A SUPPLEMENTARY MATERIAL

A.1 Proof of Theorem 4.1

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

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

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 $\mathcal{F}_k$

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

A simple computation shows

$$\begin{aligned}\mathbb{E}[\left\|C\xi_{k+1}\right\|_A^2] &= \mathbb{E}[(C\xi_{k+1})^\top A (C\xi_{k+1})] \\ &= \mathbb{E}[\xi_{k+1}^\top C^\top A C \xi_{k+1}] \\ &= \mathrm{Tr}(C^\top A C)\end{aligned} \tag{3}$$

Moreover, we have

$$\left\|A\theta_k\right\|_A^2 \le \lambda_{\max} \left\|A\theta_k\right\|_2^2, \tag{4}$$

where $\lambda_{\max}$ denotes maximum eigenvalue of A (and $\lambda_{\min}$ correspondingly denotes minimum eigenvalues of A). This comes from the fact that for any symmetric, positive-definite matrix B,

$$\left\|x\right\|_B^2 \le \lambda_{\max,B} \left\|x\right\|_2^2, \tag{5}$$

where $\lambda_{\max,B}$ denotes maximum eigenvalue of B. Using the results in Eqns. 3 and 4 as well as the assumption on the step-size schedule for all k: $\alpha_k < \alpha_0 < \frac{1}{\lambda_{\max}}$, we rewrite Eqn. 2 as

$$\begin{aligned}\mathbb{E}[\mathcal{L}(\theta_{k+1})|\mathcal{F}_k] &\le \mathcal{L}(\theta_k) + \left(\frac{\alpha_k}{2}\lambda_{\max} - 1\right)\alpha_k \left\|A\theta_k\right\|_2^2 \\ &\quad + \frac{\alpha_k^2}{2}\mathrm{Tr}(C^\top A C) \\ &\le \mathcal{L}(\theta_k) - \frac{\alpha_k}{2}\left\|A\theta_k\right\|_2^2 + \frac{\alpha_k^2}{2}\mathrm{Tr}(C^\top A C).\end{aligned} \tag{6}$$

Now, taking $x = A\theta_k$ and $B = A^{-1}$ in Eqn. 5, we have

$$\left\|A\theta_k\right\|_{A^{-1}}^2 \le \lambda_{\max,A^{-1}} \left\|A\theta_k\right\|_2^2$$

Note that $\lambda_{\max,A^{-1}} = \lambda_{\min}$ and simplifying the above,

$$\begin{aligned}\left\|A\theta_k\right\|_2^2 &\ge \lambda_{\min}\left\|A\theta_k\right\|_{A^{-1}}^2 \\ &= \lambda_{\min}(\theta_k^\top A)A^{-1}(A\theta_k) \\ &= \lambda_{\min}\left\|\theta_k\right\|_A^2 \\ &= 2\lambda_{\min}\mathcal{L}(\theta_k).\end{aligned}$$

Using the above inequality in Eqn. 6 and then taking expectation yields

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

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.,

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

We prove the $k+1$ case.

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

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

A.2 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

$$\theta_{k+1} = \theta_k - \alpha_k\nabla\mathcal{L}(\theta_k) + \alpha_k C\xi_k,\ \xi_k \sim \mathcal{N}(0, I_d). \tag{7}$$

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 and 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., 2017). We present the precise definition.

Definition A.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 and Elisseeff (2002) shows that uniform stability implies generalization.

Theorem A.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. 1. 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

$$\epsilon_{\text{SGLD}} \leq \sup_{\mathcal{S},\mathcal{S}'} \sqrt{H^2(\pi_t, \pi_t')}. \tag{8}$$

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. 1. Our usage of stochastic differential equations will be very soft but we refer to reader to Gardiner (2009); Pavliotis (2014) for necessary background. For the two datasets $\mathcal{S}$ and $\mathcal{S}'$, the continuous-time analogue of Eqn. 1 are Ornstein-Uhlenbeck processes (Uhlenbeck and 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

$$\Sigma_{\mathcal{S}}(t) = \alpha P_{\mathcal{S}}\left(\int_0^t e^{-D_{\mathcal{S}}(t-u)}P_{\mathcal{S}}^{-1}C_{\mathcal{S}}C_{\mathcal{S}}^\top P_{\mathcal{S}}e^{-D_{\mathcal{S}}(t-u)}du\right)P_{\mathcal{S}}^{-1}. \tag{9}$$

We are in the position to directly apply the framework of (Mou et al., 2017). Choosing π_t and $\pi_{t'}$ in Eqn. 8 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. 1. 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

$$H^2(\pi_t, \pi_t') = 1 - \frac{\det(\Sigma_{\mathcal{S}})^{\frac{1}{4}}\det(\Sigma_{\mathcal{S}'})^{\frac{1}{4}}}{\det(\frac{\Sigma_{\mathcal{S}}+\Sigma_{\mathcal{S}'}}{2})^{\frac{1}{2}}}\Lambda_{\mathcal{S},\mathcal{S}'} \tag{10}$$

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. 10 above, it is evident that to ensure good generalization error for Eqn. 1, 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. 1 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. 10. 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 $C = \sqrt{\operatorname{diag}(A)}$ 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. 9 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).

A.3 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.

A.4 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,

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

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

$$F_{i,j}(w) = \mathbb{E}[\operatorname{vec}(\mathcal{D}W_i)\operatorname{vec}(\mathcal{D}W_j)^\top] = \mathbb{E}[a_{i-1}a_{j-1}^\top \otimes g_i g_j^\top]. \tag{11}$$

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,

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

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

A.5 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 and Martens, 2016; Luk and Grosse, 2018).

A convolution layer l takes as input a layer of activations $a_{j,t}$ where $j \in \{1, \dots, J\}$ 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 $i \in \{1, \dots, I\}$ 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

$$z_{i,t} = \sum_{\delta \in \Delta} w_{i,j,\delta} a_{j,t+\delta}. \tag{12}$$

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

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

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. 12 can be reformulated as the matrix multiplication

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

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

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

A.6 Kronecker-Factored Approximate Curvature (K-FAC)

Later in Section A.7, we will compare the diagonal approximation of the Fisher matrix to the Kronecker-factored approximate curvature (K-FAC) (Martens and 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. 11. 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

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

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.

A.7 Supplementary Experiments Details

Learning rate: We tuned the learning rate schedule for each method in Table 2 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.

Table 1: Validation accuracy results on classification tasks using BatchChange, Multiplicative, K-FAC and Fisher Trace. Results are averaged over 3 random seeds. For the reader's 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

A.8 Validation Accuracy Results

We provide additional validation accuracy results to complement Table 2 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 $\sqrt{\text{Tr}(F(\theta_k))}$ instead

The results are reported in Table 1 above.

A.9 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 $\text{Cov}(\nabla_\theta \mathcal{L}, \nabla_\theta \mathcal{L}) = \mathbb{E}_x[J_f^\top H_\mathcal{L} J_f]$, 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., $P(\sigma_i = 1) = P(\sigma_i = -1) = \frac{1}{2}$. Then, the mean $\mathbb{E}_\sigma[\sum_{i=1}^M \sigma_i \nabla l_i] = 0$. The covariance is empirical Fisher.

References

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

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

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

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

Moritz Hardt, Benjamin Recht, and Yoram Singer. Train faster, generalize better: Stability of stochastic gradient descent. *CoRR*, abs/1509.01240, 2015. URL http://arxiv.org/abs/1509.01240.

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

Kevin Luk and Roger Grosse. A coordinate-free construction of scalable natural gradient, 2018.

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

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

Wenlong Mou, Liwei Wang, Xiyu Zhai, and Kai Zheng. Generalization bounds of sgld for non-convex learning: Two theoretical viewpoints. *arXiv preprint arXiv:1707.05947*, 2017.

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

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

Samuel L Smith, Pieter-Jan Kindermans, and Quoc V Le. Don't decay the learning rate, increase the batch size. *arXiv preprint arXiv:1711.00489*, 2017.

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