Interplay Between Optimization and Generalization of Stochastic Gradient Descent with Covariance Noise

Yeming Wen [*1234] **Kevin Luk** [*3] **Maxime Gazeau** [*3] **Guodong Zhang** [24] **Harris Chan** [24] **Jimmy Ba** [24]

Abstract

The choice of batch-size in a stochastic optimization algorithm plays a substantial role for both optimization and generalization. Increasing the batch-size used typically improves optimization but degrades generalization. To address the problem of improving generalization while maintaining optimal convergence in large-batch training, we propose to add covariance noise to the gradients. We demonstrate that the optimization performance of our method is more accurately captured by the structure of the noise covariance matrix rather than by the variance of gradients. Moreover, over the convex-quadratic, we prove in theory that it can be characterized by the Frobenius norm of the noise matrix. Our empirical studies with standard deep learning model-architectures and datasets shows that our method not only improves generalization performance in large-batch training, but furthermore, does so in a way where the optimization performance remains desirable and the training duration is not elongated.

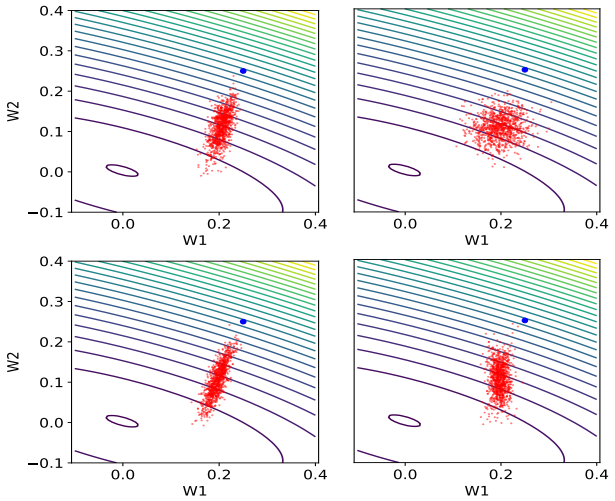


Figure 1. Noise structure in a simple two-dimensional regression problem. **Top-left**: One-step SGD update. **Top-right**: One-step SGD update with isotropic Gaussian ($\sigma = 0.1$) noise. **Bottom-left**: One-step SGD update with full Fisher noise. **Bottom-right**: One-step SGD update with diagonal Fisher noise. The full Fisher noise almost recovers the SGD noise. Observe that the full Fisher noise direction is perpendicular to the contours of the loss surface. Moreover, full Fisher exhibits slower convergence than diagonal Fisher; we refer to Section 4 for a more detailed analysis.

1 Introduction

From a strictly mathematical perspective, training neural networks is a high-dimensional non-convex optimization problem and the dynamics of the training process is incredibly complicated. Despite this, Stochastic Gradient Descent (SGD) and its variants have proven to be extremely effective for training neural networks in practice. Much of the recent empirical successes of deep learning in application tasks such as image recognition (He et al., 2016), speech recognition (Amodei et al., 2016), natural language processing (Wu et al., 2016) and game playing (Mnih et al., 2015) can be seen as testaments to the effectiveness of SGD.

The choice of a batch-size plays an important role in the resulting learning behavior of SGD. Taking larger batch-sizes ensures better gradient estimation which typically leads to faster training convergence. However, there is a tradeoff from the viewpoint of generalization; the intrinsic noise stemming from mini-batch gradients provides regularization effects (Chaudhari & Soatto, 2017; Smith & Le, 2017) and by increasing batch-sizes, we lose such generalization benefits. It is then an interesting question to ask whether large-batch can be engineered in a way such that generalization significantly improves but at the same time not sacrificing too much the training convergence. This is exactly the central objective of our paper.

To address this question, we propose to add a noise term whose covariance structure is given by the diagonal Fisher matrix to the large-batch gradient updates. We discuss the motivations underlying our approach. Under the standard

[*]Equal contribution [1]Work done as an intern at Borealis AI [2]University of Toronto [3]Borealis AI [4]Vector Institute. Correspondence to: Yeming Wen <ywen@cs.toronto.edu>, Kevin Luk <kevin.luk@borealisai.com>, Maxime Gazeau <maxime.gazeau@borealisai.com>.

log-likelihood loss assumption, the difference of large-batch gradients and small-batch gradients can be modeled as a Fisher noise. We can expect that adding this noise directly to large-batch gradients will yield small-batch performance. While this may resolve generalization issues associated with large-batch training (Keskar et al., 2016; Hoffer et al., 2017), the resulting convergence performance is undesirable. To attain our end goal of designing a method which enjoys desirable optimization and generalization performance simultaneously, we reduce the noise level by changing the covariance structure from full Fisher to diagonal Fisher.

Variance is commonly regarded as a criteria of optimization performance. However, for our proposed method in this paper, studying the gradient variance is not sufficient in deducing any information on the optimization behavior. Rather, it is the structure of the noise covariance matrix which plays a critical role. For large-batch training with diagonal Fisher, we find that despite having a high gradient variance, it still attains an ideal optimization performance.

Outline and Summary of Main Contributions. We begin in Section 2 by introducing the basic framework and necessary definitions. We consider different noise covariance structures for large-batch training in Section 2.2 and then propose the choice of diagonal Fisher.

Sections 3 and 4 constitute the central contributions of the paper. The primary takeaways are:

1. Gradient variance is not an accurate indicator of optimization. In Fig. 2, large-batch with diagonal Fisher noise has roughly the same gradient variance as large-batch with full Fisher noise and small-batch. However, in Fig. 3, we see that the convergence performance of large-batch with diagonal Fisher is much better compared to the other two.

2. The main theoretical contribution is Theorem 4.1. We show over the convex quadratic setting, the convergence can be characterized by the Frobenius norm of the noise covariance matrix. In Fig. 5(a), we show empirically that this carries over to the non-convex deep learning context.

In Section 5, we apply our methodology to address the "generalization gap" problem. We show that within the same number of training epochs, large-batch with diagonal Fisher can attain generalization performance roughly comparable to that of small-batch. Related works is given in Section 6 and we close the paper in Section 7.

2 Preliminaries and Approach

2.1 Preliminary Background

Excess Risk Decomposition. We work in the standard framework of supervised learning. Let $\mathcal{D}$ be the unknown joint probability distribution over the data domain $\mathcal{X} \times \mathcal{Y}$ where $\mathcal{X}$ is the input space and $\mathcal{Y}$ is the target space. We have a training set $\mathcal{S} = \{(x_1, y_1), \ldots, (x_N, y_N)\}$ of N input-target samples drawn i.i.d. from $\mathcal{D}$. The family of classifiers of interest to us are neural network outputs $f(x_i, \theta)$, where $\theta \in \mathbb{R}^d$ are parameters of the network. Let $\ell : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$ be the loss function measuring the disagreement between outputs $f(x_i, \theta)$ and targets y_i. For convenience, we use the notation $\ell_i(\theta)$ to denote $\ell(f(x_i, \theta), y_i)$. The expected risk and empirical risk functions are defined to be

$$\mathscr{L}(\theta) := \mathbb{E}_{(x,y)\sim\mathcal{D}}[\ell(f(x,\theta),y)],\ \mathcal{L}(\theta) := \frac{1}{N}\sum_{i=1}^{N}\ell_i(\theta).$$

The standard technique to analyze the interplay between optimization and generalization in statistical learning theory is through excess risk decomposition. The expected excess risk, after k iterations, is defined as:

$$\Delta_k := \mathbb{E}_{\theta\sim Q_k}[\mathscr{L}(\theta)] - \inf_{\theta\in\mathbb{R}^d}\mathscr{L}(\theta),$$

where by the law of total expectation $\mathbb{E}_{\theta\sim Q_k}[\mathscr{L}(\theta)] = \mathbb{E}_{\mathcal{S}}[\mathbb{E}_{\theta\sim Q_k(\cdot|\mathcal{S})}[\mathscr{L}(\theta)]]$ and Q_k is the distribution of θ_k given the data.

Assume that there exists θ^* and θ_* such that $\mathcal{L}(\theta^*) = \inf_{\theta\in\mathbb{R}^d}\mathcal{L}(\theta)$ and $\mathscr{L}(\theta_*) = \inf_{\theta\in\mathbb{R}^d}\mathscr{L}(\theta)$. Then, we have the decomposition:

$$\mathbb{E}_{\theta\sim Q_k(.|\mathcal{S})}[\mathscr{L}(\theta)] - \inf_{\theta\in\mathbb{R}^d}\mathscr{L}(\theta) = T_1 + T_2 + T_3 + T_4$$

where

$$\begin{aligned}
T_1 &:= \mathbb{E}_{\theta\sim Q_k(\cdot|\mathcal{S})}[\mathscr{L}(\theta) - \mathcal{L}(\theta)],\\
T_2 &:= \mathbb{E}_{\theta\sim Q_k(\cdot|\mathcal{S})}[\mathcal{L}(\theta)] - \mathcal{L}(\theta^*),\\
T_3 &:= \mathcal{L}(\theta^*) - \mathcal{L}(\theta_*),\\
T_4 &:= \mathcal{L}(\theta_*) - \mathscr{L}(\theta_*).
\end{aligned}$$

The first term T_1 is the generalization error and the second term T_2 is the optimization error. The third term T_3 is negative by definition and the last term T_4 is zero in expectation with respect to the data distribution. From Bottou & Bousquet (2008); Chen et al. (2018), the expected excess risk can be upper-bounded by

$$\Delta_k \leq \underbrace{\mathbb{E}_{\mathcal{S}}[T_1]}_{\mathcal{E}_{\text{gen}}} + \underbrace{\mathbb{E}_{\mathcal{S}}[T_2]}_{\mathcal{E}_{\text{opt}}}. \tag{1}$$

The terms $\mathcal{E}_{\text{gen}}$ and $\mathcal{E}_{\text{opt}}$ are the expected generalization and optimization errors respectively. It is often the case that optimization algorithms are studied from one perspective: either optimization or generalization. The excess risk decomposition in Eqn. 1 indicates that both aspects should be analyzed together (Chen et al., 2018; Bottou & Bousquet, 2008) since the goal of a good optimization-generalization algorithm in machine learning is to minimize the excess risk in the least amount of iterations.

2.2 Motivations and Approach

We begin by formalizing the setup. Let B_L denote large-batch and $M_L = |B_L|$ denote the size of the large-batch. We consider the following modification of large-batch SGD updates

$$\theta_{k+1} = \theta_k - \alpha_k \nabla \mathcal{L}_{M_L}(\theta_k) + \alpha_k C(\theta_k)\xi_k. \quad (2)$$

where α_k is the learning rate, $\xi_k \sim \mathcal{N}(0, I_d)$ is the multivariate Gaussian distribution with mean zero and identity covariance, and $\nabla \mathcal{L}_{M_L}(\theta_k) = \frac{1}{M_L}\sum_{i=1}^{M_L} \nabla \ell_i(\theta_k)$ is the large-batch gradient. We can understand Eqn. 2 as modifying large-batch SGD by injecting Gaussian noise with mean zero and covariance $C(\theta_k)C(\theta_k)^\top$ to the gradients. The central goal of this paper is to determine a suitable matrix $C(\theta_k)$ such that the excess risk of the algorithm in Eqn. 2 is minimized; in more concrete terms, it achieves low optimization and generalization error simultaneously within a reasonable computational budget.

Intrinsic SGD Noise. Let $B \subset \mathcal{S}$ be a mini-batch drawn uniformly and without replacement from $\mathcal{S}$ and $M = |B|$ be the size of this chosen mini-batch. We can write the SGD update rule here as

$$\begin{aligned}\theta_{k+1} &= \theta_k - \alpha_k \nabla \mathcal{L}_M(\theta_k) \\ &= \theta_k - \alpha_k \nabla \mathcal{L}(\theta_k) + \alpha_k \underbrace{(\nabla \mathcal{L}(\theta_k) - \nabla \mathcal{L}_M(\theta_k))}_{\delta_k}\end{aligned}$$

where $\nabla \mathcal{L}(\theta_k) = \frac{1}{N}\sum_{i=1}^{N} \nabla \ell_i(\theta_k)$ is the full-batch gradient. The difference $\delta_k = \nabla \mathcal{L}(\theta_k) - \nabla \mathcal{L}_M(\theta_k)$ is the intrinsic noise stemming from mini-batch gradients. The covariance of δ_k is given by

$$\frac{1}{N}\sum_{i=1}^{N} \left(\nabla \mathcal{L}(\theta_k) - \nabla \ell_i(\theta_k)\right)\left(\nabla \mathcal{L}(\theta_k) - \nabla \ell_i(\theta_k)\right)^\top \quad (3)$$

This result can be found in Hu et al. (2017); Hoffer et al. (2017). For the purposes of this paper, we assume that the loss is taken to be negative log-likelihood, $\ell_i(\theta_k) = -\log p(y_i|x_i, \theta_k)$ where $p(y|x, \theta)$ is the density function for the model's predictive distribution. Furthermore, we assume that the gradient covariance matrix above can be well-approximated by

$$\frac{1}{N}\sum_{i=1}^{N} \nabla \log p(y_i|x_i, \theta_k) \nabla \log p(y_i|x_i, \theta_k)^\top, \quad (4)$$

where (x_i, y_i) are sampled from the empirical data distribution. In fact, in the literature, this is referred to as the empirical Fisher matrix (Martens, 2014).

For the remainder of this paper, unless otherwise specified, all mentions of "Fisher matrix" or $F(\theta)$ refers to the empirical Fisher. A detailed description of Fisher matrices for feed-forward and convolutional network architectures in given in Section C of the Supplementary Material.

Naive choices of covariance matrix for Eqn. 2. We begin by considering the choice of $C(\theta_k) = 0$. In this case, Eqn. 2 is just standard large-batch gradient descent. Since large-batches provide better gradient estimation, we can expect better training error per parameter update. Indeed, in the convex setting if we view large-batch as full-batch in the extreme case, there are strong convergence guarantees established by the convex optimization community; we refer to Bottou et al. (2018); Bubeck et al. (2015) for a comprehensive overview. However, from the perspective of generalization, it has been observed in LeCun et al. (1998); Keskar et al. (2016); Hoffer et al. (2017) that using larger batch-sizes can lead to a decay in generalization performance of the model.

Next, let B_S denote small-batch, $M_S = |B_S|$ denote the size of small-batch and consider $C(\theta_k)$ to be

$$C(\theta_k) = \sqrt{\frac{M_L - M_S}{M_L M_S}}\sqrt{F(\theta_k)}. \quad (5)$$

Now, if the intrinsic SGD noise is reasonably approximated as a Gaussian distribution with mean zero and covariance given by $C(\theta_k)$ above, then Eqn. 2 with this choice of $C(\theta_k)$ should exhibit similar behavior as small-batch. If this is the case, then we can expect that Eqn. 2 exhibits poor convergence. Indeed, as shown on a 2D convex example in Fig. 1(c), choosing $C(\theta_k)$ as in Eqn. 5 essentially recovers SGD behavior. Futhermore, on the CIFAR-10 image classification task trained using ResNet-44 in Fig. 3, we find that adding this noise significantly worsens the training convergence. Thus, using $C(\theta_k)$ as in Eqn. 5 does not satisfy our objective of simultaneously attaining desirable convergence and generalization for large-batch training.

Using Diagonal Fisher. Previously, we observed that taking $C(\theta_k) = 0$ in Eqn. 2 yields good optimization but poor generalization and vice versa when $C(\theta_k)$ is the square-root of Fisher as defined in Eqn. 5. To achieve our goal, we propose to take a "middle ground" and choose $C(\theta_k)$ to be

$$C(\theta_k) = \sqrt{\frac{M_L - M_S}{M_L M_S}}\sqrt{\operatorname{diag}\left(F(\theta_k)\right)}.$$

A formal statement is given in Algorithm 1. In our empirical analysis in Sections 3 and 5, we show that Algorithm 1 can achieve both desirable convergence and generalization performance within an epoch training budget; which implies that the excess risk is minimized. Since most of our experiments later uses feed-forward and convolutional networks; for completeness purposes, we provide explicit expressions of $\sqrt{\operatorname{diag}\left(F(\theta_k)\right)}$ for these architectures in Section C of Supplementary Material.

Algorithm 1 Adding diagonal Fisher noise to large-batch SGD. Differences from standard large-batch SGD are highlighted in blue

Require: Number of iterations K, initial step-size α_0, large-batch B_L of size M_L, small-batch B_S of size M_S, initial condition $\theta_0 \in \mathbb{R}^d$

for $k = 1$ to K **do**

$\xi_k \sim \mathcal{N}(0, I_d)$

$\epsilon_k = \alpha_k \sqrt{\frac{M_L - M_S}{M_L M_S}} \sqrt{\text{diag}\left(F(\theta_k)\right)} \xi_k$

$\theta_{k+1} = \theta_k - \alpha_k \nabla \mathcal{L}_{M_L}(\theta_k) + \epsilon_k$

end for

Changing $C(\theta_k)$ from $\sqrt{F(\theta_k)}$ to $\sqrt{\text{diag}\left(F(\theta_k)\right)}$ has important implications for both optimization and generalization behavior. In Section 4 and Section B of Supplementary Material, we analyze this difference in the special case where the loss is a convex quadratic. Even in this simple example, it is not obvious how to compare the generalization behavior between the two. We provide an extensive theoretical discussion in Section B of Supplementary Material regarding how different choices of $C(\theta_k)$ can affect the generalization performance of Eqn. 2.

Working with the assumption that the generalization error is comparable between using diagonal Fisher and full Fisher, the excess risk in Eqn. 1 can then be minimized by focusing solely on the optimization error. In the next two sections, we analyze the optimization properties of choosing different noise covariance matrices.

3 Optimization Observations

The objective of this section is to examine the optimization performance of the four regimes: large-batch with $C(\theta_k)$ equal to 0 (standard large-batch), large-batch with $C(\theta_k)$ equal to diagonal Fisher $\sqrt{\text{diag}\left(F(\theta_k)\right)}$, large-batch with $C(\theta_k)$ equal to full Fisher $\sqrt{F(\theta_k)}$ and small-batch. In the experimentation, we fix large-batch to be 4096 and small-batch to be 128. For conciseness, we set forth the notation **LB** and **SB** for large-batch and small-batch respectively.

We begin our study by analyzing the gradient variance of each regime above. It is tempting to think that two optimization methods will have similar convergence if their gradient variance is similar. Our experiments below show that this is not necessarily true.

Variance of Gradients. In this experiment, we give an estimation of the variance of gradients of the four regimes outlined above. The experiment is performed as follows: we freeze a partially-trained network and compute Monte-Carlo estimates of gradient variance with respect to each parameter over different mini-batches. This variance is then averaged over the parameters within each layer. The results

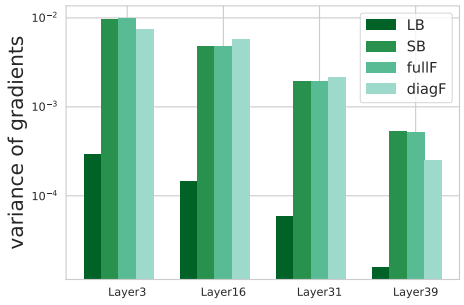


Figure 2. Average variance of gradients for **LB**, **SB**, **LB** with full Fisher and **LB** with diagonal Fisher.

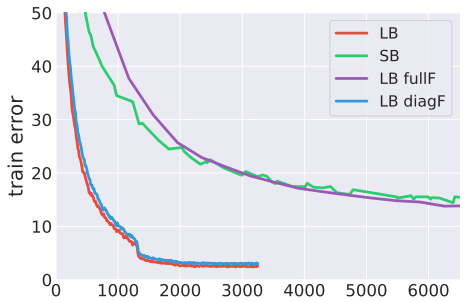


Figure 3. Training error per parameter update for **SB**, **LB**, **LB** with full Fisher and **LB** with diagonal Fisher.

are presented in Fig. 2.

Using diagonal Fisher and full Fisher for **LB** results in roughly the same average gradient variance as **SB**. From the perspective of generalization, we can view this as a heuristic which suggests that the generalization performance between diagonal Fisher and full Fisher is comparable. Indeed, in Section 5, we show that adding diagonal Fisher noise improves the generalization performance considerably. We now proceed to analyze the optimization performance of each training regime.

Convergence. We compare the training error (measured per parameter update) of ResNet44 (CIFAR-10) on the following four regimes: **SB**, **LB**, **LB** with diagonal Fisher noise and **LB** with full Fisher noise. For a fair comparison, we use the same learning rate schedule across all four regimes. Fig. 3 shows that **LB** with diagonal Fisher trains much faster than **LB** with full Fisher. More interestingly, **LB** with full Fisher matches the convergence performance of **SB**, indicating that the intrinsic noise of **SB** is accurately modeled by the full Fisher noise. In contrast, **LB** with diagonal Fisher attains a convergence similar to **LB**, demonstrating that adding this particular form of covariance noise does not hamper much the optimization performance. We illustrate how to sample a random noise with full Fisher covariance in Appendix F.

Combining Fig. 2 and Fig. 3 reveals rather surprising findings. The following three methods: **LB** with diagonal Fisher, **LB** with full Fisher and **SB** share roughly the same gradient variance. However, the convergence of **LB** with diagonal Fisher is much faster than **LB** with full Fisher. In fact, it is similar to the convergence of **LB**.

The above two experiments indicates that methods which share the same gradient variance can converge in very different manners. Hence, knowledge of gradient variance is not sufficient by any means for analyzing optimization behavior. This raises the question: Is there a more appropriate heuristic for understanding optimization? In the following section, we show that the structure of the noise covariance matrix can inform us much more than gradient variance.

4 Case Study: Convex Quadratic Example

In this section, we work within the example of a convex quadratic. We stress here that approximating the loss surface of a neural network with a quadratic often serves as a fertile "testing ground" when introducing new methodologies in deep learning. Analyzing the toy quadratic problem has led to important advances; for example, in learning rate scheduling (Schaul et al., 2013) and formulating SGD as approximate Bayesian inference (Mandt et al., 2017). With regards to optimization, Martens (2010) observed that much of the difficulty in neural network optimization can be captured by using quadratic models.

For strongly-convex objective functions and diminishing step-sizes, the expected optimality gap is bounded in terms of the second-order moment of the gradients (Bottou et al., 2018). However, in practice, different algorithms having the same gradient moments might not need the same number of iterations to converge to the minimum.

Consider the loss function

$$\mathcal{L}(\theta) = \frac{1}{2}\theta^\top A \theta,$$

where A is a symmetric, positive-definite matrix. We focus on the algorithm in Eqn. 2 and consider a constant $d \times d$ covariance matrix C. The following theorem, adapted from Bottou et al. (2018), analyzes the convergence of this optimization method. The proof is relegated to Section A of Supplementary Material.

Theorem 4.1. *Let $\lambda_{\max}$ and $\lambda_{\min}$ denote the maximum and minimum eigenvalue of A respectively. For a chosen $\alpha_0 \leq \lambda_{\max}^{-1}$, suppose that we run the algorithm in Eqn. 2 according to the decaying step-size sequence*

$$\alpha_k = \frac{2}{(k+\gamma)\lambda_{\min}},$$

for all $k \in \mathbb{N}_{>0}$ and where γ is chosen such that $\alpha_k \leq \alpha_0$.

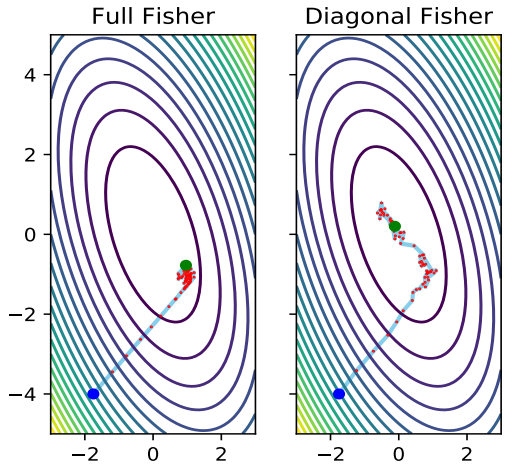


Figure 4. Trajectory using full Fisher versus diagonal Fisher noise for the algorithm in Eqn. 2 used to minimize a two-dimensional quadratic function. Blue dot indicates the initial parameter value and the green dot shows the final parameter value. We used a learning rate of 0.1 for 500 iterations (plotting every 10 iterations). Observe that adding diagonal Fisher to the gradient achieves faster convergence than full Fisher.

Then for all $k \in \mathbb{N}$,

$$\mathbb{E}[\mathcal{L}(\theta_k)] \leq \frac{\nu}{k+\gamma}$$

where

$$\nu = \max\left(\frac{2\,\mathrm{Tr}(C^\top A C)}{\lambda_{\min}^2}, \gamma \mathcal{L}(\theta_0)\right).$$

We make a couple of observations concerning this bound. First, the convergence rate is optimal when $C = 0$ which is expected. In this case, there is no noise and hence we obtain no regularization benefits which leads to poor generalization. A more formal discussion is given at the end of Section B in the Supplementary Material where if we employ a scaling factor $C_\lambda := \lambda C$; as $\lambda \to 0$, the expected generalization error becomes worse.

The second observation is that the term of importance in this theorem is $\mathrm{Tr}(C^\top A C)$. While the overall convergence rate of the algorithm is $O(1/k)$, the discrepancy in convergence performance for different choices of the matrix C rests entirely on this term. The number of iterations for the algorithm in Eqn. 2 to reach the unique minimum depends entirely on $\mathrm{Tr}(C^\top A C)$ and not on the second-order moment.

We analyze two specific cases which are relevant for us: the first case where C is square-root of A, $C = \sqrt{A}$, and the second case where C is the square-root of the diagonal of A, $C = \sqrt{\mathrm{diag}\,(A)}$. The second-order moment of the noise

perturbation is the same for both and is given by

$$\mathbb{E}_{\xi}[\|C\xi_k\|^2] = \mathrm{Tr}(C^\top C) = \mathrm{Tr}(A). \tag{6}$$

However, it is different for $\mathrm{Tr}(C^\top AC)$; in the case of $C = \sqrt{A}$, we get

$$\mathrm{Tr}(C^\top AC) = \mathrm{Tr}(A^2) = \|A\|^2_{\mathrm{Frob}}\,,$$

and for the case of $C = \sqrt{\mathrm{diag}\,(A)}$,

$$\mathrm{Tr}(C^\top AC) = \mathrm{Tr}(\mathrm{diag}\,(A)^2) = \|\mathrm{diag}\,(A)\|^2_{\mathrm{Frob}}\,.$$

Thus, the difference in training performance between the two cases can be measured by the difference of their respective Frobenius norms and less number of iterations are needed with the choice of $\sqrt{\mathrm{diag}\,(A)}$. This suggests that the off-diagonal elements of A play a role in the optimization performance. In Fig. 4, we provide a visualization of the difference between $C = \sqrt{A}$ and $C = \sqrt{\mathrm{diag}\,(A)}$ over a two-dimensional quadratic function.

In the previous analysis, the matrix A above is the Hessian which is also, in this specific setup, equal to the (exact true) Fisher defined as the expectation of the outer product of log-likelihood gradients,

$$\mathbb{E}_{P_x, P_{y|x}}[\nabla \log p(y|x,\theta) \nabla \log p(y|x,\theta)^\top] \tag{7}$$

The expectation here is taken with respect to the data distribution P_x for inputs x and the model's predictive distribution $P_{y|x}$ for targets y.

However in this paper, we are working with the empirical Fisher (which is the empirical covariance of the gradients) instead of an approximation of the (exact true) Fisher matrix. We believe that the insight of this section can be carried over to the empirical Fisher in the sense that the off-diagonal terms of the matrix will degrade the optimization performance.

5 Further Experiments

The primary objective of our empirical study is to analyze how taking $C(\theta_k) = \sqrt{\mathrm{diag}\,(F(\theta_k))}$ in Eqn. 2 results in desirable convergence and generalization performance; thereby minimizing the excess risk in Eqn. (1). We adopt the **LB** and **SB** notation from Section 3.

5.1 Experimentation Details

Batch Normalization. For all experiments involving **LB**, we adopt Ghost Batch Normalization (GBN) (Hoffer et al., 2017) and hence **LB** throughout stands for **LB** with Ghost Batch Normalization. This allows a fair comparison between **LB** and **SB**, as it ensures that batch normalization statistics are computed on the same number of training examples. Using standard batch normalization for large batches can lead to degradation in model quality (Hoffer et al., 2017).

Models and Datasets. The network architectures we use are fully-connected networks, shallow convolutional networks (LeNet (LeCun et al., 1998), AlexNet (Krizhevsky et al., 2012)), and deep convolutional networks (VGG16 (Simonyan & Zisserman, 2014), ResNet44 (He et al., 2016), ResNet44X2 (the number of filters are doubled)). These models are evaluated on the standard deep-learning datasets: MNIST, Fashion-MNIST (LeCun et al., 1998; Xiao et al., 2017), CIFAR-10 and CIFAR-100 (Krizhevsky & Hinton, 2009).

5.2 Frobenius Norm

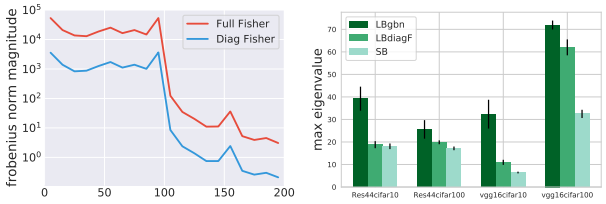
Over the convex quadratic setting in Section 4, Theorem 4.1 tells us that the number of iterations to reach optimum is characterized by the Frobenius norm. Hence, the optimization difference between using large-batch with diagonal Fisher versus full Fisher lies in the difference of their respective Frobenius norms.

We now give an empirical verification of this phenomena in the non-convex setting of deep neural networks. We compute the Frobenius norms during the training of the ResNet44 network on CIFAR-10. Fig. 5(a) shows that the full Fisher matrix has much larger Frobenius norm than the diagonal Fisher matrix, which suggests that using diagonal Fisher noise should have faster convergence than full Fisher noise in the deep neural network setting. Indeed, Fig. 3 shows that **LB** with full Fisher converges as slow as **SB** whereas **LB** with diagonal Fisher converges much faster; and in fact, roughly the same as **LB**. This indicates that adding diagonal Fisher noise to **LB** does not degrade the optimization performance of **LB**.

5.3 Maximum Eigenvalue of Hessian

While the relationship between the curvature of the loss surface landscape and generalization is not completely explicit, numerous works have suggested that the maximum eigenvalue of the Hessian is possibly correlated with generalization performance (Keskar et al., 2016; Chaudhari et al., 2016; Chaudhari & Soatto, 2017; Yoshida & Miyato, 2017; Xing et al., 2018). In this line of research, the magnitudes of eigenvalues of the Hessian may be interpreted as a heuristic measure for generalization; the smaller the magnitude the better the model generalizes. To situate our method with this philosophy, we compute the maximum eigenvalue of the Hessian of the final model for the following three regimes: **SB**, **LB**, and **LB** with diagonal Fisher.

We provide the details of this experiment. Computing maximum eigenvalue without any modification to the model



(a) Frobenius Norm (b) Maximum Eigenvalue

Figure 5. **a)** Frobenius norms of full Fisher and diagonal Fisher along the training trajectory. The model is trained on ResNet44 with CIFAR-10. **b)** Maximum eigenvalue of the Hessian matrix at the end of training for **LB** with Ghost Batch Normalization, **LB** with Ghost Batch Normalization + diagonal Fisher and **SB**. Error bar is computed over 3 random seeds.

gives inconsistent estimates even between different runs of the same training configuration. To make the maximum eigenvalue of the Hessian comparable over different training trajectories, the Hessian needs to be invariant under typical weight reparameterizations; for example, affine transformations (Liao et al., 2018). To achieve this, we make the following modification to the trained model: (1) For the layers with batch normalization, we can just push the batch-norm layer parameters and running statistics into the layer-wise parameter space so that the the layer is invariant under affine transformations; and (2) For the layers without batch normalization, reparameterization changes the prediction confidence while the prediction remains the same. Hence, we train a temperature parameter on the cross-entropy loss on the test data set, this encourages the model to make a calibrated prediction (prevent it from being over-confident). In Fig. 5(b), we give the error bar of the maximum eigenvalue of the Hessian over different runs, which indicates the modification gives a roughly consistent estimate.

The central takeaway from Fig. 5(b) is that across all models with which we experiment, we consistently find that the maximum eigenvalue of the Hessian for **LB** with diagonal Fisher is lower than that of **LB** and in some cases, comparable to **SB**.

5.4 Generalization Gap

In this last part of our empirical studies, we apply our methodology to address the "generalization gap" problem in stochastic optimization. It has been observed in LeCun et al. (1998); Keskar et al. (2016); Hoffer et al. (2017) that using larger batch-sizes can lead to a decay in generalization capability of the model.

We experiment with three regimes: **SB**, **LB** and **LB** with diagonal Fisher. All regimes are trained for the same number for epochs. **LB** with diagonal Fisher here refers to training **LB** with diagonal Fisher noise for the first one-fourth of the total number of epochs used and then using standard **LB** for the remainder. For example, if we use 200 epochs total, **LB** with diagonal Fisher noise is used in the first 50 epochs and then standard **LB** for the last 150 epochs. The reasoning behind this comes from the fact that typically in neural network training, there is an early "transient" learning phase where adding gradient noise can be highly beneficial (Sutskever et al., 2013; Neelakantan et al., 2015). After this transient phase, the benefits of noise becomes increasingly limited. We also point out that we do not experiment with **LB** with full Fisher due to its exceeding long training time. This can be seen from Fig. 3 where for ResNet44 trained on CIFAR-10, **LB** with full Fisher does not achieve good convergence even after 8000 parameter updates. According to Shallue et al. (2018), there is no optimal learning-rate scaling rule for large-batch training across different models and datasets. Hence, we tune the learning rate schedule to obtain optimal results for each method.

Table 1. Validation accuracy results on classification tasks for **SB**, **LB** + Ghost Batch Norm and **LB** + Ghost Batch Norm + diagonal Fisher noise. The results are averaged over 3 random seeds. All methods in each row are trained with the same number of epochs. While it is indeed not feasible to experiment **LB** with full Fisher for all the models below, we note that this reaches roughly the same validation accuracy (93.22) as **SB** in the case of ResNet44 (CIFAR-10).

DATASET	MODEL	SB	LB	LB+Diag
MNIST	MLP	98.10	97.95	**98.10**
MNIST	LENET	99.10	98.88	**99.11**
FASHION	LENET	91.15	88.89	**90.79**
CIFAR-10	ALEXNET	87.80	86.42	**87.61**
CIFAR-100	ALEXNET	59.21	56.79	**59.10**
CIFAR-10	VGG16	93.25	91.81	**93.19**
CIFAR-100	VGG16	72.83	69.45	**72.11**
CIFAR-10	RESNET44	93.42	91.93	**92.88**
CIFAR-100	RESNET44X2	75.55	73.13	**74.26**

The final validation accuracy numbers are reported in Table 1. While it is true that using **LB** with diagonal Fisher cannot completely close the "generalization gap" in some cases, it yields definite improvements over **SB** within an epoch-training budget. Such a training regime typically favors small-batch training as they perform more parameter updates (Shallue et al., 2018). This highlights that our approach is a data-efficient way to improve generalization for large-batch training.

In addition, we experimented with other regimes such as injecting multiplicative Gaussian noise with constant diagonal Fisher (Hoffer et al., 2017), replacing diagonal Fisher with the block-diagonal Kronecker-Factored Approximate Curvature (K-FAC) (Martens & Grosse, 2015) noise [1], and

[1]This corresponds to the matrix-variate Gaussian noise in Zhang et al. (2018a)

using the square-root trace of Fisher $\sqrt{\mathrm{Tr}(F(\theta_k))}$ as the covariance matrix $C(\theta_k)$ in Eqn. 2. We delegate these results to Section E of Supplementary Material.

6 Related Works

Variance and Optimization. In the context of large-scale learning, stochastic algorithms are very popular compared to full-batch methods due to lower computational overhead (Bottou et al., 2018; Bottou, 1991). The tradeoff is that stochastic algorithms exhibit slower convergence asymptotically due to the inherent noise present in their gradients (Moulines & Bach, 2011; Bottou et al., 2018; Wen et al., 2018). For smooth and strongly-convex functions, variance reduction is a common technique to improve convergence rates (Johnson & Zhang, 2013; Defazio et al., 2014).

However, in non-convex optimization, increasing the variance by adding noise is often times beneficial. Unlike the convex setting, the loss surface is much more complicated and there is an abundance of global minima (Choromanska et al., 2015). Adding noise can significantly improve training since it enables the dynamics to escape saddle points or shallow local minima (Ge et al., 2015; Jin et al., 2017). More specifically for deep learning, injecting annealed gradient noise has been shown to speed up training of very deep neural networks (Neelakantan et al., 2015).

Variance and Generalization. The inherent noise in stochastic optimization methods is also conducive to generalization performance. There are vast bodies of literature devoted to this in deep learning; for example, scaling the learning rate or batch-size (Smith & Le, 2017; Goyal et al., 2017; Hoffer et al., 2017) to augment gradient noise in order to encourage better generalization. More direct approaches of studying the covariance structure of mini-batch gradients have also been explored (Jastrzebski et al., 2017; Xing et al., 2018; Zhu et al., 2018; Li et al., 2015).

Interplay Between Optimization and Generalization. A closely-related approach to ours is the Stochastic Gradient Langevin Dynamics (SGLD) (Gelfand et al., 1992; Welling & Teh, 2011); a modification of SGD where an annealed isotropic Gaussian noise is injected to the gradients. There have been a surge of works on the optimization and generalization of SGLD; Xu et al. (2017) obtained global convergence results and the work of Mou et al. (2017) introduced generalization bounds. Raginsky et al. (2017b) analyzed both aspects through the excess risk decomposition.

The recent systematic empirical study of Shallue et al. (2018) demonstrates that links between optimization, generalization, and the choice of batch-size in SGD is extremely complex. It underscores the necessity of a more foundational understanding of the interaction between batch-sizes, model architectures, and other optimization metaparameters.

In a more broader context, optimization and generalization in deep learning are interwined with one another and cannot be decoupled. The choice of an optimization algorithm and the choice of optimization metaparameters induces an implicit bias (Neyshabur et al., 2017a;b; Gunasekar et al., 2018; Wilson et al., 2017) which can lead to solutions of varying generalization quality. Standard regularization methods can also change the optimization landscape; for example, in Santurkar et al. (2018) shows that batch-norm provides a smoothening effect for the loss surface and in Arora et al. (2018) shows that increasing depth can accelerate optimization. Zhang et al. (2018b) demonstrated that weight decay can be understood as increasing the effective learning rate and reducing the Jacobian norm.

7 Conclusion

In this paper, we explored using covariance noise in designing optimization algorithms for deep neural networks that could potentially exhibit ideal learning behavior. We proposed to add diagonal Fisher noise to large-batch gradient updates. Our empirical studies showed that this yield significant improvements in generalization while retaining desirable convergence performance. Furthermore, we demonstrated that the structure of the noise covariance matrix encodes much more information about optimization than the variance of gradients. Over the convex-quadratic setting, we theoretically showed that it is captured exactly by the Frobenius norm.

In much of the paper, we have pinpointed the role of the noise covariance matrix towards optimization. However, as optimization and generalization are inherently tied together in deep learning, the mathematical structure of this matrix should encode generalization information as well. For example, in the special cases of diagonal Fisher and full Fisher, our experiments appear to indicate that the generalization performance is somewhat comparable. This seems to suggest that the diagonal elements contribute much more to the generalization performance than the off-diagonal elements. It is interesting to investigate as to why this is the case.

The analysis presents a starting point and we hope it will pave the way towards the design of more efficient algorithms for non-convex learning. We believe that the diagonal Fisher matrix is neither the optimal or unique choice of covariance matrix that could be used. A more systematic empirical study as well as a more in-depth theoretical study is needed to determine more appropriate choices. The theoretical framework developed to analyze the behavior of SGLD could be useful in that context.

References

Amodei, D., Ananthanarayanan, S., Anubhai, R., Bai, J., Battenberg, E., Case, C., Casper, J., Catanzaro, B., Cheng, Q., Chen, G., et al. Deep speech 2: End-to-end speech recognition in english and mandarin. In *International Conference on Machine Learning*, pp. 173–182, 2016.

Arora, S., Cohen, N., and Hazan, E. On the optimization of deep networks: Implicit acceleration by overparameterization. *arXiv preprint arXiv:1802.06509*, 2018.

Bottou, L. Stochastic gradient learning in neural networks. In *In Proceedings of Neuro-Nmes. EC2*, 1991.

Bottou, L. and Bousquet, O. The tradeoffs of large scale learning. In *Advances in neural information processing systems*, pp. 161–168, 2008.

Bottou, L., Curtis, F. E., and Nocedal, J. Optimization methods for large-scale machine learning. *SIAM Review*, 60(2):223–311, 2018.

Bousquet, O. and Elisseeff, A. Stability and generalization. *Journal of machine learning research*, 2(Mar):499–526, 2002.

Bubeck, S. et al. Convex optimization: Algorithms and complexity. *Foundations and Trends® in Machine Learning*, 8(3-4):231–357, 2015.

Chaudhari, P. and Soatto, S. Stochastic gradient descent performs variational inference, converges to limit cycles for deep networks. *arXiv preprint arXiv:1710.11029*, 2017.

Chaudhari, P., Choromanska, A., Soatto, S., LeCun, Y., Baldassi, C., Borgs, C., Chayes, J., Sagun, L., and Zecchina, R. Entropy-sgd: Biasing gradient descent into wide valleys. *arXiv preprint arXiv:1611.01838*, 2016.

Chen, Y., Jin, C., and Yu, B. Stability and Convergence Trade-off of Iterative Optimization Algorithms. *ArXiv e-prints*, April 2018.

Choromanska, A., Henaff, M., Mathieu, M., Arous, G., and LeCun, Y. The loss surfaces of multilayer networks. *Journal of Machine Learning Research*, 38:192–204, 2015.

Defazio, A., Bach, F., and Lacoste-Julien, S. Saga: A fast incremental gradient method with support for non-strongly convex composite objectives. In *Advances in Neural Information Processing Systems*, pp. 1646–1654, 2014.

Gardiner, C. *Stochastic methods*, volume 4. springer Berlin, 2009.

Ge, R., Huang, F., Jin, C., and Yuan, Y. Escaping from saddle pointsnline stochastic gradient for tensor decomposition. In *Conference on Learning Theory*, pp. 797–842, 2015.

Gelfand, S. B., Doerschuk, P. C., and Nahhas-Mohandes, M. Theory and application of annealing algorithms for continuous optimization. In *Proceedings of the 24th Conference on Winter Simulation*, WSC '92, pp. 494–499, New York, NY, USA, 1992. ACM. ISBN 0-7803-0798-4. doi: 10.1145/167293.167407.

Goyal, P., Dollár, P., Girshick, R., Noordhuis, P., Wesolowski, L., Kyrola, A., Tulloch, A., Jia, Y., and He, K. Accurate, large minibatch sgd: training imagenet in 1 hour. *arXiv preprint arXiv:1706.02677*, 2017.

Grosse, R. and Martens, J. A kronecker-factored approximate fisher matrix for convolution layers. In *International Conference on Machine Learning*, pp. 573–582, 2016.

Gunasekar, S., Lee, J., Soudry, D., and Srebro, N. Characterizing implicit bias in terms of optimization geometry. *arXiv preprint arXiv:1802.08246*, 2018.

Hardt, M., Recht, B., and Singer, Y. Train faster, generalize better: Stability of stochastic gradient descent. *CoRR*, abs/1509.01240, 2015. URL `http://arxiv.org/abs/1509.01240`.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

Hoffer, E., Hubara, I., and Soudry, D. Train longer, generalize better: closing the generalization gap in large batch training of neural networks. In *Advances in Neural Information Processing Systems*, pp. 1731–1741, 2017.

Hu, W., Li, C. J., Li, L., and Liu, J.-G. On the diffusion approximation of nonconvex stochastic gradient descent. *arXiv preprint arXiv:1705.07562*, 2017.

Jastrzebski, S., Kenton, Z., Arpit, D., Ballas, N., Fischer, A., Bengio, Y., and Storkey, A. Three factors influencing minima in sgd. *arXiv preprint arXiv:1711.04623*, 2017.

Jin, C., Ge, R., Netrapalli, P., Kakade, S. M., and Jordan, M. I. How to escape saddle points efficiently. In *International Conference on Machine Learning*, pp. 1724–1732, 2017.

Johnson, R. and Zhang, T. Accelerating stochastic gradient descent using predictive variance reduction. In *Advances in neural information processing systems*, pp. 315–323, 2013.

Keskar, N. S., Mudigere, D., Nocedal, J., Smelyanskiy, M., and Tang, P. T. P. On large-batch training for deep learning: Generalization gap and sharp minima. *arXiv preprint arXiv:1609.04836*, 2016.

Krizhevsky, A. and Hinton, G. Learning multiple layers of features from tiny images. Technical report, Citeseer, 2009.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pp. 1097–1105, 2012.

LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

Li, Q., Tai, C., et al. Stochastic modified equations and adaptive stochastic gradient algorithms. *arXiv preprint arXiv:1511.06251*, 2015.

Liao, Q., Miranda, B., Banburski, A., Hidary, J., and Poggio, T. A surprising linear relationship predicts test performance in deep networks. *arXiv preprint arXiv:1807.09659*, 2018.

Luk, K. and Grosse, R. A coordinate-free construction of scalable natural gradient, 2018.

Mandt, S., Hoffman, M. D., and Blei, D. M. Stochastic gradient descent as approximate bayesian inference. *The Journal of Machine Learning Research*, 18(1):4873–4907, 2017.

Martens, J. Deep learning via hessian-free optimization. 2010.

Martens, J. New insights and perspectives on the natural gradient method. *arXiv preprint arXiv:1412.1193*, 2014.

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

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540): 529, 2015.

Mou, W., Wang, L., Zhai, X., and Zheng, K. Generalization bounds of sgld for non-convex learning: Two theoretical viewpoints. *arXiv preprint arXiv:1707.05947*, 2017.

Moulines, E. and Bach, F. R. Non-asymptotic analysis of stochastic approximation algorithms for machine learning. In *Advances in Neural Information Processing Systems*, pp. 451–459, 2011.

Neelakantan, A., Vilnis, L., Le, Q. V., Sutskever, I., Kaiser, L., Kurach, K., and Martens, J. Adding Gradient Noise Improves Learning for Very Deep Networks. *arXiv e-prints*, art. arXiv:1511.06807, November 2015.

Neyshabur, B., Bhojanapalli, S., Mcallester, D., and Srebro, N. Exploring generalization in deep learning. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 30*, pp. 5947–5956. Curran Associates, Inc., 2017a.

Neyshabur, B., Tomioka, R., Salakhutdinov, R., and Srebro, N. Geometry of optimization and implicit regularization in deep learning. *CoRR*, abs/1705.03071, 2017b.

Pavliotis, G. A. *Stochastic processes and applications: diffusion processes, the Fokker-Planck and Langevin equations*, volume 60. Springer, 2014.

Raginsky, M., Rakhlin, A., and Telgarsky, M. Non-convex learning via stochastic gradient langevin dynamics: a nonasymptotic analysis. *CoRR*, abs/1702.03849, 2017a.

Raginsky, M., Rakhlin, A., and Telgarsky, M. Non-convex learning via stochastic gradient langevin dynamics: a nonasymptotic analysis. In *Proceedings of the 30th Conference on Learning Theory, COLT 2017, Amsterdam, The Netherlands, 7-10 July 2017*, pp. 1674–1703, 2017b.

Santurkar, S., Tsipras, D., Ilyas, A., and Madry, A. How does batch normalization help optimization?(no, it is not about internal covariate shift). *arXiv preprint arXiv:1805.11604*, 2018.

Schaul, T., Zhang, S., and LeCun, Y. No more pesky learning rates. In *International Conference on Machine Learning*, pp. 343–351, 2013.

Shallue, C. J., Lee, J., Antognini, J., Sohl-Dickstein, J., Frostig, R., and Dahl, G. E. Measuring the effects of data parallelism on neural network training. *arXiv preprint arXiv:1811.03600*, 2018.

Simonyan, K. and Zisserman, A. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

Smith, S. L. and Le, Q. V. A Bayesian Perspective on Generalization and Stochastic Gradient Descent. *ArXiv e-prints*, October 2017.

Smith, S. L. and Le, Q. V. Understanding generalization and stochastic gradient descent. *arXiv preprint arXiv:1710.06451*, 2017.

Smith, S. L., Kindermans, P.-J., and Le, Q. V. Don't decay the learning rate, increase the batch size. *arXiv preprint arXiv:1711.00489*, 2017.

Sutskever, I., Martens, J., Dahl, G., and Hinton, G. On the importance of initialization and momentum in deep learning. In *International conference on machine learning*, pp. 1139–1147, 2013.

Uhlenbeck, G. E. and Ornstein, L. S. On the theory of the brownian motion. *Physical review*, 36(5):823, 1930.

Welling, M. and Teh, Y. W. Bayesian learning via stochastic gradient langevin dynamics. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, pp. 681–688, 2011.

Wen, Y., Vicol, P., Ba, J., Tran, D., and Grosse, R. Flipout: Efficient pseudo-independent weight perturbations on mini-batches. *arXiv preprint arXiv:1803.04386*, 2018.

Wilson, A. C., Roelofs, R., Stern, M., Srebro, N., and Recht, B. The marginal value of adaptive gradient methods in machine learning. In *Advances in Neural Information Processing Systems*, pp. 4148–4158, 2017.

Wu, Y., Schuster, M., Chen, Z., Le, Q. V., Norouzi, M., Macherey, W., Krikun, M., Cao, Y., Gao, Q., Macherey, K., et al. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*, 2016.

Xiao, H., Rasul, K., and Vollgraf, R. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*, 2017.

Xing, C., Arpit, D., Tsirigotis, C., and Bengio, Y. A walk with sgd. *arXiv preprint arXiv:1802.08770*, 2018.

Xu, P., Chen, J., Zou, D., and Gu, Q. Global Convergence of Langevin Dynamics Based Algorithms for Nonconvex Optimization. *ArXiv e-prints*, July 2017.

Yoshida, Y. and Miyato, T. Spectral norm regularization for improving the generalizability of deep learning. *arXiv preprint arXiv:1705.10941*, 2017.

Zhang, G., Sun, S., Duvenaud, D., and Grosse, R. Noisy natural gradient as variational inference. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80, pp. 5852–5861, 2018a.

Zhang, G., Wang, C., Xu, B., and Grosse, R. Three mechanisms of weight decay regularization. *arXiv preprint arXiv:1810.12281*, 2018b.

Zhu, Z., Wu, J., Yu, B., Wu, L., and Ma, J. The Anisotropic Noise in Stochastic Gradient Descent: Its Behavior of Escaping from Minima and Regularization Effects. *ArXiv e-prints*, 2018.

A Proof of Theorem 4.1

The proof of this theorem follows the spirit of Bottou et al. (2018). The algorithm

$$\theta_{k+1} = \theta_k - \alpha_k \nabla \mathcal{L}(\theta_k) + \alpha_k C \xi_{k+1},\ \xi_{k+1} \sim \mathcal{N}(0, I_d). \tag{8}$$

falls into the Robbins-Monro setting where the true gradient is perturbed by random noise. This perturbation can be considered as a martingale difference in the sense that

$$\mathbb{E}[C\xi_{k+1}|\mathcal{F}_k] = 0$$

where $(\mathcal{F}_k)_{k\in\mathbb{N}}$ is a increasing filtration generated by the sequence of parameters $(\theta_k)_{k\in\mathbb{N}}$. When the step size is constant $\alpha_k = \alpha$ for all k, it corresponds to the Euler discretization of a gradient flow with random perturbation. We begin the proof by considering the equality,

$$\begin{aligned}\mathcal{L}(\theta_{k+1}) &= \mathcal{L}(\theta_k) + \langle \mathcal{L}(\theta_k), \theta_{k+1} - \theta_k \rangle \\ &\quad + \frac{1}{2}(\theta_{k+1} - \theta_k)^\top \nabla^2 \mathcal{L}(\theta_k)(\theta_{k+1} - \theta_k).\end{aligned}$$

Using the fact that $\nabla \mathcal{L}(\theta_k) = A\theta_k$, $\nabla^2 \mathcal{L}(\theta_k) = A$, and from the definition of θ_{k+1}, we can rewrite the above equation as

$$\begin{aligned}\mathcal{L}(\theta_{k+1}) &= \mathcal{L}(\theta_k) + \langle A\theta_k, -\alpha_k A\theta_k + \alpha_k C\xi_{k+1} \rangle \\ &\quad + \frac{1}{2}\left\|\alpha_k A\theta_k - \alpha_k C\xi_{k+1}\right\|_A^2.\end{aligned}$$

Now, taking the conditional expectation $\mathbb{E}[\cdot|\mathcal{F}_k]$ on both sides of the equality, we obtain by independence of the noise ξ_{k+1} to $\mathcal{F}_k$

$$\begin{aligned}\mathbb{E}[\mathcal{L}(\theta_{k+1})|\mathcal{F}_k] &= \mathcal{L}(\theta_k) - \alpha_k \left\|A\theta_k\right\|_2^2 + \frac{\alpha_k^2}{2}\left\|A\theta_k\right\|_A^2 \\ &\quad + \frac{\alpha_k^2}{2}\mathbb{E}[\left\|C\xi_{k+1}\right\|_A^2]\end{aligned} \tag{9}$$

A simple computation shows

$$\begin{aligned}\mathbb{E}[\left\|C\xi_{k+1}\right\|_A^2] &= \mathbb{E}[(C\xi_{k+1})^\top A(C\xi_{k+1})] \\ &= \mathbb{E}[\xi_{k+1}^\top C^\top AC\xi_{k+1}] \\ &= \mathrm{Tr}(C^\top AC)\end{aligned} \tag{10}$$

Moreover, we have

$$\begin{aligned}\left\|A\theta_k\right\|_A^2 &= (\theta_k^\top A)A(A\theta_k) \\ &= A\left\|A\theta_k\right\|_2^2 \\ &\leq \lambda_{\max}\left\|A\theta_k\right\|_2^2.\end{aligned} \tag{11}$$

Using the results in Eqns. 10 and 11 as well as the assumption on the step-size schedule for all k: $\alpha_k < \alpha_0 < \frac{1}{\lambda_{\max}}$, we rewrite Eqn. 9 as

$$\begin{aligned}\mathbb{E}[\mathcal{L}(\theta_{k+1})|\mathcal{F}_k] &\leq \mathcal{L}(\theta_k) + \left(\frac{\alpha_k}{2}\lambda_{\max} - 1\right)\alpha_k \left\|A\theta_k\right\|_2^2 \\ &\quad + \frac{\alpha_k^2}{2}\mathrm{Tr}(C^\top AC) \\ &\leq \mathcal{L}(\theta_k) - \frac{\alpha_k}{2}\left\|A\theta_k\right\|_2^2 + \frac{\alpha_k^2}{2}\mathrm{Tr}(C^\top AC).\end{aligned} \tag{12}$$

Furthermore,

$$\left\|A\theta_k\right\|_2^2 = A(\theta_k^\top A\theta_k) \geq \lambda_{\min}\left\|\theta_k\right\|_A^2 = 2\lambda_{\min}\mathcal{L}(\theta_k)$$

Using this above fact and then taking the expectation of Eqn. 12 leads to

$$\mathbb{E}[\mathcal{L}(\theta_{k+1})] \leq (1 - \alpha_k\lambda_{\min})\mathbb{E}[\mathcal{L}(\theta_k)] + \frac{\alpha_k^2}{2}\mathrm{Tr}(C^\top AC).$$

We proceed by induction to prove the final result. By definition of ν, the result is obvious for $k = 0$. For the inductive step, suppose that the induction hypothesis holds for k, i.e.,

$$\alpha_k = \frac{2}{(k+\gamma)\lambda_{\min}}, \quad \mathbb{E}[\mathcal{L}(\theta_k)] \leq \frac{\nu}{k+\gamma}.$$

We prove the $k+1$ case.

$$\begin{aligned}\mathbb{E}[\mathcal{L}(\theta_{k+1})] &\leq \left(1 - \frac{2}{k+\gamma}\right)\frac{\nu}{k+\gamma} + \frac{2}{(k+\gamma)^2\lambda_{\min}^2}\mathrm{Tr}(C^\top AC) \\ &\leq \frac{\nu}{(k+\gamma+1)}\end{aligned}$$

This comes from the definition of ν and also the inequality $(k+\gamma-1)(k+\gamma+1) \leq (k+\gamma)^2$. This conclude the proof. □

B Relationship Between Noise Covariance Structures and Generalization

As in the previous section, we work entirely in the convex quadratic setting. In this case, Eqn. 2 becomes

$$\theta_{k+1} = \theta_k - \alpha_k \nabla \mathcal{L}(\theta_k) + \alpha_k C\xi_k,\ \xi_k \sim \mathcal{N}(0, I_d). \tag{13}$$

Our aim in this section is to provide some theoretical discussions on how the choice of covariance structure C influences the generalization behavior.

Uniform stability. Uniform stability (Bousquet & Elisseeff, 2002) is one of the most common techniques used in statistical learning theory to study generalization of a learning algorithm. Intuitively speaking, uniform stability measures how sensitive an algorithm is to perturbations of the sampling data. The more stable an algorithm is, the better its generalization will be. Recently, the uniform stability has been investigated for Stochastic Gradient methods (Hardt et al., 2015) and also for Stochastic Gradient Langevin Dynamics (SGLD) (Mou et al., 2017; Raginsky et al., 2017a). We present the precise definition.

Definition B.1 (Uniform stability)**.** A randomized algorithm $\mathcal{A}$ is ϵ-stable if for all data sets $\mathcal{S}$ and $\mathcal{S}'$ where $\mathcal{S}$ and $\mathcal{S}'$ differ in at most one sample, we have

$$\sup_{(x,y)} |\mathbb{E}_{\mathcal{A}}[\mathcal{L}(\theta_{\mathcal{S}}) - \mathcal{L}(\theta_{\mathcal{S}'})]| \leq \epsilon,$$

where $\mathcal{L}(\theta_{\mathcal{S}})$ and $\mathcal{L}(\theta_{\mathcal{S}'})$ highlight the dependence of parameters on sampling datasets. The supremum is taken over input-target pairs (x, y) belonging to the sample domain.

The following theorem from Bousquet & Elisseeff (2002) shows that uniform stability implies generalization.

Theorem B.2 (Generalization in expectation)**.** *Let $\mathcal{A}$ be a randomized algorithm which is ϵ-uniformly stable, then*

$$|\mathbb{E}_{\mathcal{A}}[\mathcal{E}_{\text{gen}}]| \leq \epsilon,$$

where $\mathcal{E}_{\text{gen}}$ is the expected generalization error as defined in Eqn. 1 of Section 2.

Continuous-time dynamics. We like to use the uniform stability framework to analyze generalization behavior of Eqn. 8. To do this, we borrow ideas from the recent work of Mou et al. (2017) which give uniform stability bounds for Stochastic Gradient Langevin Dynamics (SGLD) in non-convex learning. While the authors in that work give uniform stability bounds in both the discrete-time and continuous-time setting, we work with the continuous setting since this conveys relevant ideas while minimizing technical complications. The key takeaway from Mou et al. (2017) is that uniform stability of SGLD may be bounded in the following way

$$\epsilon_{\text{SGLD}} \leq \sup_{\mathcal{S},\mathcal{S}'} \sqrt{H^2(\pi_t, \pi_t')}. \tag{14}$$

Here, π_t and π_t' are the distributions on parameters θ trained on the datasets $\mathcal{S}$ and $\mathcal{S}'$. The H^2 refers to the Hellinger distance.

We now proceed to mirror the approach of Mou et al. (2017) for Eqn. 8. Our usage of stochastic differential equations will be very soft but we refer to reader to Gardiner (2009); Pavliotis (2014) for necessary backgrounds. For the two datasets $\mathcal{S}$ and $\mathcal{S}'$, the continuous-time analogue of Eqn. 8 are Ornstein-Uhlenbeck processes (Uhlenbeck & Ornstein, 1930):

$$d\theta_{\mathcal{S}}(t) = -A_{\mathcal{S}}\theta_{\mathcal{S}}(t)dt + \sqrt{\alpha}C_{\mathcal{S}}dW(t)$$
$$d\theta_{\mathcal{S}'}(t) = -A_{\mathcal{S}'}\theta_{\mathcal{S}'}(t)dt + \sqrt{\alpha}C_{\mathcal{S}'}dW(t).$$

The solution is given by

$$\theta_{\mathcal{S}}(t) = e^{-A_{\mathcal{S}}t}\theta_{\mathcal{S}}(0) + \sqrt{\alpha}\int_0^t e^{-A_{\mathcal{S}}(t-u)}C_{\mathcal{S}}dW(u),$$

In fact, this yields the Gaussian distribution

$$\theta_{\mathcal{S}}(t) \sim \mathcal{N}(\mu_{\mathcal{S}}(t), \Sigma_{\mathcal{S}}(t)),$$

where

$$\mu_{\mathcal{S}}(t) = e^{-A_{\mathcal{S}}t}\theta_{\mathcal{S}}(0)$$

and $\Sigma_{\mathcal{S}}(t)$ satisfies the Ricatti equation,

$$\frac{d}{dt}\Sigma_{\mathcal{S}}(t) = -(A_{\mathcal{S}}\Sigma_{\mathcal{S}}(t) + \Sigma_{\mathcal{S}}(t)A_{\mathcal{S}}) + \alpha C_{\mathcal{S}}C_{\mathcal{S}}^{\top}.$$

Observe that $A_{\mathcal{S}}$ is symmetric and positive-definite which means that it admits a diagonalization $A_{\mathcal{S}} = P_{\mathcal{S}}D_{\mathcal{S}}P_{\mathcal{S}}^{-1}$. Solving the equation for the covariance matrix gives

$$\Sigma_{\mathcal{S}}(t) = \alpha P_{\mathcal{S}}\left(\int_0^t e^{-D_{\mathcal{S}}(t-u)}P_{\mathcal{S}}^{-1}C_{\mathcal{S}}C_{\mathcal{S}}^{\top}P_{\mathcal{S}}e^{-D_{\mathcal{S}}(t-u)}du\right)P_{\mathcal{S}}^{-1}. \tag{15}$$

We are in the position to directly apply the framework of (Mou et al., 2017). Choosing π_t and $\pi_{t'}$ in Eqn. 14 to be the Gaussians $\mathcal{N}(\mu_{\mathcal{S}}(t), \Sigma_{\mathcal{S}}(t))$ and $\mathcal{N}(\mu_{\mathcal{S}'}(t), \Sigma_{\mathcal{S}'}(t))$ respectively, we obtain a uniform stability bound for Eqn. 8. We compute the right-hand side of the bound to obtain insights on generalization. Using the standard formula for Hellinger distance between two Gaussians, we have

$$H^2(\pi_t, \pi_t') = 1 - \frac{\det(\Sigma_{\mathcal{S}})^{\frac{1}{4}}\det(\Sigma_{\mathcal{S}'})^{\frac{1}{4}}}{\det(\frac{\Sigma_{\mathcal{S}}+\Sigma_{\mathcal{S}'}}{2})^{\frac{1}{2}}}\Lambda_{\mathcal{S},\mathcal{S}'} \tag{16}$$

where $\Lambda_{\mathcal{S},\mathcal{S}'}$ is

$$\exp\left\{-\frac{1}{8}(\mu_{\mathcal{S}} - \mu_{\mathcal{S}'})^{\top}\left(\frac{\Sigma_{\mathcal{S}} + \Sigma_{\mathcal{S}'}}{2}\right)^{-1}(\mu_{\mathcal{S}} - \mu_{\mathcal{S}'})\right\}.$$

Choosing the noise covariance. From Eqn. 16 above, it is evident that to ensure good generalization error for Eqn. 8, we want to choose a covariance $C_{\mathcal{S}}$ such that the Hellinger distance H^2 is minimized. Since we are working within the uniform stability framework, a good choice of $C_{\mathcal{S}}$ should be one where Eqn. 8 becomes less data-dependent. This is intuitive after all – the less data-dependent an algorithm is; the better suited it should be for generalization.

We study Eqn. 16. Note that as time $t \to \infty$, the exponential term goes to 1. Hence, we focus our attention on the ratio of the determinants. Suppose that we choose $C_{\mathcal{S}} = \sqrt{A_{\mathcal{S}}}$ and note that $A_{\mathcal{S}}$ is the Fisher in this convex quadratic example. Simplifying the determinant of $\Sigma_{\mathcal{S}}(t)$ in this case,

$$\det(\Sigma_{\mathcal{S}}(t)) = \left(\frac{\alpha}{2}\right)^d \det(I_d - e^{-2D_{\mathcal{S}}t})$$

Suppose that we choose $C = I_d$. Proceeding analogously,

$$\det(\Sigma_{\mathcal{S}}(t)) = \left(\frac{\alpha}{2}\right)^d \frac{\det(I_d - e^{-2D_{\mathcal{S}}t})}{\det(D_{\mathcal{S}})}$$

We can think of choosing $C = I_d$ or $C = \sqrt{A}$ to be extreme cases and it is interesting to observe that the Hellinger distance is more sensitive to dataset perturbation when $C = I_d$. Our proposed method of this paper was to choose $C = \sqrt{\text{diag}\,(A)}$ and our experiments seem to suggest that choosing the square-root of diagonal captures much of the generalization behavior of full Fisher. Understanding precisely why this is the case poses an interesting research direction to pursue in the future.

A simple scaling argument also highlights the importance of the trade-off between optimization and generalization. Consider $C_\lambda = \lambda C$. Then Theorem 4.1 suggests to take λ small to reduce the variance and improve convergence. However, in that case $\Sigma_\lambda = \lambda^2 \Sigma$ where Σ is given by the Eqn. 15 for C and

$$H^2(\pi_t, \pi_t') = 1 - \frac{\det(\Sigma_{\mathcal{S}})^{\frac{1}{4}} \det(\Sigma_{\mathcal{S}'})^{\frac{1}{4}}}{\det(\frac{\Sigma_{\mathcal{S}} + \Sigma_{\mathcal{S}'}}{2})^{\frac{1}{2}}} \Lambda_{\mathcal{S},\mathcal{S}',\lambda},$$

where $\Lambda_{\mathcal{S},\mathcal{S}',\lambda}$ is

$$\exp\left\{ -\frac{1}{8\lambda^2} (\mu_{\mathcal{S}} - \mu_{\mathcal{S}'})^\top \left(\frac{\Sigma_{\mathcal{S}} + \Sigma_{\mathcal{S}'}}{2} \right)^{-1} (\mu_{\mathcal{S}} - \mu_{\mathcal{S}'}) \right\}.$$

The Hellinger distance gets close to one in the limit of small λ (which intuitively corresponds to the large batch situation).

C Fisher Information Matrix for Deep Neural Networks

In this section, we give a formal description of the Fisher information matrix for both feed-forward networks and convolutional networks. In addition, we give the diagonal expression for both networks. Note that these expressions are valid for both the empirical and the exact Fisher; in the empirical case, the expectation will be taken over the empirical data distribution whereas in the exact case, the expectation will be taken over the predictive distribution for targets y.

C.1 Feed-forward networks

Consider a feed-forward network with L layers. At each layer $i \in \{1, \ldots, L\}$, the network computation is given by

$$z_i = W_i a_{i-1}$$
$$a_i = \phi_i(z_i),$$

where a_{i-1} is an activation vector, z_i is a pre-activation vector, W_i is the weight matrix, and $\phi_i : \mathbb{R} \to \mathbb{R}$ is a nonlinear activation function applied coordinate-wise. Let w be the parameter vector of network obtained by vectorizing and then concatenating all the weight matrices W_i,

$$w = [\text{vec}(W_1)^\top \ \text{vec}(W_2)^\top \ \ldots \ \text{vec}(W_L)^\top]^\top.$$

Furthermore, let $\mathcal{D}v = \nabla_v \log p(y|x, w)$ denote the log-likelihood gradient. Using backpropagation, we have a decomposition of the log-likelihood gradient $\mathcal{D}W_i$ into the outer product:

$$\mathcal{D}W_i = g_i a_{i-1}^\top,$$

where $g_i = \mathcal{D}z_i$ are pre-activation derivatives. The Fisher matrix $F(w)$ of this feed-forward network is a $L \times L$ matrix where each (i, j) block is given by

$$F_{i,j}(w) = \mathbb{E}[\text{vec}(\mathcal{D}W_i)\,\text{vec}(\mathcal{D}W_j)^\top] = \mathbb{E}[a_{i-1}a_{j-1}^\top \otimes g_i g_j^\top]. \tag{17}$$

Diagonal version. We give an expression for the diagonal of $F_{i,i}(w)$ here. The diagonal of $F(w)$ follows immediately afterwards. Let a_{i-1}^2 and g_i^2 be the element-wise product of a_{i-1} and g_i respectively. Then, in vectorized form,

$$\text{diag}\,(F_{i,i}(w)) = \mathbb{E}[\text{vec}((a_{i-1}^2)(g_i^2)^\top)],$$

where $(a_{i-1}^2)(g_i^2)^\top$ is the outer product of a_{i-1}^2 and g_i^2.

C.2 Convolutional networks

In order to write down the Fisher matrix for convolutional networks, it suffices to only consider convolution layers as the pooling and response normalization layers typically do not contain (many) trainable weights. We focus our analysis on a single layer. Much of the presentation here follows (Grosse & Martens, 2016; Luk & Grosse, 2018).

A convolution layer l takes as input a layer of activations $a_{j,t}$ where $j \in \{1, \ldots, J\}$ indexes the input map and $t \in \mathcal{T}$ indexes the spatial location. $\mathcal{T}$ here denotes the set of spatial locations, which we typically take to be a 2D-grid. We assume that the convolution here is performed with a stide of 1 and padding equal to the kernel radius R, so that the set of spatial locations is shared between the input and output feature maps. This layer is parameterized by a set of weights $w_{i,j,\delta}$, where $i \in \{1, \ldots, I\}$ indexes the output map and $\delta \in \Delta$ indexes the spatial offset. The numbers of spatial locations and spatial offsets are denoted by $|\mathcal{T}|$ and $|\Delta|$ respectively. The computation of the convolution layer is given by

$$z_{i,t} = \sum_{\delta \in \Delta} w_{i,j,\delta} a_{j,t+\delta}. \tag{18}$$

The pre-activations $z_{i,t}$ are then passed through a nonlinear activation function ϕ_l. The log-likelihood derivatives of the weights are computed through backpropagation:

$$\mathcal{D}w_{i,j,\delta} = \sum_{t \in \mathcal{T}} a_{j,t+\delta} \mathcal{D}z_{i,t}.$$

Then, the Fisher matrix here is

$$\mathbb{E}[\mathcal{D}w_{i,j,\delta} \mathcal{D}w_{i',j',\delta'}] = \mathbb{E}\left[\left(\sum_{t \in \mathcal{T}} a_{j,t+\delta} \mathcal{D}z_{i,t} \right) \left(\sum_{t' \in \mathcal{T}} a_{j',t'+\delta'} \mathcal{D}z_{i',t'} \right) \right].$$

Table 2. Validation accuracy results on classification tasks using BatchChange, Multiplicative, K-FAC and Fisher Trace. Results are averaged over 3 random seeds. For the readers convenience, we report again the result of Diag-F.

DATASET	MODEL	SB	BATCHCHANGE	MULTIPLICATIVE	K-FAC	FISHER TRACE	DIAG-F
CIFAR-10	VGG16	93.25	93.18	90.98	93.06	92.91	93.19
CIFAR-100	VGG16	72.83	72.44	68.77	71.86	71.35	72.11
CIFAR-10	RESNET44	93.42	93.02	91.28	92.81	92.33	92.88
CIFAR-100	RESNET44X2	75.55	75.16	71.98	73.84	73.77	74.26

Diagonal version. To give the diagonal version, it will be convenient for us to express the computation of the convolution layer in matrix notation. First, we represent the activations $a_{j,t}$ as a $J \times |\mathcal{T}|$ matrix A_{l-1}, the pre-activations $z_{i,t}$ as a $I \times |\mathcal{T}|$ matrix Z_l, and the weights $w_{i,j,\delta}$ as a $I \times J|\Delta|$ matrix W_l. Furthermore, by extracting the patches surrounding each spatial location $t \in \mathcal{T}$ and flattening these patches into column vectors, we can form a $J|\Delta| \times |\mathcal{T}|$ matrix A_{l-1}^{exp} which we call the expanded activations. Then, the computation is Eqn. 18 can be reformulated as the matrix multiplication

$$Z_l = W_l A_{l-1}^{\text{exp}}.$$

Readers familiar with convolutional networks can immediately see that this is the Conv2D operation.

At a specific spatial location $t \in \mathcal{T}$, consider the $J|\Delta|$-dimensional column vectors of A_{l-1}^{exp} and I-dimensional column vectors of Z_l. Denote these by $a_{l-1}^{(:,t)}$ and $z_l^{(t)}$ respectively. The matrix W_l maps $a_{l-1}^{(:,t)}$ to $z_l^{(t)}$. In this case, we find ourselves in the exact same setting as the feed-forward case given earlier. The diagonal is simply

$$\mathbb{E}\left[\text{vec}\left((a_{l-1}^{(:,t)})^2 (\mathcal{D}z_l^{(t)})^2\right)\right]$$

D Kronecker-Factored Approximate Curvature (K-FAC)

Later in Section E, we will compare the diagonal approximation of the Fisher matrix to the Kronecker-factored approximate curvature (K-FAC) (Martens & Grosse, 2015) approximation of the Fisher matrix. We give a brief overview of the K-FAC approximation in the case of feed-forward networks.

Recall that the Fisher matrix for a feed-forward network is a $L \times L$ matrix where each of the (i, j) blocks are given by Eqn. 17. Consider the diagonal (i, i) blocks. If we approximate the activations a_{i-1} and pre-activation derivatives g_i as statistically independent, we have

$$\begin{aligned} F_{i,i}(w) &= \mathbb{E}[\text{vec}(\mathcal{D}W_i)\,\text{vec}(\mathcal{D}W_i)^\top] \\ &= \mathbb{E}[a_{i-1}a_{i-1}^\top \otimes g_i g_i^\top] \\ &\approx \mathbb{E}[a_{i-1}a_{i-1}^\top] \otimes \mathbb{E}[g_i g_i^\top]. \end{aligned}$$

Let $A_{i-1} = \mathbb{E}[a_{i-1}a_{i-1}^\top]$ and $G_i = \mathbb{E}[g_i g_i^\top]$. The K-FAC approximation $\hat{F}$ of the Fisher matrix F is

$$\hat{F} = \begin{bmatrix} A_0 \otimes G_1 & & & 0 \\ & A_1 \otimes G_2 & & \\ & & \ddots & \\ 0 & & & A_{L-1} \otimes G_L \end{bmatrix}.$$

The K-FAC approximation of the Fisher matrix can be summarized in the following way: (1) keep only the diagonal blocks corresponding to individual layers, and (2) make the probabilistic modeling assumption where the activations and pre-activation derivatives are statistically independent.

E Supplementary Experiments

E.1 Supplementary Experiments Details

Learning rate: We tuned the learning rate schedule for each method in Table 1 of Section 5 to obtain best performance. As a result, for both **LB** and **LB** with diagonal Fisher method, we need to scale up the learning rate and use the linear warmup strategy in the first 10 epochs. For **LB**, the optimal learning rate on CIFAR-10 and CIFAR-100 with ResNet44 is 3.2 while is 1.6 for **LB** with diagonal Fisher. With VGG16 network on CIFAR-10 and CIFAR-100, the optimal learning rates are 1.6 for both methods. We decay the learning rate by 0.1 at the epoch of 100, 150 for all above methods.

Noise Termination: For all training regimes involving noise injection, we found terminating the noise at a quarter of the training trajectory and using standard **LB** for the remainder of training achieves the best performance. This finding is consistent to the result of BatchChange in Table 1, which suggests that noise only helps generalization in the beginning of the training.

E.2 Validation Accuracy Results

We provide additional validation accuracy results to complement Table 1 of Section 5. The additional regimes are:

- **BatchChange:** Here, we use **SB** for the first 50 epochs and then use **LB** for the remainder. This experimental setup was inspired by Smith et al. (2017).

- **Multiplicative:** Here, we multiply the gradients with a Gaussian noise with constant diagonal covariance structure. This experimental setup was inspired by Hoffer et al. (2017).
- **K-FAC:** Instead of choosing diagonal Fisher as the noise covariance structure, we use the block-diagonal approximation of Fisher given by K-FAC instead
- **Fisher Trace:** Instead of choosing diagonal Fisher as the noise covariance structure, we use square-root of the trace of Fisher $\sqrt{\mathrm{Tr}(F(\theta_k))}$ instead

The results are reported in Table 2 above.

F Sampling Full Fisher Noise

Sampling True Fisher Random Vector. We describe a method to sample a random vector with Fisher covariance efficiently. We obtain prediction $f(x, \theta)$ by a forward-pass. If we randomly draw labels from the model's predictive distribution and obtain back-propagated gradients $\nabla_\theta \mathcal{L}$, then we have $\mathrm{Cov}(\nabla_\theta \mathcal{L}, \nabla_\theta \mathcal{L}) = \mathbb{E}_x[J_f^\top H_\mathcal{L} J_f]$, which is the exact true Fisher (Martens, 2014). Here, J_f is the Jacobian of outputs with respect to parameters and $H_\mathcal{L}$ is the Hessian of the loss function with respect to the outputs.

Sampling Empirical Fisher Random Vector. Let M be the size of the mini-batch and from the M-forward passes we obtain the back-propagated gradients $\nabla l_1, \ldots, \nabla l_M$ for each data-point. Consider independent random variables $\sigma_1, \ldots, \sigma_M$ drawn from Rademacher distribution, i.e., $P(\sigma_i = 1) = P(\sigma_i = 1) = 1/2$. Then, the mean $\mathbb{E}_\sigma[\sum_{i=1}^M \sigma_i \nabla l_i] = 0$. The covariance is empirical Fisher.