# Does the Data Induce Capacity Control in Deep Learning?

**Rubing Yang** [1] **Jialin Mao** [1] **Pratik Chaudhari** [2]

## Abstract

We show that the input correlation matrix of typical classification datasets has an eigenspectrum where, after a sharp initial drop, a large number of small eigenvalues are distributed uniformly over an exponentially large range. This structure is mirrored in a network trained on this data: we show that the Hessian and the Fisher Information Matrix (FIM) have eigenvalues that are spread uniformly over exponentially large ranges. We call such eigenspectra "sloppy" because sets of weights corresponding to small eigenvalues can be changed by large magnitudes without affecting the loss. Networks trained on atypical datasets with non-sloppy inputs do not share these traits and deep networks trained on such datasets generalize poorly. Inspired by this, we study the hypothesis that sloppiness of inputs aids generalization in deep networks. We show that if the Hessian is sloppy, we can compute non-vacuous PAC-Bayes generalization bounds analytically. By exploiting our empirical observation that training predominantly takes place in the non-sloppy subspace of the FIM, we develop data-distribution dependent PAC-Bayes priors that lead to accurate generalization bounds using numerical optimization.

## 1. Introduction

In Fig. 1 (top), for a wide residual network (with 10 layers) on CIFAR-10, we calculated the eigenspectrum of the input correlation matrix ($n^{-1}XX^\top$ where each column of $X$ is one input image) and compared it to the eigenspectra of the Fisher Information Matrix (FIM) and the Hessian. We find that this decay pattern for the input correlation matrix is mirrored in that of the FIM and the Hessian. There are very few (less than 5% of the input dimensionality) large eigenvalues (stiff) after which there is a sharp drop and a long tail of small eigenvalues (we call them sloppy as defined in Def. 8). Other quantities, e.g., correlations of activations of different layers, Jacobians of different logits with respect to the weights, and gradients of the loss with respect to activations of different layers, have a similar decay pattern. Eigenvalues span exponentially large ranges—about 7 orders of magnitude in this experiment. Sloppy eigenvalues are distributed uniformly across such exponentially large ranges.

Eigenspectra of many typical datasets and networks are similar. In Fig. 1 bottom, we created synthetic inputs with varying slopes for the decay of sloppy eigenvalues. We labeled such inputs using a teacher network with randomly generated, but fixed, weights and trained different student networks on such datasets. Each student was trained to have zero training error, i.e., it interpolated its training dataset perfectly. We find in Fig. 1 (left) that, again, the decay pattern of the inputs is mirrored in the FIM/Hessian of the students—sloppier the inputs, sloppier the FIM and the Hessian. Sloppier the input correlations, better the generalization error of the student (Fig. 1 bottom right).

The Hessian governs the local geometry of the loss function in the weight space; small eigenvalues correspond to directions along which the loss is insensitive to changes in the weights. The FIM governs the local geometry in the prediction space; if we think of a network as a parameterized distribution $p_w(y \mid x)$, eigenvectors corresponding to small eigenvalues of the FIM correspond to sets of weights which can be changed significantly without affecting the distribution $p_w(y \mid x)$ much. A sloppy eigenspectrum for these matrices indicates that the trained network is in some sense, "simple": few sets of weights dominate its predictions while there exists a large set of sets that improve the predictions marginally. Both these matrices play a role in determining the generalization error of a neural network.

This paper investigates how sloppiness of the inputs causes the sloppiness of the FIM and the Hessian and how such sloppiness aids generalization.

### 1.1. Contributions

(1) We show that **for typical datasets and deep networks, eigenspectra of correlation matrices of the inputs, ac-**

[1]Applied Mathematics and Computational Science, University of Pennsylvania [2]Electrical and Systems Engineering, University of Pennsylvania. Correspondence to: Rubing Yang <rubingy@sas.upenn.edu>.

*Proceedings of the $39^{th}$ International Conference on Machine Learning*, Baltimore, Maryland, USA, PMLR 162, 2022. 

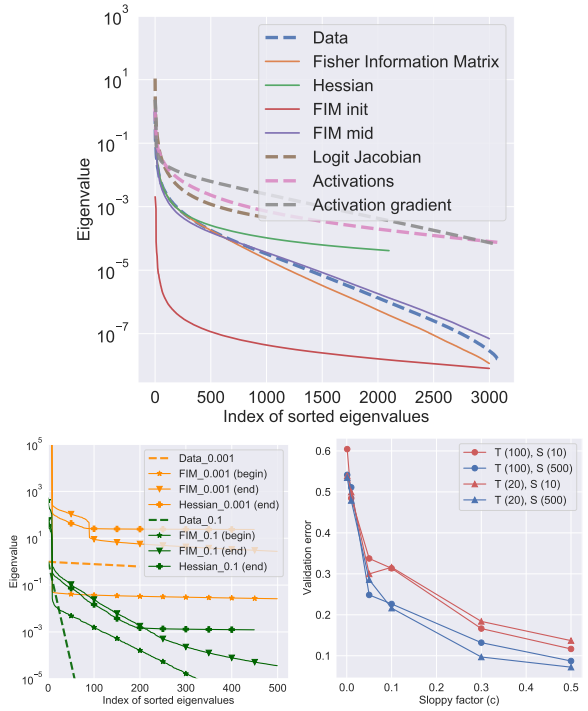


*Figure 1.* **Top:** Eigenspectra of the correlations of the inputs, activations and activation gradients, logit Jacobians and the FIM, Hessian at the end of training; for FIM we also calculate the spectra at initialization and middle of training. All eigenspectra are scaled by the largest eigenvalue of the input correlations (activation gradients are scaled up by $10^{12}$). Eigenspectra corresponding to activations/activation gradients of all layers of the network, and logit Jacobians of all logits are very similar (see Appendix G). Eigenspectra of these quantities are also qualitatively the same at initialization, at the middle of training (see Appendix G.4 Fig. S-15). This plot is drawn for a wide residual network with 10 layers on CIFAR-10 (WRN-10-8 Zagoruyko & Komodakis (2016)), eigenspectra of other networks/datasets are qualitatively the same (see Fig. S-7 and Appendix G).

**Bottom Left:** Eigenspectra of the input correlation matix, FIM and Hessian at begining and end of training for sloppy factor (slope of the sloppy eigenvalue decay) $c = 10^{-3}$ (orange) and $c = 10^{-1}$ (green). If inputs are not sloppy (small $c$) then even if there is a sharp drop after the top few eigenvalues (around 100 for orange lines), the eigenspectrum is flat. In comparison, the FIM/Hessian decay by about 3 orders of magnitude for $c = 0.1$. The details of the experiments can be found at Appendix A.

**Bottom Right:** Validation error of a student (S) network on synthetic datasets of different sloppiness (X-axis) labeled by a teacher network (T). Numbers in brackets indicate number of hidden neurons in two-layer teachers/students. All students in this plot interpolate the training data perfectly. For non-sloppy inputs, interpolation leads to poor generalization, whereas interpolation is not detrimental to generalization for sloppy inputs. As the number of student neurons increases, fixed the teacher's size and the sloppiness factor, the validation error is better. Fixed teacher size, say 20, if inputs are sloppier (sloppy factor of 0.5 vs. 0.1) then we can generalize—roughly equally well—even if the student is smaller (10 vs. 500).

**tivations of different layers, Jacobian of logits with respect to the weights, gradients of the loss with respect to the activations, as also the Hessian and the FIM, are sloppy**. These eigenspectra consists of few large eigenvalues and a large number of small eigenvalues that are distributed uniformly across an exponentially large range. We call such eigenspectra (or the corresponding quantities) "sloppy" and define this notion in Def. 8. Synthetic datasets can be constructed where these quantities are not sloppy; interpolating networks do not generalize well for such datasets. We prove that (a) the trace of the correlation of the activations, logit Jacobians, Hessian and the FIM can be upper bounded by the trace of the input correlation matrix, (b) if we assume that the activations are sloppy then the eigenspectrum of a block-diagonal approximation of the FIM is also sloppy, (c) under the assumption of weights with bounded norm, eigenvalues of activations decays faster than $\mathcal{O}(1/i)$. (2) For a Gaussian isotropic prior $N(w_0, \epsilon^{-1}I)$ centered at the initialized weights $w_0$, we calculate the optimal covariance of a Gaussian posterior $N(w, \Sigma_q)$ (where $w$ are weights of the trained network) that minimizes a PAC-Bayes generalization bound. If the Hessian at $w$ is sloppy, then we obtain a non-vacuous generalization bound. For example, for MNIST, we get a bound of 32.4% for a fully-connected network and 5.7% for LeNet. This indicates that sloppiness of inputs controls the capacity of the model. **To our knowledge, this is the only analytical, non-vacuous generalization bound for deep networks that does not use weight compression.**

(3) We characterize the **effective dimensionality of a deep network as**

$$p(n, \epsilon) = \sum_{i=1}^{p} \mathbf{1}\left\{|\lambda_i| > \frac{\epsilon}{2(n-1)}\right\},$$

where $\epsilon$ is the inverse covariance of the PAC-Bayes prior and $n$ is the number of samples. Roughly speaking, $\epsilon/(2(n-1))$ is the elbow of the eigenspectrum in Fig. 1 (top); eigenvalues of the optimal PAC-Bayes posterior beyond this threshold are dominated by the complexity term in a PAC-Bayes bound while eigenvalues before this threshold are dominated by the training error. **For sloppy eigenspectra, this dimensionality is typically a tiny fraction of the number of weights**, e.g., it is less than 0.5% of the number of weights for all networks/datasets considered in this paper, and much smaller than, say the VC-dimension.

(4) We find that the **stiff sub-space of the FIM at initialization has a strong overlap with its counterpart at the end of training, and weight updates during training primarily happen in this stiff subspace**. We exploit this observation to numerically compute a PAC-Bayes bound using a Gaussian prior whose covariance is proportional to the FIM and a Gaussian posterior whose eigenvectors are the same as those of the FIM at initialization. This is a remarkably accurate estimate of generalization gap, e.g., for LeNet on MNIST, it is 0.9% whereas the gap is about 0.5%.

All the code for experiments in this paper is provided at https://github.com/grasp-lyrl/sloppy.

## 2. Background

Consider a dataset $D_n = \{(x_i, y_i)\}_{i=1}^n$ with $n$ samples, $x_i \in X \subset \mathbb{R}^d$ and $y_i \in Y = \{1, \ldots, m\}$. We assume that this dataset is drawn from a joint distribution $D$ on $X \times Y$. A classifier $h_w : X \mapsto [0,1]^m$ parameterized by weights $w \in \mathbb{R}^p$ belongs to a hypothesis space $\{h_w : w \in \mathbb{R}^p\}$; this classifier maps inputs $x \in X$ to $m$-dimensional categorical distributions $p_w(y \mid x) \in [0,1]^m$. Let $Q$ be a distribution on hypotheses, which is implicitly a measure on $\mathbb{R}^p$. We define

(a) training error of a hypothesis $\hat{e}(h_w, D_n) = \frac{1}{n}\sum_{i=1}^n \mathbf{1}\{y_i \neq \text{argmax}_y(p_w(y \mid x_i))\}$;
(b) population error $e(h_w) = \mathbb{E}_{D_n \sim D^n}[\hat{e}(h_w, D_n)]$;
(c) training loss is $\check{e}(h_w, D_n) = -\frac{1}{n \log(2)} \sum_{i=1}^n \log p_w(y_i \mid x_i)$;
(d) empirical error and loss of the distribution $Q$ of hypotheses $\hat{e}(Q, D_n) = \mathbb{E}_{w \sim Q}[\hat{e}(h_w, D_n)]$ and $\check{e}(Q, D_n) = \mathbb{E}_{w \sim Q}[\check{e}(h_w, D_n)]$, respectively;
(e) population error of distribution $Q$ given by $e(Q) = \mathbb{E}_{D_n \sim D^n}[\hat{e}(Q, D_n)]$; and
(f) population loss is $\check{e}(Q) = \mathbb{E}_{D_n \sim D^n}[\check{e}(Q, D_n)]$.

**Hessian and Fisher Information Matrix (FIM)** The Hessian $H \in \mathbb{R}^{p \times p}$ is the second derivative of the empirical loss with respect to the weights $w$, i.e., $H_{ij} = \partial_i \partial_j \check{e}(h_w, D_n)$. The Fisher Information Matrix (FIM) $F \in \mathbb{R}^{p \times p}$ has entries

$$F_{ij} = \frac{1}{n} \sum_{k=1}^n \sum_{y=1}^m p_w(y \mid x_k) \partial_i \log p_w(y \mid x_k) \partial_j \log p_w(y \mid x_k).$$

It is important to note the expectation over the outputs $y$. The empirical FIM is an approximation of the FIM where one sets $y = y_k$. Both the Hessian and FIM are large matrices and it is difficult to compute them for modern deep networks. Therefore some of our experiments use a Kronecker-factor approximation (Martens & Grosse, 2016) of a block diagonal Hessian and FIM where cross-terms $\partial_i \partial_j$ across different layers of a deep network are set to zero.

### 2.1. PAC-Bayes Generalization Bounds

The PAC-Bayesian framework developed in Langford & Seeger (2001); McAllester (1999) allows us to estimate the population error of a randomized hypothesis with distribution $Q$ using its empirical error and its Kullback-Leibler (KL) divergence with respect to some prior distribution $P$. For any $\delta > 0$, with probability at least $1 - \delta$ over draws of the dataset $D_n$, we have

$$\text{kl}(\hat{e}(Q, D_n), e(Q)) \leq \frac{\text{KL}(Q, P) + \log(n/\delta)}{(n-1)}, \qquad (1)$$

where $\text{KL}(Q, P) = \int \text{d}Q(w) \log(\text{d}Q \,/\, \text{d}P)(w)$. We will also define a KL divergence between two Bernoulli random variables with parameters $b, a$ as $\text{kl}(b, a) = b \log(b/a) + (1 - b) \log((1-b)/(1-a))$. The right hand-side of this inequality can be minimized to compute a distribution $Q$ that has a small generalization error (Langford & Caruana, 2002; Dziugaite & Roy, 2017). Typically, we pick a simple form for distributions $Q$ and $P$, say Gaussian. We can also have hyper-parameters for the prior $P$, say the scale $\epsilon$ of the covariance of $P$ and search over this scale while optimizing the bound. See Appendix B for details.

### 2.2. Data-dependent PAC-Bayes Priors

The posterior $Q$ in (1) may depend upon the training samples $D_n$, e.g., it could be the distribution on the weight space induced by a randomized training algorithm like stochastic gradient descent (SGD). The prior $P$ can depend upon the data distribution $D$, but not the samples $D_n$ themselves. Although it is common to use priors that do not depend upon the data at all, it is has been increasingly noticed that data-distribution dependent priors may provide tighter bounds (Dziugaite & Roy, 2018). To gain intuition, recall that in the expression for the KL-divergence between two Gaussians $Q = N(w, \Sigma_q)$ and $P = N(w_0, \Sigma_p)$, we have a term of the form $(w - w_0)^\top \Sigma_p^{-1} (w - w_0)$ that depends upon the distance between trained weights $w$ and the initialization $w_0$. Priors $P$ that do not depend upon the data may therefore incur a large KL-term.

**FIM and Hessian-dependent priors** We can pick a prior using a subset of the training samples (Ambroladze et al., 2007), e.g., we can center the Gaussian prior on weights trained on this subset, to obtain a better PAC-Bayes bound—the theory allows this. Doing so leads to a worse denominator in (1), although this may be mitigated by a smaller numerator. Parrado-Hernández et al. (2012) also define expectation-priors, i.e., where we choose a prior that depends on the data *distribution* and, in practice, evaluate this prior using samples in the training dataset in lieu of the distribution. For example, PAC-Bayes theory allows both picking the prior covariance $\Sigma_p$ to be $\Sigma_p \propto F_{w_0}$ and $\Sigma_p \propto \tilde{H}_{w_0}$ where $\tilde{H}$ is the Gauss-Newton approximation of the Hessian. But while we may use all the samples to compute the FIM, we should compute the Hessian on a separate subset of the data.

## 3. Theoretical Results

We prove how sloppiness in the Hessian and the FIM is related to sloppiness of the correlations of the activations (§3.1) and the inputs (§3.2). We then exploit sloppiness to compute PAC-Bayes generalization bounds (§3.3) and develop an expression for the effective dimensionality of a deep network (§4.1). We exploit sloppiness to get effective

methods for optimizing PAC-Bayes bounds ( §5). All proofs are provided in Appendix C. The theory in this section applies for general deep networks; we will remark when restrictions are in place.

### 3.1. Sloppy Input Correlation Matrix Leads to a Sloppy FIM and Hessian

Consider a deep network with $L$ layers with weights $w = (w^0, w^1, \ldots, w^L)$. Activations of the $k^{\text{th}}$ layer are given by $h^k = \sigma(w^{k-1}h^{k-1})$, and we set $h^0 \equiv x$. The non-linearity $\sigma$ acts element-wise upon its argument and we assume that it has a bounded derivative $\left|\sigma'(x)\right| \leq a$ with $\sigma(x) = 0$ in which case $|\sigma(x)| \leq a\,|x|$; ReLU, leaky ReLUs and tanh satisfy this assumption. Preactivations (before nonlinearities) will be denoted by $u^k = w^{k-1}h^{k-1}$ for $k = 1, \cdots, L+1$, and for clarity, we use a special notation $z \equiv u^{L+1}$ to denote the logits of the network. The dimensionality of these quantities is $h^k \in \mathbb{R}^{d_k}$, $w^k \in \mathbb{R}^{d_{k+1} \times d_k}$ and $w^L \in \mathbb{R}^{m \times d_L}$. The linear map represented by $w^k$ can model both fully-connected layers and convolutional layers. For the sake of exposition, we set all the bias terms to zero.

**Theorem 1 (Trace of the FIM and Hessian are bounded by that of the input correlation matrix).** For any weights, the trace of the FIM $F_w$ and the Gauss-Newton approximation of the Hessian $\tilde{H}_w$ are both upper-bounded by

$$2ma^{2L}\text{tr}\left(\mathbb{E}[xx^\top]\right)\prod_{j=0}^{L}\left\|w^j\right\|_2^2\left(\sum_{j=0}^{L}\left\|w^j\right\|_2^{-2}\right). \quad (2)$$

The Gauss-Newton approximation which neglects the so-called $H$ terms of the Hessian (Papyan, 2019) is good towards the end of training when the logits have a small entropy. For the FIM, the above bound is remarkable however, it indicates that the trace of FIM is controlled by that the input correlations and multiplicative terms that depend upon the $\ell_2$ norm of the weights.

We can also go beyond the trace and control the entire eigenspectrum. But this is difficult to do in general because both FIM and Hessian are a result of multiple nonlinear operations on the inputs. We therefore bound the eigenvalues of a block-diagonal approximation of the FIM in terms of the eigenvalues of the activations.

**Lemma 2 (Block-diagonal approximation of the FIM is sloppy if the activations are sloppy).** Let $\text{spec}(A)$ denote the eigenvalues $(\lambda_1(A), \ldots, \lambda_p(A))$ of a matrix $A$ in descending order. For a constant $c$, let $\text{spec}(A) \preceq c\,\text{spec}(B)$ denote that $\lambda_i(A) \leq c\lambda_i(B)$ for all $i \leq p$. For any logit $z_i$, for all layers $k \leq L$, we have

$$\text{spec}\left(\mathbb{E}\left[\frac{\mathrm{d}z_i}{\mathrm{d}w^k}\frac{\mathrm{d}z_i}{\mathrm{d}w^k}^\top\right]\right) \preceq a^{2(L-k)}\prod_{j=k+1}^{L}\left\|w^j\right\|_2^2 \text{spec}(I_{d_{k+1}}) \otimes \text{spec}\left(\mathbb{E}\left[h^k h^{k\top}\right]\right), \quad (3)$$

with $\prod_{j=L+1}^{L}\left\|w_j\right\|_2^2 \equiv 1$. A similar result also holds for the sum of logits $\sum_{i=1}^{m} z_i$ as in Lemma 2 (see Corollary 15). The proof of this lemma also shows that a block-diagonal approximation of the Gauss-Newton approximation of the Hessian is sloppy if the activations are sloppy.

This lemma indicates that the eigenspectrum of the block-diagonal approximation of the FIM (concatenation of the eigenspectra of different blocks) is controlled by the eigenspectrum of the activation correlations of different layers. Our experiments show that activations of all layers (except the logits) of a trained deep network are sloppy.

### 3.2. Special Cases Where Sloppy Inputs Lead to Sloppy Activations and Thereby Sloppy FIM and Hessian

Although our experiments show that activations are sloppy if the inputs are, it seems rather difficult to prove in general. We therefore discuss two special cases where this holds. The first case is for a kernel machine with an inner product kernel while the second case assumes that the width of the network goes to infinity and weights remain bounded in $\ell_2$ norm.

**Remark 3 (Eigenspectrum of inner product kernel is controlled by that of its inputs).** Let $x_i \in \mathbb{R}^d$ for $i \leq n$ be iid random vectors. Karoui (2010, Theorem 2.1) shows that the Gram matrix of an inner product kernel $M_{i,j} = f\left(\frac{x_i^\top x_j}{d}\right)$ for some function $f$ can be approximated by

$$K = \left(f(0) + f''(0)\frac{\text{tr}(\Sigma_d^2)}{2d^2}\right)11^\top + f(0)\frac{XX^\top}{d} + v_d I_n$$

where $v_d = f\left(\frac{\text{tr}(\Sigma_d)}{d}\right) - f(0) - f'(0)\frac{\text{tr}(\Sigma_d)}{d}$.

More precisely $\|M - K\|_2 \to 0$ in probability when $d, n \to \infty$ for a fixed ratio $d/n$. Note that $v_d$ is small when $\frac{\text{tr}(\Sigma_d)}{d}$ is small. Hence, we can see that the eigenspectrum of $K$, and thereby $M$, is controlled directly by that of $XX^\top$.

Note that this argument cannot directly be used for a deep network because correlations of activations in the network are not an inner product kernel. But this indicates that even for such a kernel machine, sloppiness of the inputs leads to sloppiness of the FIM.

**Remark 4 (Infinitely wide network with bounded weight norm).** If the $\ell_2$ norm of the weights is bounded,

we show in Lemma 10 that

$$\text{tr}\left(\mathbb{E}\left[h^k h^{k\top}\right]\right) \leq a^2 \left\|w^{k-1}\right\|_2^2 \text{tr}\left(\mathbb{E}\left[h^{k-1} h^{k-1\top}\right]\right).$$

If we iterate upon this inequality down to the last layer to $\text{tr}\,\mathbb{E}[xx^\top]$ on the right hand-side (which is a constant). If the width of the $k^{\text{th}}$ layer goes to infinity, for the trace to be summable, we have that the eigenvalues of $\mathbb{E}[h^k h^{k\top}]$ decay faster than $\mathcal{O}(1/i)$.

### 3.3. Analytical Bound on Generalization (Method 1)

Consider a deep network trained to minimize the loss $\check{e}(h_w, D_n)$. Assume that $w$ is a local minimum of the objective and thus the Hessian $H_w$ is positive semi-definite. We can write $H_w$ as its orthonormal decomposition $H_w = U_w \Lambda_w U_w^\top$ where $\Lambda_w = \text{diag}(\lambda_1, \ldots, \lambda_p)$ with eigenvalues $\lambda_1, \geq \cdots \geq \lambda_p \geq 0$ arranged in descending order. Consider a Gaussian posterior $Q = N(\mu_q, \Sigma_q)$ with the mean $\mu_q = w$ fixed. We would like to compute the best $\Sigma_q$ that gives a tight PAC-Bayes bound.

We use a loose version of the bound $e(Q) \leq L(\Sigma_q) := \check{e}(Q, D_n) + \text{KL}(Q, P)/(2(n-1))$ to simplify the analytical calculation and show in Appendix B.2 that

$$\Sigma_q = U_w (\bar{\Lambda}_w)^{-1} U_w^\top, \tag{4}$$

$$\text{where } \bar{\lambda}_i = 2(n-1)\lambda_i + \epsilon \quad \forall i \leq p. \tag{5}$$

This posterior gives a non-vacuous bound on the generalization error (as explained in §4.2) and to our knowledge, this is the only analytical bound that is non-vacuous and does not use weight compression (e.g., (Zhou et al., 2018)). For example, the bound for a fully-connected network on MNIST with one hidden layer of 600 neurons is 0.32 while the test error $e(Q)$ is $\approx 0.089$. For comparison, Dziugaite & Roy (2017) *numerically optimize* (1) to get a bound of 0.161.

**Remark 5 (PAC-Bayes posterior is more spread out along sloppy eigenvectors).** In (5), we can think of the scaled prior inverse variance $\epsilon/(2(n-1))$ as a threshold beyond which the sloppy eigenvalues of the Hessian $\lambda_i$ are small enough and the loss changes so little that the optimal PAC-Bayes posterior in (1) focuses on accurately capturing the prior's covariance to obtain a small KL-term. For eigenvalues above this threshold, e.g., the stiff eigenvalues, the optimal posterior has to ensure that the empirical loss is not large. We will see in §6.3 that this phenomenon also holds for cases when posteriors are optimized.

## 4. Effective Dimensionality of a Deep Network

### 4.1. Definition of Effective Dimensionality

Motivated by Remark 5, we define the effective dimensionality for a deep network at weights $w$ as the number of eigenvalues of the Hessian $H_w$ with magnitude at least $\epsilon/(2(n-1))$, i.e.,

$$p(n, \epsilon) = \sum_{i=1}^{p} \mathbf{1}\left\{|\lambda_i(H_w)| \geq \tfrac{\epsilon}{2(n-1)}\right\}. \tag{6}$$

Appendix B.3 gives the calculation for why this is a good definition of the dimensionality. It indicates that the threshold $\epsilon/(2(n-1))$ can be thought of as the "elbow" in the eigenspectra in Fig. 1 (top), which separates the stiff eigenvalues which decrease quickly and the sloppy eigenvalues. This gives an easy way to compute the effective dimensionality, e.g., for the purposes of model selection. This is also true if we use more sophisticated, numerical, methods for optimizing the PAC-Bayes bound as shown in Fig. 2 and Table 1.

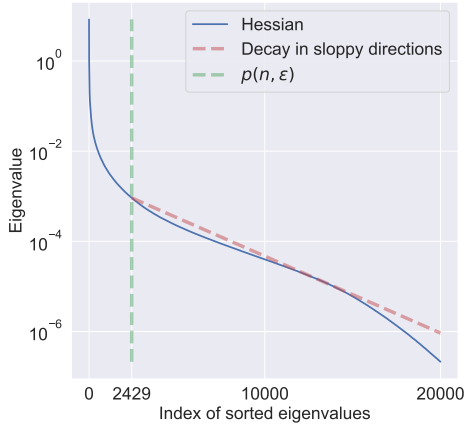


*Figure 2.* For two layer fully connected network (FC-600-2), we calculated the eigenspectrum (blue) of Kronecker-factored approximation of the Hessian at the mean of the posterior $Q$ (using numerical optimization of the PAC-Bayes bound using Method 3 in §5 but that is not important at the present). The dimensionality $p(n, \epsilon)$ (green) was calculated using the $\epsilon$ obtained by the same procedure. The red line shows the linear decay of sloppy eigenvalues (slope is 0.0004). The green line is close to the elbow and effectively splits the stiff and sloppy eigenvalues.

**Remark 6 (Why does the effective dimensionality depend on $\epsilon$?).** Our definition in (6) may seem unusual because $\epsilon$ is a user-chosen parameter but this is only an artifact of PAC-Bayes theory. As $\epsilon \to 0$, the effective dimensionality converges to the number of weights $p$, but for non-zero values of $\epsilon$, where the PAC-Bayes theory effectively restricts its predictions to a subset of the hypothesis space, this expression coupled with the analytical calculation in (5) may provide a useful way to perform model selection.

**Remark 7 (Why does the effective dimensionality depend on $n$?).** The fact that $p(n, \epsilon)$ depends upon $n$ is reminiscent of the Bayesian Information Criterion (BIC) where the the model complexity term scales with $\log n$ (Schwarz, 1978). The dependence on $n$ in our cases also arises for

| Model | #weights ($p$) | $p(n,\epsilon)/p$ (%) |
|---|---|---|
| FC-600-1 | 472,202 | 0.487 |
| FC-600-2 | 832,802 | 0.292 |
| FC-1200-1 | 944,402 | 0.245 |
| FC-1200-2 | 2,385,602 | 0.095 |
| LeNet | 44,429 | 0.184 |
| Synthetic ($c = 10^{-1}$) | 211,010 | 0.256 |
| Synthetic ($c = 10^{-3}$) | 211,010 | 0.820 |

*Table 1.* **Effective dimensionality of different models** calculated using the $\epsilon$ and Hessian from Method 3 in §5 for different networks on MNIST (except last two rows which use fully-connected networks for synthetic datasets created in Fig. 1 with different slopes of the eigenspectra $c$). We see that in all cases, $p(n,\epsilon)$ is a very small fraction of the number of weights.

similar reasons, from a balance between the training error $\hat{e}(Q, D_n)$ and the KL-term in (1). As $n \to \infty$, we see that $p(n,\epsilon) \to p$. This is because for inputs with sloppy dimensions the model needs to capture *all* the dimensions to predict accurately.

### 4.2. Definition of Sloppiness

We next build upon §4.1 to define sloppiness.

**Definition 8 (Strength factor and sloppy factor).** Let $\lambda_i(A)$ denote eigenvalues of a positive semi-definite matrix $A \in \mathbb{R}^{p\times p}$ in descending order $\lambda_1 \geq \cdots \geq \lambda_p$. The strength factor for a model with effective dimensionality $p(n,\epsilon)$ at a local minimum $w$ (where $H_w$ is positive semi-definite) is defined to be

$$s(n,\epsilon) = \sum_{i=1}^{p(n,\epsilon)} 1 + \log\left(\frac{2(n-1)\lambda_i(H_w)}{\epsilon} + 1\right). \quad (7)$$

The strength factor characterizes the stiff eigenvalues of the eigenspectrum. For a matrix $A$, the sloppy factor for such a model at index $r$ is defined to be

$$c(A,r) = \sup\{c' \geq 0 : \lambda_i(A) \leq \lambda_r(A) e^{-c'(i-r)} \forall i \geq r \geq 1\} \quad (8)$$

This definition implicitly means that the small eigenvalues beyond $\lambda_r(A)$ are uniformly distributed across an exponentially large range $(\lambda_r, \lambda_p)$ if $c(A,r) > 0$. We will be primarily interested in setting the index $r$ to be simply $p(n,\epsilon)$. Note that sloppiness is a phenomenon pertaining to the *non-zero eigenvalues* of a matrix and is relevant even if the matrix is singular, e.g., the FIM loses rank for non-identifiable models like deep networks (Amari et al., 2002).

**How do the strength and sloppy factor affect generalization?** Let us simplify notation to write the sloppy factor as $c(n,\epsilon) \equiv c(H_w, p(n,\epsilon))$. Under the assumption that the $c(n,\epsilon)$ is non-negative, when the training error $\check{e}(h_w, D_n)$ is close to zero, we show in Appendix B a loose version of PAC-Bayes bound $\check{e}(Q, D_n) + \mathrm{KL}(Q,P)/(2(n-1))$ (this was also used in Method 1 in §3.3) is

$$\frac{s(n,\epsilon) + 2/c(n,\epsilon) + \epsilon\|w - w_0\|_2^2}{4(n-1)}. \quad (9)$$

Thus, the strength and sloppy factor together determine the generalization performance. If the Hessian $H_w$ is sloppy, then the effective dimensionality $p(n,\epsilon)$ is small. This ensures that both $s(n,\epsilon)$ and $1/c(n,\epsilon)$ are small compare to $n$. The third term $\epsilon\|w - w_0\|^2$ comes from the the fact that the mean of $P$ and $Q$ are different. It is typically not large compared to $n$. For example, for a two-layer fully-connected network on MINST, $p(n,\epsilon) = 2429$, $s(n,\epsilon) = 6810$, $1/c(n,\epsilon) = 2545$, and $\epsilon\|w - w_0\|^2 = 8526$, with $n = 55000, \epsilon = 101.3$). For comparison, if we have an isotropic Hessian $\lambda_i \equiv \lambda$, either $s(n,\epsilon)$ or $1/c(n,\epsilon)$ will be $\mathcal{O}(p)$ and $p$ is about 0.8 million.

This suggests that **even if the hypothesis class of deep networks is very large, sloppiness of $H_w$, which is inherited from sloppiness of the input data, restricts the set of hypotheses that the trained model belongs to**. The three quantities that we have defined here $p(n,\epsilon)$, $s(n,e)$ and $c(n,\epsilon)$ together help understand this phenomenon; see Appendix F their values for other models.

## 5. Numerical Methods to Compute PAC-Bayes Bounds

We next discuss three methods to numerically optimize the PAC-Bayes bound. These methods exploit the observation in our experiments that there is a large overlap between the subspace spanned by the stiff eigenvectors of the FIM at the end of training with the corresponding subspace at the beginning of training (Fig. 4). Similarly, there is a large overlap between the subspace spanned by the stiff eigenvectors of the Hessian with that of the FIM (Fig. 3). We will use the notation $\mathrm{Ev}(A)$ to denote the set of eigenvectors of the matrix $A$, arranged in decreasing order of eigenvalues.

**Method 2: Eigenvectors of covariance of $Q$ are fixed to those of FIM at initialization** ($\mathrm{Ev}(\Sigma_q) = \mathrm{Ev}(F_{w_0})$) We pick the posterior covariance matrix to have the same eigenvectors as that of FIM at initialization and optimize only its eigenvalues, i.e., we set

$$P = N(w_0, \epsilon^{-1} I),\ Q = N(w, \Sigma_q = U_{w_0} \bar{\Lambda}_w U_{w_0}^\top), \quad (10)$$

where $F_{w_0} = U_{w_0} \Lambda U_{w_0}^\top$ is the orthonormal decomposition of the FIM at initialization $w_0$. We can optimize the bound in (1) numerically over the mean of the posterior $w$, eigenvalues of the covariance $\bar{\Lambda}_w$ and the scale of the prior $\epsilon$.

**Method 3: Eigenvectors of covariance of $Q$ are the same as those of the Hessian** ($\mathrm{Ev}(\Sigma_q) = \mathrm{Ev}(H_w)$) We show

in Appendix B.2 that the covariance of the optimal Gaussian posterior has the same eigenvectors as those of the Hessian. Building upon this, we modify the eigenvectors of $\Sigma_q$ while optimizing the bound as

$$P = N(w_0, \epsilon^{-1} I),\ Q = N(w, \Sigma_q = U_w \bar{\Lambda}_w U_w^\top), \qquad (11)$$

where $H_w = U_w \Lambda U_w^\top$ is the orthonormal decomposition of the Hessian at weights $w$. The variables of optimization are the mean of the posterior $w$, eigenvalues of the covariance $\bar{\Lambda}_w$, and the scale of the prior $\epsilon$. Note that this involves recomputing the Hessian at every candidate weight $w$ during optimization of the bound.

**Method 4: Covariance of $P$ is proportional to FIM at initialization; eigenvectors of covariance of $Q$ are the same as those of FIM at initialization** ($\Sigma_p = aF_{w_0} + \epsilon^{-1}$; $\text{Ev}(\Sigma_q) = \text{Ev}(F_{w_0})$) This is a data-distribution dependent prior. We set

$$P = N(w_0, aF_{w_0} + \epsilon^{-1} I),\ Q = N(w, \Sigma_q = U_{w_0} \bar{\Lambda}_w U_{w_0}^\top), \qquad (12)$$

where $F_{w_0} = U_{w_0} \Lambda U_{w_0}^\top$ is the orthonormal decomposition of the FIM at initialization $w_0$. The variables of optimization of the bound are the posterior mean $w$, eigenvalues $\bar{\Lambda}_w$, and scalar constants $a$ and $\epsilon^{-1}$.

## 6. Empirical Validation

We use fully-connected networks (of varying widths, and up to two hidden layers), convolutional networks (LeNet, ALL-CNN of Springenberg et al. (2015) and wide residual network of Zagoruyko & Komodakis (2016)) of varying sizes on MNIST (LeCun et al., 1990) and CIFAR-10 (Krizhevsky, 2009) for empirical validation of our theoretical results. See Appendices A and D for further details.

To be able to work with Hessian/FIM of large networks, in some cases, e.g., Fig. 1 we compute fewer eigenvalues, but compute them exactly without any approximations. When the Hessian/FIM are used for optimizing the PAC-Bayes bound (e.g., Methods 3–4) we use Kronecker-factor (KFAC) approximation of the Gauss-Newton matrix in Backpack (Dangel et al., 2020). We have also developed a trick in PyTorch (see Appendix E) that allows us to quickly estimate $\check{e}(Q, D_n)$ using a large number of samples (we use 150, for comparison Dziugaite & Roy (2017) use 1). This allows us to optimize the PAC-Bayes bound with much fewer iterations. This trick that we have developed is also useful for Bayesian deep learning (Wilson, 2020).

### 6.1. Sloppiness of the Hessian, FIM and Other Related Quantities in the Network

Appendix G.1 shows the eigenspectra of the Hessian, FIM and correlations of the activations, logit Jacobians and activation gradients for two and three-layer fully-connected networks on MNIST and All-CNN and a wide residual network on CIFAR-10; Appendix G.4 shows some eigenspectra at the middle of training; Appendix G.5 shows eigenspectra for synthetic datasets. The eigenspectra are qualitatively the same as those in Fig. 1 so we do not repeat them in the main text. Fig. 3 studies how eigenspectra of FIM and Hessian compare to their KFAC approximations.

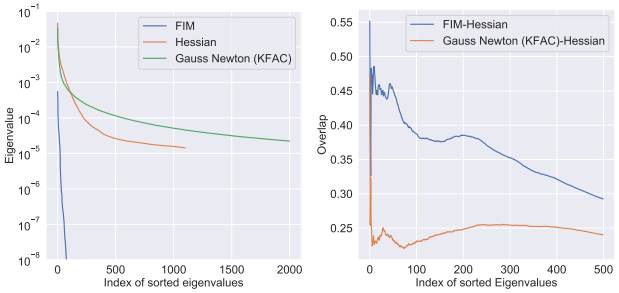


*Figure 3.* (Left) **Eigenspectra of FIM, Hessian and a KFAC approximation of the Gauss-Newton matrix for a two-layer fully-connected network on MNIST.** Even if FIM's eigenvalues are quite different, its eigenvectors have a large inner product with those of the Hessian (right), much larger than a random vector. KFAC is a good approximation for the eigenvalues of the Hessian but eigenvectors computed from KFAC are quite different from those of the Hessian. This also shows that eigenvectors of the FIM have a strong overlap with those of the Hessian.

### 6.2. Overlap of the Stiff Subspaces of the FIM/Hessian at the End of Training with that at Initialization

Fig. 4 (left) computes the overlap of the subspace spanned by the top $k$ eigenvectors of the FIM at the end of training with that at the initialization. Fig. 4 (right) shows a projection of the change in the weights (difference between weights at the and that at initialization) into the subspace spanned by the top $k$ eigenvectors of the FIM. Both these overlaps are large, e.g., projection into a random subspace of dimension $k$ for the latter. This suggests that during training, weights change predominantly in the stiff subspace of the FIM at initialization.

We also constructed a third network (denoted -v2) as follows. Given a trained network $w$ from initialization $w_0$, we train $w$ for more epochs to minimize the training loss and a penalty $\|w - w_0\|^2$ which pulls it closer to $w_0$—without changing the training/validation error much. We find in Fig. 4 (right) that this variant has a much larger projection into the stiff eigenspace, and thereby a smaller overlap with the sloppy eigenvectors. Thus, weights can effectively "come back" towards the initialization in the sloppy subspace although they evolved during training in the stiff subspace. See Fig. S-14 for more details.

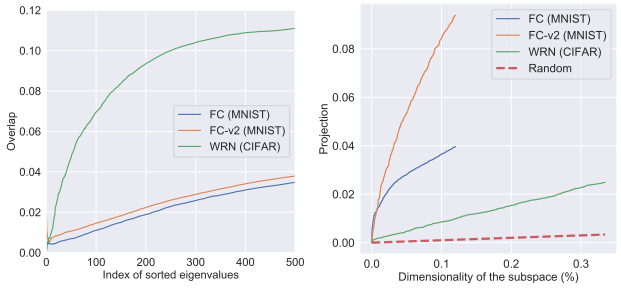


*Figure 4.* (Left) **Overlap between the subspace of the top $k$ eigenvectors (X-axis) of the FIM at the end of training with that at initialization** ($\left\|\mathrm{Ev}_k(F_w)^\top \mathrm{Ev}_k(F_{w_0})\right\|_F^2/k$) for fully-connected (FC) and convolutional networks (WRN) is far larger than overlap of two random subspaces in $\mathbb{R}^p$, which is approximately $10^{-6}$. (Right) **Projection $\|\mathrm{Ev}_k(F_{w_0})\Delta w\|_2^2/\|\Delta w\|_2^2$ of the change in weights** (where $\Delta w = w - w_0$) **into the subspace of the top $k$** (shown as percentage of weights because different networks have different sizes) eigenvectors of the FIM is much larger than the projection into a random subspace.

### 6.3. PAC-Bayes Bounds

Table 2 shows results of using different methods to calculate PAC-Bayes bounds. It is remarkable that for all networks, the analytical method using Method 1 obtains a non-vacuous bound. Methods 2–4 obtain bounds that are comparable to those of existing methods, e.g.,Dziugaite & Roy (2017); Wu et al. (2021). Appendix D discusses a number of technical details in how each method are implemented numerically, e.g., sampling weights from posteriors whose covariance is represented as a KFAC approximation. As discussed in §5, our methods in Table 2 exploit the fact that the stiff subspaces of the Hessian/FIM have a strong overlap at the beginning and that training predominantly takes place in the stiff subspace of the FIM at initialization. This experiment therefore shows that we can exploit sloppiness to compute non-vacuous generalization bounds for deep networks.

**We next study how similar the optimal PAC-Bayes posterior covariance computed by numerical optimization and the one obtained analytically are.** We will contrast two numerical methods, our Method 3 with $\mathrm{Ev}(\Sigma_q) = \mathrm{Ev}(H_w)$, and the method of Dziugaite & Roy (2017) in Table 2 which uses a diagonal posterior covariance. We find in Fig. 5 that the eigenvalues of the posterior computed by Method 3 match remarkably well with our analytical expression in (5) for Method 1. This sheds light into why our analytical PAC-Bayes bound is non-vacuous—essentially numeric optimization finds a very similar posterior as that of the analytical method. It is however important to run even the numerical optimization in the appropriate basis. The bounds obtained by Dziugaite & Roy (2017) for a diagonal posterior are worse than Method 3 in Table 2 and as Fig. 5 (left) indicates, this is because their posterior has a smaller

| Model | Method | | | | | |
|---|---|---|---|---|---|---|
| | 1 | 2 | 3 | 4 | A | B |
| FC-600-1 | 0.3241 | 0.1590 | 0.1357 | 0.1323 | 0.161 | 0.1198 |
| FC-600-2 | 0.3794 | 0.1767 | 0.1540 | 0.1397 | 0.186 | 0.1443 |
| FC-1200-1 | 0.3509 | 0.1523 | 0.1515 | 0.1486 | 0.179 | 0.1413 |
| FC-1200-2 | 0.3915 | 0.2017 | 0.1817 | 0.1702 | 0.223 | - |
| LeNet-5 | 0.0572 | 0.0099 | 0.0188 | 0.0092 | - | - |

*Table 2.* **PAC-Bayes bounds on MNIST for different methods.** Methods 1–4 are ours, described in §§3.3 and 5. The prior for Method 4 is $\Sigma_p = aF_{w_0} + \epsilon^{-1}$; all other methods use $P = N(w_0, \epsilon^{-1}I)$. The penultimate column (A) is from Dziugaite & Roy (2017) and optimizes the diagonal of the covariance of $Q$ numerically. The final column (B) which sets eigenvectors of covariance of $Q$ to be the same as that of the block-diagonal Hessian is from Wu et al. (2021). For fully-connected nets, the error $e(Q)$ ranges from 6–8 $\times 10^{-2}$ for Method 1 and 1–4 $\times 10^{-2}$ for all other methods. For LeNet-5 the error $e(Q)$ ranges from 1–2 $\times 10^{-2}$ for all methods. See Appendix F for the extended version.

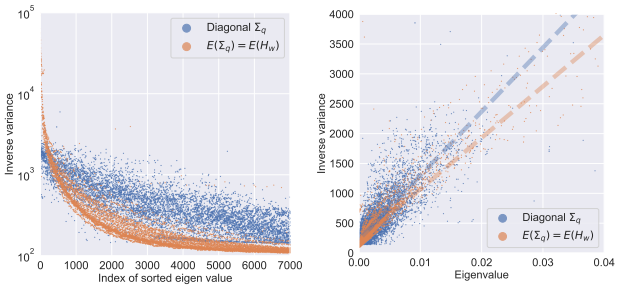


*Figure 5.* **Posterior covariance computed by numerically optimizing PAC-Bayes bound is aligned with sloppy directions.** (Left) Inverse eigenvalues of the posterior covariance ($\bar{\lambda}_i^{-1}$) *indexed by eigenvalues* of the Hessian (X-axis) for $\mathrm{Ev}(\Sigma_q) = \mathrm{Ev}(H_w)$ (orange, our Method 3) and a diagonal-$\Sigma_q$ (blue, this is the method of Dziugaite & Roy (2017)). For the latter, abscissa are the indices of the sorted diagonal of the Hessian. (Right) Inverse eigenvalues of the posterior covariance ($\bar{\lambda}_i^{-1}$) *plotted against eigenvalues* of the Hessian for $\mathrm{Ev}(\Sigma_q) = \mathrm{Ev}(H_w)$ (orange) and for diagonal-$\Sigma_q$ (blue). For the latter the X-axis is the corresponding entry on the diagonal of the Hessian. For the orange pointcloud, we obtain a surprisingly accurate fit for our analytical expression ($\bar{\lambda}_i^{-1} = 2(n-1)\lambda_i + \epsilon$: we get $n \sim 43,000$ (true $n = 55,000$) and $\epsilon \sim 210.6$ (true $\epsilon = 101.3$) with $R^2 = 0.972$, which indicates good fit for regression.

variance in the sloppy subspace (blue cloud).

## 7. Related Work

**Sloppy models in physics and biology** Our work is inspired by Brown et al. (2004); Gutenkunst et al. (2007) who noticed that regression models fitted to systems biology data have few stiff parameters that determine the outcome and a large number of sloppy parameters which only weakly determine the outcome. These authors have developed an elaborate geometric understanding of this phenomenon, see Transtrum et al. (2011) and references therein. While sloppiness is thought to be a universal property of parametric models (Waterfall et al., 2006), the mechanism that causes models to be sloppy has not been studied yet. This work has also exclusively focused on the under-parameterized regime. We connect the sloppiness of a deep network to the sloppiness of inputs and show that if the inputs are sloppy, then key quantities pertaining to the model, e.g., activations, FIM and Hessian etc., are also sloppy.

**Hessian and the FIM of deep networks** have been studied to understand the local geometry of the energy landscape and the behavior of SGD, see Hochreiter & Schmidhuber (1997); Chaudhari et al. (2017); Fort & Ganguli (2019), among others. FIM has been used to study optimization (Amari, 1998; Martens & Grosse, 2016; Karakida et al., 2019), gradient diversity (Yin et al., 2018; Chaudhari & Soatto, 2018), and generalization (Achille et al., 2019). A number of these works have pointed out that the Hessian and the FIM have spiky/large eigenvalues (Papyan, 2019) along with a bulk of near-zero eigenvalues (Papyan, 2018; Pennington & Bahri), and that this indicates that the energy landscape, or the prediction space, is locally flat. We focus on the decay pattern of the eigenspectra of these matrices and discover that it mirrors the decay pattern of the inputs for typical datasets. We see a strong overlap of the stiff subspace of the Hessian/FIM at initialization with that at the end of training; this is consistent with the analysis in Gur-Ari et al. (2018); Chizat et al. (2019).

**Generalization** PAC-Bayes bounds for deep networks have been obtained using the methods of (Langford & Caruana, 2002) by Dziugaite & Roy (2017); Dziugaite (2020); Zhou et al. (2018). While analytical generalization bounds are often vacuous (Bartlett et al., 2017; 2021; Neyshabur et al., 2017), we show that if we exploit the sloppiness of the Hessian, then we can obtain non-vacuous analytical bounds. We show that the posterior computed by the method of Dziugaite & Roy (2017) aligns well with sloppy eigenvalues of the Hessian/FIM. We build upon this work and show the benefits of sloppiness by providing data-distribution dependent PAC-Bayes bounds (also see Dziugaite & Roy (2018)).

Our Method 3 is related to the work of Wu et al. (2021). The difference is that they assume that the block-diagonal approximation of the Hessian decouples into a Kronecker product of the Hessian of the activations and the input correlation matrix; we instead optimize the PAC-Bayes prior using the top few eigenvectors of the full Hessian for some models (LeNet) and the Kronecker-factored approximation of the blocks for others. They analyze the case when the data matrix has rank 1 and Hessian has rank $m-1$ ($m$ is the number of classes). Our experiments show that they are both full-rank but sloppy, and we therefore analyze this instead.

Bartlett et al. (2020) show that a minimum norm interpolating solution of over-parameterized linear regression can predict accurately if the data matrix has a long tail of small eigenvalues. Our notion of effective dimensionality is also seen in their calculations: roughly speaking, larger our sloppiness factor $c$ in Def. 8, better the excess risk in their linear regression, which is consistent with Fig. 1 (bottom right). (Liang & Rakhlin, 2018) show similar results on the minimum-norm interpolating solution for kernel regression.

## 8. Discussion

We showed that for typical datasets, the sloppy decay pattern of eigenvalues of the input correlation matrix is mirrored in key quantities of the deep network, e.g., eigenspectra of the activation correlations, activation gradients, logit Jacobians, Hessian and the FIM. This suggests that the "simplicity", more precisely, the sloppiness, of inputs of high-dimensional datasets controls the representations learned by the network. We validated this hypothesis by providing non-vacuous PAC-Bayes generalization bounds for deep networks, including analytical ones. Our calculations also provided a simple definition of the effective dimensionality of a deep network and we showed how this number can be much smaller than the number of weights.

## 9. Acknowledgments

This work was supported by grants from the National Science Foundation (2145164) and the Office of Naval Research (N00014-22-1-2255), and cloud computing credits from Amazon Web Services.

## References

Achille, A., Rovere, M., and Soatto, S. Critical learning periods in deep networks. In *International Conference on Learning Representations*, 2018.

Achille, A., Paolini, G., and Soatto, S. Where is the information in a deep neural network? *arXiv preprint arXiv:1905.12213*, 2019.

Amari, S.-i. Natural gradient works efficiently in learning.

*Neural Computation*, 10(2):251–276, 1998. doi: 10.1162/089976698300017746.

Amari, S.-i., Park, H., and Ozeki, T. Geometrical singularities in the neuromanifold of multilayer perceptrons. *Advances in neural information processing systems*, 1:343–350, 2002.

Ambroladze, A., Parrado-Hernández, E., and Shawe-Taylor, J. Tighter pac-bayes bounds. *Advances in neural information processing systems*, 19:9, 2007.

Bartlett, P. L., Foster, D. J., and Telgarsky, M. J. Spectrally-normalized margin bounds for neural networks. In *Advances in Neural Information Processing Systems*, pp. 6240–6249, 2017.

Bartlett, P. L., Long, P. M., Lugosi, G., and Tsigler, A. Benign overfitting in linear regression. *Proceedings of the National Academy of Sciences*, 117(48):30063–30070, 2020.

Bartlett, P. L., Montanari, A., and Rakhlin, A. Deep learning: A statistical viewpoint. *arXiv preprint arXiv:2103.09177*, 2021.

Botev, A., Ritter, H., and Barber, D. Practical Gauss-Newton Optimisation for Deep Learning. In *Proceedings of the 34th International Conference on Machine Learning*, pp. 557–565. PMLR, 2017.

Brown, K. S., Hill, C. C., Calero, G. A., Myers, C. R., Lee, K. H., Sethna, J. P., and Cerione, R. A. The statistical mechanics of complex signaling networks: Nerve growth factor signaling. *Physical biology*, 1(3):184, 2004.

Chaudhari, P. and Soatto, S. Stochastic gradient descent performs variational inference, converges to limit cycles for deep networks. *Proc. of International Conference of Learning and Representations (ICLR), Apr 30-May 3, 2018*, 2018.

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. In *Proc. of the International Conference on Learning Representations (ICLR)*, 2017.

Chizat, L., Oyallon, E., and Bach, F. On lazy training in differentiable programming. *Advances in Neural Information Processing Systems*, 32:2937–2947, 2019.

Dangel, F., Kunstner, F., and Hennig, P. BackPACK: Packing more into backprop. In *International Conference on Learning Representations*, 2020.

Dziugaite, G. K. *Revisiting Generalization for Deep Learning: PAC-Bayes, Flat Minima, and Generative Models*. PhD thesis, University of Cambridge, 2020.

Dziugaite, G. K. and Roy, D. M. Computing Nonvacuous Generalization Bounds for Deep (Stochastic) Neural Networks with Many More Parameters than Training Data. In *Proc. of the Conference on Uncertainty in Artificial Intelligence (UAI)*, 2017.

Dziugaite, G. K. and Roy, D. M. Data-dependent PAC-Bayes priors via differential privacy. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, pp. 8440–8450, 2018.

Fort, S. and Ganguli, S. Emergent properties of the local geometry of neural loss landscapes. *arXiv:1910.05929 [cs, stat]*, October 2019.

Gur-Ari, G., Roberts, D. A., and Dyer, E. Gradient Descent Happens in a Tiny Subspace. *arXiv:1812.04754 [cs, stat]*, December 2018.

Gutenkunst, R. N., Waterfall, J. J., Casey, F. P., Brown, K. S., Myers, C. R., and Sethna, J. P. Universally sloppy parameter sensitivities in systems biology models. *PLoS computational biology*, 3(10):e189, 2007.

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

Karakida, R., Akaho, S., and Amari, S.-i. Universal Statistics of Fisher Information in Deep Neural Networks: Mean Field Approach. *arXiv:1806.01316 [cond-mat, stat]*, October 2019.

Karoui, N. E. The spectrum of kernel random matrices. *Annals of Statistics*, 38:1–50, 2010.

Krizhevsky, A. *Learning Multiple Layers of Features from Tiny Images*. PhD thesis, Computer Science, University of Toronto, 2009.

Langford, J. and Caruana, R. (Not) bounding the true error. *Advances in Neural Information Processing Systems*, 2:809–816, 2002.

Langford, J. and Seeger, M. Bounds for averaging classifiers. 2001.

LeCun, Y., Boser, B. E., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W. E., and Jackel, L. D. Handwritten digit recognition with a back-propagation network. In *Advances in Neural Information Processing Systems*, pp. 396–404, 1990.

Liang, T. and Rakhlin, A. Just interpolate: Kernel "ridgeless" regression can generalize. *CoRR*, abs/1808.00387, 2018. URL http://arxiv.org/abs/1808.00387.

Martens, J. and Grosse, R. Optimizing Neural Networks with Kronecker-factored Approximate Curvature. *arXiv:1503.05671 [cs, stat]*, May 2016.

McAllester, D. A. PAC-Bayesian model averaging. In *Proceedings of the Twelfth Annual Conference on Computational Learning Theory*, pp. 164–170, 1999.

Neyshabur, B., Bhojanapalli, S., McAllester, D., and Srebro, N. Exploring Generalization in Deep Learning. *arXiv:1706.08947 [cs]*, July 2017.

Papyan, V. The full spectrum of deepnet hessians at scale: Dynamics with sgd training and sample size. *arXiv preprint arXiv:1811.07062*, 2018.

Papyan, V. Measurements of three-level hierarchical structure in the outliers in the spectrum of deepnet hessians. *arXiv preprint arXiv:1901.08244*, 2019.

Parrado-Hernández, E., Ambroladze, A., Shawe-Taylor, J., and Sun, S. PAC-Bayes bounds with data dependent priors. *The Journal of Machine Learning Research*, 13(1):3507–3531, 2012.

Pennington, J. and Bahri, Y. Geometry of Neural Network Loss Surfaces via Random Matrix Theory. pp. 9.

Sagun, L., Bottou, L., and LeCun, Y. Singularity of the hessian in deep learning. *arXiv:1611:07476*, 2016.

Schwarz, G. Estimating the dimension of a model. *The annals of statistics*, pp. 461–464, 1978.

Springenberg, J. T., Dosovitskiy, A., Brox, T., and Riedmiller, M. Striving for Simplicity: The All Convolutional Net. *arXiv:1412.6806 [cs]*, April 2015.

Transtrum, M. K., Machta, B. B., and Sethna, J. P. The geometry of nonlinear least squares with applications to sloppy models and optimization. *Physical Review E*, 83(3):036701, March 2011. ISSN 1539-3755, 1550-2376. doi: 10.1103/PhysRevE.83.036701.

Waterfall, J. J., Casey, F. P., Gutenkunst, R. N., Brown, K. S., Myers, C. R., Brouwer, P. W., Elser, V., and Sethna, J. P. Sloppy-Model Universality Class and the Vandermonde Matrix. *Physical Review Letters*, 97(15):150601, October 2006. doi: 10.1103/PhysRevLett.97.150601.

Wilson, A. G. The case for bayesian deep learning. *arXiv preprint arXiv:2001.10995*, 2020.

Wu, Y., Zhu, X., Wu, C., Wang, A., and Ge, R. Dissecting Hessian: Understanding Common Structure of Hessian in Neural Networks. *arXiv:2010.04261 [cs, stat]*, June 2021.

Yin, D., Pananjady, A., Lam, M., Papailiopoulos, D., Ramchandran, K., and Bartlett, P. Gradient diversity: A key ingredient for scalable distributed learning. In *International Conference on Artificial Intelligence and Statistics*, pp. 1998–2007. PMLR, 2018.

Zagoruyko, S. and Komodakis, N. Wide residual networks. In *British Machine Vision Conference 2016*. British Machine Vision Association, 2016.

Zhou, W., Veitch, V., Austern, M., Adams, R. P., and Orbanz, P. Non-vacuous generalization bounds at the ImageNet scale: A PAC-Bayesian compression approach. In *International Conference on Learning Representations*, 2018.

## A. Details of the experimental setup

**Data** We use the MNIST dataset for experiments on fully-connected networks and LeNet. We setup a binary classification problem (we map {0,1,2,3,4} to label 0 and {5,6,7,8,9} to label 1). We use 55000 samples from the training set to train the model and to optimize the PAC-Bayes bound. We set aside 5000 samples for calculating the FIM, which is used in Method 4 of PAC-Bayes bound optimization. Strictly speaking, it is not required to do so because a prior that depends upon the FIM is an expectation-prior (as discussed in Parrado-Hernández et al. (2012)) but we set aside these samples to compare in a systematic manner to existing methods in the literature that use 55,000 samples. Test error of all models is estimated using the validation set of MNIST. We use the CIFAR-10 dataset for experiments using two architectures, an All-CNN network and a wide residual network. For CIFAR-10, we use 50, 000 samples for training and 10, 000 samples for estimating the test error. No data augmentation is performed for MNIST, for CIFAR-10 we randomly flip images (left to right) with probability 0.5 and select random crops of size 32×32 after adding a padding of 4 pixels on the width and height.

**Architectures** For experiments on MNIST, we use LeNet-5 (this is a network with two convolutional layers of 20 and 50 channels respectively, both of 5×5 kernel size, and a fully-connected layer with 500 hidden neurons) and fully-connected net with one or two layers and 600 or 1200 neurons on each layer. The latter are denoted as FC-600-1, or FC-1200-2 in our experimental section. For CIFAR-10, we use ALL-CNN (in order to reduce the number of weights, we reduced the number of channels in the first set of blocks to 64, and in the second set of blocks to 128; this is down from 96 and 192 respectively in the original network) and wide residual net with depth 10 and a widening factor of 8. In the latter case, in order to reduce the number of weights which makes computing Hessian amenable, we reduce the number of channels in each block of the WRN to [4, 32, 64, 128], down from [16, 128, 256, 512] for a widen factor of 8.

**Training procedure** We train for 30 epochs on MNIST and for 100 epochs on CIFAR-10. The batch-size is fixed to 500 for both datasets. For all experiments with train with Adam and reduce the learning rate using a cosine annealing schedule starting from an initial learning rate of $10^{-3}$ and ending at a learning rate of $10^{-5}$.

**Constructing the v2 model in Fig. 4** We construct the v2 model by training in two phases. The first phase proceeds as usual: we initialize the model at $w_0$ and train as discussed above to obtain the trained weights $w^1$. In the second phase, and training further for 20 epochs with an objective that is the sum of the original training objective and an addition term spring-force-like term:

$$\hat{e}(h_w, D_n) + \alpha \|w - w_0\|_2^2.$$

The second term forces the weight updates the reduce the Euclidean distance with respect to $w_0$. The coefficient $\alpha$ is set to be twice that of the learning rate.

**Hyperparameters for optimizing the PAC-Bayes bound** In Methods 2, 3, 4, we choose $b = 0.01$, $c = 0.1$ for the penalty of the scaling parameters in the prior. In method 1, we choose $b = 0.1$, $c = 0.05$. For all PAC-Bayes bound optimization experiments, we use confidence parameter $\delta = 0.025$.

**Optimizing the PAC-Bayes bound** We use batch size of 1100, we draw 150 samples from the posterior $Q$ to estimate $\hat{e}(Q, D_n)$ for each weight update; see Appendix E for some more implementation details of how to compute a large number of samples efficiently. Adam is used to optimize the PAC-Bayes bound. For Methods 2, 3, 4, we first train for 100 epochs with learning rate $10^{-3}$ and train for another 150 epochs while decaying the learning rate by a multiplicative factor of 0.95 every 5 epochs. We found that for this problem, having a constant learning rate at the beginning is beneficial, instead of decaying the learning rate immediately, say using a cosine schedule. For the reproduction of the approach of Dziugaite & Roy (2017) (which we denote as $\text{diag}(\Sigma_q) = \Lambda$), we train for 300 epochs for the second phase with decaying learning rate.

**Atypical problems** For atypical problems in, we constructed a training set of 50,000 samples and a validation set of 10,000 samples. Inputs $x_i \in \mathbb{R}^{200}$ were generated from distribution $N(0, \Lambda)$ where $\Lambda = (\lambda_1, ..., \lambda_{200})$. We set $\lambda_i = b \exp(-ci)$ where and $b/c = 50$; fixing the ratio $b/c$ to be a constant keeps the trace of the data correlation matrix to be about the same for different values of $c$. Labels were generated by $y_i = \text{argmax}_{y \in [m]} p_w^t(y|x_i)$, where $p_w^t$ is the teacher network randomly initialized with one hidden layer and ten output classes. We train fully-connected networks on these synthetic datasets for 50 epochs; Adam is used with a batch-size of 500 and a cosine learning rate schedule with learning rate that ranges from $10^{-3}$ to $10^{-5}$.

We constructed datasets of Gaussian inputs of varying degrees of sloppiness by selecting decay patterns for the eigenvalues of a diagonal data correlation matrix. For $n^{-1}\operatorname{diag}(XX^\top) = \Lambda$ where $\Lambda = (\lambda_1, \ldots, \lambda_d)$ are eigenvalues in descending order, we set $\lambda_i = b\exp(-ci)$ where $b, c$ are constants. The trace of this correlation matrix is roughly $b/c$ which we keep constant for different datasets. Larger the value of the "sloppy factor" $c$, more sharp the decay for the eigenspectrum of the data matrix. We randomly initialize a two layer fully-connected neural network with 10 output classes (called the teacher) and use it to label a dataset of such inputs. Note that since the teacher's weights in the first layer multiply the inputs, the correlation matrix of the first layer activations is non-diagonal and we are not being unduly restrictive in picking a diagonal data correlation matrix. We then fit student networks (fully-connected networks with two layers) on this data until they interpolate on the training dataset. Our goal is to study (i) how the various quantities discussed in this paper, e.g., the Hessian, FIM, activations, activation gradients, logit Jacobians , depend upon the sloppiness of the data matrix; (ii) whether the student can interpolate on sloppy datasets without over fitting. Fig. 1 shows the results of the experiment.

## B. Calculation of the effective dimensionality of a deep network

### B.1. PAC-Bayes bounds

**Theorem 9 (PAC-Bayes generalization bound McAllester (1999); Langford & Seeger (2001)).** For every $\delta > 0$, $n \in \mathbb{N}$, distribution $D$ on $\mathbb{R}^k \times \{0,1\}^m$, and distribution $P$ on $\mathcal{H}$, with probability at least $1-\delta$ over $D_n \sim D^n$, for all distributions $Q$ on $\mathcal{H}$,

$$\operatorname{kl}(\hat{e}(Q, D_n), e(Q)) \leq \frac{\operatorname{KL}(Q,P) + \log\frac{n}{\delta}}{n-1}$$

We have the following lower-bound from Pinksker's inequality on the KL-divergence between two Bernoulli random variables:

$$2(q-p)^2 \leq \operatorname{kl}(q,p).$$

We can invert this inequality to get

$$\operatorname{kl}^{-1}(q,p) \leq q + \sqrt{p/2}.$$

When this is substituted into the above PAC-Bayes bound (1), we have

$$e(Q) \leq \hat{e}(Q) + \sqrt{\frac{\operatorname{KL}(Q,P) + \log\left(\frac{n}{\delta}\right)}{2(n-1)}}.$$

Since

$$\mathbf{1}\left\{ y_i \neq \operatorname*{argmax}_y (p_w(y \,|\, x_i)) \right\} \leq -\frac{1}{\log 2} \log p_w(y_i|x_i)$$

we also have

$$\hat{e}(Q) \leq \check{e}(Q).$$

Now set $\epsilon = c\exp(j/b)$, for $j \in \mathbb{N}$ and for a fixed $b, c \geq 0$, by the calculations in Appendix D.5, we see that

$$e(Q) \leq \check{e}(Q) + \sqrt{\frac{\operatorname{KL}(Q,P) + 2\log\left(b\log\frac{c}{\epsilon}\right) + \log\left(\frac{\pi^2 n}{6\delta}\right)}{2(n-1)}},$$

holds with probability $1-\delta$.

### B.2. Calculation for the closed form expression for eigenvalues of the inverse posterior covariance in (5)

The KL-divergence between two multivariate Gaussians $Q = N(\mu_q, \Sigma_q)$, $P = N(\mu_p, \Sigma_p)$ be two multivariate Gaussians is

$$\operatorname{KL}(Q,P) = \frac{1}{2}\left( \operatorname{tr}(\Sigma_p^{-1}\Sigma_q) - p + (\mu_p - \mu_q)^\top \Sigma_p^{-1}(\mu_p - \mu_q) + \log\left(\frac{\det \Sigma_p}{\det \Sigma_q}\right)\right). \tag{S-13}$$

In order to compute the inverse posterior covariance that minimizes the right-hand side of the PAC-Bayes bound, we would like to solve the problem

$$\begin{aligned} \text{minimize} \quad & L(\Sigma_q) := \check{e}(Q, D_n) + \frac{\operatorname{KL}(Q,P)}{2(n-1)} \\ \text{such that} \quad & Q = N(w, \Sigma_q) \\ \text{and} \quad & \Sigma_q \succeq 0. \end{aligned}$$

Observe that

$$\breve{e}(h_{w'}, D_n) = \breve{e}(h_w, D_n) + \frac{1}{2}\left\langle w' - w, H_w(w' - w)\right\rangle.$$
$$P = N(w_0, \epsilon^{-1} I).$$

For $P_w = N(w, \epsilon^{-1} I)$, we have

$$\mathrm{KL}(Q, P) = \mathrm{KL}(Q, P_w) + \frac{\epsilon}{2}\|w - w_0\|^2$$

Hence,

$$\begin{aligned}
L(\Sigma_q) &= \int Q(w')\breve{e}(h_{w'}, D_n)\,\mathrm{d}w' + \frac{1}{2(n-1)}\int Q(w')\log\frac{Q(w')}{P_w(w')}\,\mathrm{d}w' + \frac{\epsilon}{4(n-1)}\|w - w_0\|^2 \\
&= \frac{1}{2(n-1)}\int\left(-\log\exp\big(-2(n-1)\breve{e}(w', D_n)\big) + \log\frac{Q(w')}{P_w(w')}\right)Q(w')\,\mathrm{d}w' + \frac{\epsilon}{4(n-1)}\|w - w_0\|^2 \\
&= \frac{1}{2(n-1)}\int\left(\log\frac{Q(w')}{\exp(-2(n-1)\breve{e}(w', D_n))P_w(w')/Z} - \log Z\right)Q(w')\,\mathrm{d}w' + \frac{\epsilon}{4(n-1)}\|w - w_0\|^2 \\
&= \frac{1}{2(n-1)}\left(\mathrm{KL}(Q, B) - \log Z\right) + \frac{\epsilon}{4(n-1)}\|w - w_0\|^2,
\end{aligned}$$

where we have defined

$$B(w') = \exp\big(-2(n-1)\breve{e}(w', D_n)\big)P_w(w')/Z, \quad \text{and}$$
$$Z = \int \exp\big(-2(n-1)\breve{e}(w', D_n)\big)P_w(w')\,\mathrm{d}w'.$$

We can now see that $L(\Sigma_q)$ attains a minimum when

$$Q = B \propto \exp\big(-2(n-1)\breve{e}(w', D_n)\big)P_w(w') \tag{S-14}$$

or $\Sigma_q^{-1} = 2(n-1)H_w + \epsilon\, I$, in other words,

$$\Sigma_q = U_w(\bar{\Lambda}_w)^{-1}U_w^\top,$$

where

$$\bar{\lambda}_i = 2(n-1)\lambda_i + \epsilon \quad \forall i \le p.$$

### B.3. Calculation for (9)

Recall that the effective dimensionality of a model at a local minimum $w$ is the number of eigenvalues of the Hessian with magnitude at least $\frac{\epsilon}{2(n-1)}$, i.e.,

$$p(n, \epsilon) = \sum_{i=1}^{p} \mathbf{1}\left\{|\lambda_i| \ge \frac{\epsilon}{2(n-1)}\right\},$$

The strength of the model at $w$ is

$$s(n, \epsilon) = \sum_{i=1}^{p(n,\epsilon)} 1 + \log\left(\frac{2(n-1)\lambda_i}{\epsilon} + 1\right).$$

We assume that $c(H_w, p(n, \epsilon)) > 0$. i.e., denote $c(H_w, p(n, \epsilon))$ as $c(n, \epsilon)$

$$\lambda_i \le \frac{\epsilon}{2(n-1)}\exp(-c(n, \epsilon)(i - p(n, \epsilon)))$$

We can also assume a weaker version of this decay pattern,

$$\sum_{i=p(n,\epsilon)+1}^{p} \lambda_i = \frac{\epsilon}{2(n-1)c(n, \epsilon)}.$$

We approximate the training objective in the neighborhood of $w$ as

$$\breve{e}(h_{w'}, D_n) = \breve{e}(h_w, D_n) + \frac{1}{2}\left\langle w' - w, H_w(w' - w)\right\rangle.$$

and we assume that the model at $w$ is a interpolation solution. In §3.3, for the posterior $Q = N(w, \Sigma_q)$ that maximizes the loose version of the PAC-Bayes bound (1), where

$$\Sigma_q = U_w \bar{\Lambda}_w^{-1} U_w{}^\top,$$
$$\bar{\lambda}_i = 2(n-1)\lambda_i + \epsilon.$$

We can now calculate

$$\begin{aligned}\check{e}(Q, D_n) - \check{e}(h_w, D_n) &= \frac{1}{2}\sum_{i=1}^{p} \frac{\lambda_i}{\bar{\lambda}_i} \\ &\leq \frac{p(n,\epsilon) + 1/c(n,\epsilon)}{4(n-1)}, \text{ and}\end{aligned}$$

$$\begin{aligned}\frac{\mathrm{KL}(Q,P)}{2(n-1)} &= \frac{1}{4(n-1)}\left(\epsilon\|w - w_0\|^2 - p + \sum_{i=1}^{p} \log\frac{\bar{\lambda}_i}{\epsilon} + \frac{\epsilon}{\bar{\lambda}_i}\right) \\ &\leq \frac{1}{4(n-1)}\left(\epsilon\|w - w_0\|^2 + \sum_{i=1}^{p(n,\epsilon)} \log\left(\frac{2(n-1)\lambda_i}{\epsilon} + 1\right) + \sum_{i=p(n,\epsilon)+1}^{p} \frac{2(n-1)\lambda_i}{\epsilon}\right) \\ &\leq \frac{1}{4(n-1)}\left(\epsilon\|w - w_0\|^2 + \sum_{i=1}^{p(n,\epsilon)} \log\left(\frac{2(n-1)\lambda_i}{\epsilon} + 1\right) + \frac{1}{c(n,\epsilon)}\right), \text{ hence}\end{aligned}$$

$$\check{e}(Q, D_n) + \frac{\mathrm{KL}(Q,P)}{2(n-1)} \leq \frac{s(n,\epsilon) + 2/c(n,\epsilon) + \epsilon\|w - w_0\|^2}{4(n-1)}.$$

For the KL-term, in the first inequality we have used the fact that $\log(1+x) \leq x$ to split the first summation into two parts; in the second inequality we have used the assumption that the eigenspectrum is sloppy to sum the series from $i = p(n,\epsilon) + 1$; the latter is also used in the inequality for the gap in the loss.

## C. Proofs of Lemmas in §3.1

We use $\mathbb{E}$ to denote the expectation over inputs $x$. The following lemmas holds for all distribution of $x$. In particular, we can choose the distribution of $x$ to be the point mass distribution on the dataset $D_n$, i.e. $x \sim \frac{1}{n}\sum_{i=1}^{n} \delta_{x_i}$, in this case, $\mathbb{E}\left[xx^\top\right] = \frac{1}{n} X X^\top \in \mathbb{R}^{d\times d}$ is the input corelation matrix.

The following lemma bounds the trace of the activation correlations and the norm of the gradient of each logit with respect to the activations.

**Lemma 10 (Bounding the trace of the correlations of activations and norm of activation gradients).** We have

$$\mathrm{tr}\left(\mathbb{E}\left[h^k h^{k\top}\right]\right) \leq a^2 \left\|w^{k-1}\right\|_2^2 \mathrm{tr}\left(\mathbb{E}\left[h^{k-1} h^{k-1\top}\right]\right), \tag{S-15}$$

and

$$\left\|\frac{\mathrm{d}z_i}{\mathrm{d}h^k}\right\|_2 \leq a \left\|\frac{\mathrm{d}z_i}{\mathrm{d}h^{k+1}}\right\|_2 \left\|w^k\right\|_2. \tag{S-16}$$

**Proof of Lemma 10**. For the first inequality in (S-15), observe that

$$\begin{aligned}
\text{tr}\left(\mathbb{E}\left[h^k {h^k}^\top\right]\right) &\leq \sum_{j=1}^{d_k} \mathbb{E}\left[\sigma(u_j^k)^2\right] \\
&\leq a^2 \sum_{j=1}^{d_k} \mathbb{E}\left[(u_j^k)^2\right] \\
&= a^2 \text{tr}\left(\mathbb{E}\left[u^k {u^k}^\top\right]\right) \\
&= a^2 \text{tr}\left(\mathbb{E}\left[\left(w^{k-1} h^{k-1}\right)\left(w^{k-1} {h^{k-1}}^\top\right)\right]\right) \\
&= a^2 \text{tr}\left(w^{k-1}\, \mathbb{E}\left[h^{k-1} {h^{k-1}}^\top\right] {w^{k-1}}^\top\right) \\
&\leq a^2 \left\|w^{k-1}\right\|_2^2 \text{tr}\left(\mathbb{E}\left[h^{k-1} {h^{k-1}}^\top\right]\right).
\end{aligned}$$

For the second inequality in (S-16), observe that

$$\begin{aligned}
\frac{\mathrm{d}z_i}{\mathrm{d}h^k} &= \frac{\mathrm{d}z_i}{\mathrm{d}u^{k+1}} w^k \\
&= a\left(\frac{\mathrm{d}z_i}{\mathrm{d}h^{k+1}} \mathbb{1}_{u^{k+1}\geq 0}\right) w^k \\
\Rightarrow \left\|\frac{\mathrm{d}z_i}{\mathrm{d}h^k}\right\|_2 &\leq a \left\|\frac{\mathrm{d}z_i}{\mathrm{d}h^{k+1}}\right\|_2 \left\|w^k\right\|_2.
\end{aligned}$$

where $\mathbb{1}_{\text{cond}}$ is a vector of 1s at elements where the condition is true. □

The above inequalities can be used in Lemma 11 to bound the trace of the gradient correlation of any logit $z_i$ with respect to weights of a layer $w^k$.

**Lemma 11 (Bounding the trace of the correlation sum-of-logit Jacobian).** For logit $z_i$, $i = 1, ..., m$

$$\text{tr}\left(\mathbb{E}\left[\frac{\mathrm{d}z_i}{\mathrm{d}w^k} \frac{\mathrm{d}z_i}{\mathrm{d}w^k}^\top\right]\right) \leq a^{2L} \text{tr}\left(\mathbb{E}\left[xx^\top\right]\right) \prod_{j=0, j\neq k}^{L} \left\|w^j\right\|_2^2. \tag{S-17}$$

for $k = 0, ..., L$. As a result,

$$\text{tr}\left(\mathbb{E}\left[\frac{\mathrm{d}z_i}{\mathrm{d}w} \frac{\mathrm{d}z_i}{\mathrm{d}w}^\top\right]\right) \leq a^{2L} \text{tr}\left(\mathbb{E}\left[xx^\top\right]\right) \prod_{j=0}^{L} \left\|w^j\right\|_2^2 \left(\sum_{j=0}^{L} \frac{1}{\|w^j\|_2^2}\right).$$

**Proof of Lemma 11**. The proof follows via an application of Lemma 10. For $k = 0, 1, ..., L-1$,

$$\begin{aligned}
\text{tr}\left(\mathbb{E}\left[\frac{\mathrm{d}z_i}{\mathrm{d}w^k}\frac{\mathrm{d}z_i}{\mathrm{d}w^k}^\top\right]\right) &= \text{tr}\left(\mathbb{E}\left[\frac{\mathrm{d}z_i}{\mathrm{d}u^{k+1}}\frac{\mathrm{d}z_i}{\mathrm{d}u^{k+1}}^\top \otimes h^k h^{k\top}\right]\right) \\
&= \mathbb{E}\left[\text{tr}\left(\frac{\mathrm{d}z_i}{\mathrm{d}u^{k+1}}\frac{\mathrm{d}z_i}{\mathrm{d}u^{k+1}}^\top\right)\text{tr}\left(h^k h^{k\top}\right)\right] \\
&\leq a^2\left\|\frac{\mathrm{d}z_i}{\mathrm{d}h^{k+1}}\right\|_2^2 \text{tr}\left(\mathbb{E}\left[h^k h^{k\top}\right]\right) \\
&\leq a^2\left\|\frac{\mathrm{d}z_i}{\mathrm{d}h^{L}}\right\|_2^2 \left(\prod_{j=k+1}^{L-1}\left\|w^j\right\|_2^2\right) a^{2(L-k-1)} \\
&\quad a^{2k}\prod_{j=0}^{k-1}\left\|w^j\right\|_2^2 \text{tr}\left(\mathbb{E}\left[xx^\top\right]\right) \\
&\leq a^{2L}\text{tr}\left(\mathbb{E}\left[xx^\top\right]\right)\prod_{j=0, j\neq k}^{L}\left\|w^j\right\|_2^2.
\end{aligned}$$

The third line comes from the fact that the matrix $\frac{\mathrm{d}z_i}{\mathrm{d}u^{k+1}}\frac{\mathrm{d}z_i}{\mathrm{d}u^{k+1}}^\top$ is rank one and its trace is the same as 2-norm. The last inequality comes from the fact that $\left\|w_i^L\right\|_2 \leq \left\|w^L\right\|_2$. For $k = L$,

$$\begin{aligned}
\text{tr}\left(\mathbb{E}\left[\frac{\mathrm{d}z_i}{\mathrm{d}w^L}\frac{\mathrm{d}z_i}{\mathrm{d}w^L}^\top\right]\right) &= \text{tr}\left(\mathbb{E}\left[\frac{\mathrm{d}z_i}{\mathrm{d}w_i^L}\frac{\mathrm{d}z_i}{\mathrm{d}w_i^L}^\top\right]\right) \\
&= \text{tr}\left(\mathbb{E}\left[h^L h^{L\top}\right]\right) \\
&\leq a^{2L}\text{tr}\left(\mathbb{E}\left[xx^\top\right]\right)\prod_{j=0}^{L-1}\left\|w^j\right\|_2^2.
\end{aligned}$$

□

**Proof of Theorem 1**. We first calculate an inequality for the Fisher Information Matrix (FIM)

$$\begin{aligned}
F &= \mathbb{E}\left[\sum_{y=1}^{m} p_w(y\,|\,x)(\partial_w \log p_w(y\,|\,x))(\partial_w \log p_w(y\,|\,x))^\top\right] \\
&= \mathbb{E}\left[\partial_w z\left[\sum_{y=1}^{m} p_w(y\,|\,x)\frac{\mathrm{d}\log p_w(y\,|\,x)}{\mathrm{d}z}\frac{\mathrm{d}\log p_w(y\,|\,x)}{\mathrm{d}z}^\top\right]\partial_w z^\top\right]
\end{aligned}$$

For an output distribution $p_w(y\,|\,x)$ obtained using the softmax operator on the logits $z_y$

$$p_y \equiv p_w(y\,|\,x) = \frac{e^{z_y}}{\sum_{y'} e^{z_{y'}}}$$

we have

$$\frac{\mathrm{d}}{\mathrm{d}z}\log p_w(y\,|\,x) = e_y - p$$

where $e_y$ is the one-hot vector of the class $y$ and $p = [p_1, ..., p_m]$.

$$\begin{aligned}
\sum_{y=1}^{m} p_w(y\,|\,x)\frac{\mathrm{d}\log p_w(y\,|\,x)}{\mathrm{d}z}\frac{\mathrm{d}\log p_w(y\,|\,x)}{\mathrm{d}z}^\top &\preceq \sum_{y=1}^{m} p_w(y\,|\,x)\left\|\frac{\mathrm{d}\log p_w(y\,|\,x)}{\mathrm{d}z}\right\|_2^2 I \\
&= (1 - \|p\|_2^2)I \\
&\preceq I
\end{aligned}$$

Hence we have

$$F \preceq \mathbb{E}\left[(\partial_w z)\,(\partial_w z)^\top\right].$$

In the case of the Hessian for the cross-entropy loss we make a similar calculation following the calculation of Fort & Ganguli (2019). For the calculation of Hessian, the expectation $\mathbb{E}$ denotes the expectation with respect to inputs and labels in the training set. We write

$$\begin{aligned}(\log 2)H &\approx \mathbb{E}\left[(\partial_w z)\,\nabla_z^2(-\log p_w(y\,|\,x))\,(\partial_w z)^\top\right]\\ &= \mathbb{E}\left[(\partial_w z)\left(\mathrm{diag}(p) - pp^\top\right)(\partial_w z)^\top\right]\\ &\preceq \mathbb{E}\left[(\partial_w z)\,(\mathrm{diag}(p))\,(\partial_w z)^\top\right]\\ &\preceq \mathbb{E}\left[(\partial_w z)\,(\partial_w z)^\top\right].\end{aligned}$$

In the above calculation, we have kept only the so-called G-term of the Hessian and neglected an additional H-term.

$$\mathbb{E}\left[\sum_{i=1}^{m}(y_i - p_i)\,\frac{\partial^2 z_i}{\partial w_\alpha \partial w_\beta}\right]$$

which is typically small in practice for a well-trained network because the terms $1-p_i$ are close to zero for all logits (Papyan, 2019; Sagun et al., 2016)($\mathbb{E}[\sum_{i=1}^{m}|y_i - p_i|$ is $5.32\times 10^{-8}$ for FC-600-2 on MNIST). Hence, both $\mathrm{tr}\,(F)$ and $(\log 2)\mathrm{tr}\,(H)$ can be bounded by

$$\mathrm{tr}\,(F)\,,(\log 2)\mathrm{tr}\,(H) \leq \sum_{i=1}^{m}\mathbb{E}\left[\frac{\mathrm{d}z_i}{\mathrm{d}w}\frac{\mathrm{d}z_i}{\mathrm{d}w}^\top\right] \leq ma^{2L}\mathrm{tr}\left(\mathbb{E}\left[xx^\top\right]\right)\prod_{j=0}^{L}\left\|w^j\right\|_2^2\left(\sum_{j=0}^{L}\frac{1}{\|w^j\|_2^2}\right). \tag{S-18}$$

Notice that the $\log 2$ factor in front of $\mathrm{tr}(H)$ comes from the rescaling factor in the definition of $\check{e}(h_w, D_n)$. □

**Remark 12.** The G-term is always positive semi-definite since the output distribution $p \in \mathbb{R}^C$ is always convex on the logits $z \in \mathbb{R}^C$, i.e., $\left(-\log\left(\frac{e^{zy}}{\sum_{y'}e^{zy'}}\right)\right)_{y=1}^{C}$ is convex in $z$.

**Remark 13.** Empirically, the trace of FIM and Hessian at the end of training (Fig. 3) is usually much smaller than the trace of correlation matrix of logit Jacobians (Fig. S-8). In this case, the prediction of the bound in (S-18) seems very loose. However from the above calculation, we also know that

$$\begin{aligned}\mathrm{tr}(F) &\leq (1-\|p\|_2^2)\,\mathrm{tr}\left(\sum_{i=1}^{m}\mathbb{E}\left[\frac{\mathrm{d}z_i}{\mathrm{d}w}\frac{\mathrm{d}z_i}{\mathrm{d}w}^\top\right]\right),\\ \mathrm{tr}(H) &\leq \mathrm{tr}\left(\mathbb{E}\left[(\partial_w z)\left(\mathrm{diag}(p) - pp^\top\right)(\partial_w z)^\top\right]\right).\end{aligned}$$

For trained network that predicts accurately, we usually get the probabilities $p$ that are very close to one-hot vectors of the correct classes. In this case, both $1-\|p\|_2^2$ and $\mathrm{diag}(p) - pp^\top$ are close to zero. This explains why in our experiments the trace of $F$ and $H$ at the end of training are much smaller than that of logit Jacobians.

**Proof of Lemma 2**. The proof depends upon Weyl's inequality to control the eigenvalues of the sum of Hermitian matrices. It states that for Hermitian matrices $A, B, C \in \mathbb{R}^{p\times p}$, if $C = A + B$, then

$$\lambda_{i+j-1}(C) \leq \lambda_i(A) + \lambda_j(B), \quad \lambda_{p-i-j}(C) \geq \lambda_{p-i}(A) + \lambda_{p-j}(B) \tag{S-19}$$

for all $1 \leq i, j \leq p$. In particular if $B \succeq 0$, then $\lambda_i(C) \geq \lambda_i(A)$ for all $i \leq p$.

We can now write,

$$\begin{aligned}\mathbb{E}\left[\frac{\mathrm{d}z_i}{\mathrm{d}w^k}\frac{\mathrm{d}z_i}{\mathrm{d}w^k}^\top\right] &= \mathbb{E}\left[\left(\frac{\mathrm{d}z_i}{\mathrm{d}h^{k+1}}\odot\frac{\mathrm{d}h^{k+1}}{\mathrm{d}u^{k+1}}\right)\left(\frac{\mathrm{d}z_i}{\mathrm{d}h^{k+1}}\odot\frac{\mathrm{d}h^{k+1}}{\mathrm{d}u^{k+1}}\right)^\top\otimes h^k h^{k\top}\right]\\ &\preceq \mathbb{E}\left[a^2\left\|\frac{\mathrm{d}z_i}{\mathrm{d}h^{k+1}}\right\|^2 I_{d_{k+1}}\otimes h^k h^{k\top}\right]\\ &= a^2\left\|\frac{\mathrm{d}z_i}{\mathrm{d}h^{k+1}}\right\|^2 I_{d_{k+1}}\otimes\mathbb{E}\left[h^k h^{k\top}\right]\\ &= a^{2(L-k)}\left(\prod_{j=k+1}^{L}||w_j||^2\right)I_{d_{k+1}}\otimes\mathbb{E}\left[h^k h^{k\top}\right]\end{aligned}$$

Hence, by (S-19)

$$\text{spec}\left(\mathbb{E}\left[\frac{\mathrm{d}z_i}{\mathrm{d}w_k}\frac{\mathrm{d}z_i}{\mathrm{d}w_k}^\top\right]\right)\preceq\text{spec}\left(a^{2(L-k)}\prod_{j=k+1}^{L}||w_j||^2 I_{d_{k+1}}\otimes\mathbb{E}\left[h^k h^{k\top}\right]\right)$$

so we have

$$\text{spec}\left(\mathbb{E}\left[\frac{\mathrm{d}z_i}{\mathrm{d}w_k}\frac{\mathrm{d}z_i}{\mathrm{d}w_k}^\top\right]\right)\preceq a^{2(L-k)}\prod_{j=k+1}^{L}||w_j||^2\text{spec}\left(I_{d_{k+1}}\right)\otimes\text{spec}(\mathbb{E}\left[h^k h^{k\top}\right])$$

□

**Remark 14 (Modification using sloppiness of activation gradients).** Fig. 1 shows that the slope of decay of FIM and the activations are essentially the same. However, in (3) if spec $\left(\mathbb{E}\left[h^k h^{k\top}\right]\right)$ decays as $\mathcal{O}(\exp(-ci))$, the decay of spec $\left(\mathbb{E}\left[\frac{\mathrm{d}z_i}{\mathrm{d}w^k}\frac{\mathrm{d}z_i}{\mathrm{d}w^k}^\top\right]\right)$ is $\mathcal{O}(\exp(-ci/d_{k+1}))$. This is a loose bound, especially when $d_{k+1}$ is large, e.g., the spectrum could decay much more faster. But note that if we can write a KFAC-approximation

$$\mathbb{E}\left[\frac{\mathrm{d}z_i}{\mathrm{d}w^k}\frac{\mathrm{d}z_i}{\mathrm{d}w^k}^\top\right]\approx\mathbb{E}\left[\frac{\mathrm{d}z_i}{\mathrm{d}u^{k+1}}\frac{\mathrm{d}z_i}{\mathrm{d}u^{k+1}}^\top\right]\otimes\mathbb{E}\left[h^k h^{k\top}\right].$$

then we obtain a stronger decay for the logit gradient when $d_{k+1}$ is large, if we assume that the activations *gradients* are sloppy. If spec $\left(\mathbb{E}\left[\frac{\mathrm{d}z_i}{\mathrm{d}u^{k+1}}\frac{\mathrm{d}z_i}{\mathrm{d}u^{k+1}}^\top\right]\right)$ decays as $\exp\{-c_1 i\}$ and spec $\left(\mathbb{E}\left[h^k h^{k\top}\right]\right)$ decays as $\exp\{-c_2 j\}$, then the $(i+j)^2$th largest eigenvalue of $\mathbb{E}\left[\frac{\mathrm{d}z_i}{\mathrm{d}w^k}\frac{\mathrm{d}z_i}{\mathrm{d}w^k}^\top\right]$ is smaller than $\exp(-\min\{c_1,c_2\}(i+j))$, hence the $k$th largest eigenvalue of $\mathbb{E}\left[\frac{\mathrm{d}z_i}{\mathrm{d}w^k}\frac{\mathrm{d}z_i}{\mathrm{d}w^k}^\top\right]$ is smaller than $\exp\left(-\min\{c_1,c_2\}\sqrt{k}\right)$. Hence, the decay rate of spec $\left(\mathbb{E}\left[\frac{\mathrm{d}z_i}{\mathrm{d}w^k}\frac{\mathrm{d}z_i}{\mathrm{d}w^k}^\top\right]\right)$ is $\mathcal{O}\left(\exp\left(-\min\{c_1,c_2\}\sqrt{k}\right)\right)$.

**Corollary 15.** Denote the FIM and Hessian with respect to the $k$th layer $F(w_k), H(w_k)$ respectively, then we have,

$$\text{spec}\left(F(w_k)\right),\text{spec}\left(H(w_k)\right)\preceq 2ma^{2(L-k)}\prod_{j=k+1}^{L}\left\|w^j\right\|_2^2\text{spec}(I_{d_{k+1}})\otimes\text{spec}\left(\mathbb{E}\left[h^k h^{k\top}\right]\right).$$

As in Lemma 2, $\prod_{j=L=1}^{L}\left\|w^j\right\|_2^2=1$.

**Proof**. From Lemma 11 we know that

$$F(w_k),H(w_k)\preceq 2\,\mathbb{E}\left[(\partial_{w_k}z)(\partial_{w_k}z)^\top\right]$$

Let $s=\sum_{i=1}^{m}z_i$ be the sum of logits, then we have

$$\begin{aligned}F(w_k),H(w_k)&\preceq 2\mathbb{E}\left[\left(\frac{\mathrm{d}s}{\mathrm{d}w_k}\right)\left(\frac{\mathrm{d}s}{\mathrm{d}w_k}\right)^\top\right]\\ &\preceq 2ma^{2(L-k)}\prod_{j=k+1}^{L}\left\|w^j\right\|_2^2\text{spec}(I_{d_{k+1}})\otimes\text{spec}\left(\mathbb{E}\left[h^k h^{k\top}\right]\right)\end{aligned}$$

The second inequality comes from a similar calculation as in Lemma 2 for network with one added layer where $h_{L+1}=u_{L+1}=z$, $u_{L+2}=w_{L+1}h_{L+1}$, and $w_{L+1}=[1,...,1]$, $\|w_{L+1}\|_2^2=m$. □

## D. Technical details of different methods for optimizing the PAC-Bayes bound

We optimize the problem,

$$\min \breve{e}(Q, D_n) + \sqrt{\frac{\mathrm{KL}(Q, P) + \varphi}{2(n-1)}} \tag{S-20}$$

where $Q$, $P$ are multivariate normal distribution, $\varphi$ is the penalty we added for including a trainable parameter in prior (say its scale), and $n$ is the number of samples. For Gaussian distributions on the weight space $Q, P$, as we saw in (S-13), the KL-divergence is

$$\frac{1}{2}\left(\mathrm{tr}(\Sigma_p^{-1}\Sigma_q) - p + (w - w_0)^\top \Sigma_p^{-1}(w - w_0) + \log(\det \Sigma_p / \det \Sigma_q)\right).$$

The penalty for the case when $P = N(0, \epsilon^{-1} I)$ comes from the union bound over the set $\epsilon = c\exp(j/b)$ for $j \in \mathbb{N}$ and is given by

$$\varphi = 2\log\left(b\log(c/\epsilon)\right) + \log\left(\pi^2 n/(6\delta)\right)$$

Note that for Method 4, we need more than one trainable parameters for the prior, and the penalty $\varphi$ should also be modified according to Appendix D.5. We calculate $\breve{e}(Q, D_n)$ using Monte Carlo samples from $Q$. After the optimization process, we calculate the PAC-Bayes bound on $e(Q)$ using

$$\mathrm{kl}(\hat{e}(Q, D_n), e(Q)) \leq \frac{\mathrm{KL}(Q, P) + \varphi}{n-1}, \tag{S-21}$$

which involves finding an approximation of $\mathrm{kl}^{-1}(b, a) := \sup\{a' \in [0,1] : \mathrm{kl}(b, a') \leq a\}$ (see (Dziugaite & Roy, 2017) for details). We next discuss the various methods for calculating PAC-Bayes bounds developed in the paper and provide their implementation details.

### D.1. Method 1

The tightest bound in this case is obtained using the v2 model described in Fig. 4 and Appendix A. To recall, this involves a second post-training phase where the trained model is updated to be closer to the initialization $w_0$. In the context of the PAC-Bayes upper bound, this reduces the distance between the means of the Gaussian prior and posterior. We choose $\Sigma_q$ as in (4) and (5). For $\epsilon = c\exp(j/b)$ and $j = 1, \ldots, 60$, we evaluate $\mathrm{KL}(Q, P)$ by using (S-13), and $\hat{e}(Q, D_n)$ is estimated by sampling. The covariance $\Sigma_q$ is approximated by the top eigenvalues and eigenvectors of the Hessian as discussed in Appendix D.4.2. The PAC-Bayes bound is calculated by (S-20) and we choose the smallest bound among all choices of $\epsilon$.

We also set $\Sigma_q = U_w \bar{\Lambda}_w U_w^\top$ and calculate $\bar{\Lambda}$ by directly minimizing (S-20) where the variables of optimization are $\bar{\lambda}_i$ for $i \leq k$ using nonlinear optimization in scipy (using the BFGS algorithm), and the PAC-Bayes bound is calculated in the same way as above. This is denoted as Method 5 (Numerical) in Table S-3.

For comparison, we also choose $\Sigma_q = \epsilon^{-1} I$ and calculate the PAC-Bayes bound. This is denoted as Method 6 (Isotropic) in Table S-3.

### D.2. Methods 2 and 3

We choose $P$ and $Q$ as described in §5. We set $\epsilon^{-1} = \exp(2\rho)$ , $\bar{\Lambda}_w = \exp(2\xi)$. The parameters $\rho$, $w$, $\xi$ are optimized while optimizing the PAC-Bayes upper bound. We initialize $\epsilon^{-1}$ at $\exp(-6)$ and $\bar{\Lambda}_w$ at $(\Lambda^F + \epsilon^{-1})/10$ where $\Lambda^F$ are the eigenvalues of the FIM at initialization. For fully-connected networks and LeNet, we evaluate $\hat{e}(Q, D_n)$ using the methods described in Appendix D.4.1 and Appendix D.4.2 respectively.

We use the Gauss-Newton matrix as an approximation of the FIM for Method 2.

### D.3. Method 4

We choose $P$ and $Q$ as described in §5. We set $a = \exp(2\rho_1)$, $\epsilon^{-1} = \exp(2\rho_2)$, $\sigma = \exp(2\xi)$ and train parameters $\rho_1$, $\rho_2$, $w$, $\xi$. In our experiments, $\epsilon^{-1}$ is initialized to $\exp(-6)$, $a$ is initialized to $\exp(-1)$ and $\Sigma_q$ is initialized to be $(aF_{w_0} + \epsilon)/10$. In this case,

$$\mathrm{KL}(Q, P) = \frac{1}{2}\left(\sum_i \frac{\sigma_i}{a\lambda_i^F + \epsilon^{-1}} - d + (w - w_0)^\top (aF_{w_0} + \epsilon^{-1})^{-1}(w - w_0) + \sum_i \log \frac{a\lambda_i^F + \epsilon^{-1}}{\sigma_i}\right)$$

where $\lambda_i^F$ are eigenvalues of $F_{w_0}$. For fully-connected networks and LeNet, we approximate $(w-w_0)^\top(aF_{w_0}+\epsilon^{-1})^{-1}(w-w_0)$ using the methods described in Appendix D.4.1 and Appendix D.4.2 respectively.

We use the Gauss-Newton matrix as an approximation of the FIM for Method 4.

### D.4. Computing the PAC-Bayes term that corresponds to the distance from initialization

In Method 4, we need to calculate

$$E = (w-w_0)^\top(aF_{w_0}+\epsilon^{-1})^{-1}(w-w_0).$$

In Methods 2 and 3, we need to sample from a posterior of the form $N(0, U\Lambda U^\top)$ for various different values of $U$ and $\Lambda$. Doing either of these is not easy for high-dimensional weight spaces. We employ two different methods to deal with this problem. For fully-connected networks we use a KFAC approximation of the Hessian/FIM while for LeNet which has much fewer weights, we approximate these matrices using their top few eigenvalues and eigenvectors.

#### D.4.1. KFAC APPROXIMATION OF THE FIM AND HESSIAN

We approximate the Hessian/FIM by a variation of Kronecker decomposition of block-diagonal Hessian/FIM (KFRA, (Botev et al., 2017)). We use the BACKPACK library for implementing this (Dangel et al., 2020). For the weight of the $k^{\text{th}}$ layer $w_k \in \mathbb{R}^{d_{k+1}\times d_k}$, the KFRA approximation of the corresponding block in the Hessian/FIM which is denoted by $F^k$ or $H^k$ can be written as $A_k \otimes B_k$. Denote by $U_{A_k}, U_{B_k}$ the eigenspaces of $A_k$ and $B_k$. To estimate $E$, we can first decompose $E$ as the summation where each term is for a particular layer $k$

$$E = \sum_{k=0}^{L} E^k$$

where

$$\begin{aligned}
E^k &= (w^k - w_0^k)^\top (a\,(F_{w_0})^k + \epsilon^{-1})^{-1}(w^k - w_0^k) \\
&= (w^k - w_0^k)^\top U^k (a\Lambda^k + \epsilon^{-1})^{-1} {U^k}^\top (w^k - w_0^k) \\
&= \left((w^k - w_0^k)^\top U^k \left(a\Lambda^k + \epsilon^{-1}\right)^{-1/2}\right)\left((w^k - w_0^k)^\top\, U^k \left(a\Lambda^k + \epsilon^{-1}\right)^{-1/2}\right)^\top
\end{aligned}$$

$(E^k)^{1/2}$ can be calculated by

$$\begin{aligned}
{E^k}^{1/2} &= (w^k - w_0^k)^\top U^k \left(a\Lambda^k + \epsilon^{-1}\right)^{-1/2} \\
&= (U_{A_k}^\top (w^k - w_0^k) U_{B_k}) \odot \left(a\Lambda^k + \epsilon^{-1}\right)^{-1/2}
\end{aligned}$$

where in the last line, $(w^k - w_0^k) \in \mathbb{R}^{d_{k+1}\times d_k}$. We use $\odot$ to denote element wise multiplication. $U_{A_k}^T (w^k - w_0^k) U_{B_k}$ can now be easily calculated using the KFAC factors.

To sample from the posterior $N(w, U\Lambda U^\top)$, we can concatenate the samples of the weights of each layer. We first sample $r^k \sim N(0, I_{d_k})$, then calculate $\sqrt{\Lambda^k} \odot r_k$ and thereby

$$\nu_k := U_k \left(\sqrt{\Lambda^k} \odot r_k\right) = U_{A_k} \left(\sqrt{\Lambda^k} \odot r_k\right) U_{B_k}^\top.$$

The final sample is therefore $w + [\nu_1, ..., \nu_k]$ which is distributed as $N(w, U\Lambda U^\top)$.

#### D.4.2. APPROXIMATE FIM AND HESSIAN USING ITS TOP EIGENVALUES AND EIGENVECTORS

For symmetric $\Sigma$ with orthogonal decomposition $\Sigma = U\Lambda U^\top$, $U = [U_1, U_2]$, $\Lambda = \text{diag}(\Lambda_1, \Lambda_2)$, we have

$$\begin{aligned}
\Sigma &= U_1\Lambda_1 U_1^\top + U_2\Lambda_2 U_2^\top \\
\text{where } I &= U_1 U_1^\top + U_2 U_2^\top.
\end{aligned}$$

In this case, to calculate $E$, we approximate $aF_{w_0} + \epsilon^{-1}$ by

$$aF_{w_0} + \epsilon^{-1} = U_1(a\Lambda_1 + \epsilon_1^{-1})U_1^\top + \epsilon_2^{-1} U_2 U_2^\top$$

where $\Lambda_1, U_1$ are the stiff (largest $k$) eigenvalues and corresponding eigenvectors for $F_{w_0}$ and $U_2, \Lambda_2$ are the sloppy ones. Notice that we use two scalar parameters $\epsilon_1$ and $\epsilon_2$ to set the additive constant in the prior covariance.

$$\begin{aligned} E &= (w-w_0)^\top U_1(a\Lambda_1 + \epsilon_1^{-1})^{-1}U_1^\top(w-w_0) + \epsilon_2(w-w_0)^\top U_2U_2^\top(w-w_0) \\ &= (w-w_0)^\top U_1(a\Lambda_1 + \epsilon_1^{-1})^{-1}U_1^\top(w-w_0) + \epsilon_2\left(\|w-w_0\|_2^2 - (w-w_0)^\top U_1U_1^\top(w-w_0)\right) \end{aligned}$$

Notice that the term $(w-w_0)^\top U_1$ is not hard to calculate because $U_1 \in \mathbb{R}^{p\times k}$ and since we are choosing the top few eigenvalues of the Hessian/FIM, the value of $k$ is small (about 300).

To sample from the posterior $N(w, U\Lambda U^\top)$, we first set $\Lambda = \text{diag}(\Lambda_1, \Lambda_2)$ where $\Lambda_1$ are the top $k$ stiff eigenvectors and $\Lambda_2$ are the $p-k$ other eigenvectors. Correspondingly, we have $U = [U_1, U_2]$. We use an isotropic variance for the sloppy subspace and set $\Lambda_2 = \epsilon^{-1}I_{p-k}$. We first sample $r \sim N(0, I_k)$, then calculate

$$\begin{aligned} \nu_1 &= U_1\sqrt{\Lambda_1}U_1^\top r \\ \nu_2 &= \epsilon^{-1/2}U_2U_2^\top r = \epsilon^{-1/2}(r - U_1U_1^\top r) \end{aligned}$$

Notice that $U_1U_1^\top r$, and $U_1\sqrt{\Lambda_1}U_1^\top r$ are easy to calculate. The result $w + [\nu_1, \nu_2]$ is distributed as $N(w, U\Lambda U^\top)$.

For cases when we recompute the FIM/Hessian while optimizing the PAC-Bayes bound (Method 2 and 3 respectively), we recompute the eigenvalues $\Lambda_1$ and the corresponding eigenvectors $U_1$. Note that the parameter $\epsilon$ in the covariance of the posterior is also optimized when we optimize the PAC-Bayes bound.

### D.5. Optimizing parameters of the prior in the PAC-Bayes bound

The prior should be fixed before looking at the training set, but for all methods above, we optimize the scale of the prior. We do this by adding an additional penalty in the KL term. Assume that $a^i$ for $i = 1, \ldots, m'$ are the number of parameters in the prior that we can select, we choose $a^i = (1/c^i)\exp\left(-j^i/b^i\right)$ for $j^i \in \mathbb{N}$. We reindex $j^i$ as a single index $k = (\sum_i j^i)^{m'}$, then if the PAC-Bayes bound for each index $k$ is designed to hold with probability at least $1 - \frac{6\delta}{\pi^2k^2}$, then by union bound, it will hold uniformly for all $k \in \mathbb{N}$ with probability at least $1-\delta$. For a bound that holds with probability $1-\delta'$, the penalty we should add is $\log\frac{n}{\delta'}$, hence, using the relation

$$a^i = (1/c^i)\exp\left(j^i/b^i\right), \quad \delta' = \frac{6\delta}{\pi^2k^2}, \quad k = (\sum_i j^i)^{m'}$$

we add the penalty

$$\varphi(a^1, ..., a^{m'}) = 2m'\log\left(\sum_i b_i \log\left(c^i a^i\right)\right) + \log\frac{\pi^2 n}{6\delta}$$

Similarly, for any positive or negative integer $j^i$, we can set $k = (\sum_i 2|j|^i)^{m'}$ to get the penalty

$$\varphi(a^1, ..., a^{m'}) = 2m'\log\left(2\sum_i \left|b_i \log\left(c^i a^i\right)\right|\right) + \log\frac{\pi^2 n}{6\delta}$$

In Methods 1, 2, 3 we choose $a^1 = \epsilon^{-1}$, in Method 4, we choose $a^1 = a$ and $a^2 = \epsilon^{-1}$.

## E. Working efficiently with Bayesian deep networks

Typically, a Bayesian neural network is implemented by programming Bayesian variants of standard layers in deep learning. For instance, one defines a BayesianLinear layer which maintains two sets of parameters, the mean weight vector and a standard deviation for each weight. At each forward pass, the layer samples a weight vector using the reparameterization trick to compute the activations. This is a reasonably efficient way to implement a Bayesian neural network but it is cumbersome because code for complex deep network architectures has to be rewritten from scratch to accommodate these Bayesian layers. We noticed that we can use the following trick (this is likely specific to PyTorch) to create a wrapper around any existing deep network code and construct its Bayesian variant. All our experiments use 150 samples from $Q$ before each update; in comparison typical implementations use 1 sample (Dziugaite & Roy, 2017; Wu et al., 2021). This strategy is potentially useful for other problems as well, e.g., for estimating the prediction uncertainty.

The code shown in Appendix E is adapted from https://github.com/pytorch/pytorch/blob/master/benchmarks/functional_autograd_benchmark/utils.py and works by first calculating the reparameterization trick (Line 45) using the mean and (logarithm of the) standard deviation of the weights (self.mu_std) and then swapping the weight of the actual model (self.w) that performs the forward propagation using the sampled weights.

```
1  def del_attr(obj, names):
2      #names: one name in the list names_all, a.b.c, split by "."
3      # list of format names = [a,b,c]
4      # delete the attribute obj.a.b.c
5      if len(names) == 1: delattr(obj, names[0])
6      else: del_attr(getattr(obj, names[0]), names[1:])
7
8  def set_attr(obj, names, val):
9      #names: one name in the list names_all, a.b.c, split by ".",
10     # list of format names = [a,b,c],
11     # set the attribute obj.a.b.c to val if obj.a.b.c is nn.Parameter
12     if len(names) == 1: setattr(obj, names[0], val)
13     else: set_attr(getattr(obj, names[0]), names[1:], val)
14
15 def get_names_params(mod):
16     # names_all: a list of all names of mod.parameters of type
17     # [a1.b1.c1, a2.b2.c2, ...]
18     # orig_params: tuple of parameters of type nn.Parameter
19     orig_params = tuple(mod.parameters())
20     names_all = []
21     for name, p in list(mod.named_parameters()):
22         names_all.append(name)
23     return orig_params, names_all
24
25 def load_weights(mod, names_all, params):
26     for name, p in zip(names_all, params):
27         set_attr(mod, name.split("."), p)
28
29 class bayesian_nn(nn.Module):
30     def __init__(self, c, args, ns=1):
31         # c: class of the model and args
32         # ns: number of Monte Carlo samples
33         super().__init__()
34         self.w = c(*args)
35         self.mu_std = nn.ModuleList([c(*args), c(*args)])
36         self.ns, self.args = ns, args
37         orig_params_w, self.names = get_names_params(self.w)
38
39     def forward(self, x):
40         ys = []
41         for _ in range(self.ns):
42             for name, m, v in zip(self.names,
43                                   list(self.mu_std[0].parameters()),
44                                   list(self.mu_std[1].parameters())):
45                 r = torch.randn_like(m).mul(torch.sqrt(torch.exp(2*v))).add(m)
46                 del_attr(self.w, name.split("."))
47                 set_attr(self.w, name.split("."), r)
48             ys.append(self.w(x))
49         return torch.stack(ys)
```

*Figure S-6.* Code for Bayesian neural networks

## F. Full results of PAC-Bayes generalization bounds and effective dimensionalities

We display the extended version of the results of PAC-Bayes bound optimization in Table S-3. Methods 1 and 5 give bounds that are similar to each other: this shows that our analytical expression (4) for the optimal posterior using a loose PAC-Bayes bound under the assumption that the loss is quadratic at the weights at the end of training is an accurate estimate of the optimal posterior (1). The bound calculated by these two methods is smaller than that of Method 6, which shows that the sloppiness of the Hessian at the end of training ($H_w$) is effective at providing non-vacuous generalization bounds. Using an isotropic posterior in Method 6 produces a remarkably good bound because almost all eigenvalues of $H_w$ for MNIST are small; even the largest eigenvalue is quite small in its magnitude as shown in Fig. 3). Methods 1, 5, 6 (which are the three methods that compute a bound without any optimization using the training dataset) give worse bounds than Methods 2, 3, 4 and also the method of Dziugaite & Roy (2017). This is because the approximation

$$\check{e}(h_{w'}, D_n) = \check{e}(h_w, D_n) + \frac{1}{2}\left\langle w' - w, H_w(w' - w)\right\rangle .$$

as we discussed in Method 1 may not be an accurate estimate of $\check{e}(w', D_n)$ in the neighborhood of $w$. As we see in Appendix B, the posterior that optimizes the loose PAC-Bayes bound *without* the approximation of the quadratic loss instead looks like (S-14). Methods 2–4 which involve optimization of the PAC-Bayes bound capture the optimal posterior better than the one corresponding to the quadratic assumption leads to a tighter PAC-Bayes bound. Method 4 gives the tightest bound since the training predominantly takes place in the stiff subspace of FIM at initialization, and the prior with covariance proportional to FIM puts less penalty than the isotropic prior on the stiff directions. Using posterior with $\mathrm{E}(\Sigma_q) = H_w$ (Method 3, which is similar to Wu et al. (2021)) works better than a diagonal posterior $\mathrm{E}(\Sigma_q) = \Lambda$ (which is the method in Dziugaite & Roy (2017)); this coincides with our calculation in Method 1 (see §3.3) that the eigenvectors of the optimal posterior is the same as that of the Hessian $H_w$.

We also calculated the effective dimensionalities, strength and sloppy factor of different models using $\epsilon$ derived in Method 3 (the $\epsilon$ calculated by PAC-Bayes bound optimization can be regarded as a sound choice), the results are displayed in the 4th block of Table S-3.

## G. Further experimental studies

### G.1. Additional results on the sloppiness of different architectures and datasets

MNIST in spite of its lower dimensionality has roughly the same range of eigenvalues but it has a very small threshold $r$ in Def. 8 which indicates that data has a lower number of effective dimensions than CIFAR-10. The FIM (empirical FIM is essentially the same line) shows a very strong decay for MNIST; since the trace of the FIM has been used as an indicator of the information stored in the weights (Achille et al., 2018), this indicates that the weights have to store very little information to predict MNIST well. The Hessian and FIM have very different eigenvalues for MNIST but as Fig. 3 indicates the two matrices have a larger overlap in their top eigenvectors. Eigenspectra of other networks on MNIST are similar to Fig. S-7 while those of CIFAR-10 are similar to Fig. 1.

In Fig. S-8, we compare the correlation matrices of logit Jacobian for different logits, which shows that the eigenspectra for different logits are similar. In Fig. S-9 and Fig. S-10 we compare the correlation matrices of activations and their gradients. From the figures, we can see that the eigenspectra are similar for different layers, which shows that the sloppiness is preserved as we getting into higher layers of neural network. In S-11 and S-12 we ploted the eigenspectra for different networks. The similarity of eigenspectra of matrices calculated on same dataset but different architectures strongly indicates that the sloppiness of Hessian, FIM, correlations of logit Jacobians, activations and gradients of activations are all inherited from the sloppiness of the data set. Fig. S-13 is a reproduction of Fig. 5 using FC-1200-1 on MNIST.

### G.2. Weights of a trained network can come back towards the initialization in the sloppy subspace even if they evolved in the stiff subspace

Fig. S-14 shows that the projection of change of weights of the model ($w - w_0$) for the v2 model (which has a second phase of training with a penalty $\propto ||w - w_0||_2^2$) onto the stiff directions is larger than that of original model (FC). This indicates that the projection onto the sloppy directions of model v2 is smaller than that of the original model because the projection onto orthogonal decompositions of the parameter space sums to one. This indicates that weights can effectively come back towards the initialization in the sloppy subspace without affecting the accuracy of the model even if the model predominantly evolves in the stiff subspace during training.

| Quantity/Model | FC-600-1 | FC-600-2 | FC-1200-1 | FC-1200-2 | LeNet |
|---|---|---|---|---|---|
| **Training and validation error of the trained model** | | | | | |
| $\hat{e}(h_w, D_n)$ | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 |
| $\log 2 * \check{e}(h_w, D_n)$ | 0.0008 | 0.0000 | 0.0010 | 0.0000 | 0.0000 |
| $e(h_w)$ | 0.0150 | 0.0143 | 0.0146 | 0.0139 | 0.0111 |
| $\log 2 * \check{e}(h_w)$ | 0.0641 | 0.0956 | 0.0584 | 0.0977 | 0.0669 |
| **Analytic (Method 1)** | | | | | |
| $\hat{e}(Q, D_n)$ | 0.0901 | 0.0766 | 0.0534 | 0.0678 | 0.0074 |
| $\log 2 * \check{e}(Q, D_n)$ | 0.2299 | 0.1997 | 0.1410 | 0.1776 | 0.0263 |
| $e(Q)$ | 0.0897 | 0.0827 | 0.0553 | 0.0729 | 0.0167 |
| $\log 2 * \check{e}(Q)$ | 0.2384 | 0.2314 | 0.1492 | 0.2015 | 0.0927 |
| PAC-Bayes bound | 0.3241 | 0.3794 | 0.3509 | 0.3915 | 0.0572 |
| $\mathrm{KL}(Q, P)$ | 8512.5098 | 13417.4023 | 14088.1738 | 15308.4170 | 1965.8048 |
| $\epsilon$ | 199.4836 | 401.7107 | 328.8929 | 443.9590 | 36.4424 |
| $\mathbf{Ev}(\Sigma_q) = \mathbf{Ev}(F_{w_0})$ **(Method 2)** | | | | | |
| $\hat{e}(Q, D_n)$ | 0.0309 | 0.0288 | 0.0267 | 0.0298 | 0.0053 |
| $\log 2 * \check{e}(Q, D_n)$ | 0.0895 | 0.0798 | 0.0742 | 0.0829 | 0.0160 |
| $e(Q)$ | 0.0346 | 0.0331 | 0.0327 | 0.0348 | 0.0147 |
| $\log 2 * \check{e}(Q)$ | 0.0995 | 0.0959 | 0.0947 | 0.0995 | 0.0590 |
| PAC-Bayes bound | 0.1590 | 0.1767 | 0.1523 | 0.2017 | 0.0099 |
| $\mathrm{KL}(Q, P)$ | 4772.4854 | 5953.1523 | 4841.5972 | 7268.2832 | 46.5822 |
| $\mathbf{E}(\Sigma_q) = \mathbf{E}(H_w)$ **(Method 3, our implementation)** | | | | | |
| $\hat{e}(Q, D_n)$ | 0.0202 | 0.0165 | 0.0169 | 0.0178 | 0.0043 |
| $\log 2 * \check{e}(Q, D_n)$ | 0.0556 | 0.0451 | 0.0466 | 0.0487 | 0.0133 |
| $e(Q)$ | 0.0268 | 0.0253 | 0.0245 | 0.0249 | 0.0141 |
| $\log 2 * \check{e}(Q)$ | 0.0781 | 0.0781 | 0.0761 | 0.0742 | 0.0564 |
| PAC-Bayes bound | 0.1357 | 0.1540 | 0.1515 | 0.1817 | 0.0188 |
| $\mathrm{KL}(Q, P)$ | 4645.1128 | 6122.5703 | 5919.6455 | 7589.6387 | 430.4026 |
| $\epsilon$ | 46 | 101 | 53 | 172 | 1360 |
| $p(n, \epsilon)$ | 2301 (0.487%) | 2429 (0.292 %) | 2315 (0.245 %) | 2287 (0.095 %) | 82 (0.184 %) |
| $s(n, \epsilon)$ | 6435 | 6810 | 6420 | 6280 | 231 |
| $1/c(n, \epsilon)$ | 2236 | 2497 | 2604 | 2841 | 38 |
| $\Sigma_p = aF_{w_0} + \epsilon^{-1}, \mathbf{E}(\Sigma_q) = \mathbf{E}(F_{w_0})$ **(Method 4)** | | | | | |
| $\hat{e}(Q, D_n)$ | 0.0237 | 0.0218 | 0.0226 | 0.0220 | 0.0048 |
| $\log 2 * \check{e}(Q, D_n)$ | 0.0663 | 0.0611 | 0.0631 | 0.0614 | 0.0147 |
| $e(Q)$ | 0.0270 | 0.0265 | 0.0266 | 0.0264 | 0.0145 |
| $\log 2 * \check{e}(Q)$ | 0.0806 | 0.0956 | 0.0789 | 0.0801 | 0.0573 |
| PAC-Bayes bound | 0.1323 | 0.1397 | 0.1486 | 0.1702 | 0.0092 |
| $\mathrm{KL}(Q, P)$ | 4090.7241 | 4679.0293 | 5074.4102 | 6369.7505 | 23.2886 |
| $\mathbf{diag}(\Sigma_q) = \Lambda$ **(our implementation)** | | | | | |
| $\hat{e}(Q, D_n)$ | 0.0283 | 0.0249 | 0.0284 | 0.0285 | 0.0079 |
| $\log 2 * \check{e}(Q, D_n)$ | 0.0795 | 0.0700 | 0.0795 | 0.0797 | 0.0236 |
| $e(Q)$ | 0.0330 | 0.0311 | 0.0326 | 0.0331 | 0.0161 |
| $\log 2 * \check{e}(Q)$ | 0.0942 | 0.0923 | 0.0940 | 0.0963 | 0.0637 |
| PAC-Bayes bound | 0.1707 | 0.1846 | 0.1886 | 0.2167 | 0.0131 |
| $\mathrm{KL}(Q, P)$ | 5674.5186 | 6854.7871 | 6668.9023 | 8332.5869 | 37.5598 |
| **Numerical optimization of Method 1 calculations (Method 5)** | | | | | |
| $\hat{e}(Q, D_n)$ | 0.0711 | 0.0630 | 0.0805 | 0.0580 | 0.0087 |
| $\log 2 * \check{e}(Q, D_n)$ | 0.1805 | 0.1673 | 0.2072 | 0.1510 | 0.0331 |
| $e(Q)$ | 0.0717 | 0.0683 | 0.0800 | 0.0644 | 0.0168 |
| $\log 2 * \check{e}(Q)$ | 0.1902 | 0.1925 | 0.2092 | 0.1811 | 0.0955 |
| PAC-Bayes bound | 0.3182 | 0.3917 | 0.3539 | 0.4366 | 0.0792 |
| $\mathrm{KL}(Q, P)$ | 9920.2510 | 15908.7813 | 11271.7246 | 20162.2891 | 2941.7917 |
| $\epsilon$ | 243.6499 | 490.6506 | 269.2748 | 599.2820 | 54.3656 |
| **Isotropic Posterior (Method 6)** | | | | | |
| $\hat{e}(Q, D_n)$ | 0.0473 | 0.0879 | 0.0653 | 0.0638 | 0.0094 |
| $\log 2 * \check{e}(Q, D_n)$ | 0.1266 | 0.2538 | 0.1661 | 0.1757 | 0.0385 |
| $e(Q)$ | 0.0524 | 0.0937 | 0.0677 | 0.0697 | 0.0191 |
| $\log 2 * \check{e}(Q)$ | 0.1409 | 0.2935 | 0.1759 | 0.2057 | 0.1076 |
| PAC-Bayes bound | 0.3694 | 0.5461 | 18533.2422 | 0.5490 | 0.1146 |
| $\mathrm{KL}(Q, P)$ | 16261.0205 | 26160.2773 | 0.4288 | 30034.0645 | 4807.3564 |
| $\epsilon$ | 401.7107 | 808.9461 | 443.9590 | 894.0237 | 89.6338 |
| $\mathbf{diag}(\Sigma_q) = \Lambda$ **(from Dziugaite & Roy (2017))** | | | | | |
| $e(h_w)$ | 0.018 | 0.016 | 0.018 | 0.015 | - |
| $e(Q)$ | 0.034 | 0.033 | 0.035 | 0.035 | - |
| PAC-Bayes bound | 0.161 | 0.186 | 0.179 | 0.223 | - |
| $\mathrm{KL}(Q, P)$ | 5144 | 6534 | 5977 | 8558 | - |
| $\mathbf{E}(\Sigma_q) = \mathbf{E}(H_w)$ **(from (Wu et al., 2021))** | | | | | |
| $e(h_w)$ | 0.0153 | 0.0148 | 0.0161 | - | - |
| $e(Q)$ | 0.02347 | 0.02523 | 0.02316 | - | - |
| PAC-Bayes bound | 0.1198 | 0.1443 | 0.1413 | - | - |
| $\mathrm{KL}(Q, P)$ | 3766.1 | 4956.8 | 5021.1 | - | - |

*Table S-3.* **Comparison of PAC-Bayes bounds on MNIST for different methods.** This table is an expansion of Table 2. The 6th block is our reproduction of (Dziugaite & Roy, 2017), the first, 7th and 8th block corresponds to the three methods of constructing posterior for PAC-Bayes bound without training described in Appendix D.1.

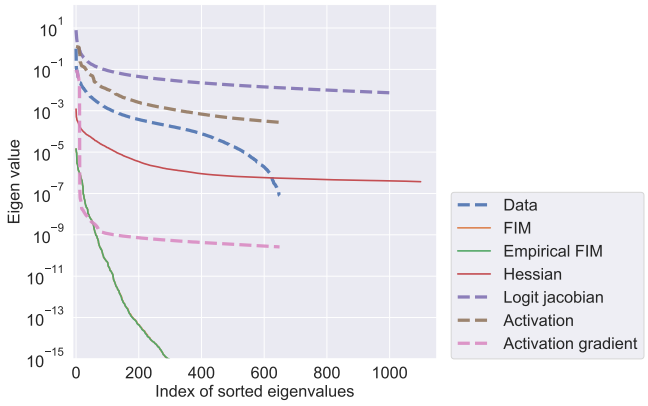


*Figure S-7.* Eigenspectra for a two-layer fully-connected network on MNIST. The eigenspectra are qualitatively the same as those of Fig. 1, e.g., there is a sharp drop at the beginning and a long, linear tail of small eigenvalues follows. Slopes of the eigenspectra of activations, activation gradients, Jacobians and Hessian mirror those of the data. In contrast to Fig. 1, the slope of the FIM is quite different here. The Empirical FIM and FIM overlaps with each other since the model is trained to nearly perfect train and validation error.

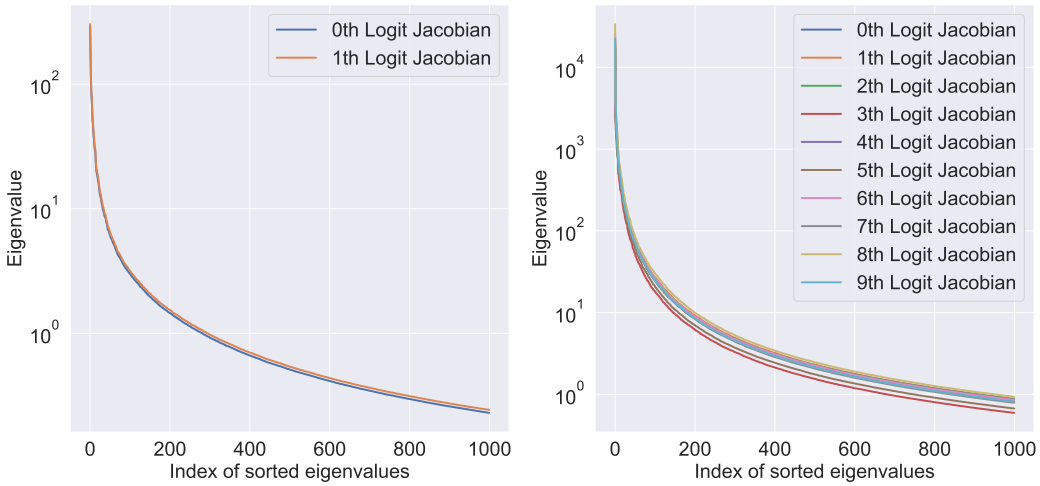


*Figure S-8.* Eigenspectra of the correlation matrices of Jacobian of logits for FC-600-2 on MNIST (Left) and wide residual net on CIFAR-10 (Right). The eigenspectra are similar for different logits.

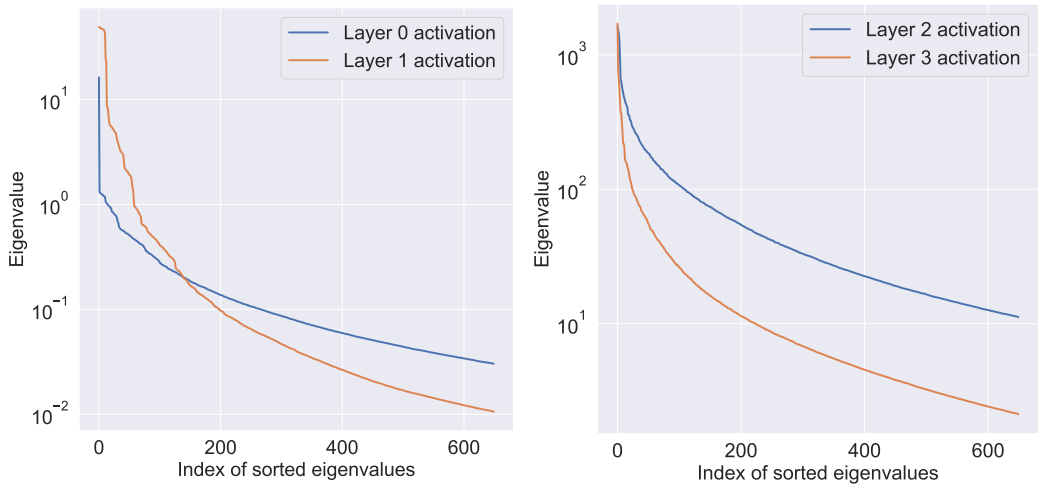


*Figure S-9.* Eigenspectra of the correlation matrices of activations of different layers for FC-600-2 on MNIST (Left) and wide residual net on CIFAR-10 (Right). For different layers, the eigenspectra are similar.

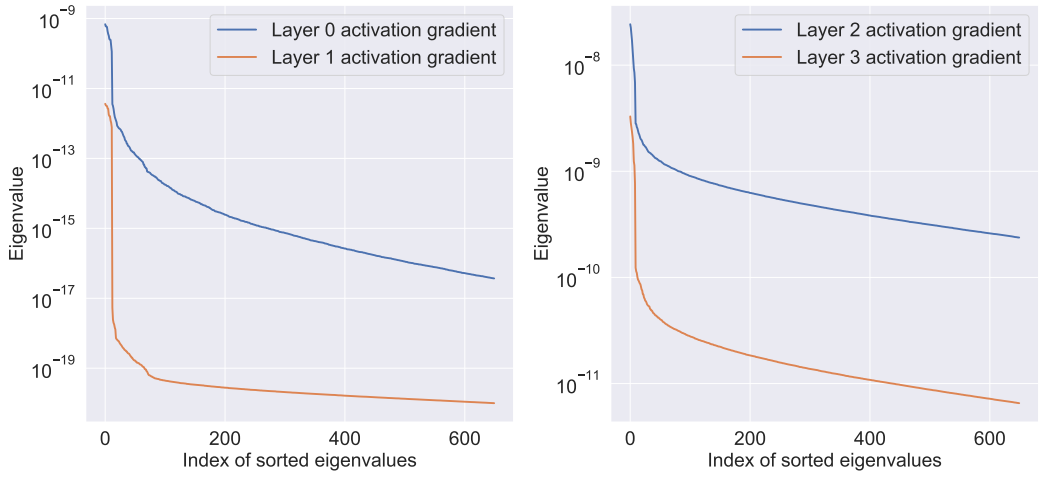


*Figure S-10.* Eigenspectra of the correlation matrices of gradients with respect to the activations of different layers for FC-600-2 on MNIST (Left) and wide residual net on CIFAR-10 (Right). For different layers, the eigenspectra are qualitatively similar, and as we move into higher layers of neural networks, the eigenvalues becomes smaller for gradient of activations.

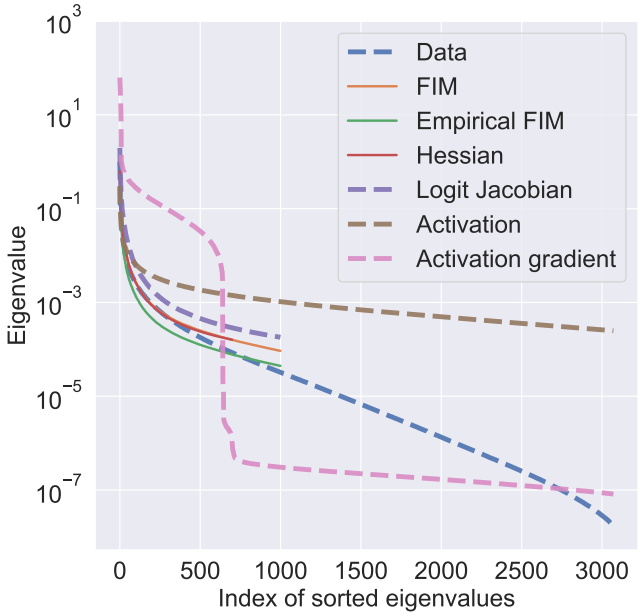


*Figure S-11.* Eigenspectra for ALL-CNN on CIFAR-10. The eigenspectra are qualitatively the same as those of Fig. 1 for a wide residual network on CIFAR-10.

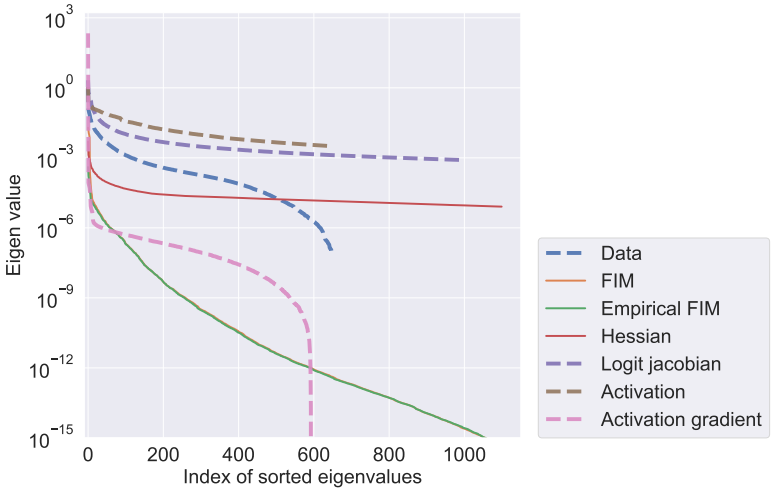


*Figure S-12.* Eigenspectra for FC-1200-1 on MNIST. The eigenspectra are qualitatively the same as those of Fig. S-7 for FC-600-2 on MNIST.

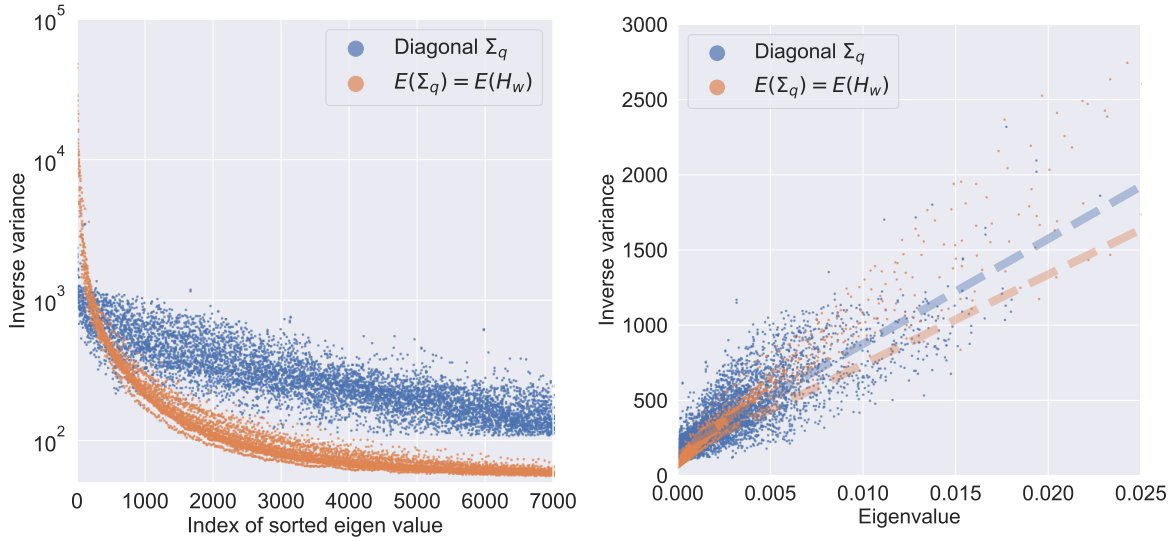


*Figure S-13.* **Posterior covariance computed by optimizing PAC-Bayes bound is aligned with sloppy directions** This plot is a reproduction of Fig. 5 for FC-1200-1 (Fig. 5 is for FC-600-2). We get $n \sim 30000$ (true $n = 55000$) and $\epsilon \sim 138.2$ (true $\epsilon = 53.3$)

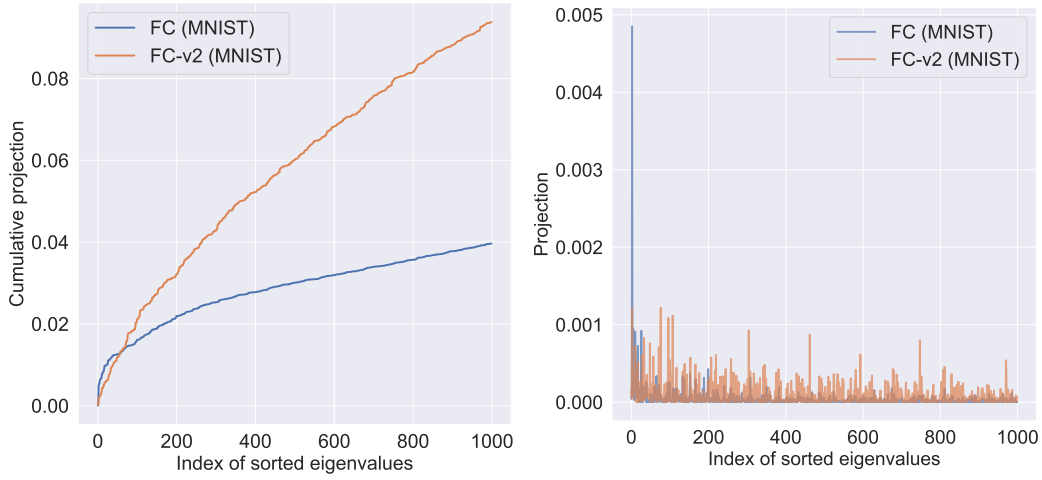


*Figure S-14.* (Left) Cumulative projection $\|E_k(F_{w_0})\Delta w\|_2^2/\|\Delta w\|_2^2$ of the change in weights during training (where $\Delta w = w - w_0$) onto the eigenspace of the top $k$th eigenvalues of $F_{w_0}$. (Right) Projection $\|\nu_k(F_{w_0})\Delta w\|_2^2/\|\Delta w\|_2^2$ of the change in weights during training onto the eigenvector $\nu_k(F_{w_0})$ of $k$th largest eigenvalue of $F_{w_0}$, which is the derivative of the curve in the left plot. We use FC-600-2 on MNIST for this experiment.

### G.3. PAC-Bayes generalization bounds and effective dimensionality for synthetic data sets

In Table S-4, we show the results for PAC-Bayes bound optimization for synthetic data sets introduced in §6.1. Using the $\epsilon$ derived by Method 3, we calculate the effective dimensionality, strength, and sloppy factor of Hessian at the end of training (showed in the third block of Table S-4), which shows that for non-sloppy data set, we have heavier tails and more stiff directions, resulting in worse generalization.

### G.4. Eigenspectra at the middle of training

Fig. S-15 shows the eigenspectra for FIM and Hessian of a wide residual net on CIFAR-10 during training, which shows that the eigenspectra are qualitatively the same throughout training.

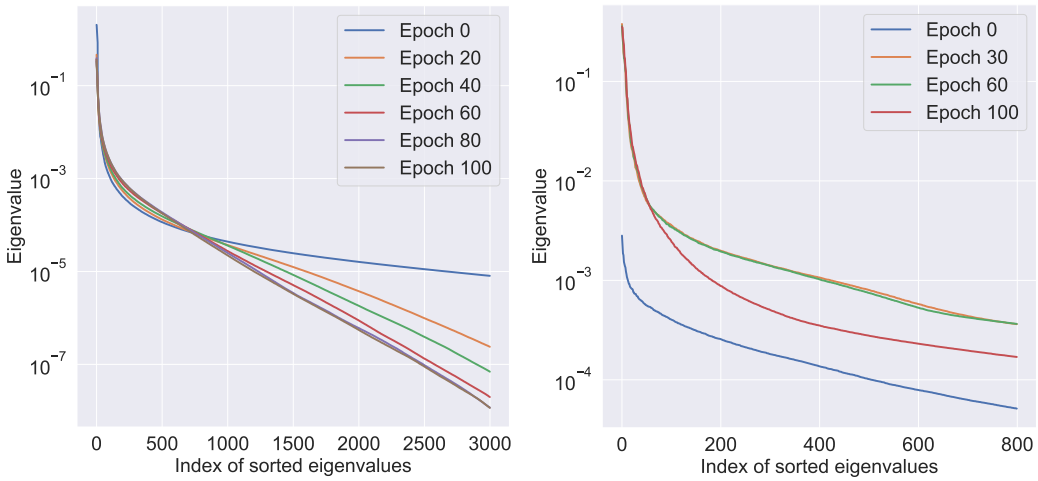


*Figure S-15.* (Left) Eigenspectra for FIM of WRN on CIFAR-10 throughout training. The eigenspectrum for epoch 0 are scaled up by $10^3$ to bring it to this scale. (Right) Eigenspectra for Hessian of WRN on CIFAR-10 throughout training. We take the absolute value of the eigenvalues. The eigenspectra are qualitatively the same.

### G.5. Eigenspectrum at the end of training for data sets with random labels

Fig. S-16 shows the eigenspectra of empirical FIM at the end of training, in which the eigenspectra is less sloppy for data sets with random labels, and the top eigenvalues increases as we increase the fraction of random labels. This shows that even if we have the same input data set, the sloppiness and the top few eigenvectors can still be affected by the task. The eigenspectra becomes less sloppy and has larger head when the the model is used to learn more difficult tasks (more random labels).

| Quantity/Model | Random-0.1 | Random-0.001 |
|---|---|---|
| **Training and validation error of the trained model** | | |
| $\hat{e}(h_w, D_n)$ | 0.0000 | 0.0749 |
| $\log 2 * \check{e}(h_w, D_n)$ | 0.0869 | 0.5745 |
| $e(h_w)$ | 0.1150 | 0.5035 |
| $\log 2 * \check{e}(h_w)$ | 0.2983 | 1.3797 |
| $\mathbf{Ev}(\Sigma_q) = \mathbf{Ev}(F_{w_0})$ **(Method 2)** | | |
| $\hat{e}(Q, D_n)$ | 0.0155 | 0.0117 |
| $\log 2 * \check{e}(Q, D_n)$ | 0.0509 | 0.2368 |
| $e(Q)$ | 0.1191 | 0.4596 |
| $\log 2 * \check{e}(Q)$ | 0.3965 | 1.2574 |
| PAC-Bayes bound | 0.3560 | 0.6781 |
| $\mathrm{KL}(Q, P)$ | 18468.6914 | 53052.6094 |
| $\mathbf{E}(\Sigma_q) = \mathbf{E}(H_w)$ **(Method 3, our implementation)** | | |
| $\hat{e}(Q, D_n)$ | 0.0102 | 0.0128 |
| $\log 2 * \check{e}(Q, D_n)$ | 0.0420 | 0.2508 |
| $e(Q)$ | 0.1133 | 0.4614 |
| $\log 2 * \check{e}(Q)$ | 0.3793 | 1.2556 |
| PAC-Bayes bound | 0.2986 | 0.6769 |
| $\mathrm{KL}(Q, P)$ | 15311.6416 | 52592.7852 |
| $\epsilon$ | 1.60 | 0.266 |
| $p(n, \epsilon)$ | 542 (0.256%) | 1730 (0.820 %) |
| $s(n, \epsilon)$ | 2325 | 4962 |
| $1/c(n, \epsilon)$ | 209 | 452 |
| $\Sigma_p = aF_{w_0} + \epsilon^{-1}, \mathbf{E}(\Sigma_q) = \mathbf{E}(F_{w_0})$ **(Method 4)** | | |
| $\hat{e}(Q, D_n)$ | 0.0093 | 0.0083 |
| $\log 2 * \check{e}(Q, D_n)$ | 0.0262 | 0.2217 |
| $e(Q)$ | 0.1199 | 0.5035 |
| $\log 2 * \check{e}(Q)$ | 0.6493 | 1.3797 |
| PAC-Bayes bound | 0.2514 | 0.6577 |
| $\mathrm{KL}(Q, P)$ | 12346.2207 | 50933.4063 |
| $\mathbf{diag}(\Sigma_q) = \Lambda$ **(our implementation)** | | |
| $\hat{e}(Q, D_n)$ | 0.0094 | 0.0083 |
| $\log 2 * \check{e}(Q, D_n)$ | 0.0275 | 0.0452 |
| $e(Q)$ | 0.1325 | 0.4599 |
| $\log 2 * \check{e}(Q)$ | 0.6041 | 2.1837 |
| PAC-Bayes bound | 0.5096 | 0.7512 |
| $\mathrm{KL}(Q, P)$ | 32949.1836 | 66678.6016 |

*Table S-4.* **Comparison of PAC-Bayes bounds on synthetic data sets.** The table shows the PAC-Bayes bound optimization results of the synthetic data sets introduced in §6.1. The first 3 blocks corresponds to our Methods 2-4 described in §3.3. The 4th block is our reproduction of (Dziugaite & Roy, 2017) on synthetic data sets.

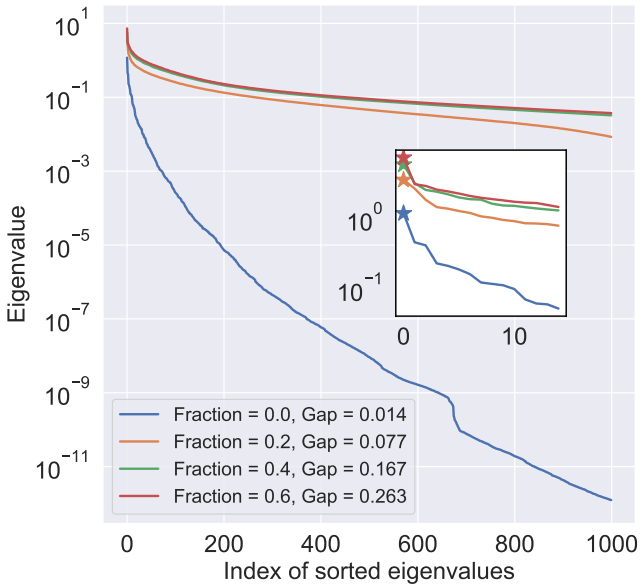


*Figure S-16.* **Eigenspectra at the end of training for data sets with random labels.** The plots shows the eigenspectra of empirical FIM at the end of training. The experiment is done on MNIST using fully connected net FC-600-2. The label "Fraction=$a$" indicates the data set with random label of fraction $a$. The inset plot shows the top 15 eigenvalues. The line for Fraction=0.0 are scaled up by $10^7$. The plots shows that the FIM at the end of training is less sloppy for data sets with random labels, and the top eigenvalues increases as we increase the fraction of random labels.