Does the Data Induce Capacity Control in Deep Learning?

Rubing Yang\textsuperscript{1}, Jialin Mao\textsuperscript{1}, and Pratik Chaudhari\textsuperscript{2,3}

\textsuperscript{1}Applied Mathematics and Computational Sciences, University of Pennsylvania
\textsuperscript{2}Electrical and Systems Engineering, University of Pennsylvania
\textsuperscript{3}Computer and Information Science, University of Pennsylvania

Email: rubingy@sas.upenn.edu, jmao@sas.upenn.edu, pratikac@seas.upenn.edu

Abstract

This paper studies how the dataset may be the cause of the anomalous generalization performance of deep networks. We show that the data 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, non-sloppy synthetic data do not share these traits. We show how this structure in the data can give to non-vacuous PAC-Bayes generalization bounds analytically; we also construct data-distribution dependent priors that lead to accurate bounds using numerical optimization.

1 Introduction

Consider the experiment in Fig. 1. For a convolutional network trained on CIFAR-10, we calculated the eigenspectrum of the data correlation matrix \((n^{-1}XX^\top)\) where each column of \(X\) is one input sample and compared it to the eigenspectra of the Fisher Information Matrix (FIM) and the Hessian of a deep network trained on this dataset. Eigenspectrum of the FIM and the Hessian have a similar decay pattern as that of the data matrix. There are very few (less than 5\% of the input dimensionality) large—let us call them stiff—eigenvalues after which there is a sharp drop in magnitude and a long tail of small—let us call them “sloppy”—eigenvalues follows. Some other correlation matrices that one can obtain from the network also show the same decay pattern, e.g., that of activations of different layers, Jacobians of any of the logits with respect to the weights, and gradients of the loss with respect to activations of different layers. All these matrices have eigenvalues that span exponentially large ranges, e.g., the top 1000 eigenvalues drop by about 5 orders of magnitude. Sloppy eigenvalues are distributed uniformly across such exponentially large ranges. Eigenspectra of many other typical datasets and neural networks share these traits. On the other hand, eigenspectra corresponding to synthetic, atypical datasets, which we create by sampling inputs randomly and labeling them either randomly, or using a random teacher network, do not share these traits (§4).

The Hessian determines the local geometry in the weight space of the loss function of a deep network, small eigenvalues correspond to directions in the weight space where the loss is insensitive to changes in the weights. Similarly, the FIM governs the local geometry in the prediction space, i.e., if we think of a deep network as a parameterized probability distribution \(p_w(y \mid x)\), the FIM is the correlation matrix of the gradient of log-likelihood \(\log p_w(y \mid x)\). Eigenvectors of small eigenvalues of the FIM correspond to sets of weights which can be modified significantly without affecting the probability distribution \(p_w(y \mid x)\) much.

This leads us to the hypothesis that the training data effectively controls the complexity of the trained model. We investigate this hypothesis in this paper and make the following contributions.
Figure 1: Eigenspectra of the correlation matrices of the data (dashed blue), the activations (dashed brown) and gradients with respect to the activations (dashed pink) of the second layer, Jacobian of one of the logits with respect to the weights (dashed purple) and the FIM (orange), empirical FIM (green), Hessian (red). Activation gradients are scaled up by $10^{12}$ to bring them to this scale. All eigenspectra are scaled by the largest eigenvalue of the data correlation matrix. All eigenspectra are qualitatively similar with eigenvalues that span exponentially large ranges. After an initial regime where the top few eigenvalues drop sharply (which we call stiff), the slope of the tail (which we call sloppy) of these eigenspectra on a logarithmic scale is almost the same as that of the data matrix. We call this phenomenon “sloppiness”. Eigenspectra corresponding to activations/activation gradients of other layers and logit Jacobians of other logits are very similar (see Appendix G). This plot shows the eigenspectra for a trained wide residual network on CIFAR-10, corresponding eigenspectra of other networks on MNIST, even those at initialization, are qualitatively the same (see Fig. 2 and Appendix G).

1. We show that for typical datasets and deep networks, eigenspectra of correlation matrices of the data, activations of different layers, Jacobian of the logits with respect to the weights, gradients of the loss with respect to the activations, as also the Hessian and the FIM, are all qualitatively the same. These eigenspectra consists of a few large eigenvalues and a large number of small eigenvalues that are distributed uniformly across an exponentially large range. We call such eigenspectra and the corresponding quantities “sloppy” and define this notion in Def. 8. Atypical datasets, e.g., those with random inputs or labels, can be constructed for whom these quantities do not have such eigenspectra. This suggests that typical inputs are “simple” (in the precise sense of having a sloppy data correlation matrix) and this results in a “simple” trained model (in the sense of a sloppy Hessian or FIM). We show using theory 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 data correlation matrix, (b) if we assume that the activations are sloppy then we can show that the eigenspectrum of the FIM is sloppy, (c) under the assumption of bounded norm of weights, we can show that the eigenvalues of activations decays faster than $O(1/i)$.

2. As a demonstration of how sloppiness of data controls the capacity of the trained model, we calculate non-vacuous PAC-Bayes bounds analytically. For a Gaussian isotropic prior $N(w_0, \epsilon^{-1}I)$ centered at the initial 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 loose version of PAC-Bayes generalization bound. We show that if the Hessian at $w$ is sloppy, we can obtain a non-vacuous generalization bound—without any numerical optimization. Specifically, we show that the optimal PAC-Bayes posterior inverse covariance has the same
eigenvectors as Hessian, and the corresponding eigenvalues are

\[ \tilde{\lambda}_i = 2(n - 1)\lambda_i + \epsilon \quad \forall i \leq p, \]

where \( \lambda_i \) are eigenvalues of the Hessian, \( n \) is the number of samples and \( p \) is the number of weights. A posterior that leads to a small bound is less spread out along stiff directions but more spread out along sloppy directions of the Hessian.

(3) We can think of the prior inverse variance scaled by the number of data \( \epsilon/(2(n - 1)) \) as a threshold beyond which the eigenvalues of the Hessian \( \lambda_i \) are small enough and the loss changes so little that the optimal PAC-Bayes posterior focuses on accurately capturing the prior covariance to obtain a small KL-term. Motivated by this, we define an **effective dimensionality of a deep network** as

\[ p(n, \epsilon) = \sum_{i=1}^{p} 1\{ |\lambda_i| > \frac{\epsilon}{2(n - 1)} \}, \]

i.e., the number of eigenvalues of the Hessian that are greater than a threshold \( \epsilon/(2(n - 1)) \) for a user-chosen PAC-Bayes prior \( N(0, \epsilon^{-1}I) \). For sloppy eigenspectra, for values of \( \epsilon \) that lead to non-vacuous bounds, this dimensionality is typically a tiny fraction of the number of weights (0.23% for a fully-connected network on MNIST).

(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**. Updates to the weights take place in the subspace spanned by the stiff eigenvectors of the FIM (as also the Hessian). We exploit this observation to compute tight data-distribution/data-dependent PAC-Bayes bounds for fully-connected and convolutional networks on MNIST 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.

## 2 Background

This section develops notation and PAC-Bayesian generalization bounds.

### 2.1 Problem Setup

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 \( h \colon X \mapsto [0, 1]^m \) parameterized by weights \( w \in \mathbb{R}^p \) belongs to a hypothesis space \( \{h_w \colon w \in \mathbb{R}^p\} \); 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 \( \hat{e}(h_w, D_n) = \frac{1}{n} \sum_{i=1}^{n} 1\{ y_i \neq \text{argmax}_y (p_w(y \mid x_i)) \} \);

(b) cross-entropy objective \( \bar{e}(h_w, D_n) = -\frac{1}{n \log(2)} \sum_{i=1}^{n} \log p_w(y_i \mid x_i) \) which is used as a differentiable surrogate and convex upper-bound of the error for purposes of training;

(c) population error of a hypothesis \( e(h_w) = \mathbb{E}_{D \sim D} [\hat{e}(h_w, D_n)] \);

(d) empirical error and loss of the distribution of hypotheses \( \bar{e}(Q, D_n) = \mathbb{E}_{w \sim Q} [\hat{e}(h_w, D_n)] \) and \( \bar{e}(Q, D_n) = \mathbb{E}_{w \sim Q} [\bar{e}(h_w, D_n)] \), respectively; and

(e) with some abuse of notation, population error of \( Q \) given by \( e(Q) = \mathbb{E}_{D \sim D} [\hat{e}(Q, D_n)] \),

(f) we use \( \bar{e}(Q) = \mathbb{E}_{D_n \sim D} [\bar{e}(Q, D_n)] \) to denote the rescaled population loss.

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

\[ \frac{1}{n} \sum_{k=1}^{n} \sum_{y=1}^{m} p_w(y \mid x_k) \partial_i \log p_w(y \mid x_k) \partial_j \log p_w(y \mid x_k); \]
it is important to note the expectation over the outputs \( y \). A related quantity is the empirical FIM where one sets \( y = y_k \) in the expression above. Although the FIM and the empirical FIM are different in general, especially when used in algorithms like natural gradient descent (Kunstner et al., 2019), their eigenspectra are qualitatively similar (Fig. 1). Both the Hessian and FIM are large matrices and it is prohibitive to compute them for modern deep networks. Some of our experiments use a Kronecker-factor approximation (Martens and Grosse, 2016) of block diagonal Hessian and FIM where cross-terms \( \partial_i \partial_j \) across different layers of a deep network are set to zero.

### 2.2 PAC-Bayes generalization bounds

The PAC-Bayesian framework developed in Langford and 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

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

where \( \text{KL}(Q, P) = \int dQ(w) \log(dQ / dP)(w) \). We will also define a KL divergence between two Bernoulli random variables with parameters \( b, a \) as \( \text{kl}(b, a) = b \log(b/a) + (1 - b) \log((1 - b)/(1 - a)) \). \( Q \) such a bound can be turned into an optimization algorithm (Langford and Caruana, 2002; Dziugaite and Roy, 2017), i.e., we can optimize the right-hand side to compute a distribution \( Q \) that gives the tightest bound. This typically involves picking a prior and posterior from, say, the Gaussian family, and inverting the left-hand side using \( \text{kl}^{-1}(b, a) \leq b + \sqrt{a/2} \) to get a bound on \( e(Q) \).

Consider a Gaussian prior \( N(w_0, \epsilon^{-1} I) \) and posterior \( N(w, \Sigma_q) \). We can work around the facts that the scale \( \epsilon \) of the prior is unknown by searching over a discrete set \( \epsilon = c \exp(j/b) \) for \( j \in \mathbb{N} \) and that \( \hat{e}(Q, D_n) \leq \hat{e}(Q, D_n) \) to get a differentiable bound that is able to be optimized using SGD. See Appendix B for details.

\[
e(Q) \leq \hat{e}(Q, D_n) + \left( \frac{\text{KL}(Q, P) + 2 \log(b \log(c/\epsilon)) + \log(\pi^2 n/(6 \delta))}{2(n-1)} \right)^{1/2}, \tag{2}
\]

with probability at least \( 1 - \delta \).

### 2.3 Data-dependent 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 and Roy, 2018). To gain intuition, notice that in the expression for the KL-divergence between two Gaussians, we have a term of the form \( (w - w_0)^\top \Sigma_p^{-1} (w - w_0) \) depends upon the distance between trained weights \( w \) and initialization \( w_0 \) (weighted by the inverse prior covariance \( \Sigma_p^{-1} \)). It is difficult to pick a prior \( P \) such that this term is small.

**Priors that depend upon the FIM and the Hessian.** In order to compute a tight PAC-Bayes bound, one may choose a prior using a subset of the training samples (Ambroladze et al., 2007). For instance, we can center the Gaussian prior on weights pre-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 but in practice, we evaluate this prior using the dataset. For example, choosing the prior covariance to be proportional to the FIM, or the Gauss-Newton approximation of the Hessian,

\[
\Sigma_p \propto F_{w_0}, \text{ or, } \Sigma_p \propto \tilde{H}_{w_0}, \tag{3}
\]
are both valid choices in the PAC-Bayes framework. The crucial distinction between these two choices is that while we may use all the training samples to compute the FIM in the former case, we should compute the Hessian on a separate subset of the data in the latter case.

3 Methods

In §3.1, we prove how such a sloppiness in the Hessian and the FIM arises if it is present in the input correlation matrix. We then discuss different methods to compute PAC-Bayes bounds on the generalization error using sloppiness (§3.2) and develop an expression for the effective dimensionality of a deep network (§3.3).

3.1 A sloppy data correlation matrix leads to a sloppy FIM and Hessian

Consider a deep network with $L$ layers with the input of the network $x \in \mathbb{R}^d$ and $m$ outputs. The activations of the $k$th layer are given by $h^k = \sigma(u^{k-1}h^{k-1})$, for $k = 1, \cdots, L$. Preactivations 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 $z = u^{L+1}$ to denote the logits of the network. The weights of the network are $w = (w^0, w^1, \ldots, w^L)$. We have $h^k \in \mathbb{R}^{d_k}$, $u^k \in \mathbb{R}^{d_{k+1} \times d_k}$ and $w^k \in \mathbb{R}^{m \times d_k}$. In this notation, we have $h_0 = x \in \mathbb{R}^d$. 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. The non-linearity $\sigma$ acts element-wise upon its argument and we assume that it has a bounded derivative, $|\sigma'(x)| \leq a$ with $\sigma(x) = 0$, in which case, $|\sigma(x)| \leq a|x|$. Popular nonlinearities like rectified linear units (ReLU), leaky ReLUs and tanh satisfy this assumption. In this section, we use $\mathbb{E}$ to denote the expectation over $x$. 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. $x \sim \frac{1}{n} \sum_{i=1}^n \delta_x$, in this case, $\mathbb{E}[xx^T] = \frac{1}{n}XX^T \in \mathbb{R}^{d \times d}$ is the input correlation matrix. See Appendix C for proofs of the results in this section.

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 1 (Bounding the trace of the correlations of activations and norm of activation gradients). We have

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

and

$$\left\| \frac{dz_i}{dh^k} \right\|_2 \leq a \left\| \frac{dz_i}{dh^{k+1}} \right\|_2 \left\| w^k \right\|_2.$$

We first bound the trace of the FIM and the Hessian in terms of the trace of the input correlation matrix.

Theorem 2 (Trace of the FIM and Hessian are bounded by that of the data correlation matrix). For the FIM $F$ and the Hessian $H$, we have

$$\text{tr}(F), (\log 2)\text{tr}(H) \leq ma^2 \text{tr} \left( \mathbb{E}[xx^T] \right) \prod_{j=0}^L \left\| w^j \right\|_2^2 \left( \sum_{j=0}^L \frac{1}{\left\| w^j \right\|_2^2} \right).$$

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

We would next like to show that if the eigenvalues of the input correlation matrix $\frac{1}{n}XX^T \in \mathbb{R}^{d \times d}$ are sloppy, then the FIM and the Hessian are also sloppy. Note that for a regression problem with a deep linear network, the Hessian and the FIM are actually the input matrix itself and therefore have the same eigenspectrum. It is however quite difficult to control the eigenspectrum of these matrices because they are a result of multiple nonlinear operations on the inputs. In general, the eigenspectrum of even the correlation of the activations can evolve arbitrarily. We therefore first bound the eigenvalues of a block-diagonal approximation of the FIM in terms of the eigenvalues of the activations in the following lemma.
we next discuss our different ways that exploit sloppiness of a deep network to pick the prior and posterior in

\[ \text{where we use the notation} \]

\[
3.2 \text{ PAC-Bayes generalization bounds that exploit sloppiness}
\]

We next discuss four different ways that exploit sloppiness of a deep network to pick the prior and posterior in PAC-Bayes theory.
Method 1: Posterior covariance has the same eigenvectors as that of the Hessian at end (Analytical). Consider a deep network trained to minimize the loss \( \bar{e}(h_{w'}, D_n) \). Assume that \( w \) is a local minimum of the objective. In this case, the Hessian at \( w, H_w \), is positive semi-definite. We can write \( H_w \) as its orthonormal decomposition \( H_w = U_w \Lambda_w U_w^T \) 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, our goal is to find the best \( \Sigma_q \) give a tight PAC-Bayes bound. We write the training objective in the neighborhood of \( w \) as
\[
\bar{e}(h_{w'}, D_n) = \bar{e}(h_w, D_n) + \frac{1}{2} \langle w' - w, H_w(w' - w) \rangle.
\]
We use a loose PAC-Bayes bound
\[
e(Q) \leq L(\Sigma_q) := \bar{e}(Q, D_n) + \frac{\text{KL}(Q, P)}{2(n-1)}
\]
for the purposes of deriving the posterior analytically. For the prior \( P = N(w_0, \epsilon^{-1} I) \), this gives
\[
L(\Sigma_q) \leq \frac{1}{4(n-1)} \| w - w_0 \|^2_2 = \frac{1}{2(n-1)} \left( \int Q(w') \log \frac{Q(w')}{2(n-1) \bar{e}(w', D_n)} P_w(w') \, dw' \right)
\]
where \( P_w = N(w, \epsilon^{-1} I) \) and \( L(\Sigma_q) \) attains a minimum when \( Q \propto \exp(-2(n-1)\bar{e}(w', D_n)) P_w(w') \), i.e. \( \Sigma_q^{-1} = 2(n-1)H_w + \epsilon I \), i.e.,
\[
\Sigma_q = U_w (\bar{\Lambda}_w)^{-1} U_w^T,
\]
where
\[
\bar{\lambda}_i = 2(n-1)\lambda_i + \epsilon \quad \forall i \leq p.
\]
which proves that the optimal posterior covariance has the same eigenvectors as that of the Hessian (see Appendix B for details).

Using this posterior, we obtain a non-vacuous bound on the generalization error (see §4). For example, the bound for a fully-connected network on MNIST with one hidden layer of 600 neurons is 0.32, \( \epsilon(Q) \approx 0.089 \) while an upper bound computed by Dziugaite and Roy (2017) by numerically optimizing the right hand-side of (1) is 0.161.

Note that even if we computed the posterior covariance analytically using a loose PAC-Bayes bound, we could substitute this posterior into (2). Given eigenvalues of the Hessian \( \Lambda_w \), if we fix the eigenvectors of \( \Sigma_q \) to be that of \( H_w \), we can also optimize the eigenvalues of the inverse posterior covariance \( \bar{\Lambda}_w \) directly using nonlinear optimization.

Remark 7 (PAC-Bayes posterior is more spread out along sloppy eigenvectors). 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 focuses on accurately capturing the prior covariance to obtain a small KL-term. For eigenvalues above this threshold, e.g., the stiff eigenvalues, the optimal posterior has to strike a balance between the empirical loss term and the KL term. We will see in §4 that this observation also holds for cases when posteriors are optimized.

3.3 Effective dimensionality of a deep network
Motivated by Remark 7, we define an effective dimensionality for a deep network as
\[
p(n, \epsilon) = \sum_{i=1}^{p} \mathbf{1}\left\{ |\lambda_i| \geq \frac{\epsilon}{2(n-1)} \right\}
\]
i.e., as the number of eigenvalues of the Hessian with magnitude at least \( \frac{\epsilon}{2(n-1)} \). We next assume—motivated by our experiments in Fig. 1 and §4—that the logarithm of the eigenvalues of the Hessian decays linearly beyond this threshold, i.e.,
\[
\lambda_i \leq \frac{\epsilon}{2(n-1)} e^{-c(i-p(n, \epsilon))} \quad \text{for } p(n, \epsilon) \leq i \leq p,
\]
\[
\leq \frac{\epsilon}{2(n-1)} e^{-c(\epsilon 2(n-1))} \quad \text{for } p(n, \epsilon) \leq i \leq p.
\]
for some constant $c > 0$. With this assumption, we can calculate (Appendix B)

\[
\tilde{c}(Q, D_n) - \tilde{c}(h_w, D_n) = \mathcal{O}(p(n, \epsilon)/n), \quad \text{and} \\
\text{KL}(Q, P)/(2(n - 1)) = \mathcal{O}(p(n, \epsilon)/n).
\]

(15)

for the optimal posterior $Q$ derived in Section 3.2. Therefore, for sloppy models, the PAC-Bayes bound in (2) is $\mathcal{O}(\frac{p(n, \epsilon)}{n} + \sqrt{\frac{p(n, \epsilon)}{n}})$. If we repeat the calculation for an isotropic Hessian $\lambda_i \equiv \lambda$, the PAC-Bayes upper bound in (2) is $\mathcal{O}(\frac{p}{n} + \frac{p}{n})$. Therefore if $p(n, \epsilon) \ll n$, we get better generalization bound than what would be given by assuming a non-zero lower bound on the eigenvalues of the Hessian. In particular, for $p(n, \epsilon) \ll n$, we can expect a non-vacuous bound. Note that for deep networks, we have $p \gg n$.

**Definition 8 (Sloppy eigenspectrum).** If $\lambda_i(A)$ denote eigenvalues of a symmetric matrix $A \in \mathbb{R}^{p \times p}$ in descending order $\lambda_1 \geq \cdots \lambda_p$, then the eigenspectrum of $A$ is sloppy (or $A$ is sloppy in short) if there exists an $r < p$ such that

\[
\lambda_i(A) \leq \lambda_r(A) e^{-c(i-r)}, \quad \text{for } i \geq r \geq 1.
\]

We call $c$ the “sloppiness factor”.

Note that this definition implicitly means that the small eigenvalues beyond $\lambda_r$ are uniformly distributed across an exponentially large range $(\lambda_r, \lambda_p)$. Let us emphasize 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).

**Remark 9 (Why does the effective dimensionality depend upon $\epsilon$?).** Our definition of the effective dimensionality may seem unusual because it depends upon $\epsilon$, which is essentially a user-chosen parameter. This is an artifact of the PAC-Bayes analysis. As $\epsilon \to 0$, the effective dimensionality converges to the number of weights $p$, but for non-zero values of $\epsilon$, where the prior covariance restricts the set of hypotheses that the PAC-Bayes learner searches over, this expression coupled with the analytical calculation in (12) may provide a useful way to perform model selection. For practical purposes, we can also replace eigenvalues of the Hessian in (12) by those of the FIM (see Fig. 1), which is easier to compute.

Tight upper and lower bounds for the VC-dimension of deep networks are of the form $\mathcal{O}(L p \log p)$ which is likely much larger than $\mathcal{O}(p(n, \epsilon))$: for comparison while the VC-dimension (up to constant) of an MNIST network (FC-600-2) with one hidden layer is about 66 million, the value of $p(n, \epsilon)$ is 1982 for $\epsilon = 101.3$. This suggests that even if the hypothesis class of deep networks is very large, sloppiness of the input data severely restricts the set of hypotheses that are accessible during training.

### 3.4 Numerical methods to compute PAC-Bayes bounds

**Method 2: Eigenvectors of the posterior covariance are fixed to those of FIM at initialization.** ($E(\Sigma_q) = E(F_{w_0})$) Our experiments show 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. 3). Similarly, there is a large overlap between the subspace spanned by the stiff eigenvectors of the Hessian with that of the FIM (Fig. 4). Therefore, as the simplest extension of the analytical computation of the posterior in Method 1, we pick the posterior covariance matrix to have the same eigenvectors as that of FIM at initialization and optimize only its eigenvalues, i.e.,

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

(17)

where $F_{w_0} = U_{w_0} \Lambda U_{w_0}^{\top}$ is the orthonormal decomposition of the FIM at initialization $w_0$. We use the notation $E(A)$ to denote the set of eigenvectors of the matrix $A$, arranged in decreasing order of eigenvalues. We can optimize the PAC-bound in (2) numerically. The variables of optimization are the mean of the posterior $w$, eigenvalues of the covariance $\Lambda_w$, and the scale of the prior $\epsilon$. Note that in this method, we do not modify the eigenvectors of the posterior covariance even if the mean changes during optimization.
Method 3: Eigenvectors of the posterior covariance are the same as those of the Hessian, but the Hessian is reevaluated at different instants while optimizing the bound. \( E(\Sigma_q) = E(H_w) \) This method is similar to Method 2 except that we update the eigenvectors of the posterior covariance matrix while optimizing the bound

\[
P = \mathcal{N}(w_0, \epsilon^{-1} I), \quad Q = \mathcal{N}(w, \Sigma_q = U_w \Lambda_w U_w^\top),
\]

where \( H_w = U_w \Lambda_w U_w^\top \) is the orthonormal decomposition of the Hessian at weights \( w \). The variables of optimization here the same, i.e., mean of the posterior \( w \), eigenvalues of the covariance \( \Lambda_w \), and the scale of the prior \( \epsilon \).

Method 4: Prior covariance is proportional to FIM at initialization, eigenvectors of posterior covariance the same as those of FIM at initialization. \( \Sigma_p = a F_{w_0} + \epsilon^{-1} I \); \( E(\Sigma_q) = E(F_{w_0}) \) This is a data-distribution dependent prior. We exploit the fact that the subspace spanned by the stiff eigenvectors of the FIM has a strong overlap with the corresponding subspace at the end of training:

\[
P = \mathcal{N}(w_0, a F_{w_0} + \epsilon^{-1} I), \quad Q = \mathcal{N}(w, \Sigma_q = U_{w_0} \Lambda_w U_{w_0}^\top),
\]

where \( F_{w_0} = U_{w_0} \Lambda_{w_0} U_{w_0}^\top \) is the orthonormal decomposition of the FIM at initialization \( w_0 \). The variables that are modified during optimization of the bound are the posterior mean \( w \), the eigenvalues \( \Lambda_w \), and scalar constants \( a, \epsilon^{-1} \). In this case, we incur a larger penalty from the union bound (see Appendix D.5) because there are two constants being optimized in the prior.

4 Empirical Study

4.1 Setup

We use fully-connected (of varying widths and up to two hidden layers) and convolutional (LeNet, ALL-CNN of Springenberg et al. (2015) and wide residual network of Zagoruyko and Komodakis (2016)) deep networks of varying sizes on MNIST (LeCun et al., 1990) and CIFAR-10 (Krizhevsky, 2009) datasets to study sloppy eigenspectra. PAC-Bayes bounds are computed on MNIST for binary classification problems (digits 0–4 and 5–9, respectively). See Appendices A and D for further details.

Implementation details. We make approximations while working with large matrices like Hessian and FIM. In some cases we compute the full Hessian/FIM, for larger networks we compute only the top 1000–3000 eigenvalues of the Hessian. We also use Kronecker-factor (KFAc) approximation of the Gauss-Newton matrix in Backpack (Dangel et al., 2020) as a replacement for FIM/Hessian; this is the so-called G-term Appendix C. See Appendix D for details. We also exploit a trick in PyTorch (see Appendix E) that allows us to draw a large number of samples to estimate the gradient of \( \hat{e}(Q, D_n) \) more accurately (we use 150, for comparison Dziugaite and Roy (2017) use 1) and thereby optimize the PAC-Bayes bound for much fewer iterations.

4.2 Hessian and FIM are sloppy if the data correlation matrix is sloppy

Eigenspectra for MNIST. Fig. 2 is similar to Fig. 1 and shows the eigenspectra for a two-layer fully-connected network with 600 hidden neurons on MNIST. The two figures are qualitatively similar: MNIST in spite of its lower dimensionality has roughly the same range of eigenvalues but it has a very small threshold \( \epsilon \) 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. 4 indicates the two matrices have a larger overlap in their top eigenvectors. Eigenspectra of other networks on MNIST are similar to Fig. 2 while those of CIFAR-10 are similar to Fig. 1; see Appendix G.1.

\footnote{The code for all experiments in this paper can be found at https://github.com/rubingy/sloppy.git}
Figure 2: 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.

Overlap of the stiff subspace of the FIM/Hessian at the end of training with that at initialization is large. Fig. 3 shows the results of this experiment. 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 error much. We find in Fig. 3 (right) that this variant has a much larger projection into the stiff eigenspace, and thereby a smaller overlap with the sloppy eigenvectors. 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 details.

Figure 3: (Left) Overlap between the subspace of the top $k$ eigenvectors (X-axis) of the FIM at the end of training with that at initialization ($\| E_k(F_w)^\top E_k(F_{w_0}) \|_F^2 / k$) for fully-connected (FC) and convolutional networks (WRN). (Right) Projection $\| E_k(F_{w_0}) \Delta w \|_2^2 / \| \Delta w \|_2^2$ of the change in weights during training (where $\Delta w = w - w_0$) into the sub-space of the top $k$ eigenvectors (shown as percentage because different networks have different size) of the Hessian for these networks is much larger than that of a random vector. Thus weights change predominantly in the stiff eigenspace of the FIM; also see Fig. 4 which shows that the FIM eigenvectors have a strong overlap with those of the Hessian.

A study of the data correlation matrix, FIM and Hessian for atypical problems. 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 $n^{-\frac{1}{2}} \text{diag}(XX^\top) = \Lambda$ 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 $b/c$. 

10
Figure 4: (Left) Eigenspectra of a two-layer fully-connected network on MNIST (same network as Fig. 2) for the FIM, Hessian and a KFAC approximation of the Gauss-Newton matrix. 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. Even if KFAC is a good approximation for the eigenvalues of the Hessian, the eigenvectors computed from KFAC are quite different from those of the Hessian.

Table 1: Comparison of PAC-Bayes bounds on MNIST for different methods. The first four methods correspond to our Methods 1–4 described in §3.2. The prior for Method 4 is $\Sigma_p = a F w_0 + \epsilon^{-1}$; all other methods use $P = \mathcal{N}(w_0, \epsilon^{-1} I)$. The notation $E(A)$ denotes eigenvectors of the matrix $A$. The penultimate column where the posterior covariance is diagonal corresponds to numbers from Dziugaite and Roy (2017). The final column corresponds to the approach of Wu et al. (2021) who set the eigenvectors of the posterior covariance to be the same as those of the layer-wise Hessian. The error $e(Q)$ ranges from $6–8 \times 10^{-2}$ for Method 1 and $1–4 \times 10^{-2}$ for all other methods. All bounds hold with probability at least 0.975. Also see Appendix F.

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 (i) to study how the various quantities discussed in this paper, e.g., the Hessian, FIM, at the beginning or the end of training, depend upon the sloppiness of the data matrix, and (ii) whether the student can interpolate on sloppy datasets without over fitting. Fig. 5 show the results of this experiment.

4.3 PAC-Bayes bounds

Table 1 shows different methods to calculate PAC-Bayes bounds. For all networks, the analytical method using Method 1 obtains a non-vacuous bound. We see that Methods 2–4 obtain bounds that are comparable to these existing methods. The tightest bounds in our case are obtained by Method 4 which uses a data-distribution dependent prior and a posterior that depends upon the FIM at initialization; this method involves inverting the FIM at each step of optimization, and we discuss the technical details of doing so in Appendix D. Our priors and posteriors exploit the fact that the stiff eigenspaces of the Hessian/FIM have a strong overlap at
the beginning and end of training, and the posterior should have smaller variance in stiff directions. The sloppiness of the Hessian $H_w$ leads to the exponentially large range of the distribution of the inverse posterior covariance eigenvalues, which makes it possible to find a posterior with small $\text{KL}(Q, P)$ while still have small training loss $\mathcal{L}(Q, D_n)$, which leads to a non-vacuous generalization bound. This explains why sloppiness can be effective at providing good generalization bounds.

Fig. 6 studies the posterior covariance computed while optimizing the bound. We observe that our Method 3, where the posterior covariance has the same eigenvectors as those of the Hessian (which we prove in §3.2 Method 1 to optimize pac-bayes bound), puts a large variance along sloppy directions of the Hessian. The method of Dziugaite and Roy (2017) can be thought of as assuming a diagonal Hessian. As Fig. 6 (left) shows, the latter approach has a smaller variance in the sloppy subspace, and correspondingly their KL-term and the total bound is worse than ours, even if they have a sharper posterior. This coincides with our calculation in Method 1 that the eigenvectors of the optimal posterior is the same as that of the Hessian $H_w$. It is interesting to note that both methods clearly follow the trend that sloppier eigenvalues correspond to larger variance in the posterior. The inverse eigenvalues of covariance matrix has a strong linear relation with the eigenvalues of Hessian $H_w$, which also coincides with our analytical calculation (12) in Method 1.

5 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 data and show that if the data 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 and Schmidhuber (1997); Chaudhari et al. (2017); Fort and

Figure 5: (Left, Middle) Eigenspectra of the data correlation matrix (blue), FIM (orange for beginning of training, green for the end) and the Hessian at the end of training (red), for sloppy factor $c = 0.1$ (left) and $c = 10^{-3}$ (right). We see that if the data is sloppy, both FIM/Hessian are sloppy but if data is not sloppy then even if there is a sharp drop after the top few eigenvalues (around 100 for middle panel), the eigenspectrum is flat. In comparison, the FIM/Hessian decay by about 3 orders of magnitude on the left. (Right panel) Validation error on synthetic datasets of different sloppiness (X-axis) for different number of hidden neurons (numbers in brackets in the legend) for the two-layer teacher (T) and student networks (S). All students in this plot interpolate the training data perfectly. For isotropic data correlation matrices, i.e., small sloppiness factor, interpolation results in poor generalization. For sloppy data, interpolation is not detrimental to generalization. Also note that 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. 0.3) then we can generalize—roughly equally well in this experiment—even if the student is smaller (10 vs. 500).
Ganguli (2019); Sagun et al. (2016); Dinh et al. (2017), among others. Similarly, FIM has been used to study optimization (Amari, 1998; Martens and Grosse, 2016; Karakida et al., 2019), gradient diversity (Yin et al., 2018; Chaudhari and Soatto, 2018), and also generalization (Amari et al., 2002; Sun and Nielsen, 2020). 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 and 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 observed a strong overlap of the subspace spanned by the stiff eigenvectors of the Hessian/FIM at initialization with the corresponding subspaces at the end of training; this is consistent with the analysis in Gur-Ari et al. (2018) and the literature on the lazy training regime (Chizat et al., 2019).

Generalization. PAC-Bayes bounds for deep networks have been obtained using the methods of Langford and Caruana (2002) by Dziugaite and 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 and 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 and Roy (2018)).

In particular, one of our methods (Method 3) is closely related to the work of Wu et al. (2021). The key 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. While they analyze the case when the data matrix has rank $m - 1$ and Hessian has rank $m - 1$ ($m$ is the number of classes), we are motivated by our experiments and instead consider the setting when both inputs and Hessian are sloppy.

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. 5. Liang and Rakhlin (2018) shows similar results on minimum norm interpolating solution of kernel regression.
6 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, low-dimensional nature, of inputs of high-dimensional datasets directly affects the representations learned by the network. We showcased the benefit of sloppiness by providing non-vacuous PAC-Bayes generalization bounds for deep networks, including analytical ones.

7 Acknowledgements

We would like to thank Anirudh Cowlagi whose project as a first year undergraduate student in the Vagelos Integrated Program in Energy Research (VIPER) undergraduate research program at the University of Pennsylvania gave us the first insights into the connection between the sloppiness of the input correlation matrix and the Fisher Information Matrix. We would also like to thank James P. Sethna and Mark Transtrum for numerous discussions during the conception of this paper.

References

Alessandro Achille, Matteo Rovere, and Stefano Soatto. Critical learning periods in deep networks. In International Conference on Learning Representations, 2018.

Shun-ichi Amari. Natural gradient works efficiently in learning. Neural Computation, 10(2):251–276, 1998. doi: 10.1162/089976698300017746.

Shun-ichi Amari, Hyeyoung Park, and Tomoko Ozeki. Geometrical singularities in the neuromanifold of multilayer perceptrons. Advances in neural information processing systems, 1:343–350, 2002.

Amiran Ambroladze, Emilio Parrado-Hernández, and John Shawe-Taylor. Tighter pac-bayes bounds. Advances in neural information processing systems, 19:9, 2007.

Peter L Bartlett, Dylan J Foster, and Matus J Telgarsky. Spectrally-normalized margin bounds for neural networks. In Advances in Neural Information Processing Systems, pages 6240–6249, 2017.

Peter L Bartlett, Philip M Long, Gábor Lugosi, and Alexander Tsigler. Benign overfitting in linear regression. Proceedings of the National Academy of Sciences, 117(48):30063–30070, 2020.

Peter L Bartlett, Andrea Montanari, and Alexander Rakhlin. Deep learning: A statistical viewpoint. arXiv preprint arXiv:2103.09177, 2021.

Aleksandar Botev, Hippolyt Ritter, and David Barber. Practical Gauss-Newton Optimisation for Deep Learning. In Proceedings of the 34th International Conference on Machine Learning, pages 557–565. PMLR, 2017.

Kevin S Brown, Colin C Hill, Guillermo A Calero, Christopher R Myers, Kelvin H Lee, James P Sethna, and Richard A Cerione. The statistical mechanics of complex signaling networks: Nerve growth factor signaling. Physical biology, 1(3):184, 2004.

Pratik Chaudhari and Stefano Soatto. 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.

Pratik Chaudhari, Anna Choromanska, Stefano Soatto, Yann LeCun, Carlo Baldassi, Christian Borgs, Jennifer Chayes, Levent Sagun, and Riccardo Zecchina. Entropy-SGD: Biasing gradient descent into wide valleys. In Proc. of the International Conference on Learning Representations (ICLR), 2017.

Lénaïc Chizat, Edouard Oyallon, and Francis Bach. On lazy training in differentiable programming. Advances in Neural Information Processing Systems, 32:2937–2947, 2019.
Felix Dangel, Frederik Kunstner, and Philipp Hennig. BackPACK: Packing more into backprop. In *International Conference on Learning Representations*, 2020.

Laurent Dinh, Razvan Pascanu, Samy Bengio, and Yoshua Bengio. Sharp minima can generalize for deep nets. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 1019–1028. PMLR, August 2017.

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

Gintare Karolina Dziugaite and Daniel M. Roy. 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.

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

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

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

Ryan N Gutenkunst, Joshua J Waterfall, Fergal P Casey, Kevin S Brown, Christopher R Myers, and James P Sethna. Universally sloppy parameter sensitivities in systems biology models. *PLoS computational biology*, 3(10):e189, 2007.

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

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

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

Frederik Kunstner, Philipp Hennig, and Lukas Balles. Limitations of the empirical Fisher approximation for natural gradient descent. In *Advances in Neural Information Processing Systems*, pages 4156–4167, 2019.

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

John Langford and Matthias Seeger. Bounds for averaging classifiers. 2001.

Yann LeCun, Bernhard E Boser, John S Denker, Donnie Henderson, Richard E Howard, Wayne E Hubbard, and Lawrence D Jackel. Handwritten digit recognition with a back-propagation network. In *Advances in Neural Information Processing Systems*, pages 396–404, 1990.

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

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

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

Behnam Neyshabur, Srinadh Bhojanapalli, David McAllester, and Nathan Srebro. Exploring Generalization in Deep Learning. *arXiv:1706.08947 [cs]*, July 2017.

Vardan Papyan. The full spectrum of deepnet hessians at scale: Dynamics with sgd training and sample size. *arXiv preprint arXiv:1811.07062*, 2018.
Vardan Papyan. Measurements of three-level hierarchical structure in the outliers in the spectrum of deepnet hessians. *arXiv preprint arXiv:1901.08244*, 2019.

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

Jeffrey Pennington and Yasaman Bahri. Geometry of Neural Network Loss Surfaces via Random Matrix Theory. page 9.

Levent Sagun, Leon Bottou, and Yann LeCun. Singularity of the hessian in deep learning. *arXiv:1611:07476*, 2016.

Jost Tobias Springenberg, Alexey Dosovitskiy, Thomas Brox, and Martin Riedmiller. Striving for Simplicity: The All Convolutional Net. *arXiv:1412.6806 [cs]*, April 2015.

Ke Sun and Frank Nielsen. Lightlike Neuromaniolds, Occam’s Razor and Deep Learning. *arXiv:1905.11027 [cs, stat]*, February 2020.

Mark K. Transtrum, Benjamin B. Machta, and James P. Sethna. 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.

Joshua J. Waterfall, Fergal P. Casey, Ryan N. Gutenkunst, Kevin S. Brown, Christopher R. Myers, Piet W. Brouwer, Veit Elser, and James P. Sethna. Sloppy-Model Universality Class and the Vandermonde Matrix. *Physical Review Letters*, 97 (15):150601, October 2006. doi: 10.1103/PhysRevLett.97.150601.

Yikai Wu, Xingyu Zhu, Chenwei Wu, Annie Wang, and Rong Ge. Dissecting Hessian: Understanding Common Structure of Hessian in Neural Networks. *arXiv:2010.04261 [cs, stat]*, June 2021.

Dong Yin, Ashwin Pananjady, Max Lam, Dimitris Papailiopoulos, Kannan Ramchandran, and Peter Bartlett. Gradient diversity: A key ingredient for scalable distributed learning. In *International Conference on Artificial Intelligence and Statistics*, pages 1998–2007. PMLR, 2018.

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

Wenda Zhou, Victor Veitch, Morgane Austern, Ryan P Adams, and Peter Orbanz. 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 55,000 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. 3.** 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:

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

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 $\hat{c}(Q, D_n)$ 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 and Roy (2017) (which we denote as diag($\Sigma_q$) = $\Lambda$), we train for 300 epochs for the second phase with decaying learning rate.

**Atypical problems.** For atypical problems in Fig. 5, 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
\[ y_i = \argmax_{y \in [m]} P_{t_w}^i(y|x_i), \] where \( P_{t_w}^i \) 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 \( 10^{-5} \).

**B Calculation of the effective dimensionality of a deep network**

**B.1 PAC-Bayes bound in (2)**

**Theorem 10 (PAC-Bayes generalization bound McAllester (1999); Langford and Seeger (2001)).** For every \( \delta > 0, n \in \mathbb{N} \), distribution \( \mu \) 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 \( \mathcal{H} \),

\[
\text{KL}(Q, P) + \frac{n}{n-1} \log \frac{\text{Vol}(Q)}{\text{Vol}(P)}
\]

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

\[
2(q - p)^2 \leq \text{KL}(q, p).
\]

We can invert this inequality to get

\[
\text{KL}^{-1}(q, p) \leq q + \sqrt{p/2}.
\]

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

\[
e(Q) \leq \dot{e}(Q) + \sqrt{\text{KL}(Q, P) + \log \frac{n}{\delta}}.
\]

Since

\[
1 \{ y_i \neq \argmax_y P_w(y|x_i) \} \leq -\frac{1}{\log 2} \log P_w(y_i|x_i)
\]

we also have

\[
\dot{e}(Q) \leq \ddot{e}(Q).
\]

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

\[
e(Q) \leq \dot{e}(Q) + \sqrt{\text{KL}(Q, P) + 2 \log (b \log \frac{\epsilon}{\delta}) + \log \frac{\pi^2 n}{64 \delta^2}},
\]

holds with probability \( 1 - \delta \).

**B.2 Calculation for the closed form expression for eigenvalues of the inverse posterior covariance in (12)**

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

\[
\text{KL}(Q, P) = \frac{1}{2} \left( \text{tr}(\Sigma^{-1}_p \Sigma_q) - P + (\mu_p - \mu_q)^\top \Sigma^{-1}_p (\mu_p - \mu_q) + \log \left( \frac{\det \Sigma_p}{\det \Sigma_q} \right) \right).
\]

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

\[
\text{minimize} \quad L(\Sigma_q) := \dot{e}(Q, D_n) + \frac{\text{KL}(Q, P)}{2(n-1)}
\]

such that \( Q = N(w, \Sigma_q) \)

and \( \Sigma_q \geq 0 \).
Observe that
\[
\tilde{e}(h_{w'}, D_n) = \tilde{e}(h_w, D_n) + \frac{1}{2} \langle w' - w, H_w(w' - w) \rangle.
\]

For \( P_w = N(w, \epsilon^{-1} I) \), we have
\[
\text{KL}(Q, P) = \text{KL}(Q, P_w) + \frac{\epsilon}{2} \| w - w_0 \|^2.
\]

Hence,
\[
L(\Sigma_q) = \int Q(w') \tilde{e}(h_{w'}, D_n) \, dw' + \frac{1}{2(n - 1)} \int Q(w') \log \frac{Q(w')}{P_w(w')} \, dw' + \frac{\epsilon}{4(n - 1)} \| w - w_0 \|^2
\]
\[
= \frac{1}{2(n - 1)} \int \left( - \log \exp(-2(n - 1)\tilde{e}(w', D_n)) + \log \frac{Q(w')}{P_w(w')} \right) Q(w') \, dw' + \frac{\epsilon}{4(n - 1)} \| w - w_0 \|^2
\]
\[
= \frac{1}{2(n - 1)} (\text{KL}(Q, B) - \log Z) + \frac{\epsilon}{4(n - 1)} \| w - w_0 \|^2,
\]
where we have defined
\[
B(w') = \exp(-2(n - 1)\tilde{e}(w', D_n))P_w(w')/Z, \quad \text{and} \quad Z = \int \exp(-2(n - 1)\tilde{e}(w', D_n))P_w(w') \, dw'.
\]

We can now see that \( L(\Sigma_q) \) attains a minimum when
\[
Q = B \propto \exp(-2(n - 1)\tilde{e}(w', D_n))P_w(w') \quad \text{(S-21)}
\]
or \( \Sigma_q^{-1} = 2(n - 1)H_w + \epsilon I \), in other words,
\[
\Sigma_q = U_w(\tilde{\Lambda}_w)^{-1}U_w^\top,
\]
where
\[
\tilde{\lambda}_i = 2(n - 1)\lambda_i + \epsilon \quad \forall i \leq p.
\]

### B.3 Calculation for (15)

We define 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.,
\[
p(n, \epsilon) = \sum_{i=1}^{p} \mathbf{1} \left\{ |\lambda_i| \geq \frac{\epsilon}{2(n - 1)} \right\}.
\]

We assume that eigenvalues of the Hessian decay according to the following pattern after the above threshold
\[
\lambda_i \leq \frac{\epsilon}{2(n - 1)} \exp(-c(i - p(n, \epsilon)))
\]
for some constant \( c > 0 \). We can also assume a weaker version of this decay pattern,
\[
\sum_{i=p(n, \epsilon)+1}^{p} \lambda_i = \frac{\epsilon}{2(n - 1)c}.
\]
We approximate the training objective in the neighborhood of \( w \) as
\[
\hat{e}(h_w, D_n) = \hat{e}(h_w, D_n) + \frac{1}{2} \langle w' - w, H_w(w' - w) \rangle.
\]
In §3.2, for the posterior \( Q = N(w, \Sigma_q) \) that maximizes the loose version of the PAC-Bayes bound (9), where
\[
\Sigma_q = U_w \Lambda_w^{-1} U_w^T, \\
\bar{\lambda}_i = 2(n - 1) \lambda_i + \epsilon.
\]
We can now calculate
\[
\hat{e}(Q, D_n) - \hat{e}(h_w, D_n) = \frac{1}{2} \sum_{i=1}^{p} \frac{\lambda_i}{\bar{\lambda}_i} \\
\leq \frac{p(n, \epsilon) + 2\epsilon}{4(n - 1)} \\
= \mathcal{O} \left( \frac{p(n, \epsilon)}{n} \right), \text{ and}
\]
\[
\frac{\text{KL}(Q, P)}{2(n - 1)} = \frac{1}{4(n - 1)} \left( \epsilon \|w - w_0\|^2 - p + \sum_{i=1}^{p} \log \frac{\bar{\lambda}_i}{\epsilon} + \frac{\epsilon}{\lambda_i} \right) \\
\leq \frac{1}{4(n - 1)} \left( \epsilon \|w - w_0\|^2 + \sum_{i=1}^{p(n, \epsilon)} \log \frac{2(n - 1) \lambda_i}{\epsilon} + 1 \right) + \sum_{i=p(n, \epsilon) + 1}^{p} \frac{2(n - 1) \lambda_i}{\epsilon} \\
\leq \frac{1}{4(n - 1)} \left( \epsilon \|w - w_0\|^2 + \sum_{i=1}^{p(n, \epsilon)} \log \left( \frac{2(n - 1) \lambda_i}{\epsilon} + 1 \right) + \frac{1}{c} \right) \\
= \mathcal{O} \left( \frac{p(n, \epsilon)}{n} \right)
\]
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 \( \mathbb{E} \) in this section to denote the expectation with respect to the point mass distribution of input data \( x_i \) for \( i = 1, \ldots, n \) where \( n \) is the number of samples.

Proof of Lemma 1. For the first inequality in (4), observe that
\[
\text{tr} \left( \mathbb{E} \left[ h^k h^k^T \right] \right) \leq \sum_{j=1}^{d_k} \mathbb{E} \left[ \sigma(u_j^k)^2 \right] \\
\leq a^2 \sum_{j=1}^{d_k} \mathbb{E} \left[ (u_j^k)^2 \right] \\
= a^2 \text{tr} \left( \mathbb{E} \left[ u^k u^k^T \right] \right) \\
= a^2 \text{tr} \left( \mathbb{E} \left[ (w^{k-1} h^{k-1}) (w^{k-1} h^{k-1}^T) \right] \right) \\
= a^2 \text{tr} \left( w^{k-1} \mathbb{E} \left[ h^{k-1} h^{k-1}^T \right] w^{k-1}^T \right) \\
\leq a^2 \|w^{k-1}\|^2 \text{tr} \left( \mathbb{E} \left[ h^{k-1} h^{k-1}^T \right] \right).
\]
For the second inequality in (5), observe that

\[
\frac{dz_i}{dh^k} = \frac{dz_i}{d\mu^{k+1}} w^k
\]

\[
= a \left( \frac{dz_i}{dh^{k+1}} \right) \mu^{k+1} \geq a \left\| \frac{dz_i}{dh^{k+1}} \right\|_2 \|w^k\|_2.
\]

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 \( z_i, i = 1, \ldots, m \)

\[
\text{tr} \left( \mathbb{E} \left[ \frac{dz_i}{dw^k} \frac{dz_i}{dw^k} \right] \right) \leq a^{2L} \text{tr} \left( \mathbb{E} \left[ xx^T \right] \right) \prod_{j=0, j \neq k}^L \|w^j\|_2^2.
\]

for \( k = 0, \ldots, L \). As a result,

\[
\text{tr} \left( \mathbb{E} \left[ \frac{dz_i}{dw^k} \frac{dz_i}{dw^k} \right] \right) \leq a^{2L} \text{tr} \left( \mathbb{E} \left[ xx^T \right] \right) \prod_{j=0, j \neq k}^L \|w^j\|_2^2 \left( \sum_{j=0}^L \frac{1}{\|w^j\|_2} \right).
\]

**Proof of Lemma 11.** The proof follows via an application of Lemma 1. For \( k = 0, 1, \ldots, L - 1 \),

\[
\text{tr} \left( \mathbb{E} \left[ \frac{dz_i}{dw^k} \frac{dz_i}{dw^k} \right] \right) = \text{tr} \left( \mathbb{E} \left[ \frac{dz_i}{d\mu^{k+1}} \frac{dz_i}{d\mu^{k+1}} \right] \otimes h^k h^{k+1} \right)
\]

\[
= \mathbb{E} \left[ \text{tr} \left( \frac{dz_i}{d\mu^{k+1}} \frac{dz_i}{d\mu^{k+1}} \right) \text{tr} \left( h^k h^{k+1} \right) \right]
\]

\[
\leq a^2 \left\| \frac{dz_i}{d\mu^{k+1}} \right\|_2^2 \text{tr} \left( \mathbb{E} \left[ h^k h^{k+1} \right] \right)
\]

\[
\leq a^2 \left\| \frac{dz_i}{d\mu^L} \right\|_2^2 \left( \prod_{j=k+1}^{L-1} \|w^j\|_2^2 \right) \left( \sum_{j=0}^{L-k-1} \frac{1}{\|w^j\|_2} \right) a^{2(L-k-1)}
\]

\[
a^{2k} \prod_{j=0}^{k-1} \|w^j\|_2^2 \text{tr} \left( \mathbb{E} \left[ xx^T \right] \right)
\]

\[
\leq a^{2L} \text{tr} \left( \mathbb{E} \left[ xx^T \right] \right) \prod_{j=0, j \neq k}^L \|w^j\|_2^2.
\]

The third line comes from the fact that the matrix \( \frac{dz_i}{d\mu^{k+1}} \frac{dz_i}{d\mu^{k+1}} \) is rank one and its trace is the same as 2-norm. The last inequality comes from the fact that \( \|w^L\|_2 \leq \|w^L\|_2 \). For \( k = L \),

\[
\text{tr} \left( \mathbb{E} \left[ \frac{dz_i}{dw^L} \frac{dz_i}{dw^L} \right] \right) = \text{tr} \left( \mathbb{E} \left[ \frac{dz_i}{dw^L} \frac{dz_i}{dw^L} \right] \right)
\]

\[
= \text{tr} \left( \mathbb{E} \left[ h^L h^{L+1} \right] \right)
\]

\[
\leq a^{2L} \text{tr} \left( \mathbb{E} \left[ xx^T \right] \right) \prod_{j=0}^{L-1} \|w^j\|_2^2.
\]
Proof of Theorem 2. We first calculate an inequality for the Fisher Information Matrix (FIM)

\[ F = \mathbb{E} \left[ \sum_{y=1}^{m} p_w(y \mid x) (\partial_{w} \log p_w(y \mid x)) (\partial_{w} \log p_w(y \mid x))^\top \right] \]

\[ = \mathbb{E} \left[ \partial_{w} \left[ \sum_{y=1}^{m} p_w(y \mid x) \frac{d \log p_w(y \mid x)}{dz} \frac{d \log p_w(y \mid x)}{dz} \right] \partial_{w} \right] \]

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

\[ p_y \equiv p_w(y \mid x) = \frac{e^{z_y}}{\sum_{y'} e^{z_{y'}}} \]

we have

\[ \frac{d}{dz} \log p_w(y \mid x) = e_y - p \]

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

\[ \sum_{y=1}^{m} p_w(y \mid x) \frac{d \log p_w(y \mid x)}{dz} \frac{d \log p_w(y \mid x)}{dz} \leq \sum_{y=1}^{m} p_w(y \mid x) \left\| \frac{d \log p_w(y \mid x)}{dz} \right\|^2 \bar{I} \]

\[ = (1 - \|p\|_2^2) \bar{I} \]

\[ \leq \bar{I} \]

Hence we have

\[ F \leq \mathbb{E} \left[ (\partial_{w} z) (\partial_{w} z)^\top \right]. \]

In the case of the Hessian for the cross-entropy loss we make a similar calculation following the calculation of Fort and 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

\[ (\log 2) H \approx \mathbb{E} \left[ (\partial_{w} z) \left( \nabla^2_z \log p_w(y \mid x) \right) (\partial_{w} z)^\top \right] \]

\[ \leq \mathbb{E} \left[ (\partial_{w} z) \left( \text{diag}(p) - pp^\top \right) (\partial_{w} z)^\top \right] \]

\[ \leq \mathbb{E} \left[ (\partial_{w} z) \left( \text{diag}(p) \right) (\partial_{w} z)^\top \right] \]

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

\[ \mathbb{E} \left[ \sum_{i=1}^{m} (y_k - p_k) \frac{\partial^2 z_i}{\partial w_{a_i} \partial w_{b_i}} \right] \]

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). Hence, both \( \text{tr} (F) \) and \( (\log 2) \text{tr} (H) \) can be bounded by

\[ \text{tr} (F), (\log 2) \text{tr} (H) \leq \sum_{i=1}^{m} \mathbb{E} \left[ \frac{dz_i}{dw} \frac{dz_i}{dw} \right] \leq ma^2 L \text{tr} (\mathbb{E} \left[ xx^\top \right]) \prod_{j=0}^{L} \left\| w_j \right\|_2^2 \left( \sum_{j=0}^{L} \left\| w_j \right\|_2^2 \right). \] (S-23)

Notice that the log 2 factor in front of \( \text{tr}(H) \) comes from the rescaling factor in the definition of \( \hat{c}(h_w, D_n) \).  

\[ \square \]
Remark 12. We realized while finalizing this proof that the expression in (S-23) is the corrected version of Lemma 3 in the main paper which had a mistake in term that contains the products of the weight matrices. We will fix this error in the final version of the paper.

Remark 13. Empirically, the trace of FIM and Hessian at the end of training (Fig. 4) is usually much smaller than the trace of correlation matrix of logit Jacobians (Fig. S-8). In this case, both $1 - \|p\|_2^2$ and $\text{diag}(p) - pp^T$ 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 3.

$$
\mathbb{E} \left[ \frac{dz_i}{dw^k} \frac{dz_i}{dw^k}^\top \right] = \mathbb{E} \left[ \left( \frac{dz_i}{dh^{k+1}} \right) \left( \frac{dz_i}{dh^{k+1}} \right) \right] = \mathbb{E} \left[ \frac{dz_i}{dh^{k+1}} \frac{dz_i}{dh^{k+1}} \right] = \mathbb{E} \left[ \frac{dz_i}{dh^{k+1}} \frac{dz_i}{dh^{k+1}} \right] = a^2 L \prod_{j=k+1}^L \|w_j\|^2 \mathbb{E} \left[h^k h^k \right] = a^2 L \prod_{j=k+1}^L \|w_j\|^2 \mathbb{E} \left[h^k h^k \right]
$$

Hence, by (8)

$$
\text{spec} \left( \mathbb{E} \left[ \frac{dz_i}{dw^k} \frac{dz_i}{dw^k}^\top \right] \right) \leq \text{spec} \left( a^2 L \prod_{j=k+1}^L \|w_j\|^2 \mathbb{E} \left[h^k h^k \right] \right)
$$

so we have

$$
\text{spec} \left( \mathbb{E} \left[ \frac{dz_i}{dw^k} \frac{dz_i}{dw^k}^\top \right] \right) \leq a^2 L \prod_{j=k+1}^L \|w_j\|^2 \text{spec} \left( \mathbb{E} \left[h^k h^k \right] \right)
$$

Calculations of Remark 5. As we discussed in Remark 5, the expression in (7) gives a very loose bound when $d_k$ is large. Write $\mathbb{E} \left[ \frac{dz_i}{dw^k} \frac{dz_i}{dw^k}^\top \right]$ using KFAC approximation

$$
\mathbb{E} \left[ \frac{dz_i}{dw^k} \frac{dz_i}{dw^k}^\top \right] \approx \mathbb{E} \left[ \frac{dz_i}{dw^{k+1}} \frac{dz_i}{dw^{k+1}}^\top \right] \mathbb{E} \left[ h^k h^k \right].
$$

If spec $\left( \mathbb{E} \left[ \frac{dz_i}{dw^{k+1}} \frac{dz_i}{dw^{k+1}}^\top \right] \right)$ decays as $\exp\{c_1 i\}$ and spec $\left( \mathbb{E} \left[ h^k h^k \right] \right)$ decays as $\exp\{-c_2 j\}$, then the $(i + j)^{th}$ largest eigenvalue of $\mathbb{E} \left[ \frac{dz_i}{dw^k} \frac{dz_i}{dw^k}^\top \right]$ is smaller than $\exp\{-c_1 c_2 (i + j)\}$, hence the $k^{th}$ largest eigenvalue of $\mathbb{E} \left[ \frac{dz_i}{dw^k} \frac{dz_i}{dw^k}^\top \right]$ is smaller than $\exp\{-c_1 c_2 \sqrt{k}\}$. Hence, the decay rate of spec $\left( \mathbb{E} \left[ \frac{dz_i}{dw^k} \frac{dz_i}{dw^k}^\top \right] \right)$ is $O \left( \exp\{-c_1 c_2 \sqrt{k}\} \right)$. 

\]
**Corollary 14.** Denote the FIM and Hessian w.r.t the kth layer \(F(w_k), H(w_k)\) respectively, then we have,

\[
\text{spec}(F(w_k)) , \text{spec}((\log 2)H(w_k)) \leq ma^{2(L-k)} \prod_{j=k+1}^{L} \|w_j\|^2 \text{spec}(I_{d_k+1}) \otimes \text{spec} \left( \mathbb{E}[h^k h^k]^T \right).
\]

As in Lemma 3, \(\prod_{j=L+1}^{L} \|w_j\|^2 = 1\).

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

\[
F(w_k), (\log 2)H(w_k) \leq \mathbb{E} \left[ (\partial_{w_k} z)(\partial_{w_k} z)^T \right]
\]

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

\[
F(w_k), (\log 2)H(w_k) \leq \mathbb{E} \left[ \left( \frac{ds}{dw_k} \right) \left( \frac{ds}{dw_k} \right)^T \right]
\]

\[
\leq ma^{2(L-k)} \prod_{j=k+1}^{L} \|w_j\|^2 \text{spec}(I_{d_k+1}) \otimes \text{spec} \left( \mathbb{E}[h^k h^k]^T \right)
\]

The second inequality comes from a similar calculation as in Lemma 3 for network with one added layer where \(h_{L+1} = u_{L+1} = z, u_{L+2} = w_{L+1} h_{L+1},\) and \(w_{L+1} = [1,...,1], \|w_{L+1}\|^2 = m\).

\[\square\]

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

We optimize the problem,

\[
\min \hat{e}(Q, D_n) + \sqrt{\frac{\text{KL}(Q, P)}{2(n-1)} + \varphi}
\]

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-20), the KL-divergence is

\[
\frac{1}{2} \left( \text{tr}(\Sigma_p^{-1}\Sigma_q) - p + (w - w_0)^\top \Sigma_p^{-1}(w - w_0) + \log(\det \Sigma_p / \det \Sigma_q) \right)
\]

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

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

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 \(\hat{e}(Q, D_n)\) using Monte Carlo samples from \(Q\). After the optimization process, we calculate the PAC-Bayes bound on \(\hat{e}(Q, D_n)\) using

\[
\text{kl}(\hat{e}(Q, D_n), e(Q)) \leq \frac{\text{KL}(Q, P) + \varphi}{n-1},
\]

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

24
D.1 Method 1

The tightest bound in this case is obtained using the v2 model described in Fig. 3 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 (11) and (12). For \( \epsilon = c \exp(j/b) \) and \( j = 1, \ldots, 60 \), we evaluate \( KL(Q, P) \) by using (S-20), and \( \hat{e}(Q, D_u) \) 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 (2) and we choose the smallest bound among all choices of \( \epsilon \).

We also set \( \Sigma_q = U_w \Lambda_w U_w^T \) and calculate \( \hat{\Lambda} \) by directly minimizing (2) where the variables of optimization are \( \lambda_i \) 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 (Numerical) in Table S-2.

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-2.

D.2 Methods 2 and 3

We choose \( P \) and \( Q \) as described in §3.4. We set \( \epsilon^{-1} = \exp(2 \rho) \), \( \hat{\Lambda}_w = \exp(2 \xi) \). The parameters \( \rho \), \( w \), \( \xi \) are optimized while minimizing the PAC-Bayes upper bound. We initialize \( \epsilon^{-1} \) at \( \exp(-6) \) and \( \hat{\Lambda}_w \) 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 \( \hat{e}(Q, D_u) \) 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 §3.4. 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 \( (a F_{w_0} + \epsilon)/10 \). In this case,

\[
KL(Q, P) = \frac{1}{2} \left( \sum_i \frac{\sigma_i}{a \lambda_i^F + \epsilon^{-1}} - d + (w - w_0)^T (a F_{w_0} + \epsilon^{-1})^{-1} (w - w_0) + \sum_i \log \frac{a \lambda_i^F + \epsilon^{-1}}{\sigma_i} \right)
\]

where \( \lambda_i^F \) are eigenvalues of \( F_{w_0} \). For fully-connected networks and LeNet, we approximate \( (w - w_0)^T (a F_{w_0} + \epsilon^{-1})^{-1} (w - w_0) \) 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

\[
E = (w - w_0)^T (a F_{w_0} + \epsilon^{-1})^{-1} (w - w_0).
\]

In Methods 2 and 3, we need to sample from a posterior of the form \( N(0, U_\Lambda U^T) \) 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 $k$th layer $w_k \in \mathbb{R}^{d_k \times d_k}$, 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 $U_{Ak}, U_{Bk}$ 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

$$E^k = (w_k - w_0^k)^\top (a(F_{w_0^k}) + \epsilon^{-1})^{-1} (w_k - w_0^k)$$

$$= (w_k - w_0^k)^\top U_k (a\Lambda^k + \epsilon^{-1})^{-1} U_k^\top (w_k - w_0^k)$$

$$= \left((w_k - w_0^k)^\top U_k (a\Lambda^k + \epsilon^{-1})^{-1/2}\right) \left((w_k - w_0^k)^\top U_k (a\Lambda^k + \epsilon^{-1})^{-1/2}\right)^\top$$

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

$$E^{k/2} = (w_k - w_0^k)^\top U_k (a\Lambda^k + \epsilon^{-1})^{-1/2}$$

$$= (U_{Ak}^\top (w_k - w_0^k)U_{Bk}) \odot (a\Lambda^k + \epsilon^{-1})^{-1/2}$$

where in the last line, $(w_k - w_0^k)^k \in \mathbb{R}^{d_k \times d_k}$. We use $\odot$ to denote element wise multiplication. $U_{Ak}^\top (w_k - w_0^k)U_{Bk}$ 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 $\sqrt{\Lambda} \odot r_k$ and thereby

$$v_k := U_k \left(\sqrt{\Lambda} \odot r_k\right) = U_{Ak} \left(\sqrt{\Lambda} \odot r_k\right) U_{Bk}^\top.$$ 

The final sample is therefore $w + [v_1, \ldots, v_k]$ 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

$$\Sigma = U_1 \Lambda_1 U_1^\top + U_2 \Lambda_2 U_2^\top$$

where $I = U_1 U_1^\top + U_2 U_2^\top$.

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.

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

$$= (w - w_0)^\top U_1 (a\Lambda_1 + \epsilon_1^{-1})^{-1} U_1^\top (w - w_0) + \epsilon_2 \left(\|w - w_0\|^2 - (w - w_0)^\top U_1 U_1^\top (w - w_0)\right)$$

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

$$\nu_1 = U_1 \sqrt{\Lambda_1} U_1^\top r$$

$$\nu_2 = \epsilon^{-1/2} U_2 U_2^\top r = \epsilon^{-1/2} (r - U_1 U_1^\top r)$$

26
Notice that $U_1 U_1^\top r$, 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 $a_i = (1/\epsilon^i) \exp(-j^i/b^i)$ 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 $1 - \frac{6\delta}{\pi^2}$, 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 $\log \frac{\pi}{\delta'}$, hence, using the relation

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

we add the penalty

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

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

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

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 and 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/inputs.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.

F Full results of PAC-Bayes generalization bounds

We display the extended version of the results of PAC-Bayes bound optimization in Table S.2. Methods 1 and 5 give bounds that are similar to each other: this shows that our analytical expression (11) for the optimal
```python
def del_attr(obj, names):
    # names: one name in the list names_all, a.b.c, split by "."
    # list of format names = [a,b,c]
    # delete the attribute obj.a.b.c
    if len(names) == 1:
        delattr(obj, names[0])
    else:
        del_attr(getattr(obj, names[0]), names[1:])

def set_attr(obj, names, val):
    # names: one name in the list names_all, a.b.c, split by ".",
    # list of format names = [a,b,c],
    # set the attribute obj.a.b.c to val if obj.a.b.c is nn.Parameter
    if len(names) == 1:
        setattr(obj, names[0], val)
    else:
        set_attr(getattr(obj, names[0]), names[1:], val)

def get_names_params(mod):
    # names_all: a list of all names of mod.parameters of type
    # [a1,b1,c1, a2,b2,c2, ...]
    # orig_params: tuple of parameters of type nn.Parameter
    orig_params = tuple(mod.parameters())
    names_all = []
    for name, p in list(mod.named_parameters()):
        names_all.append(name)
    return orig_params, names_all

def load_weights(mod, names_all, params):
    for name, p in zip(names_all, params):
        setattr(mod, name.split(".")[-1], p)

class bayesian_nn(nn.Module):
    def __init__(self, c, args, ns=1):
        # c: class of the model and args
        # ns: number of Monte Carlo samples
        super().__init__()
        self.w = c(*args)
        self.mu_std = nn.ModuleList([c(*args), c(*args)])
        self.ns, self.args = ns, args
        orig_params_w, self.names = get_names_params(self.w)

    def forward(self, x):
        ys = []
        for _ in range(self.ns):
            for name, m, v in zip(self.names, 
                                   list(self.mu_std[0].parameters()),
                                   list(self.mu_std[1].parameters()):
                r = torch.randn_like(m).mul(torch.sqrt(torch.exp(2*v))).add(m)
                del_attr(self.w, name.split("."))
                setattr(self.w, name.split(".")[-1], r)
                ys.append(self.w(x))
        return torch.stack(ys)
```

Figure S-7: Code for Bayesian neural networks
posterior using a loose PAC-Bayes bound (9) under the assumption that the loss is quadratic at the weights at the end of training is an accurate estimate of the optimal posterior (2). 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. 4). 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 and Roy (2017). This is because the approximation
\[
\bar{e}(h_w', D_n) = \bar{e}(h_w, D_n) + \frac{1}{2} \langle w' - w, H_w (w' - w) \rangle.
\]
as we discussed in §3.2 may not be an accurate estimate of $\bar{e}(w', D_n)$ 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-21). 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 $E(\Sigma_q) = H_w$ (Method 3, which is similar to Wu et al. (2021)) works better than a diagonal posterior $E(\Sigma_q) = \Lambda$ (which is the method in Dziugaite and Roy (2017)); this coincides with our calculation in Method 1 (see §3.2) that the eigenvectors of the optimal posterior is the same as that of the Hessian $H_w$.

G Further experimental studies

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

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 plotted 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. 6 using FC-1200-1 on MNIST.

![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.](image-url)
Table S-2: Comparison of PAC-Bayes bounds on MNIST for different methods. This table is an expansion of Table 1. The first 4 blocks corresponds to our Methods 1–4 described in § 3.2. The 5th block is our reproduction of Dziugaite and Roy (2017), the 6th, 7th and 8th block corresponds to the three methods of constructing posterior for PAC-Bayes bound without training described in Appendix D.1, the last two blocks are statistics from Dziugaite and Roy (2017) and Wu et al. (2021).
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. 2 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. 6 for FC-1200-1 (Fig. 6 is for FC-600-2). We get \( n \sim 30000 \) (true \( n = 55000 \)) and \( \epsilon \sim 138.2 \) (true \( \epsilon = 53.3 \)).

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.

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 \( \|v_k(F_{w_0})\Delta w\|_2^2 / \|\Delta w\|_2^2 \) of the change in weights during training onto the eigenvector \( v_k(F_{w_0}) \) 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.