ - sharpness

```
η_max =FLOAT_EPSILON_MIN
while TRUE do
  θ = θ* + η_max ∇L(θ*)
  d = L(θ) − L(θ*)
  if d < ε then
    η_max = η_max * 10
  end if
end while
η_min =FLOAT_EPSILON_MIN
while TRUE do
  η = (η_max + η_min)/2
  θ = θ* + η∇L(θ*) \\ step in full-data gradient direction
  d = L(θ) − L(θ*)
  if ε − ψ ≤ d ≤ ε + ψ then
    return 1/||θ−θ*||
  end if
  if d < ε − ψ then
    η_min = η
  else if d > ε + ψ then
    η_max = η
  end if
end while
```

9.3 Fisher Rao Norm

Definition 5 (Fisher Rao Norm (Liang et al., 2019)). *The Fisher-Rao Norm (FRN), under appropriate regularity conditions (Liang et al., 2019), is approximated as $\theta^{*T}\mathbb{E}[\nabla^2 L(\theta^*)]\theta^*$.*

FRN is calculated as $\theta^{*T} hvp(\theta^*)$ where hvp is the hessian vector product function.

9.4 Hessian based measures ($\lambda_{max}(H)$, $Trace(H)$, $d_{eff}(H)$, and $\|H\|_F$)

Definition 6. *The Hessian-based sharpness measures computed at the solution θ^* are: the Frobenius norm of the Hessian ($||H||_F$), trace of the Hessian (Trace (H), the largest eigenvalue of the Hessian ($\lambda_{max}(H)$), and the effective dimensionality of the Hessian (Maddox et al., 2020; MacKay, 1992) defined as $d_{eff} = \sum_{i=1}^{n} \frac{\lambda_i}{\lambda_i + 1}$, where λ_i is the i^{th} eigenvalue of the Hessian.*

We compute 100 eigenvalues of the Hessian of the loss function using Stochastic Lanczos quadrature algorithm as described in (Ghorbani et al., 2019). $\lambda_{max}(H)$, $Trace(H)$, and $d_{eff}(H)$ can be easily estimated from the set of 100 eigen values. Note that for any matrix A, $||A||_F^2 = \mathbf{E}_v[||Av||_2^2]$, where $v \sim \mathcal{N}(0, I)$. Therefore, we use the algorithm the following algorithm to efficiently compute the Frobenius norm.

Algorithm 9.3 $||H||_F$

Input: M: number of iterations, $hvp(v)$: Hessian-vector product
Output: $||H||_F$

```
out ← 0
for k = 1 to M do
    v^k ~ N(0, I)
    out += ||hvp(v^k)||_2^2
end for
return sqrt(out/M)
```

9.5 Shannon Entropy

Shannon entropy is a classical measure of sharpness which does not consider sharpness of the loss function in the parameter space and therefore is fundamentally different from the rest of the measures.

Definition 7 (Shannon Entropy (Pereyra et al., 2017)). *Let $f_{\theta^*}(x)[j]$ denote the probability of j^{th} class predicted by the deep learning model f_{θ^*} for input data x, and let κ be the total number of classes. The Shannon entropy at θ^* is given as:*

$$\mu_{entropy} = -\frac{1}{m}\sum_{i=1}^{m}\sum_{j=1}^{\kappa} f_{\theta^*}(x_i)[j]\log(f_{\theta^*}(x_i)[j]) \tag{9.3}$$

Algorithm 9.4 $\mu_{entropy}$

Input: f_{θ^*}: trained model
Output: Shannon Entropy

```
out ← 0
for i = 1 to N do
    for j = 1 to K do
        out+ = f_θ*(x_i)[j] × log(f_θ*(x_i)[j])
    end for
end for
return -out / N
```

9.6 Norm of the Gradient of Local Entropy

Definition 8 (Gradient of the Local Entropy (Chaudhari et al., 2017)). *The sharpness of the loss landscape at the solution θ^* is computed as the norm of the gradient of the local entropy (LE). The local entropy is*

$$F(\theta^*, \gamma) = \log\Big(\int_{\theta'} \exp(-L(\theta') - \frac{\gamma}{2}||\theta^* - \theta'||_2^2)d\theta'\Big),$$

where γ is the scoping parameter and the norm of its gradient is given as

$$\mu_{LE} = ||\nabla_{\theta^*} F(\theta^*, \gamma)|| = ||\gamma(\theta^* - \mathbb{E}_{\theta' \sim \mathcal{G}}[\theta'])|| \tag{9.4}$$

where $\mathcal{G}(\theta'; \theta, \gamma) \propto \exp\Big(-\Big(\frac{1}{m}\sum_{i=1}^{m} L(\theta')\Big) - \frac{\gamma}{2}||\theta - \theta'||_2^2\Big)$ is the Gibbs distribution.

It is prohibitive to compute the local entropy, as opposed to its gradient. We utilized the norm of the gradient of the local entropy to describe the sharpness at the minimum θ^*, instead of using the local entropy. We utilize the EntropySGD algorithm, however instead of updating weight we compute the norm of the gradient.

Algorithm 9.5 μ_{LE}

Input: θ^*: final weights, L: Langevin iterations, γ : scope, η : step size, ϵ : noise level
Output: μ_{LE}

$\theta', \mu \leftarrow \theta^*$
for k = 1 to L **do**
  B $\leftarrow$ sample mini batch
  $g = \nabla_{\theta'} L(\theta', B) - \gamma(\theta - \theta')$
  $\theta' \leftarrow \theta' - \eta g + \sqrt{\eta}\epsilon\mathcal{N}(0, I)$
  $\mu \leftarrow (1 - \alpha)\mu + \alpha\theta'$
end for
return $||\gamma(\theta^* - \mu)||$

9.7 LPF based measure

Algorithm 9.6 LPF

Input: θ^*: final weights, σ: standard deviation of Gaussian filter kernel, M: MC iterations
Output: $(L \circledast K)(\theta^*)$

$out \leftarrow 0.0$
for k = 1 to M **do**
  $\tau = \mathcal{N}(0, \sigma I)$
  $out+ = L(\theta^* + \tau)$
end for
return $out/ = M$

Remark 2. *σ is set to $\sigma = 0.01$ in PAC Bayes measure so that the deviation of loss is ≈ 0.1, as recommended by Jiang et al. (2020) (see their discussion). To be consistent, $\sigma = 0.01$ in LPF and $\epsilon = 0.1$ in the ϵ-sharpness measure.*

10 SENSITIVITY OF THE SHARPNESS MEASURES TO THE CHANGES IN THE CURVATURE OF THE SYNTHETICALLY GENERATED LANDSCAPES

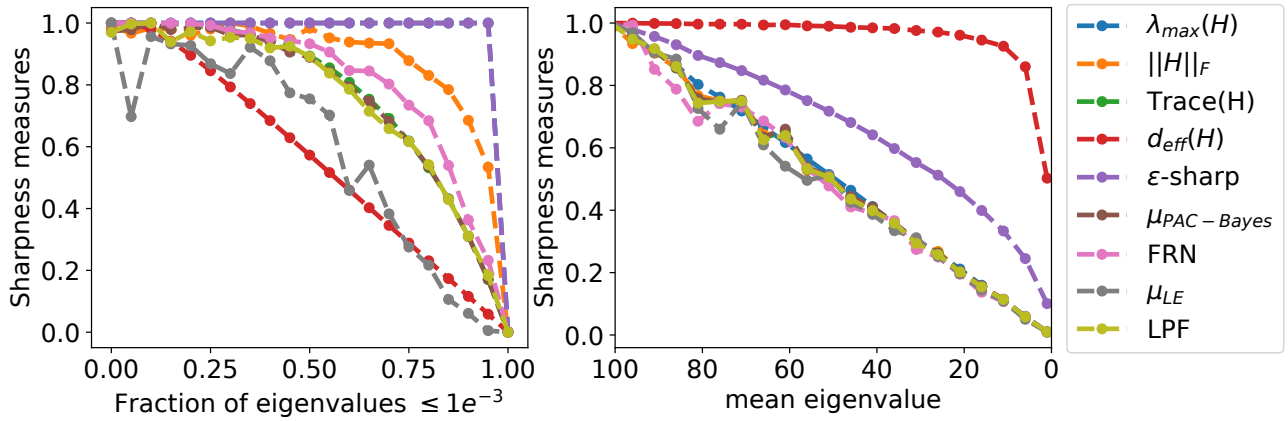

Figure 5: **Left**: The behavior of the normalized sharpness measures when the fraction of the eigenvalues of the Hessian below $1e-3$ increases from 0 to 1. **Right**: The behavior of the normalized sharpness measures when the mean eigenvalue of the Hessian is decreased from 100 to 1.

We consider a quadratic minimization problem:

$$\min_{\theta} f(\theta), \quad \text{where} \quad f(\theta) = \frac{\theta^T H \theta}{2}. \tag{10.1}$$

Note that $\nabla f = H\theta$, $\nabla^2 f = H$ and $\theta^* = \arg\min_\theta \frac{\theta^T H \theta}{2} = 0$.

In the first experiment, we randomly sample the Hessian matrix H of dimension 100 and set its K smallest eigenvalues uniformly in the interval $[1e-5,\ 1e-3]$. As the value of K is increased from 0 to 100, the loss surface becomes flatter. In the second experiment, we set the eigenvalues of Hessian H uniformly as $\mathcal{U}(K-0.10*K, K+0.10*K)$, where K is the mean eigenvalue. Intuitively, as the value of K is decreased from 100 to 1, the loss surface becomes flatter.

On the left plot in Figure 5 we show the value of the normalized sharpness measures against the fraction of eigenvalues that are $< 1e-3$. Thus as we move on the x-axis of this plot from left to right, the number of directions along which the loss landscape is flat increases. In this case LPF is the second best measure, after $d_{eff}(H)$, where by a good measure we understand the one that is sensitive to the changes in the loss landscape. On the right plot in Figure 5 we show the value of the normalized sharpness measures against the mean eigenvalue of the Hessian. Thus as we move on the x-axis of this plot from left to right, the landscape along all directions becomes flatter. In this case all measure, except ϵ-sharpness and $d_{eff}(H)$, are sensitive to the changes in the loss landscape. $d_{eff}(H)$ shows poor sensitivity to those changes. These experiments also well justify the choice of LPF based sharpness measure for the algorithm proposed in this paper.

11 TRAINING DETAILS FOR SECTION 3

11.1 Sharpness vs Generalization (training details for Section 3.1)

Following the experimental framework presented in (Jiang et al., 2020; Dziugaite et al., 2020), we trained 2916 ResNet18 models on CIFAR-10 data set by varying different model and optimizer hyper-parameters and 3 random seeds. Each model was trained using cross entropy loss function and mSGD optimizer for 300 epochs. The learning rate set to 0.1 and dropped by a factor of 0.1 at epoch 100 and 200. Since each model is trained with different hyper-parameters it is easy to overfit some models while under-fitting others. To mitigate this effect, we train each model until the cross entropy loss reaches the value of ≈ 0.01. Any model that does not reach this threshold is discarded from further analysis. We compute Kendall ranking correlation coefficient between the hyper-parameters and generalization gap and report the results in Table 10). After convergence, we balance each network according to the normalization scheme presented in Section 8 and compute sharpness measures using algorithms presented in Section 9. All the models were trained on NVIDIA RTX8000, V100, and GTX1080 GPUs on our high performance computing cluster. The total computational time is ~ 9000 GPU hours.

Measure	mo	width	wd	lr	bs	skip	bn
Emp order	-0.9712	-0.6801	-0.3135	-0.7930	0.9877	-0.2692	-0.0955

Table 10: Ranking correlation between hyper-parameter and generalization gap. The correlation sign is consistent with our intuitive understanding.

11.2 Sharpness vs Hyper-parameters (additional experimental results for Section 3.1)

As highlighted in section 3.1, we compute the Kendall ranking correlation between hyper-parameter and sharpness measures (Table 11) on ResNet18 models trained on CIFAR-10 data set as described section 11.1. We observe that momentum and weight decay are strongly negatively correlated to sharpness i.e increasing both hyper-parameters leads to flatter solution. It is also widely observed that increasing both these parameters also lead to lower generalization gap. Therefore, the table can provide us guidelines on how to design or modify deep architectures. This direction of research will be investigated in the future work.

11.3 Training details and additional experimental results for Section 3.2 (Sharpness versus generalization under data and label noise)

In order to evaluate the performance of sharpness measures to explain generalization in presence of data and label noise, we trained 10 ResNet18 models with varying level of label noise and 20 ResNet18 model with varying level of data noise on the CIFAR-10 (Krizhevsky et al., 2014a) data set (Section 3.2 in the main paper). All models were trained for 350 epochs using cross entropy loss and mSGD optimizer with a batch size of 128, weight decay of $5e^{-4}$ and momentum set to 0.9. The learning rate was set to 0.1 and dropped by a factor of 0.1 at epoch 150 and 200. The models were trained on NVIDIA RTX8000, V100, and GTX1080 GPUs on our high performance computing cluster. The total computational time is ~ 600 GPU hours. In Figure 6, we plot the

Measure	mo	width	wd	lr	bs	skip	bn
$\lambda_{\max}(H)$	-0.891	-0.063	-0.291	-0.692	0.981	0.263	0.996
$\|H\|_F$	-0.930	0.029	-0.474	-0.826	0.994	0.218	0.996
Trace (H)	-0.942	-0.127	-0.381	-0.745	0.984	-0.199	0.987
d_{eff}	-0.360	-0.137	-0.147	-0.139	0.335	-0.268	0.047
ϵ-sharpness	-0.781	0.147	-0.321	-0.772	0.967	0.509	1.000
$\mu_{PAC-Bayes}$	-0.994	0.981	-0.669	-0.971	0.996	0.322	0.996
FRN	-0.824	-0.226	-0.037	-0.545	0.855	-0.605	1.000
$\mu_{entropy}$	-0.723	-0.174	0.246	-0.352	0.718	0.613	0.950
μ_{LE}	-0.169	0.954	-0.036	-0.112	0.117	0.013	0.241
LPF	-0.994	0.874	-0.767	-0.934	0.998	-0.543	0.954

Table 11: Kendall rank correlation coefficient between various sharpness measures (rows) and hyper-parameters (columns).

values of normalized sharpness measures and generalization gap (averaged over 5 seeds) for varying level of data noise. We also report the Kendall rank correlation coefficient in the figure title.

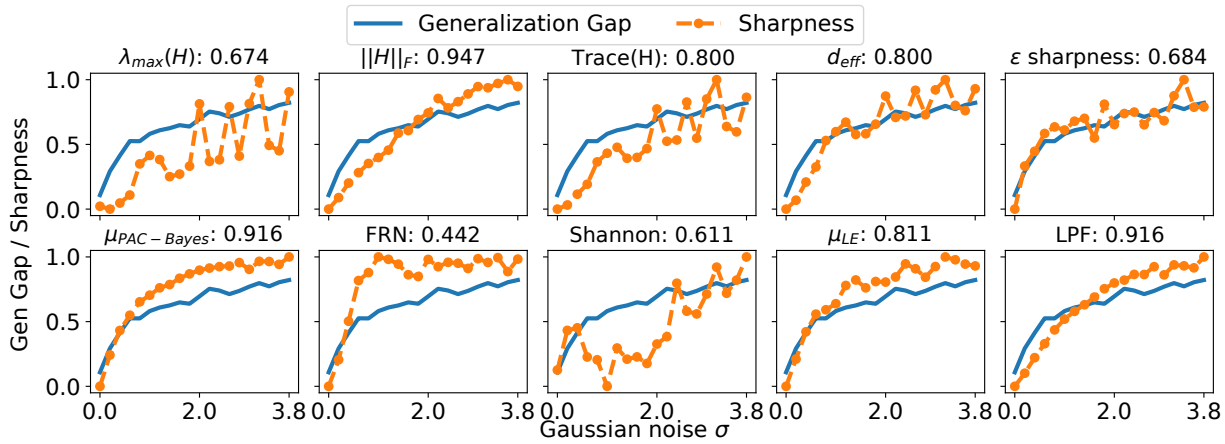

Figure 6: Normalized sharpness measures and generalization gap for varying levels of data noise. Kendall rank correlation coefficient between generalization gap and sharpness with increasing data noise are provided in the parenthesis of figure titles.

11.4 Sharpness and double descent phenomenon

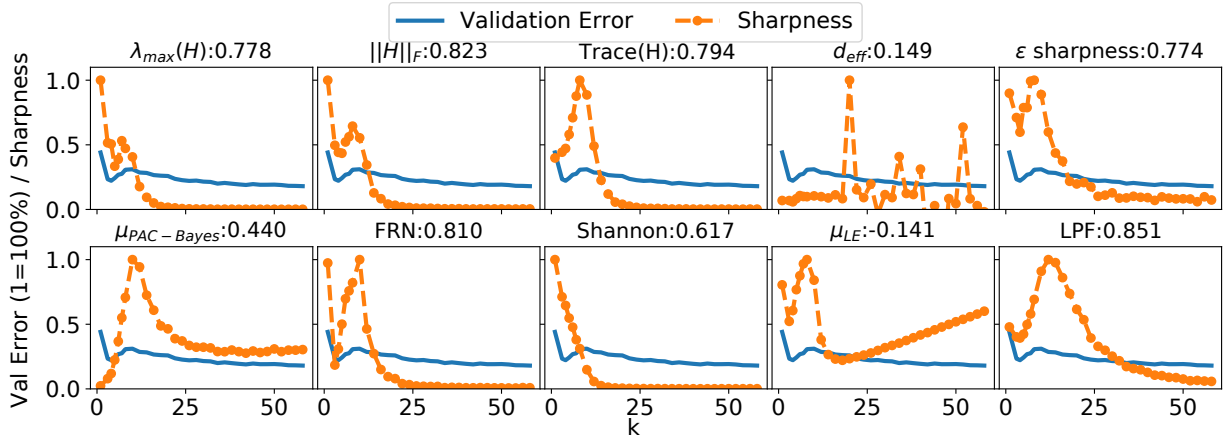

Figure 7: Normalized sharpness measures and validation error for ResNet18 when varying network width. Specifically, the widths of the consecutive layers of the network were set to $[k, 2k, 4k, 8k]$, where $k = 2, 4, \ldots, 64$. Kendall rank correlation coefficient is provided in the titles above the figures.

As we increase the complexity of deep networks, we sometimes observe the double descent phenomenon, where increasing the network capacity may lead to the temporal increase of the test loss. We evaluate the sharpness measures against the double descent behavior for ResNet18 model. We follow the experimental framework presented in (Nakkiran et al., 2020) and set the widths of the consecutive layers of the ResNet18 model as $[k, 2k, 4k, 8k]$. The value of k is varied in the range $[1, 64]$. Note that for a standard ResNet18 model $k = 64$. The models were trained on CIFAR-10 data set for 4000 epochs with 20% label noise using an Adam optimizer (Kingma and Ba, 2014) with a batch size of 128 and a constant learning rate set to $1e^{-4}$. All the models were trained on NVIDIA RTX8000, V100, and GTX1080 GPUs on our high performance computing cluster. The total compute time is ~ 300 GPU hours. In Figure 7 we plot the validation error and normalized sharpness measures when varying network width. It can be observed that most of the flatness measures can track the double descent phenomenon, including $\lambda_{\max}(H)$, $\|H\|_F$, Trace (H), ϵ-sharpness, FRN, and LPF. At the same time, LPF based measure admits the highest Kendall rank correlation coefficient from among all the flatness measures.

12 TRAINING DETAILS FOR SECTION 6

12.1 Codes

We coded all our experiments in PyTorch (Paszke et al., 2017). In all the experiments, we utilized the code for the SAM optimizer (Foret et al., 2021) available at https://github.com/davda54/sam that we treated as a baseline code for developing Adaptive Smooth with Denoising (ASO) and LPF-SGD. In terms of mSGD, this optimizer is included in the PyTorch environment. For E-SGD, we utilized public codes released by Pittorino et al. (2021) available at https://openreview.net/forum?id=xjXg0bnoDmS. Finally, the codes for our method, LPF-SGD, will be open-sourced and publicly released.

12.2 Image Classification

12.2.1 Image Classification: ResNets on CIFAR 10/100 data set with no data augmentations

Image classification models such as ResNets (He et al., 2016) are prone to extreme over-fitting when trained without any data augmentation. The flatness based optimizers such as SAM and E-SGD prevent model overfitting by seeking flat minimizers. Therefore, we first compare the performance of mSGD, E-SGD, ASO, SAM, and LPF-SGD on ResNet architectures without performing any data augmentation. Specifically, we trained ResNet18 model available in torchvision (Paszke et al., 2017) on TinyImageNet (Russakovsky et al., 2015) and ImageNet (Russakovsky et al., 2015) data sets, and a modified version of ResNet18,50,101 (He et al., 2016) available at https://github.com/kuangliu/pytorch-cifar on CIFAR-10 and CIFAR-100 (Krizhevsky et al., 2014a) data sets. The modification was small and was only done to accommodate 32×32 image sizes in CIFAR data set. We also train a LeNet (LeCun et al., 1989a) model available at https://github.com/pytorch/examples/blob/master/mnist/main.py on MNIST (LeCun et al., 1989b) data set.

We set the radius for SAM to $\rho = 0.05$ and keep it fixed for the entire training procedure, as suggested by the authors (note that the authors did extreme grid search and found little to no difference when changing parameter ρ). For E-SGD, we performed a grid search over $\gamma_0 = \{0.1, 0.5, 0.05\}$, $\gamma_1 = \{0.0001, 0.0005, 0.005\}$ and $\eta = \{0.05, 0.01, 0.1\}$. The parameters $(\gamma_0, \gamma_1, \eta) = (0.5, 0.0001, 0.05)$ were found to be the best performing parameters. The SGLD iterations (M) were set to 5. The learning rate of the base optimizer was also increased by a factor of M. For ASO, the parameter a was tuned by performing a grid search over the set $\{0.001, 0.005, 0.009, 0.0375, 0.07\}$. We find $a = 0.009$ to be the best performing hyper-parameter as suggested by the authors. The hyper-parameters common to mSGD, E-SGD, ASO, SAM, and LPF-SGD optimizers are provided in the Table 12, while the individual hyper-parameters for each optimizer are provided in Table 13. Note that for E-SGD the total number of training epochs and the epoch at which we drop the learning rate are divided by $M = 5$, i.e.: the total number of stochastic gradient Langevin dynamics steps per iteration. The models were trained on NVIDIA RTX8000, V100, and GTX1080 GPUs on our high performance computing cluster. The total computational time is ~ 1500 GPU hours. The plots showing epoch vs error curves are captured in Figure 8 and Figure 9.

Finally, after convergence, we balance our network using the normalization scheme described in section 8 and compute various sharpness measures on the best performing model on CIFAR-10 and CIFAR-100 data set for baseline mSGD and top two performing optimizers (SAM and LPF-SGD). Table 14 shows normalized sharpness measures and the corresponding validation error. Note that LPF-SGD leads to a lower value of the LPF based

Dataset	Model	BS	WD	MO	Epochs	LR (Policy)
MNIST	LeNet	128	$5e^{-4}$	0.9	150	0.01 (x 0.1 at ep=[50,100])
CIFAR10, 100	ResNet-18, 50, 101	128	$5e^{-4}$	0.9	200	0.1 (x 0.1 at ep=[100,120])
TinyImageNet	ResNet-18	128	$1e^{-4}$	0.9	100	0.1 (x 0.1 at ep=[30, 60, 90])
ImageNet	ResNet-18	256	$1e^{-4}$	0.9	100	0.1 (x 0.1 at ep=[30, 60, 90])

Table 12: Training hyper-parameters common to all optimizers used for obtaining Table 3. BS: batch size, WD: weight decay, and MO: momentum coefficient.

Dataset	Model	E-SGD	ASO	SAM	LPF-SGD
		(γ_0, γ_1, η, M)	a	ρ	(M, γ (policy))
MNIST	LeNet	(0.5,0.0001,0.05,5)	0.009	0.05	(1,0.001 (fixed))
CIFAR	ResNet18,50,101	(0.5,0.0001,0.05,5)	0.009	0.05	(1,0.002 (fixed))
TinyImageNet	ResNet18	(0.5,0.0001,0.05,5)	0.009	0.05	(1,0.001 (fixed))
ImageNet	ResNet18	(0.5,0.00001,0.05,5)	0.009	0.05	(1,0.0005 (fixed))

Table 13: Summary of E-SGD, ASO, SAM and LPF-SGD hyper-parameters used for obtaining Table 3.

sharpness measure and a smaller test error.

Data	Model	Opt	$\|\|H\|\|_F$	ϵ-sharp	μ_{PAC}	FRN	$\mu_{entropy}$	LPF	val-err
CIFAR10	ResNet18	mSGD	1.00	0.52	1.00	1.00	1.00	1.00	11.49
		SAM	0.34	0.36	0.70	0.45	0.61	0.40	10.00
		LPF-SGD	0.71	1.00	0.58	0.46	0.21	0.17	9.04
	ResNet50	mSGD	1.00	0.43	1.00	1.00	1.00	1.00	10.21
		SAM	0.37	0.41	0.66	0.64	0.60	0.45	8.81
		LPF-SGD	0.79	1.00	0.88	0.56	0.24	0.28	8.60
	ResNet101	mSGD	1.00	0.59	1.00	1.00	1.00	1.00	9.49
		SAM	0.36	0.46	0.70	0.59	0.67	0.51	8.33
		LPF-SGD	0.75	1.00	0.99	0.49	0.33	0.34	8.69
CIFAR100	ResNet18	mSGD	0.18	0.64	1.00	0.29	0.65	1.00	38.29
		SAM	0.09	0.47	0.78	0.20	0.50	0.52	36.17
		LPF-SGD	1.00	1.00	0.59	1.00	1.00	0.90	30.02
	ResNet50	mSGD	0.54	0.50	1.00	0.62	1.00	1.00	35.55
		SAM	0.18	0.54	0.78	0.27	0.64	0.46	33.15
		LPF-SGD	1.00	1.00	0.53	1.00	0.65	0.41	30.64
	ResNet101	mSGD	1.00	0.56	1.00	1.00	0.34	1.00	32.73
		SAM	0.46	0.43	0.64	0.55	0.25	0.41	30.70
		LPF-SGD	0.59	1.00	0.44	0.44	1.00	0.12	29.89

Table 14: Normalized sharpness measures and validation error for ResNet18,50,101 models trained on CIFAR-10 and CIFAR-100 data sets using standard SGD, SAM and LPF-SGD.

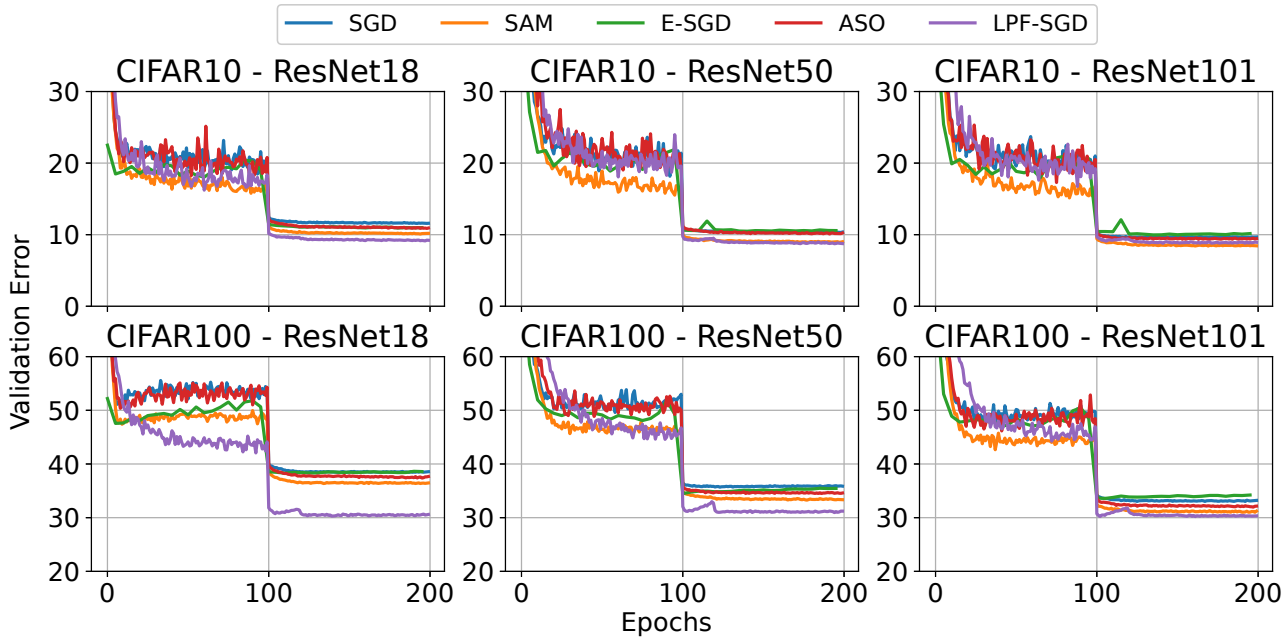

Figure 8: Validation error vs epochs for ResNet18 (left), ResNet50 (middle), and ResNet101 (right) models trained on CIFAR-10 (top) and CIFAR-100 (bottom) data sets using mSGD, SAM, and LPF-SGD.

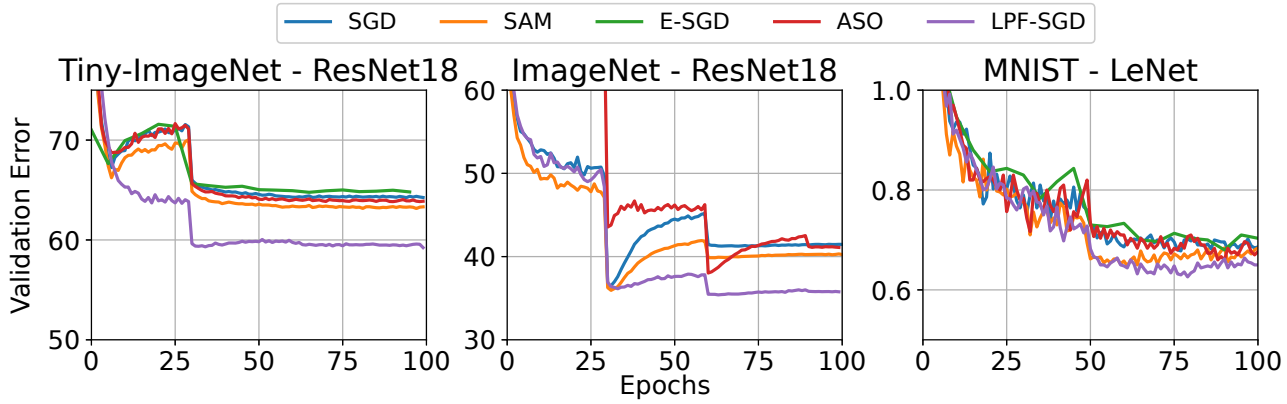

Figure 9: Validation error vs epochs for TinyImageNet (left), ImageNet (middle), and MNIST (right) data sets. TinyImage and ImageNet data sets was used to train the ResNet18 model while MNIST data set was used to train the LeNet model.

12.2.2 Image Classification: ResNet18 on ImageNet data set with basic data augmentation

In the second set of experiments, we train a ResNet18 model on ImageNet data set utilizing the standard training hyper-parameters available in pytorch (Paszke et al., 2017). The hyper-parameters for mSGD, ASO, SAM, and LPF-SGD optimizers are provided in the Table 15. The models were trained on NVIDIA RTX8000, V100, and GTX1080 GPUs on our high performance computing cluster. The total computational time is $\sim$ 1500 GPU hours.

Dataset	Model	BS	WD	MO	Epochs	LR (Policy)	ASO	SAM	LPF-SGD
							a	ρ	(M, γ, α)
ImageNet	ResNet-18	256	$1e^{-4}$	0.9	100	0.1 (x 0.1 at ep=[30, 60, 90])	0.009	0.05	(8, 0.0001, 5)

Table 15: Training hyper-parameters used for obtaining Table 4. BS: batch size, WD: weight decay, and MO: momentum coefficient.

12.2.3 Image Classification: Wide ResNets, ShakeShake and PyramidNets on CIFAR 10/100 data set with various data augmentations schemes

In the third set of experiments, we trained WRN16-8 and WRN28-10 (Zagoruyko and Komodakis, 2016) models available at https://github.com/xternalz/WideResNet-pytorch, ShakeShake (26 2x96d) (Gastaldi, 2017) available at https://github.com/hysts/pytorch_shake_shake, and PyramidNet-110($\alpha = 270$) (Dongyoon et al., 2017) and PyramidNet-272($\alpha = 200$) (Dongyoon et al., 2017) models available at https://github.com/dyhan0920/PyramidNet-PyTorch. We utilized three progressively increasing augmentation schemes: basic (random cropping and horizontal flipping), basic + cutout (DeVries and Taylor, 2017), and basic + cutout + auto-augmentation (Cubuk et al., 2019). The cutout scheme is available at https://github.com/davda54/sam and the auto-augmentation scheme is available at https://github.com/4uiiurz1/pytorch-auto-augment.

The hyper-parameters for SAM, E-SGD, and ASO were tuned as described in the Subsection 12.2.1. For LPF-SGD, we perform a grid search over the radius γ_0 and number of MC iterations (M) for experiments with WRN16-8 and WRN28-10 model and use the best performing hyper-parameters ($M = 8$, $\gamma_0 = 5e^{-4}$, and $\alpha = 15$) for the remaining experiments. Table 16 shows various hyper-parameters common to mSGD, E-SGD, ASO, SAM, and LPF-SGD optimizers, and Table 17 shows individual hyper-parameters for LPF-SGD optimizer (for SAM hyperparameter ρ is fixed to 0.05, for ASO hyperparameter a is fixed to 0.009, and for E-SGD hyper-parameters (γ_0, γ_1, η, L) are fixed to $(0.5, 0.0001, 0.05, 5)$, and thus we do not report these hyper-parameters in the table).

Model	BS	WD	MO	Epochs	LR(Policy)	
					CIFAR-10	CIFAR-100
WRN16-8	128	$5e^{-4}$	0.9	200	0.1(× 0.2 at [60,120,160])	
WRN28-10	128	$5e^{-4}$	0.9	200	0.1(× 0.2 at [60,120,160])	
ShakeShake (26 2x96d)	128	$1e^{-4}$	0.9	1800	0.2(cosine decrease)	
PyNet110	128	$1e^{-4}$	0.9	300	0.1(× 0.1 at [100,150])	0.5(× 0.1 at [100,150])
PyNet272	128	$1e^{-4}$	0.9	300	0.1(× 0.1 at [100,150])	

Table 16: Training hyper-parameters common to all optimizers used to obtain Table 5. BS: batch size, WD: weight decay, MO: momentum coefficient, and LR: learning rate.

Model	Aug	M	CIFAR-10		CIFAR-100	
			γ_0	α (policy)	γ_0	α (policy)
WRN16-8	Basic	8	0.0005	15	0.0005	15
	Basic+Cut	8	0.0005	15	0.0005	15
	Basic+Cut+AA	8	0.0005	15	0.0005	15
WRN28-10	Basic	8	0.0005	35	0.0005	25
	Basic+Cut	8	0.0005	35	0.0005	25
	Basic+Cut+AA	8	0.0005	35	0.0007	15
ShakeShake 26 2x96d	Basic	8	0.0005	15	0.0005	15
	Basic+Cut	8	0.0005	15	0.0005	15
	Basic+Cut+AA	8	0.0005	15	0.0005	15
PyNet110	Basic	8	0.0005	15	0.0005	15
	Basic+Cut	8	0.0005	15	0.0005	15
	Basic+Cut+AA	8	0.0005	15	0.0005	15
PyNet272	Basic	8	0.0005	15	0.0005	15
	Basic+Cut	8	0.0005	15	0.0005	15
	Basic+Cut+AA	8	0.0005	15	0.0005	15

Table 17: Hyper-parameters for LPF-SGD optimizer.

In Figures 10, 11, 12, 13, and 14 we provide error vs epoch curves. All the models were trained on NVIDIA RTX8000, V100, and GTX1080 GPUs on our high performance computing cluster. The total computational time is ~ 6000 GPU hours.

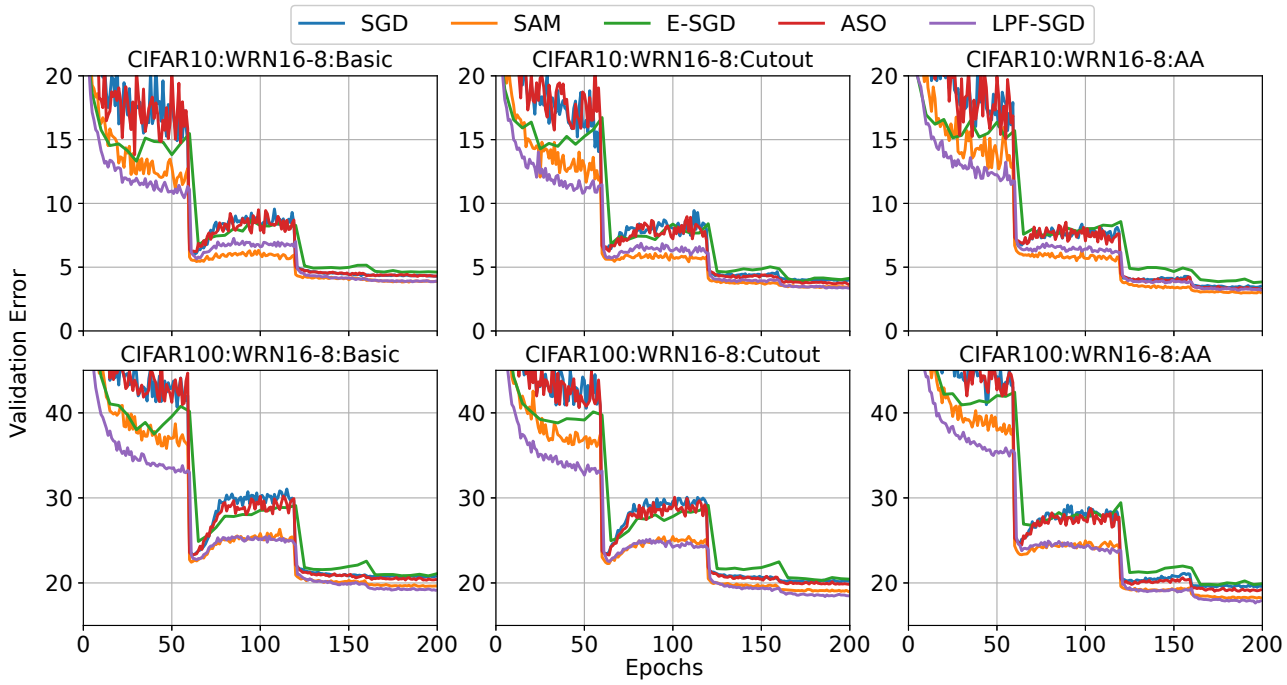

Figure 10: Validation error vs epochs for WideResNet 16-8 model trained on CIFAR-10 (top) and CIFAR-100 (bottom) data sets with Basic (left), Basic + Cutout (middle) and Basic+AutoAugmentation+Cutout (right) augmentation schemes using mSGD, SAM, E-SGD (E-SGD), ASO, and LPF-SGD optimization algorithms.

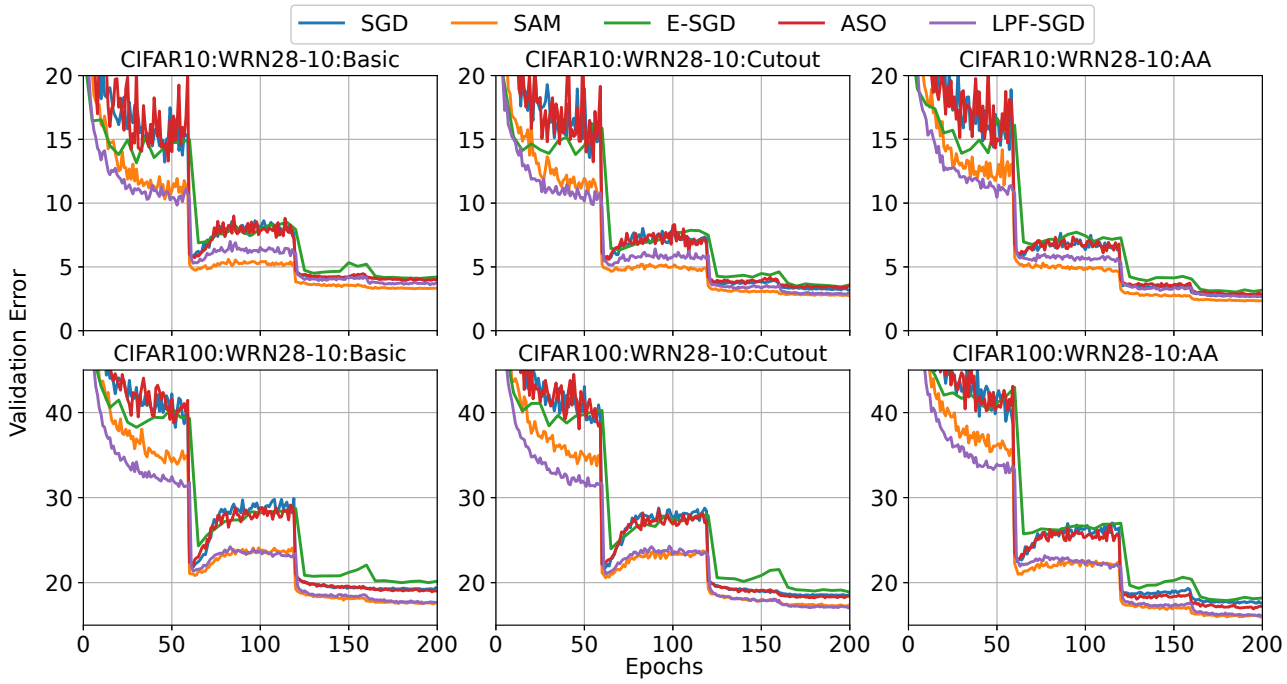

Figure 11: Validation error vs epochs for WideResNet 28-10 model trained on CIFAR-10 (top) and CIFAR-100 (bottom) data sets with Basic (left), Basic + Cutout (middle) and Basic+AutoAugmentation+Cutout (left) augmentation schemes using mSGD, SAM, E-SGD, ASO, and LPF-SGD optimization algorithms.

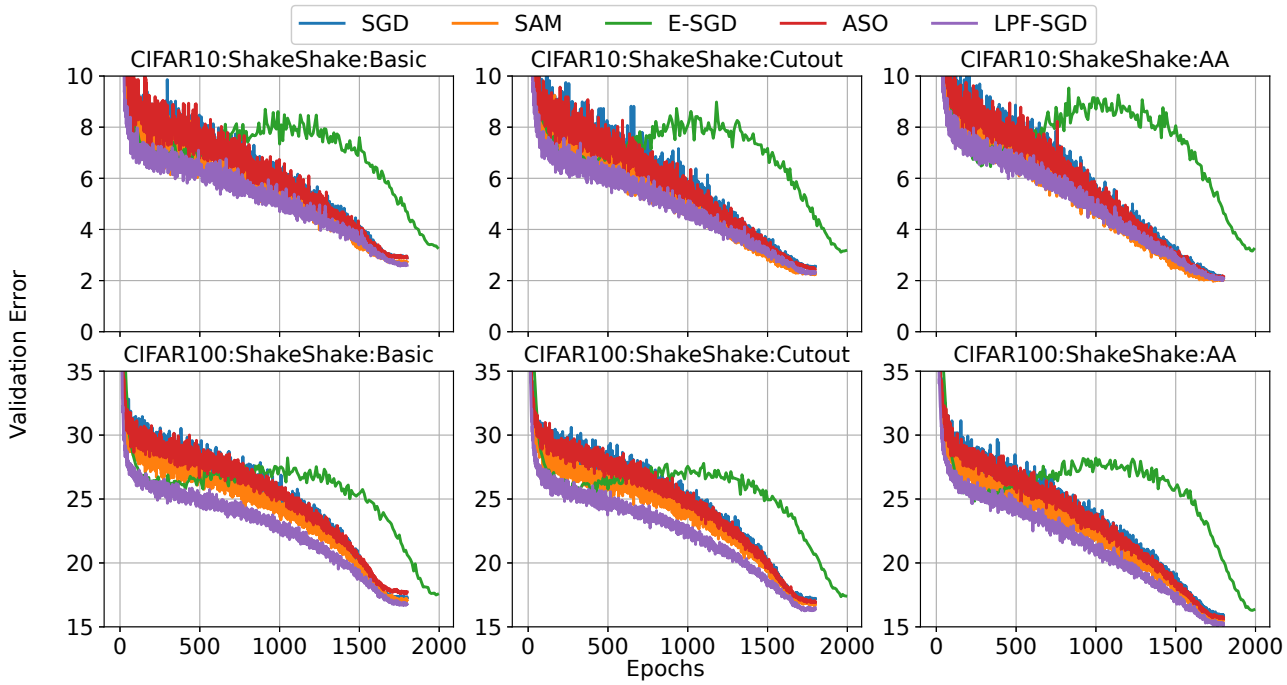

Figure 12: Validation error vs epochs for ShakeShake (26 2x96d) model trained on CIFAR-10 (top) and CIFAR-100 (bottom) data sets with Basic (left), Basic + Cutout (middle) and Basic+AutoAugmentation+Cutout (right) augmentation schemes using mSGD, SAM, E-SGD, ASO, and LPF-SGD optimization algorithms.

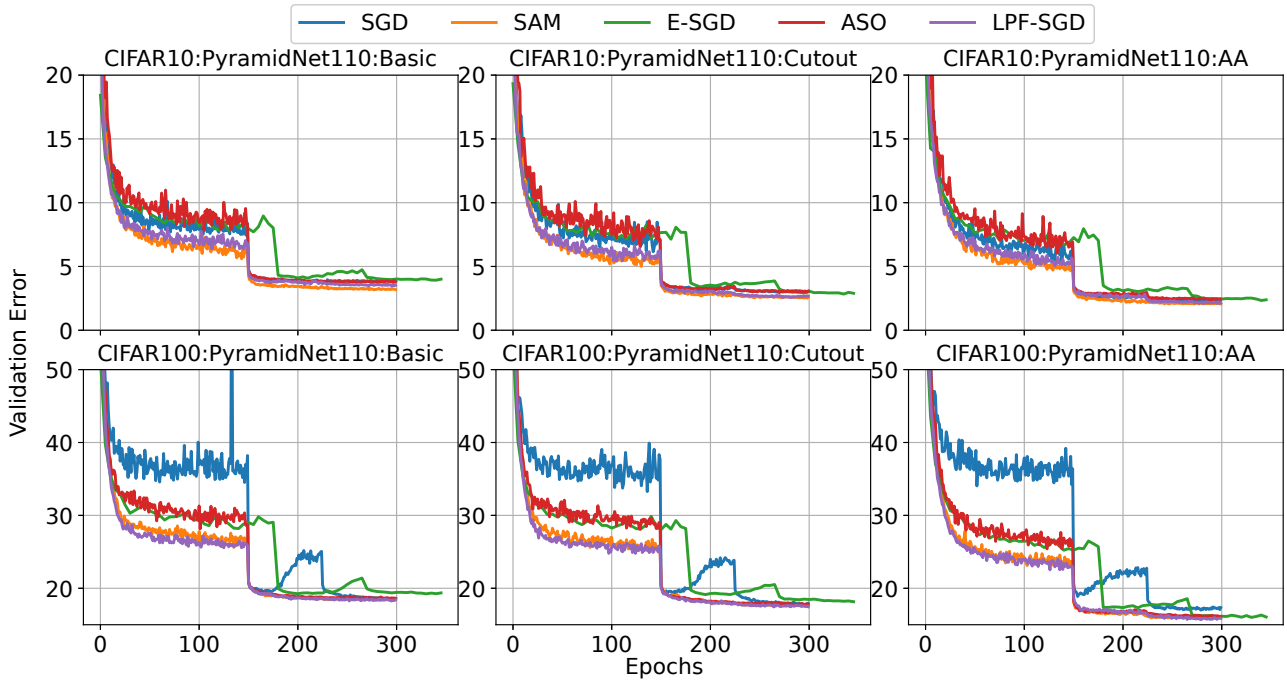

Figure 13: Validation error vs epochs for PyramidNet110 ($\alpha = 270$) model trained on CIFAR-10 (top) and CIFAR-100 (bottom) data sets with Basic (left), Basic + Cutout (middle) and Basic+AutoAugmentation+Cutout (right) augmentation schemes using mSGD, SAM, E-SGD, ASO, and LPF-SGD optimization algorithms.

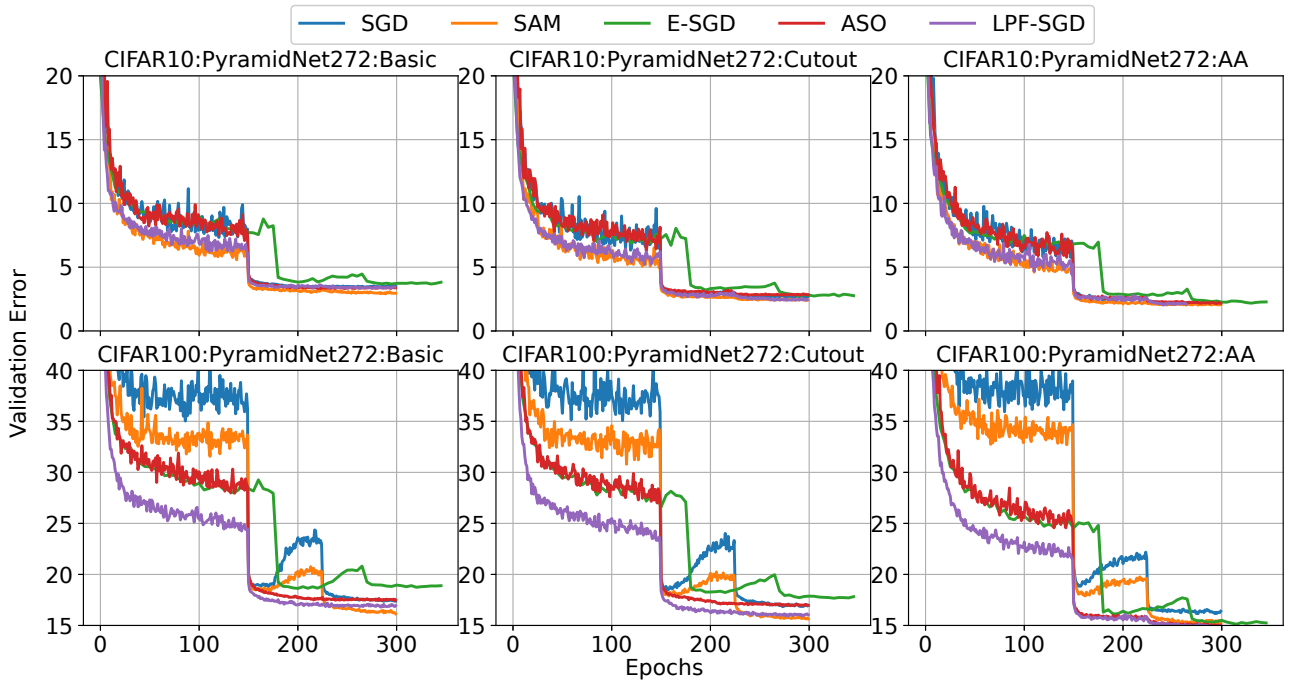

Figure 14: Validation error vs epochs for PyramidNet272 ($\alpha = 200$) model trained on CIFAR-10 (top) and CIFAR-100 (bottom) data sets with Basic (left), Basic + Cutout (middle) and Basic+AutoAugmentation+Cutout (right) augmentation schemes using mSGD, SAM, E-SGD, ASO, and LPF-SGD optimization algorithms.

12.2.4 Comparison with ASAM

The hyper-parameter ρ for ASAM was tuned by performing grid search over the set $\{0.1, 0.5, 1.0\}$ and $\rho = 1.0$ was found to be the best performing parameter. The ASAM codes are available at https://github.com/davda54/sam.

12.3 Machine Translation

We train a mini-transformer model based on Vaswani et al. (2017) with 3 encoder layers and 3 decoder layers to perform German to English translation on WMT 2014 data set (Bojar et al., 2014). The dropout rate of the model was set to 0.1 while the size of the embedding layer was set to 512. Similarly to Vaswani et al. (2017), we utilize byte-pair encoding and construct a common vocabulary of 32,000 tokens. The model was trained using Adam (Kingma and Ba, 2014) as the base optimizer, natively available in pytorch. For all other optimizers, we modify the codes described in section 12.1 to change the base optimizer from SGD to Adam. The hyper-parameters common to Adam, E-Adam, ASO-Adam, SAM-Adam, and LPF-Adam are provided in the Table 18. The hyper-parameter ρ in SAM-Adam was tuned by performing a grid search over the set $\{0.01, 0.1, 0.25, 0.5, 1.0\}$ and $\rho = 0.01$ was found to be the best performing hyper-parameter. For E-Adam, we perform grid search over $\gamma_0 = \{0.1, 0.5, 0.05\}$, $\gamma_1 = \{0.0001, 0.0005, 0.005\}$, and $\eta = \{0.05, 0.01, 0.1\}$. $(g_0, g_1, \eta) = (0.5, 0.0001, 0.1)$ were found to be the best performing hyper-parameters. The SGLD iterations (M) were set to 5 and the learning rate of the base optimizer (Adam) was also increased by a factor of M. The smooth out parameter a in ASO-Adam was tuned by performing grid search over the set $\{0.001, 0.005, 0.009, 0.05\}$ and $a = 0.009$ was found to be the best performing hyper-parameter. Finally, the LPF-Adam parameters were tuned by performing grid search over the set $\gamma = \{0.001, 0.0001, 0.0005\}$ and $\gamma = 0.0005$ was found to be the best performing hyper-parameters. The value of M was set to 8 and α was set to 15.

Dataset	Model	BS	WD	(β_1, β_2)	Itr	LR (Policy)
WMT2014	Mini-Trans	256	$1e^{-4}$	$(0.9, 0.99)$	100k	0.0005 ($\times$ 0.1 ReduceLROnPlateau)

Table 18: Training hyper-parameters common to all optimizers used for obtaining results in Table 8. BS: batch size, WD: weight decay.

13 ABLATION STUDIES

In order to understand the impact of various hyper-parameters of the LPF-SGD optimizer on its performance, we perform three sets of ablation studies. First, we study the impact of the number of MC iterations (M). Specifically, we train WRN16-8, WRN28-10, and Pyramidnet-110 models on CIFAR-10 and CIFAR-100 data sets with various different data augmentation techniques using LPF-SGD optimizer while varying the number of MC iterations (M) over the set $\{1, 2, 4, 8\}$. For $M > 8$ the computational cost is $> 2\times$SGD which may not be viable for empirical purposes. Table 19 shows that increasing M leads to the drop in the test error, as expected, though the differences in performance between different values of M are not drastic.

		CIFAR-10				CIFAR-100			
		LPF-SGD (M)				LPF-SGD (M)			
Model	Augmentation	M=1	M=2	M=4	M=8	M=1	M=2	M=4	M=8
WRN16-8	Basic	$4.1_{\pm 0.1}$	$4.0_{\pm 0.1}$	$3.8_{\pm 0.1}$	$\mathbf{3.7_{\pm <0.1}}$	$19.3_{\pm 0.2}$	$19.1_{\pm 0.2}$	$19.1_{\pm 0.1}$	$\mathbf{18.9_{\pm 0.1}}$
	Basic+Cut	$3.6_{\pm 0.0}$	$3.5_{\pm <0.1}$	$3.4_{\pm 0.1}$	$\mathbf{3.2_{\pm 0.1}}$	$18.9_{\pm 0.2}$	$18.7_{\pm 0.2}$	$18.5_{\pm 0.2}$	$\mathbf{18.3_{\pm 0.1}}$
	Basic+AA+Cut	$3.1_{\pm 0.1}$	$3.2_{\pm 0.0}$	$\mathbf{3.1_{\pm <0.1}}$	$\mathbf{3.1_{\pm 0.1}}$	$17.9_{\pm 0.2}$	$17.8_{\pm 0.0}$	$17.6_{\pm 0.2}$	$\mathbf{17.6_{\pm 0.1}}$
WRN28-10	Basic	$3.8_{\pm 0.1}$	$3.7_{\pm 0.0}$	$3.6_{\pm 0.1}$	$\mathbf{3.5_{\pm 0.1}}$	$17.7_{\pm 0.1}$	$17.7_{\pm 0.2}$	$17.5_{\pm 0.1}$	$\mathbf{17.4_{\pm 0.1}}$
	Basic+Cut	$3.1_{\pm <0.1}$	$3.0_{\pm <0.1}$	$3.0_{\pm 0.1}$	$\mathbf{2.7_{\pm <0.1}}$	$17.1_{\pm 0.1}$	$17.2_{\pm 0.2}$	$16.9_{\pm 0.2}$	$\mathbf{16.9_{\pm 0.2}}$
	Basic+AA+Cut	$2.5_{\pm 0.1}$	$2.6_{\pm <0.1}$	$2.6_{\pm <0.1}$	$\mathbf{2.5_{\pm 0.1}}$	$15.9_{\pm 0.1}$	$16.2_{\pm 0.2}$	$15.8_{\pm 0.2}$	$\mathbf{15.9_{\pm 0.1}}$
PyNet110 ($\alpha = 270$)	Basic	$3.7_{\pm 0.1}$	$3.7_{\pm <0.1}$	$3.6_{\pm 0.1}$	$\mathbf{3.4_{\pm 0.1}}$	$18.5_{\pm 0.2}$	$18.5_{\pm 0.3}$	$\mathbf{18.1_{\pm 0.1}}$	$18.2_{\pm 0.3}$
	Basic+Cut	$2.9_{\pm 0.1}$	$2.9_{\pm 0.1}$	$2.9_{\pm 0.1}$	$\mathbf{2.5_{\pm <0.1}}$	$17.8_{\pm 0.4}$	$17.8_{\pm 0.2}$	$\mathbf{17.2_{\pm 0.1}}$	$17.3_{\pm 0.2}$
	Basic+AA+Cut	$2.4_{\pm 0.1}$	$2.4_{\pm 0.1}$	$2.5_{\pm <0.1}$	$\mathbf{2.1_{\pm 0.0}}$	$15.8_{\pm 0.1}$	$15.7_{\pm 0.1}$	$15.8_{\pm 0.2}$	$\mathbf{15.6_{\pm 0.1}}$

Table 19: Validation error rate with 95% confidence interval for LPF-SGD optimizer trained with different values of MC iterations (M).

		CIFAR-10		CIFAR-100	
		LPF-SGD (γ pol)		LPF-SGD (γ pol)	
Model	Aug	Fixed	Cosine	Fixed	Cosine
WRN16-8	Basic	$3.9_{\pm 0.1}$	$\mathbf{3.7_{\pm <0.1}}$	$20.1_{\pm 0.1}$	$\mathbf{18.9_{\pm 0.1}}$
	Basic+Cut	$3.5_{\pm 0.1}$	$\mathbf{3.2_{\pm 0.1}}$	$19.4_{\pm 0.2}$	$\mathbf{18.3_{\pm 0.1}}$
	Basic+AA+Cut	$3.1_{\pm 0.1}$	$\mathbf{3.1_{\pm 0.1}}$	$19.0_{\pm 0.1}$	$\mathbf{17.6_{\pm 0.1}}$
WRN28-10	Basic	$3.8_{\pm 0.1}$	$\mathbf{3.5_{\pm 0.1}}$	$18.6_{\pm 0.1}$	$\mathbf{17.4_{\pm 0.1}}$
	Basic+Cut	$2.9_{\pm 0.1}$	$\mathbf{2.7_{\pm <0.1}}$	$18.0_{\pm 0.2}$	$\mathbf{16.9_{\pm 0.2}}$
	Basic+AA+Cut	$2.6_{\pm 0.1}$	$\mathbf{2.5_{\pm 0.1}}$	$17.1_{\pm 0.1}$	$\mathbf{15.9_{\pm 0.1}}$
PyNet110 ($\alpha = 270$)	Basic	$3.6_{\pm 0.1}$	$\mathbf{3.4_{\pm 0.1}}$	$\mathbf{18.0_{\pm 0.1}}$	$18.2_{\pm 0.3}$
	Basic+Cut	$2.8_{\pm 0.1}$	$\mathbf{2.5_{\pm 0.1}}$	$\mathbf{17.0_{\pm 0.1}}$	$17.3_{\pm 0.2}$
	Basic+AA+Cut	$2.3_{\pm 0.1}$	$\mathbf{2.1_{\pm <0.1}}$	$15.8_{\pm 0.3}$	$\mathbf{15.6_{\pm 0.1}}$

Table 20: Validation error rate with 95% confidence interval for LPF-SGD optimizer trained with different policy of γ hyper-parameter.

In the second set of experiments, we study the impact of γ policy on the test error rate. Specifically, we consider two policies for setting up γ parameter, a fixed value as well as cosine policy given in Equation 4.4 (γ parameter is chosen via hyper-parameter search). Similarly to the previous study, we train WRN16-8, WRN28-10, and Pyramidnet-110 models on CIFAR-10 and CIFAR-100 data sets with various different data augmentation techniques. Table 20 shows that cosine policy is superior and corresponds to progressively increasing the area of the loss surface that is explored (as we converge to the flat valley).

In the third set of experiments, we study the impact of two different kinds of co-variance matrix (Σ) on the test error rate. Specifically, we train WRN16-8 and WRN28-10 models on CIFAR-10 and CIFAR-100 data sets with isotropic co-variance (identity) matrix and compare its performance with the parameter dependent anisotropic approach as in LPF-SGD (note that the parameter-dependent anisotropic strategy is equivalent to using isotropic covariance on a balanced network). Table 21 captures the results and reveals that anisotropic strategy is superior to isotropic in terms of the final performance.

Finally, in order to verify whether the performance gain of the proposed method can be achieved by prolonging the number of training epochs for the mSGD method, we trained WRN-16-8 model and WRN-28-10 model on CIFAR-10 and CIFAR-100 data set for 400 epochs (instead of the current setting of 200 epochs). In Table 22, we

show that even with prolonged training, the mSGD is not able to match the performance of LPF-SGD.

Model	Aug	CIFAR-10		CIFAR-100	
		Iso	An-Iso	Iso	An-Iso
WRN 16-8	Basic	$3.8_{\pm 0.0}$	$\mathbf{3.7_{\pm <0.1}}$	$19.7_{\pm 0.2}$	$\mathbf{18.9_{\pm 0.1}}$
	Basic+Cut	$3.3_{\pm 0.0}$	$\mathbf{3.2_{\pm 0.1}}$	$19.0_{\pm 0.2}$	$\mathbf{18.3_{\pm 0.1}}$
	Basic+AA+Cut	$3.3_{\pm 0.1}$	$\mathbf{3.1_{\pm 0.1}}$	$18.1_{\pm 0.2}$	$\mathbf{17.6_{\pm 0.1}}$
WRN 28-10	Basic	$3.7_{\pm 0.1}$	$\mathbf{3.5_{\pm 0.1}}$	$18.1_{\pm 0.1}$	$\mathbf{17.4_{\pm 0.1}}$
	Basic+Cut	$2.9_{\pm 0.0}$	$\mathbf{2.7_{\pm <0.1}}$	$17.6_{\pm 0.1}$	$\mathbf{16.9_{\pm 0.2}}$
	Basic+AA+Cut	$2.8_{\pm 0.1}$	$\mathbf{2.5_{\pm 0.1}}$	$16.5_{\pm 0.1}$	$\mathbf{15.9_{\pm 0.1}}$

Table 21: Validation error rate with 95% confidence interval for LPF-SGD optimizer trained with isotropic and parameter dependent anisotropic covariance matrix.

Model	Aug	CIFAR-10			CIFAR-100		
		mSGD (200 epochs)	mSGD (400 epochs)	LPF-SGD (200 epochs)	mSGD (200 epochs)	mSGD (400 epochs)	LPF-SGD (200 epochs)
WRN16-8	B	$4.2_{\pm 0.2}$	$4.1_{\pm 0.0}$	$\mathbf{3.7_{\pm <0.1}}$	$20.6_{\pm 0.2}$	$20.4_{\pm 0.3}$	$\mathbf{18.9_{\pm 0.1}}$
	B+C	$3.9_{\pm 0.1}$	$3.6_{\pm 0.1}$	$\mathbf{3.2_{\pm 0.1}}$	$20.0_{\pm 0.1}$	$19.8_{\pm 0.1}$	$\mathbf{18.3_{\pm 0.1}}$
	B+A+C	$3.3_{\pm 0.1}$	$\mathbf{3.1_{\pm 0.1}}$	$\mathbf{3.1_{\pm 0.1}}$	$19.3_{\pm 0.2}$	$19.0_{\pm 0.2}$	$\mathbf{17.6_{\pm 0.1}}$
WRN28-10	B	$4.0_{\pm 0.1}$	$3.7_{\pm 0.1}$	$\mathbf{3.5_{\pm 0.1}}$	$19.1_{\pm 0.2}$	$18.5_{\pm 0.1}$	$\mathbf{17.4_{\pm 0.1}}$
	B+C	$3.1_{\pm <0.1}$	$2.9_{\pm 0.2}$	$\mathbf{2.7_{\pm <0.1}}$	$18.3_{\pm 0.1}$	$17.6_{\pm 0.1}$	$\mathbf{16.9_{\pm 0.2}}$
	B+A+C	$2.6_{\pm 0.1}$	$\mathbf{2.4_{\pm 0.0}}$	$2.5_{\pm 0.1}$	$17.3_{\pm 0.2}$	$17.0_{\pm 0.2}$	$\mathbf{15.9_{\pm 0.1}}$

Table 22: Mean validation error with 95% confidence interval. B refers to basic data augmentation, C refers to Cutout augmentation, and A refers to AutoAugmentation.

14 ADVERSARIAL ROBUSTNESS

In this section, we evaluate the performance of models trained with various flatness based optimizers under different adversarial attacks that cab be found in the standard foolbox library (Rauber et al., 2017). Specifically, we evaluate the robust error rate (Table 23) of WRN16-8 and WRN28-10 models trained using mSGD, E-SGD, ASO, SAM, and LPF-SGD on CIFAR-10 and CIFAR-100 data sets with different data augmentation schemes under simultaneous four different adversarial attacks: L2 Fast Gradient Method (Goodfellow et al., 2015), Linf Projected Gradient Descent (PGD) (Madry et al., 2017), Linf Additive Uniform Noise Attack (AUNA) (Inci et al., 2018), and Linf Deep Fool Attack (Moosavi-Dezfooli et al., 2016)) with standard step size of $\epsilon = 8/255$.

Model	Aug	CIFAR-10					CIFAR-100				
		mSGD	E-SGD	ASO	SAM	LPF-SGD	mSGD	E-SGD	ASO	SAM	LPF-SGD
WRN 16-8	B	$10.3_{\pm 0.1}$	$\mathbf{9.0_{\pm 0.3}}$	$10.4_{\pm 0.1}$	$9.9_{\pm 0.3}$	$9.7_{\pm 0.2}$	$29.5_{\pm 0.3}$	$\mathbf{28.6_{\pm 0.3}}$	$29.4_{\pm 0.4}$	$28.9_{\pm 0.2}$	$\mathbf{28.6_{\pm 0.3}}$
	B+C	$10.8_{\pm 0.2}$	$\mathbf{9.4_{\pm 0.5}}$	$10.8_{\pm 0.3}$	$9.9_{\pm 0.1}$	$9.9_{\pm 0.2}$	$29.1_{\pm 0.1}$	$28.7_{\pm 0.4}$	$29.3_{\pm 0.1}$	$28.7_{\pm 0.4}$	$\mathbf{28.3_{\pm 0.3}}$
	B+A+C	$10.9_{\pm 0.3}$	$9.9_{\pm 0.3}$	$10.2_{\pm 0.4}$	$\mathbf{9.4_{\pm 0.1}}$	$10.2_{\pm 0.1}$	$30.4_{\pm 0.2}$	$29.3_{\pm 0.2}$	$30.0_{\pm 0.3}$	$29.6_{\pm 0.2}$	$\mathbf{28.7_{\pm 0.4}}$
WRN 28-10	B	$9.0_{\pm 0.2}$	$9.5_{\pm 0.2}$	$8.6_{\pm 0.3}$	$\mathbf{8.1_{\pm 0.1}}$	$8.6_{\pm 0.1}$	$26.4_{\pm 0.2}$	$\mathbf{25.8_{\pm 0.2}}$	$26.9_{\pm 0.1}$	$\mathbf{25.8_{\pm 0.2}}$	$26.7_{\pm 0.1}$
	B+C	$8.3_{\pm 0.3}$	$8.6_{\pm 0.1}$	$8.4_{\pm 0.1}$	$8.4_{\pm 0.2}$	$\mathbf{7.7_{\pm 0.4}}$	$26.2_{\pm 0.1}$	$25.6_{\pm 0.2}$	$26.2_{\pm 0.3}$	$25.8_{\pm 0.2}$	$\mathbf{25.4_{\pm 0.3}}$
	B+A+C	$8.3_{\pm 0.3}$	$8.1_{\pm 0.1}$	$8.2_{\pm 0.3}$	$\mathbf{7.9_{\pm 0.2}}$	$\mathbf{7.9_{\pm 0.3}}$	$26.7_{\pm 0.2}$	$\mathbf{25.6_{\pm 0.2}}$	$26.9_{\pm 0.6}$	$25.8_{\pm 0.3}$	$26.0_{\pm 0.2}$

Table 23: Robust error rate with 95% confidence interval under simultaneous four different adversarial attacks: L2 Fast gradient method (FGM), Linf Projected gradient descent (PGD) attack, Linf Additive uniform noise attack, and Linf Deep fool attack with standard step size $\epsilon = 8/255$. B refers to basic data augmentation, C refers to Cutout augmentation, and A refers to AutoAugmentation.

15 THEORETICAL PROOFS

15.1 Proof for Theorem 1

Proof in this section is inspired by the analysis in (Duchi et al., 2012).

Lemma 1. *Let $l_o(\theta;\xi)$ be α Lipschitz continuous with respect to l_2-norm. Let variable Z be distributed according*

to the distribution μ. Then

$$\begin{aligned}\|\nabla l_\mu(x;\xi) - \nabla l_\mu(y;\xi)\| =& \mathbb{E}_{Z\sim\mu}[\nabla l_\mu(x+Z;\xi) - \nabla l_\mu(y+Z;\xi)] \\ \leq& \alpha \int |\mu(z-x) - \mu(z-y)|dz.\end{aligned} \tag{15.1}$$

If distribution μ is rotationally symmetric and non-increasing, the bound is tight and can be attained by the function

$$l_o(\theta;\xi) = \alpha \frac{\|x\|^2 + \|y\|^2}{\|x-y\|} \left| < \frac{x-y}{\|x\|^2 + \|y\|^2}, \theta > -\frac{1}{2} \right|.$$

Proof. Let Z be the random variable satisfies distribution μ.

$$\begin{aligned}&\mathbb{E}_{Z\sim\mu}[\nabla l_\mu(x+Z;\xi) - \nabla l_\mu(y+Z;\xi)] \\ =& \int \nabla l_\mu(x+z;\xi)\mu(z)dz - \int \nabla l_\mu(y;\xi)\mu(z)dz \\ =& \int \nabla l_\mu(x;\xi)\mu(z)dz - \int \nabla l_\mu(y;\xi)\mu(z)dz \\ =& \int_{I_>} \nabla l_o(z)[\mu(z-x) - \mu(z-y)]dz - \int_{I_<} \nabla l_0(z)[\mu(z-y) - \mu(z-x)]dz\end{aligned}$$

where

$$\begin{aligned}I_> =& \{z \in \mathbb{R}^d | \mu(z-x) > \mu(z-y)\}, \\ I_< =& \{z \in \mathbb{R}^d | \mu(z-x) < \mu(z-y)\}.\end{aligned}$$

Obviously,

$$\begin{aligned}&\|\mathbb{E}_{Z\sim\mu}[\nabla l_\mu(x+Z;\xi) - \nabla l_\mu(y+Z;\xi)]\| \\ \leq& \sup_{z\in I_> \cup I_<} \|\nabla l_o(z)\| \left| \int_{I_>} [\mu(z-x) - \mu(z-y)]dz - \int_{I_<} l(z)[\mu(z-y) - \mu(z-x)]dz \right| \\ \leq& \alpha \left| \int_{I_>} [\mu(z-x) - \mu(z-y)]dz - \int_{I_<} l(z)[\mu(z-y) - \mu(z-x)]dz \right| \\ =& \alpha \int |\mu(z-x) - \mu(z-y)|dz.\end{aligned}$$

We already prove the inequality 15.1. We are going to show that the bound is tight and could be attained. Since μ is an rotationally symmetric and non-increasing, the set $I_>$ could be rewritten as

$$\begin{aligned}I_> =& \{z \in \mathbb{R}^d | \mu(z-x) > \mu(z-y)\} \\ =& \{z \in \mathbb{R}^d | \, \|z-x\|^2 > \|z-y\|^2\} \\ =& \{z \in \mathbb{R}^d | \langle z, x-y\rangle > \frac{1}{2}(\|x\|^2 + \|y\|^2)\},\end{aligned}$$

similarly,

$$I_< = \{z \in \mathbb{R}^d | \langle z, x-y\rangle < \frac{1}{2}(\|x\|^2 + \|y\|^2)\}.$$

For given x, y, define function l_o as

$$l_o(\theta;\xi) = \alpha \frac{\|x\|^2 + \|y\|^2}{\|x-y\|} \left| < \frac{x-y}{\|x\|^2 + \|y\|^2}, \theta > -\frac{1}{2} \right|.$$

Therefore, the gradient of function f is

$$\nabla l_o(\theta;\xi) = \begin{cases} \alpha\dfrac{x-y}{\|x-y\|} & \text{if}\langle\theta, x-y\rangle > \dfrac{1}{2}(\|x\|^2 + \|y\|^2) \\ -\alpha\dfrac{x-y}{\|x-y\|} & \text{if}\langle\theta, x-y\rangle < \dfrac{1}{2}(\|x\|^2 + \|y\|^2) \end{cases} \tag{15.2}$$

Hence,

$$\begin{aligned} &\|\mathbb{E}_{Z\sim\mu}[\nabla l_\mu(x+Z;\xi) - \nabla l_\mu(y+Z;\xi)]\| \\ =&\left\|\int_{I_>} \nabla l_o(z)[\mu(z-x) - \mu(z-y)]dz - \int_{I_<} \nabla l_o(z)[\mu(z-y) - \mu(z-x)]dz\right\| \\ =&\left\|\int_{I_>} \alpha\frac{x-y}{\|x-y\|}[\mu(z-x) - \mu(z-y)]dz + \int_{I_<} \alpha\frac{x-y}{\|x-y\|}[\mu(z-y) - \mu(z-x)]dz\right\| \\ =&\left\|\alpha\frac{x-y}{\|x-y\|}\int |\mu(z-x) - \mu(z-y)|dz\right\| \\ =&\alpha\int |\mu(z-x) - \mu(z-y)|dz \left\|\frac{x-y}{\|x-y\|}\right\| \\ =&\alpha\int |\mu(z-x) - \mu(z-y)|dz \end{aligned}$$

We already show that the equality holds for given function l_o. Therefore the bound is tight. □

Theorem 1. *Let μ be the $\mathcal{N}(0, \sigma^2 I_{d\times d})$ distribution. Assume the differentiable loss function $l_o(\theta;\xi) : \mathbb{R}^d \to \mathbb{R}$ is α-Lipschitz continuous and β-smooth with respect to l_2-norm. The smoothed loss function $l_\mu(\theta;\xi)$ is defined as (5.1). Then the following properties hold:*

i) l_μ is α-Lipschitz continuous.

ii) l_μ is continuously differentiable; moreover, its gradient is $\min\{\frac{\alpha}{\sigma}, \beta\}$-Lipschitz continuous, i.e., f_μ is $\min\{\frac{\alpha}{\sigma}, \beta\}$-smooth.

iii) If l_o is convex, $l_o(\theta;\xi) \le l_\mu(\theta;\xi) \le l_o(\theta;\xi) + \alpha\sigma\sqrt{d}$.

In addition, for each bound i)-iii), there exists a function l_o such that the bound cannot be improved by more than a constant factor.

Proof. We are going the prove the properties one by one.

i) Since $\nabla l_\mu(\theta;\xi) = \mathbb{E}_{Z\sim\mu}[\nabla l_o(\theta+Z;\xi)]$, we have

$$\|\nabla l_\mu(\theta;\xi)\| = \|\mathbb{E}_{Z\sim\mu}[\nabla l_o(\theta+Z;\xi)]\| \le \mathbb{E}_{Z\sim\mu}[\|\nabla l_o(\theta+Z;\xi)\|] \le \alpha.$$

Therefore, l_μ is α-Lipschitz continuous. To prove the bound is tight, we define

$$l_o(\theta;\xi) = \frac{1}{2}v^T\theta,$$

where $v \in \mathbb{R}^d$ is a scalar. Hence, we have

$$l_\mu(\theta;\xi) = \mathbb{E}_{Z\sim\mu}[l_o(\theta+Z;\xi)] = \mathbb{E}_{Z\sim\mu}[\frac{1}{2}v^T(\theta-Z)] = \frac{1}{2}v^T\theta = l_o(\theta;\xi).$$

Both l_o and smoothed l_μ have the gradient v and l_μ is exactly α-Lipschitz.

ii) The proof scheme for this part is organized as follow: Firstly we show that l_μ is $\frac{\alpha}{\sigma}$-smooth and the bound can not be improved by more than a constant factor. Then we show that l_μ is β-smooth and the bound can not be improved by more than a constant factor as well. In all, we could draw the conclusion that l_μ is $\min\{\frac{\alpha}{\sigma}, \beta\}$-smooth and the bound is tight.

By Lemma 1, for $\forall x, y \in \mathbb{R}^n$,

$$\|\nabla l_\mu(x;\xi) - \nabla l_\mu(y;\xi)\| \leq \alpha \underbrace{\int |\mu(z-x) - \mu(z-y)| dz}_{I_2}. \tag{15.3}$$

Denote the integral as I_2. We follow a technique used in (Lakshmanan and Pucci De Farias, 2008; Duchi et al., 2012). Since $\mu(z-x) \geq \mu(z-y)$ is equivalent to $\|z-x\| \geq \|z-y\|$,

$$\begin{aligned}
I_2 =& \int |\mu(z-x) - \mu(z-y)| dz \\
=& \int_{z:\|z-x\| \geq \|z-y\|} [\mu(z-x) - \mu(z-y)] dz + \int_{z:\|z-x\| \leq \|z-y\|} [\mu(z-y) - \mu(z-x)] dz \\
=& 2 \int_{z:\|z-x\| \leq \|z-y\|} [\mu(z-x) - \mu(z-y)] dz \\
=& 2 \int_{z:\|z-x\| \leq \|z-y\|} \mu(z-x) dz - 2 \int_{z:\|z-x\| \leq \|z-y\|} \mu(z-y) dz.
\end{aligned}$$

Denote $u = z - x$ for $\mu(z-x)$ term and $u = z - y$ for $\mu(z-y)$ term, we have

$$\begin{aligned}
I_2 =& 2 \int_{z:\|u\| \leq \|u-(x-y)\|} \mu(u) dz - 2 \int_{z:\|u\| \geq \|u-(x-y)\|} \mu(u) dz \\
=& 2\mathbb{P}_{Z\sim\mu}(\|Z\| \leq \|Z - (x-y)\|) - 2\mathbb{P}_{Z\sim\mu}(\|Z\| \geq \|Z - (x-y)\|).
\end{aligned}$$

Obviously,

$$\begin{aligned}
& \mathbb{P}_{Z\sim\mu}(\|Z\| \leq \|Z - (x-y)\|) \\
=& \mathbb{P}_{Z\sim\mu}(\|Z\|^2 \leq \|Z - (x-y)\|^2) \\
=& \mathbb{P}_{Z\sim\mu}(2\langle z, x-y\rangle \leq \|x-y\|^2) \\
=& \mathbb{P}_{Z\sim\mu}(2\langle z, \frac{x-y}{\|x-y\|}\rangle \leq \|x-y\|),
\end{aligned}$$

$\frac{x-y}{\|x-y\|}$ has norm 1 and $Z \sim \mathcal{N}(0, \sigma^2 I)$ implies $\langle z, \frac{x-y}{\|x-y\|}\rangle \sim \mathcal{N}(0, \sigma^2 I)$. Hence, we have

$$\begin{aligned}
& \mathbb{P}_{Z\sim\mu}(\|Z\| \leq \|Z - (x-y)\|) \\
=& \mathbb{P}_{Z\sim\mu}(\langle z, \frac{x-y}{\|x-y\|}\rangle \leq \frac{\|x-y\|}{2}) \\
=& \int_{-\infty}^{\frac{\|x-y\|}{2}} \frac{1}{\sqrt{2\pi\sigma^2}} \exp(-\frac{u^2}{2\sigma^2}) du.
\end{aligned}$$

Similarly,

$$\begin{aligned}
& \mathbb{P}_{Z\sim\mu}(\|Z\| \geq \|Z - (x-y)\|) \\
=& \mathbb{P}_{Z\sim\mu}(\langle z, \frac{x-y}{\|x-y\|}\rangle \geq \frac{\|x-y\|}{2}) \\
=& \int_{\frac{\|x-y\|}{2}}^{+\infty} \frac{1}{\sqrt{2\pi\sigma^2}} \exp(-\frac{u^2}{2\sigma^2}) du.
\end{aligned}$$

Therefore,

$$\begin{aligned} I_2 &= 2\int_{-\infty}^{\frac{\|x-y\|}{2}} \frac{1}{\sqrt{2\pi\sigma^2}} \exp(-\frac{u^2}{2\sigma^2})du - 2\int_{\frac{\|x-y\|}{2}}^{+\infty} \frac{1}{\sqrt{2\pi\sigma^2}} \exp(-\frac{u^2}{2\sigma^2})du \\ &= 2\int_{-\frac{\|x-y\|}{2}}^{\frac{\|x-y\|}{2}} \frac{1}{\sqrt{2\pi\sigma^2}} \exp(-\frac{u^2}{2\sigma^2})du \\ &\leq \frac{\sqrt{2}\|x-y\|}{\sigma\sqrt{\pi}} \end{aligned} \tag{15.4}$$

In conclusion, combine formula (15.3) and (15.4) we have

$$\|\nabla l_\mu(x;\xi) - \nabla l_\mu(y;\xi)\| \leq \alpha\frac{\sqrt{2}\|x-y\|}{\sigma\sqrt{\pi}} \leq \frac{\alpha}{\sigma}\|x-y\|.$$

We finish proving that l_μ is $\frac{\alpha}{\sigma}$-smooth. We are going to show the bound is tight. For any given x, y, define function l_o as

$$l_o(\theta;\xi) = \alpha\frac{\|x\|^2 + \|y\|^2}{\|x-y\|}\left| < \frac{x-y}{\|x\|^2 + \|y\|^2}, \theta > -\frac{1}{2}\right|,$$

Uniform Lemma 1 and former proof, we know that

$$\begin{aligned} \|\nabla l_\mu(x;\xi) - \nabla l_\mu(y;\xi)\| =& \alpha\int |\mu(z-x) - \mu(z-y)|dz \\ =& \frac{\sqrt{2}\alpha}{\sigma\sqrt{\pi}}\int_{-\frac{\|x-y\|}{2}}^{\frac{\|x-y\|}{2}} \exp(-\frac{u^2}{2\sigma^2})d \end{aligned} \tag{15.5}$$

Because

$$\frac{\sqrt{2}\alpha}{\sigma\sqrt{\pi}}\exp(-\frac{\|x-y\|^2}{8\sigma^2})\|x-y\| \leq \frac{\sqrt{2}\alpha}{\sigma\sqrt{\pi}}\int_{-\frac{\|x-y\|}{2}}^{\frac{\|x-y\|}{2}} \exp(-\frac{u^2}{2\sigma^2})d \leq \frac{\sqrt{2}\alpha}{\sigma\sqrt{\pi}}\|x-y\|$$

Obviously, taking x, y such that $\|x-y\| \leq 2\sqrt{2}\sigma$,

$$\frac{\sqrt{2}\alpha}{e\sigma\sqrt{\pi}}\|x-y\| \leq \|\nabla l_\mu(x;\xi) - \nabla l_\mu(y;\xi)\| \leq \frac{\sqrt{2}\alpha}{\sigma\sqrt{\pi}}\|x-y\|$$

we could conclude the Lipschitz bound for ∇l_μ cannot be improved by more than a constant factor.

Then we are going to show smooth objective l_μ is β smooth and the bound is tight.

$$\begin{aligned} \|\nabla l_\mu(x;\xi) - \nabla l_\mu(y;\xi)\| &= \|\nabla\mathbb{E}_{Z\sim\mu}[l_o(x+Z)] - \nabla\mathbb{E}_{Z\sim\mu}[l_o(x+Z)]\| \\ &= \|\mathbb{E}_{Z\sim\mu}[\nabla l_o(x+Z) - \nabla l_o(y+Z)]\| \\ &= \left\|\int[\nabla l_o(x+Z) - \nabla l_o(y+Z)]\mu(z)dz\right\| \\ &\leq \int\|\nabla l_o(x+Z) - \nabla l_o(y+Z)\|\mu(z)dz \\ &\leq \int\beta\|(x+z)-(y+z)\|\mu(z)dz \\ &= \beta\|x-y\|\int\mu(z)dz \\ &= \beta\|x-y\| \end{aligned}$$

Therefore, l_μ is β-smooth. Then we are going to show the bound is tight and cannot be improved. Define α Lipschitz continuous and β-smooth function $l_o : \mathbb{R}^d \to \mathbb{R}$ as

$$l_o(\theta;\xi) = \frac{1}{2}\beta\|w\|^2 \qquad \theta \in B(0, \frac{\alpha}{\beta}).$$

Hence, we have

$$\begin{aligned}\|\nabla l_\mu(x;\xi) - \nabla l_\mu(y;\xi)\| &= \left\| \int (\beta x - \beta y)\mu(z)dz \right\| \\ &= \left\| \beta(x-y) \int \mu(z)dz \right\| \\ &= \beta \|x-y\| .\end{aligned}$$

Therefore. l_μ is exactly β-smooth.

iii) By Jensen's inequality, for left hand side:

$$l_\mu(\theta;\xi) = \mathbb{E}_{Z\sim\mu}[l_o(\theta+Z;\xi)] \geq l_o(\theta + \mathbb{E}_{Z\sim\mu}[Z];\xi) = l_o(\theta;\xi).$$

For the tightness proof, defining $l_o(\theta;\xi) = \frac{1}{2}v^T\theta$ for $v \in \mathbb{R}^d$ leads to $l_\mu = l_o$.

For right hand side:

$$\begin{aligned} l_\mu(\theta;\xi) =& \mathbb{E}_{Z\sim\mu}[l_o(\theta+Z;\xi)] \\ \leq& l_o(\theta;\xi) + \alpha\mathbb{E}_{Z\sim\mu}[\|Z\|] \qquad (\alpha\text{-Lipchitz continuous}) \\ \leq& l_o(\theta;\xi) + \alpha\sqrt{\mathbb{E}[\|Z\|^2]} \qquad (\frac{\|Z\|^2}{\sigma^2} \sim \mathcal{X}^2(d)) \\ =& l_o(\theta;\xi) + \alpha\sigma\sqrt{d}. \end{aligned}$$

For the tightness proof, taking $l_o(\theta;\xi) = \alpha\|\theta\|$.Since $l_\mu(\theta;\xi) \geq c\alpha\sigma\sqrt{d}$ for some constant c. Therefore, the bound cannot be improved by more than a constant factor.

□

15.2 Brief justification for Theorem 3

Note that former works also use the ratio of upper-bounds to approximate the ratio of generalization error. For example, in (Chaudhari et al., 2017), they use the ratio of upper-bounds to compare the generalization error of Entropy-SGD and Vanilla-SGD (see their Equation 9). Also, in (Sokolic et al., 2017) they use the ratio of upper-bounds (see their Equation 15) to compare the generalization error of the model with and without invariant method. Finally, comparing the properties, including convergence rate or generalization error, of learning algorithms using $\mathcal{O}$ notation is common in the field.

15.3 Proof for Theorem 3

We first consider the generalization error in the context of the original loss L, and then we analyze smoothed loss function $L \circledast K$. The true loss is defined as

$$L^{true}(\theta) := \mathbb{E}_{\xi\sim D} l(\theta;\xi). \tag{15.6}$$

where l is an arbitrary loss function (i.e., l_o for SGD case and l_μ for LPF-SGD case). Since the distribution $\mathcal{D}$ is unknown, we replace the true loss by the empirical loss given as

$$L^{\mathcal{S}}(\theta) := \frac{1}{m}\sum_{i=1}^{m} l(\theta;\xi). \tag{15.7}$$

In order to bound the generalization error ϵ_g, we consider the following stability bound.

Definition 2 (ϵ_s-uniform stability (Hardt et al., 2016)). *Let $\mathcal{S}$ and $\mathcal{S}'$ denote two data sets from input data distribution $\mathcal{D}$ such that $\mathcal{S}$ and $\mathcal{S}'$ differ in at most one example. Algorithm A is ϵ_s-uniformly stable if and only if for all data sets $\mathcal{S}$ and $\mathcal{S}'$ we have*

$$\sup_{\xi} \mathbb{E}[l(A(\mathcal{S});\xi) - l(A(\mathcal{S}');\xi)] \leq \epsilon_s. \tag{15.8}$$

The following theorem, proposed in (Hardt et al., 2016), implies that the generalization error could be bounded using the uniform stability bound.

Theorem 2. *If A is an ϵ_s-uniformly stable algorithm, then the generalization error (the gap between the true risk and the empirical risk) of A is upper-bounded by the stability factor ϵ_s:*

$$\epsilon_g := \mathbb{E}_{\mathcal{S},A}[L^{true}(A(\mathcal{S})) - L^{\mathcal{S}}(A(\mathcal{S}))] \leq \epsilon_s \tag{15.9}$$

Denote the original true loss and empirical loss as:

$$L_o^{true}(\theta) := \mathbb{E}_{\xi\sim D} l_o(\theta;\xi) \quad \text{and} \quad L_o^{\mathcal{S}}(\theta) := \frac{1}{m}\sum_{i=1}^{m} l_o(\theta;\xi).$$

Denote the stability gap and generalization error of original loss function as ϵ_s^o and ϵ_g^o, respectively. Theorem 4 bounds links the stability with Lipschitz factor α, smoothing factor β, and number of iterations T of SGD. Its proof can be found in (Hardt et al., 2016).

Theorem 4 (Uniform stability of SGD (Hardt et al., 2016))**.** *Assume that $l_o(\theta;\xi) \in [0,1]$ is a α-Lipschitz and β-smooth loss function for every example ξ. Suppose that we rum SGD for T steps with monotonically non-increasing step size $\eta_t \leq c/t$. Then SGD is uniformly stable with the stability factor ϵ_s^o satisfying:*

$$\epsilon_s^o \leq \frac{1+1/\beta c}{n-1}(2c\alpha^2)^{\frac{1}{\beta c+1}} T^{\frac{\beta c}{\beta c+1}}. \tag{15.10}$$

Now, we have already bound the stability gap ϵ_s^o on original loss. Then we will move onto the stability gap ϵ_s^μ of loss for Gaussian LPF kernel smoothed loss function. Let μ be distribution $\mathcal{N}(0,\sigma^2 I)$. By the definition of Gaussian LPF (Definition 1), the true loss and the empirical loss with respect to the Gaussian LPF smoothed function are

$$L_\mu^{true}(\theta) := (L_o^{true} \circledast K)(\theta) = \int_{-\infty}^{\infty} L_o^{true}(\theta-\tau)\mu(\tau)d\tau = \mathbb{E}_{Z\sim\mu}[L_o^{true}(\theta+Z)], \tag{15.11}$$

$$L_\mu^{\mathcal{S}}(\theta) \quad := (L_o^{\mathcal{S}} \circledast K)(\theta) = \int_{-\infty}^{\infty} L_o^{\mathcal{S}}(\theta-\tau)\mu(\tau)d\tau = \mathbb{E}_{Z\sim\mu}[L_o^{\mathcal{S}}(\theta+Z)], \tag{15.12}$$

where K is the Gaussian LPF kernel satisfies distribution μ and Z is a random variable satisfies distribution μ. Since $L_o^{true}(\theta) := \mathbb{E}_{\xi\sim D} l_o(\theta;\xi)$ and $L_o^{\mathcal{S}}(\theta) := \frac{1}{m}\sum_{i=1}^{m} l_o(\theta;\xi)$, L_μ^{true} and L_μ could be rewritten as

$$L_\mu^{true}(\theta) = \int_{-\infty}^{\infty} \mathbb{E}_{\xi\sim D}[l_o(\theta-\tau;\xi)]\mu(\tau)d\tau = \mathbb{E}_{\xi\sim D}\left[\int_{-\infty}^{\infty} l_o(\theta-\tau;\xi)\mu(\tau)d\tau\right] = \mathbb{E}_{\xi\sim D}\left[l_\mu(\theta;\xi)\right] \tag{15.13}$$

$$L_\mu^{\mathcal{S}}(\theta) = \int_{-\infty}^{\infty} \frac{1}{m}\sum_{i=1}^{m} l_o(\theta;\xi)\mu(\tau)d\tau = \frac{1}{m}\sum_{i=1}^{m}\left[\int_{-\infty}^{\infty} l_o(\theta-\tau;\xi_i)\mu(\tau)d\tau\right] = \frac{1}{m}\sum_{i=1}^{m} l_\mu(\theta;\xi_i). \tag{15.14}$$

Compare formulas (15.13-15.14) with (15.6-15.7). We could conclude that the true and empirical Gaussian LPF smoothed loss function (L_μ^{true} and $L_\mu^{\mathcal{S}}$) is exactly the formula of original true and empirical loss (L_o^{true} and $L_o^{\mathcal{S}}$) by replacing $l_o(\theta;\xi)$ with smoothed $l_\mu(\theta;\xi)$. Since LPF-SGD is exactly performing SGD iteration on Gaussian LPF smoothed loss function instead of original loss, the generalization error and stability gap of LPF-SGD also satisfies Theorem 2 and Theorem 4 after replacing l_o with l_μ.

In section 5.1 we analyze the change of Lipschitz continuous and smooth properties of the objective function after Gaussian LPF smoothing. Therefore, by Theorem 1, l_μ is α-Lipschitz continuous and $\min\{\frac{\alpha}{\sigma},\beta\}$-smooth. Define $\hat{\beta} = \min\{\frac{\alpha}{\sigma},\beta\}$, then we could bound the stability gap for LPF-SGD as

$$\epsilon_s^\mu \leq \frac{1+1/\hat{\beta} c}{n-1}(2c\alpha^2)^{\frac{1}{\beta c+1}} T^{\frac{\beta c}{\beta c+1}}.$$

Combine Theorem 1 with Theorem 4 we could have the following proposition.

Theorem 3 (Generalization error (GE) bound of LPF-SGD). *Assume that $l_o(\theta;\xi) \in [0,1]$ is a α-Lipschitz and β-smooth loss function for every example ξ. Suppose that we run SGD and LPF-SGD for T steps with non-increasing learning rate $\eta_t \leq c/t$. Denote the generalization error bound (GE bound) of SGD and LPF-SGD as $\hat{\epsilon}_g^o$ and $\hat{\epsilon}_g^\mu$, respectively. Then the ratio of GE bound is*

$$\rho = \frac{\hat{\epsilon}_g^\mu}{\hat{\epsilon}_g^o} = \frac{1-p}{1-\hat{p}} \left(\frac{2c\alpha}{T}\right)^{\hat{p}-p} = O(\frac{1}{T^{\hat{p}-p}}), \tag{15.15}$$

where $p = \frac{1}{\beta c+1}$, $\hat{p} = \frac{1}{\min\{\frac{\alpha}{\sigma},\beta\}c+1}$.

Finally, the following two properties hold:

i) If $\sigma > \frac{\alpha}{\beta}$ and $T \gg 2c\alpha^2 \left(\frac{1-p}{1-\hat{p}}\right)^{\frac{1}{\hat{p}-p}}$, $\rho \ll 1$.

ii) If $\sigma > \frac{\alpha}{\beta}$ and $T > 2c\alpha^2 \exp(\frac{2}{1-p})$, increasing σ leads to a smaller ρ.

Proof. For easy notation, denote $\hat{\beta} = \min\{\frac{\alpha}{\sigma},\beta\}$, ϵ_S^o and ϵ_s^μ are stability gaps of SGD and LPF-SGD, respectively. From Theorem 4 and based on the facts that l_o is α-Lipschitz continuous and β-smooth and that smoothed objective l_μ is α-Lipschitz continuous and $\min\{\frac{\alpha}{\sigma},\beta\}$-smooth, the upper bounds for the stability gaps are

$$\epsilon_s^o \leq \frac{1+1/\beta c}{n-1}(2c\alpha^2)^{\frac{1}{\beta c+1}} T^{\frac{\beta c}{\beta c+1}},$$
$$\epsilon_s^\mu \leq \frac{1+1/\hat{\beta} c}{n-1}(2c\alpha^2)^{\frac{1}{\hat{\beta} c+1}} T^{\frac{\hat{\beta} c}{\hat{\beta} c+1}}.$$

Denote $p = \frac{1}{\beta c+1}$, $\hat{p} = \frac{1}{\hat{\beta} c+1}$, the bound could be rewritten as

$$\epsilon_s^o \leq \frac{1}{(n-1)(1-p)}(2c\alpha^2)^p T^{1-p},$$
$$\epsilon_s^\mu \leq \frac{1}{(n-1)(1-\hat{p})}(2c\alpha^2)^{\hat{p}} T^{1-\hat{p}}.$$

By Theorem 2, the generalization errors ϵ_g^o and ϵ_g^μ are bounded by stability gaps ϵ_s^o and ϵ_s^μ:

$$\epsilon_g^o \leq \epsilon_s^o \leq \frac{1}{(n-1)(1-p)}(2c\alpha^2)^p T^{1-p},$$
$$\epsilon_g^\mu \leq \epsilon_s^\mu \leq \frac{1}{(n-1)(1-\hat{p})}(2c\alpha^2)^{\hat{p}} T^{1-\hat{p}}.$$

Therefore, the GE bound for SGD and LPF-SGD could be written as

$$\hat{\epsilon}_g^o = \frac{1}{(n-1)(1-p)}(2c\alpha^2)^p T^{1-p} \text{ and } \hat{\epsilon}_g^\mu = \frac{1}{(n-1)(1-\hat{p})}(2c\alpha^2)^{\hat{p}} T^{1-\hat{p}},$$

respectively. Then the ratio of GE bound is

$$\rho = \frac{\hat{\epsilon}_g^\mu}{\hat{\epsilon}_g^o} = \frac{1-p}{1-\hat{p}} \left(\frac{2c\alpha}{T}\right)^{\hat{p}-p} = O(\frac{1}{T^{\hat{p}-p}}).$$

i) When $\sigma > \frac{\alpha}{\beta}$ and $T > 2c\alpha^2 \left(\frac{1-p}{1-\hat{p}}\right)^{\frac{1}{\hat{p}-p}}$, $\rho = \frac{1-p}{1-\hat{p}}(\frac{2c\alpha^2}{T})^{\hat{p}-p} < 1$. Therefore, if $\sigma > \frac{\alpha}{\beta}$ and $T \gg 2c\alpha^2 \left(\frac{1-p}{1-\hat{p}}\right)^{\frac{1}{\hat{p}-p}}$, $\rho \ll 1$ and property i) holds.

ii) Denote $x := \hat{p} - p$, the reciprocal of approximated ratio could be re-written as

$$\frac{1}{\rho} \approx (1 - \frac{\hat{p}-p}{1-p})(\frac{T}{2c\alpha^2})^{\hat{p}-p} = (1 - \frac{x}{1-p})(\frac{T}{2c\alpha^2})^x$$

Define function $h(x) = (1 - ax)b^x$, where $a = \frac{1}{1-p}$ and $b = \frac{T}{2c\alpha^2}$. Compute the derivative of function h:

$$h'(x) = (-ax \ln b - a + \ln b)b^x$$

$$h'(x_0) = 0 \Longleftrightarrow x_0 = \frac{lnb - a}{a \ln b}$$

When $x \leq \frac{lnb-a}{a \ln b}$, $h'(x) \geq 0$. Otherwise $h'(x) < 0$. Since $0 < p < \hat{p} < 1$, obviously the domain of function h is in the interval $[0, 1]$. If $\frac{lnb-a}{a \ln b} > 1$, the function h is increasing in its domain. Which means that if the difference between $\hat{p}$ and p increase, the reciprocal of approximated ratio of stability gap $\frac{1}{\rho}$ increase, which is equivalent to the approximate ratio ρ of stability gap decrease. Because

$$\frac{\ln b - a}{a \ln b} > 1 \Longleftrightarrow \ln b > \frac{a}{1-a} \Longleftrightarrow T > 2c\alpha^2 e^{-p}.$$

In all, we could conclude if $T > 2c\alpha^2 e^{-p}$, $\hat{p} - p$ increase leads to the approximate ratio of stability gap ρ decrease.

What's more, we are going to analysis the relation between Gaussian filter factor σ and the difference $\hat{p} - p$. Since

$$p = \frac{1}{\beta c + 1}, \quad \hat{p} = \frac{1}{\hat{\beta} c + 1},$$

where $\hat{\beta} = \min\{\frac{\alpha}{\sigma}, \beta\}$. $\hat{p} - p$ increase is equivalent to $\hat{\beta}$ decrease. When the Gaussian factor σ is large enough ($\frac{\alpha}{\sigma} < \beta$), the smoother factor $\hat{\beta}$ for function l_μ is exactly $\frac{\alpha}{\sigma}$. Moreover, increasing the factor σ leads to the decrease of $\hat{\beta}$.

Due to the analysis above, if $T > 2c\alpha^2 e^{-p}$ and $\frac{\alpha}{\sigma} < \beta$, increasing σ will cause the approximate ratio ρ to decrease and the generalization error will be smaller. We finish the proof for condition ii).

□

15.4 Non-scalar covariance version

In this section, we analysis the case when the covariance $\Sigma = \gamma * diag(||\theta_1||, ||\theta_2|| \cdots ||\theta_k||)$ for Gaussian kernel K is no longer a scalar diagonal matrix. For easy notation, we denoted $\Sigma = diag(\sigma_1^2, \cdots, \sigma_d^2)$ where $\sigma_i^2 = \gamma * ||\theta_i||$.

Theorem 5. *Let μ be the $\mathcal{N}(0, \Sigma)$ distribution, where $\Sigma = diag(\sigma_1^2, \cdots, \sigma_d^2) \in \mathbb{R}^{d \times d}$ is diagonal. Denote $\sigma_-^2 = \min\{\sigma_1^2, \cdots, \sigma_d^2\}$. Assume the differentiable loss function $l_o(\theta; \xi) : \mathbb{R}^d \to \mathbb{R}$ is α-Lipschitz continuous and β-smooth with respect to l_2-norm. The smoothed loss function $l_\mu(\theta; \xi)$ is defined as (5.1). Then the following properties hold:*

i) l_μ is α-Lipschitz continuous.

ii) l_μ is continuously differentiable; moreover, its gradient is $\min\{\frac{\alpha}{\sigma_-}, \beta\}$-Lipschitz continuous, i.e. l_μ is $\min\{\frac{\alpha}{\sigma_-}, \beta\}$-smooth.

iii) If l is convex, $l_\mu(\theta; \xi) = l(\theta; \xi) + \alpha\sqrt{tr(\Sigma)} = l(\theta; \xi) + \alpha\sqrt{\sum_{i=1}^d \sigma_i^2}$.

In addition, for bound i) and iii), there exists a function l such that the bound cannot be improved by more than a constant factor.

Proof. We are going to prove the properties one by one.

i) The proof for properties i) is exactly the same as Theorem 1.

ii) As is shown in the proof for Theorem 1, firstly, we need to first address that l_μ is $\frac{\alpha}{\sigma}$-smooth. then show that l_μ is β-smooth. Since, the proof for second part remains the same as what in Theorem 1. We will focus on demonstrating l_μ is $\frac{\alpha}{\sigma}$-smooth.

By Lemma 1, for $\forall x, y \in \mathbb{R}^n$,

$$\|\nabla l_\mu(x;\xi) - \nabla l_\mu(y;\xi)\| \leq \alpha \underbrace{\int |\mu(z-x) - \mu(z-y)| dz}_{I_2}. \tag{15.16}$$

Denoted the integral as I_2. We follow the technique in (Lakshmanan and Pucci De Farias, 2008) and (Duchi et al., 2012). Since $\mu(z-x) \geq \mu(z-y)$ is equivalent to $\|z-x\| \geq \|z-y\|$,

$$\begin{aligned}
I_2 =& \int |\mu(z-x) - \mu(z-y)| dz \\
=& \int_{z:\|z-x\|\geq\|z-y\|} [\mu(z-x) - \mu(z-y)] dz + \int_{z:\|z-x\|\leq\|z-y\|} [\mu(z-y) - \mu(z-x)] dz \\
=& 2\int_{z:\|z-x\|\leq\|z-y\|} [\mu(z-x) - \mu(z-y)] dz \\
=& 2\int_{z:\|z-x\|\leq\|z-y\|} \mu(z-x) dz - 2\int_{z:\|z-x\|\leq\|z-y\|} \mu(z-y) dz.
\end{aligned}$$

Denote $u = z - x$ for $\mu(z-x)$ term and $u = z - y$ for $\mu(z-y)$ term, we have

$$\begin{aligned}
I_2 =& 2\int_{z:\|u\|\leq\|u-(x-y)\|} \mu(u) dz - 2\int_{z:\|u\|\geq\|u-(x-y)\|} \mu(u) dz \\
=& 2\mathbb{P}_{Z\sim\mu}(\|Z\| \leq \|Z - (x-y)\|) - 2\mathbb{P}_{Z\sim\mu}(\|Z\| \geq \|Z - (x-y)\|).
\end{aligned}$$

Obviously,

$$\begin{aligned}
&\mathbb{P}_{Z\sim\mu}(\|Z\| \leq \|Z - (x-y)\|) \\
=& \mathbb{P}_{Z\sim\mu}(\|Z\|^2 \leq \|Z - (x-y)\|^2) \\
=& \mathbb{P}_{Z\sim\mu}(2\langle z, x-y\rangle \leq \|x-y\|^2) \\
=& \mathbb{P}_{Z\sim\mu}(2\langle z, \frac{x-y}{\|x-y\|}\rangle \leq \|x-y\|),
\end{aligned}$$

Denote $p = \frac{x-y}{\|x-y\|} \in \mathbb{R}^{d\times d}$, $\frac{x-y}{\|x-y\|}$ has norm 1 implies $\sum_{i=1}^d p_i^2 = 1$. Since $Z \sim \mathcal{N}(0, \Sigma)$ and $\Sigma = diag(\sigma_1^2, \cdots, \sigma_d^2)$, each element in vector Z satisfies $z_i \sim \mathcal{N}(0, \sigma_i^2)$. Hence, we have

$$\langle z, \frac{x-y}{\|x-y\|}\rangle = \sum_{i=1}^d p_i z_i \sim \mathcal{N}(0, \sum_{i=1}^d p_i^2 \sigma_i^2).$$

Denote $\sigma^2 = \sum_{i=1}^d p_i^2 \sigma_i^2$, $\sigma_+^2 = \max\{\sigma_1^2, \cdots, \sigma_d^2\}$ and $\sigma_-^2 = \min\{\sigma_1^2, \cdots, \sigma_d^2\}$. Because $\sum_{i=1}^d p_i^2 = 1$, it is easy to know

$$\langle z, \frac{x-y}{\|x-y\|}\rangle \sim \mathcal{N}(0, \sigma^2), \quad \text{where } \sigma_-^2 \leq \sigma^2 \leq \sigma_+^2. \tag{15.17}$$

Hence, we have

$$\begin{aligned}
&\mathbb{P}_{Z\sim\mu}(\|Z\| \leq \|Z - (x-y)\|) \\
=& \mathbb{P}_{Z\sim\mu}(\langle z, \frac{x-y}{\|x-y\|}\rangle \leq \frac{\|x-y\|}{2}) \\
=& \int_{-\infty}^{\frac{\|x-y\|}{2}} \frac{1}{\sqrt{2\pi\sigma^2}} \exp(-\frac{u^2}{2\sigma^2}) du.
\end{aligned}$$

Similarly,

$$\begin{aligned}&\mathbb{P}_{Z\sim\mu}(\|Z\| \geq \|Z-(x-y)\|)\\ =&\mathbb{P}_{Z\sim\mu}(\langle z, \frac{x-y}{\|x-y\|}\rangle \geq \frac{\|x-y\|}{2})\\ =&\int_{\frac{\|x-y\|}{2}}^{+\infty} \frac{1}{\sqrt{2\pi\sigma^2}}\exp(-\frac{u^2}{2\sigma^2})du.\end{aligned}$$

Therefore,

$$\begin{aligned}I_2 &= 2\int_{-\infty}^{\frac{\|x-y\|}{2}} \frac{1}{\sqrt{2\pi\sigma^2}}\exp(-\frac{u^2}{2\sigma^2})du - 2\int_{\frac{\|x-y\|}{2}}^{+\infty} \frac{1}{\sqrt{2\pi\sigma^2}}\exp(-\frac{u^2}{2\sigma^2})du\\ &= 2\int_{-\frac{\|x-y\|}{2}}^{\frac{\|x-y\|}{2}} \frac{1}{\sqrt{2\pi\sigma^2}}\exp(-\frac{u^2}{2\sigma^2})du\\ &\leq \frac{\sqrt{2}\,\|x-y\|}{\sigma\sqrt{\pi}} \leq \frac{\sqrt{2}\,\|x-y\|}{\sigma_-\sqrt{\pi}}\end{aligned} \tag{15.18}$$

In conclusion, combine formula (15.16) and (15.18) we have

$$\|\nabla l_\mu(x;\xi) - \nabla l_\mu(y;\xi)\| \leq \alpha\frac{\sqrt{2}\,\|x-y\|}{\sigma_-\sqrt{\pi}} \leq \frac{\alpha}{\sigma_-}\|x-y\|.$$

We finish proving that l_μ is $\frac{\alpha}{\sigma_-}$-smooth. Since covariance matrix Σ for distribution μ is no longer a scalar matix and μ is not rotationally symmetric, the bound can no longer be achieved.

iii) By Jensen's inequality, for left hand side:

$$l_\mu(\theta;\xi) = \mathbb{E}_{Z\sim\mu}[l_o(\theta+Z;\xi)] \geq l_o(\theta+\mathbb{E}_{Z\sim\mu}[Z];\xi) = l_o(\theta;\xi).$$

For the tightness proof, defining $l_o(\theta;\xi) = \frac{1}{2}v^T\theta$ for $v\in\mathbb{R}^d$ leads to $l_\mu = l_o$.

For right hand side:

$$\begin{aligned}l_\mu(\theta;\xi) =&\mathbb{E}_{Z\sim\mu}[l_o(\theta+Z;\xi)]\\ \leq& l_o(\theta;\xi) + \alpha\mathbb{E}_{Z\sim\mu}[\|Z\|] \qquad (\alpha\text{-Lipchitz continuous})\\ \leq& l_o(\theta;\xi) + \alpha\sqrt{\mathbb{E}[\|Z\|^2]}.\end{aligned}$$

Letting $C^TC = \Sigma$ and $V\sim\mathcal{N}(0,I)$, because $Z\sim\mathcal{N}(0,\Sigma)$, we have

$$\mathbb{E}[\|Z\|^2] = \mathbb{E}[\|CV\|^2] = \mathbb{E}[V^TC^TCV] = tr(C^TC\mathbb{E}[V^TV]) = tr(\Sigma).$$

Therefore,

$$l_\mu(\theta;\xi) = l_o(\theta;\xi) + \alpha\sqrt{tr(\Sigma)} = l_o(\theta;\xi) + \alpha\sqrt{\sum_{i=1}^d \sigma_i^2}.$$

For the tightness proof, taking $l_o(\theta;\xi) = \alpha\|\theta\|$. Since $l_\mu(\theta;\xi) \geq c\alpha\sqrt{tr(\Sigma)}$ for some constant c. Therefore, the bound cannot be improved by more than a constant factor.

□

Proposition 1. *Let μ be the $\mathcal{N}(0,\Sigma)$ distribution, where $\Sigma = diag(\sigma_1^2,\cdots,\sigma_d^2)\in\mathbb{R}^{d\times d}$ is diagonal. Denote $\sigma_-^2 = \|\Sigma\|_\infty = \min\{\sigma_1^2,\cdots,\sigma_d^2\}$. Assume loss function $l_o(\theta;\xi):\mathbb{R}^d\to\mathbb{R}$ is α-Lipschitz and β-smooth. The smoothed loss function l_μ is defined as (5.1). Suppose we execute SGD and LPF-SGD for T steps with non-increasing*

learning rate $\eta_t \leq c/t$. Denote the generalization error bound (GE bound) of SGD and LPF-SGD as $\hat{\epsilon}_g^o$ and $\hat{\epsilon}_g^\mu$, respectively. Then the ratio of GE bound is

$$\rho = \frac{\hat{\epsilon}_g^\mu}{\hat{\epsilon}_g^o} = \frac{1-p}{1-\hat{p}} \left(\frac{2c\alpha}{T} \right)^{\hat{p}-p} = O(\frac{1}{T^{\hat{p}-p}}), \tag{15.19}$$

where $p = \frac{1}{\beta c+1}$, $\hat{p} = \frac{1}{\min\{\frac{\alpha}{\sigma},\beta\}c+1}$.

Finally, the following two properties hold:

i) If $\sigma_- > \frac{\alpha}{\beta}$ and $T \gg 2c\alpha^2 \left(\frac{1-p}{1-\hat{p}}\right)^{\frac{1}{\hat{p}-p}}$, $\rho \ll 1$.

ii) If $\sigma_- > \frac{\alpha}{\beta}$ and $T > 2c\alpha^2 e^{-p}$, increasing σ_- leads to a smaller ρ.

Proof. By Theorem 5, the smoothed loss function l_μ is α-Lipschitz continuous and $\min\{\frac{\alpha}{\sigma_-}, \beta\}$-smooth. This gives as equivalency to Theorem 1 after substituting σ_- for σ. Therefore, proof of Theorem 1 is exactly the same as that of Proposition 3 after performing this substitution and therefore will be omitted.

□