A Reparametrization-Invariant Sharpness Measure Based on Information Geometry

Cheongjae Jang
Hanyang University
cjjang@hanyang.ac.kr

Sungyoon Lee
Korea Institute for Advanced Study
sungyoonlee@kias.re.kr

Frank C. Park
Seoul National University
Saige Research
fcp@snu.ac.kr

Yung-Kyun Noh
Hanyang University
Korea Institute for Advanced Study
nohyung@hanyang.ac.kr

Abstract

It has been observed that the generalization performance of neural networks correlates with the sharpness of their loss landscape. Dinh et al. (2017) [8] have observed that existing formulations of sharpness measures fail to be invariant with respect to scaling and reparametrization. While some scale-invariant measures have recently been proposed, reparametrization-invariant measures are still lacking. Moreover, they often do not provide any theoretical insights into generalization performance nor lead to practical use to improve the performance. Based on an information geometric analysis of the neural network parameter space, in this paper we propose a reparametrization-invariant sharpness measure that captures the change in loss with respect to changes in the probability distribution modeled by neural networks, rather than with respect to changes in the parameter values. We reveal some theoretical connections of our measure to generalization performance. In particular, experiments confirm that using our measure as a regularizer in neural network training significantly improves performance.

1 Introduction

From recent discoveries in deep learning, it has been conjectured that the generalization performance of neural networks correlates with the sharpness of their loss landscape. In particular, lower generalization performance of large-batch training of stochastic gradient descent (SGD) has been attributed to its convergence to sharper minima [22, 49], and several optimization methods have shown better generalization performance by actively finding solutions with lower sharpness [7, 19, 4, 43, 12, 24]. Minimum description length (MDL)-based arguments [16, 17] and the generalization upper bound of the PAC-Bayes theory for neural networks [10] also suggest that the flatter the loss landscape, the better the generalization performance. Furthermore, in [21, 32], measures based on the sharpness concept show better performance in predicting the generalization performance of models among various generalization measures considered in the past including the margin [5] and norm [34, 33].

Various measures for sharpness and flatness have been proposed in the literature. Hochreiter & Schmidhuber (1997) [17] propose the concept of 'flat minima,' in which the flatness of a minimum is interpreted as the volume of the connected region in the parameter space over which the loss has approximately similar values. Similarly, the sharpness has often been measured by the maximum loss value (inside a Euclidean ball) near the minimum [22] or by the spectral norm of the Hessian at the minimum [49].

36th Conference on Neural Information Processing Systems (NeurIPS 2022).

The above definitions of sharpness have some notable critical flaws. Dinh et al. (2017) [8] point out that certain parameter scalings (e.g., for neural networks with ReLU activation functions [31]) and reparametrizations may have no effect on the model output or overall generalization performance, and yet lead to wildly different values of sharpness.[1] Other sharpness measures have been proposed to address the lack of scale-invariance [26, 46, 40, 39], but these measures fail to be invariant with respect to reparametrizations; Section 5 of [8] offers several nonlinear reparametrization examples. The connections between sharpness and generalization performance also remain elusive.

In this paper we address the lack of scale- and reparametrization-invariance of existing sharpness measures, as well as the problem that they often do not explain the generalization performance of deep learning networks nor lead to practical use to improve the performance. Our approach rests on the observation and insight that sharpness should be measured taking into proper account the geometry of the underlying parameter space. For this purpose we draw upon tools from information geometry, paying attention to the fact that in most classification problems, probabilistic classification models are parametrized in terms of neural networks. In information geometry, the Fisher information matrix (FIM) serves as a natural Riemannian metric for the parameter space [3] and allows for measuring the change in probability density with respect to changes in the parameter values.

The eigenspectra of the FIM of a neural network are observed to have a small number of positive outliers (the number is usually equated with the number of classes) and a bulk consisting of small eigenvalues [41, 37, 36, 48, 14]. This observation indicates that the number of principal varying components of the probability densities modeled by neural networks is significantly lower compared to the number of parameters. Therefore, identically weighting every possible parameter change direction in the parameter space – that is, applying Euclidean geometry – can be problematic, since it places too much weight on parameter change directions that are meaningless with respect to changes in the probabilistic model, or equivalently, not enough weight is placed along meaningful directions.

We argue that one should consider these implications and use the change in probabilistic models as a 'ruler' when defining a sharpness measure of neural network loss landscapes that does not depend on how the model is parametrized.

Building on the above geometric analysis of the neural network parameter space, in this paper we propose a new sharpness measure based on information geometry. This measure is by definition reparametrization-invariant. Additionally, we show that this sharpness measure simultaneously satisfies the scale-invariance for neural networks with ReLU activation functions, since it is invariant for parameter transforms that do not change the model output. Hence our measure is free from all the reparametrization- and scale-variance issues of previous sharpness measures raised by [8].

As a second contribution of this paper, we also show that our geometric sharpness measure sheds an important insight on generalization performance. As detailed in Section 3, when the FIM is replaced with the Hessian in our measure, the measure is in a form similar to Takeuchi's information criterion (TIC), which quantifies the asymptotic bias of the log-likelihood value (usually the negative loss) evaluated at the maximum likelihood estimates hence corresponding to the expectation of the generalization gap [44]. TIC for neural networks has empirically shown a strong correlation with the generalization gap [45]. One can apply the same argument to our measure since the Hessian can be approximated by the FIM under mild conditions for deep neural networks [28]. In addition, we show that our measure can be linked to generalization in another geometrical way. The measure is associated with a margin defined in an information geometric sense, enabling the interpretation that a smaller sharpness measure indicates larger margins. We also demonstrate experimentally that using our measure as a regularizer to train neural networks can significantly improve generalization performance.

Our contributions can be summarized as follows:

- We provide an information geometric analysis of the neural network parameter space by investigating the eigensubspace of the Fisher information matrix (FIM) of neural networks.
- We propose a reparametrization- and scale-invariant sharpness measure based on information geometry that is free from the problems of existing sharpness measures posed by [8], and discuss the relation between the measure and generalization performance.

[1] In this paper, unlike other works where parameter scalings are often referred to as (linear) reparametrizations, following [8], it is called a reparametrization to represent a model on a different parameter space obtained by applying a (nonlinear) bijection to the original parameters.

- We use the proposed measure as a regularizer when training neural networks, and demonstrate improved generalization performance.

We provide an information geometric analysis of the neural network parameter space in Section 2. Building on this analysis, Section 3 presents our reparametrization-invariant sharpness measure based on information geometry and connects the proposed measure to generalization. In Section 4, we perform numerical experiments using our measure as a regularizer to improve generalization performance in neural network training.

2 An information geometric analysis of the neural network parameter space

In this paper, we focus on the probabilistic classification models parametrized by neural networks. Let $x \in \mathbb{R}^D$ denote data, and $y \in \{1, \dots, C\}$ denote class labels with C as the number of classes. Let $p(x, y)$ and $p(y|x; \theta)$ respectively denote the data generating distribution and the parametric model of the probability density of classes given data. Here the parameter $\theta \in \mathbb{R}^m$ is the set of all the weight and bias parameters of neural networks. We consider a neural network as a function $f(\cdot; \theta) : \mathbb{R}^D \to \mathbb{R}^C, x \mapsto f(x; \theta) = (f_1(x; \theta), \dots, f_C(x; \theta))^\top$, where $f_j(\cdot; \theta) : \mathbb{R}^D \to \mathbb{R}$ is a function that returns the j-th logit for $j = 1, \dots, C$. Our parametric model is then represented as $p(y|x; \theta) = \frac{\exp(f_y(x))}{\sum_j \exp(f_j(x))}$, and we denote the negative log-likelihood by $l(y, f(x; \theta)) = -\log p(y|x; \theta)$.

Suppose data $\{(x_1, y_1), \dots, (x_N, y_N)\}$ are drawn i.i.d. from the data generating distribution $p(x, y)$. The cross-entropy loss, our training objective function throughout the paper, can be written as follows:[2]

$$\mathcal{L}(\theta) = \frac{1}{N} \sum_{i=1}^{N} l_i(\theta), \tag{1}$$

where $l_i(\theta) = l(y_i, f(x_i; \theta)) = -\log p(y_i|x_i; \theta)$.

2.1 The Fisher information matrix (FIM)

The Fisher information matrix (FIM) for the family of probability density functions $p(x, y; \theta) = p(x)p(y|x; \theta)$ parametrized by the neural network parameters θ is defined as

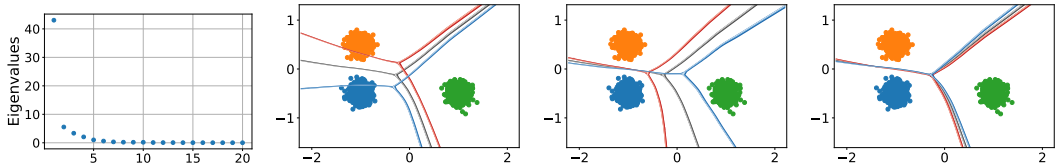
$$\begin{aligned} F(\theta) &= \mathbb{E}_{p(x)} \left[\mathbb{E}_{p(y|x;\theta)} \left[\left(\frac{\partial l}{\partial \theta} \right)^\top \left(\frac{\partial l}{\partial \theta} \right) \right] \right] = \mathbb{E}_{p(x)} \left[J(x)^\top \mathbb{E}_{p(y|x;\theta)} \left[\left(\frac{\partial l}{\partial f} \right)^\top \left(\frac{\partial l}{\partial f} \right) \right] J(x) \right] \\ &= \frac{1}{N} \sum_{i=1}^{N} J_i^\top \left(\mathrm{diag}(p_i) - p_i p_i^\top \right) J_i, \end{aligned} \tag{2}$$

where the dependence of $l(y, f(x; \theta))$ to $y, f(x; \theta)$ is omitted for simplicity and $J(x) = \frac{\partial f}{\partial \theta}(x) \in \mathbb{R}^{C \times m}$ is the Jacobian of the logits with respect to the parameters. In deriving (2), the expectation with respect to $p(x)$ is approximated by the finite sum over data points $x_1, \dots, x_N$ drawn i.i.d. from $p(x)$, $J_i = J(x_i)$, and we use the fact that $\mathbb{E}_{p(y|x_i;\theta)} \left[\left(\frac{\partial l}{\partial f} \right)^\top \left(\frac{\partial l}{\partial f} \right) \right] = \mathrm{diag}(p_i) - p_i p_i^\top$, where $p_i = (p(1|x_i; \theta), \dots, p(C|x_i; \theta))^\top \in \mathbb{R}^C$ is the model's prediction on the probability that x_i belongs to each class and $\mathrm{diag}(p_i) \in \mathbb{R}^{C \times C}$ is a diagonal matrix whose (j, j) entry is $(p_i)_j$.

In this paper, among many of the important properties and applications of the FIM (e.g., being the Cramer-Rao lower bound of estimator variances in estimation theory and statistics), we focus on the fact that the FIM can serve as a metric to measure the distance between the parametric probability density models in information geometry. Differential geometrically speaking, the FIM acts as a natural Riemannian metric on the statistical manifold, a space where the family of probability densities modeled with smoothly varying parameters $\theta \in \mathbb{R}^m$ is gathered.[3] The FIM makes it possible to

[2] We discuss the applicability of our measure to the square loss in Appendix F.

[3] Note that the family of probabilistic models from neural networks is usually not a manifold in a mathematically rigorous sense due to the inherent singularities in the neural network parameter space (see Section 12.2 of [3]), but this fact is not crucial for our discussion.



(a) Top 20 eigenvalues of $F(\theta)$ (b) Perturbed decision boundaries from $\Delta\theta_1$ (c) Perturbed decision boundaries from $\Delta\theta_2$ (d) Perturbed decision boundaries from $\Delta\theta_{10}$

Figure 1: A synthetic three-class classification example. For (b)-(d), the black, red, and blue lines correspond to decision boundaries of the neural network with the trained parameter values, and parameter values perturbed along the k-th eigenvector $\Delta\theta_k \in \mathbb{R}^m$ (associated with the k-th eigenvalue) of $F(\theta)$ with steps of 0.5 and -0.5, respectively.

measure geometric quantities such as length, angle, and volume on the manifold. (We refer the reader to [6, 11] for the backgrounds on the Riemannian manifolds and differential geometry, and [3, 35, 28] for those on the information geometry and the Fisher information matrix.) Note that when the FIM is not full-rank, e.g., in the case of overparameterized neural networks with $m \gg N$, it can be used as a pseudo metric.

The FIM is closely related to the KL-divergence, which is frequently used as an information theoretic measure of discrepancy between probability density functions. The KL-divergence between two probability density functions with infinitesimal parameter difference $d\theta \in \mathbb{R}^m$ can be approximated as

$$\mathrm{KL}(p(x, y; \theta + d\theta) || p(x, y; \theta)) \approx \frac{1}{2} d\theta^\top F(\theta) d\theta. \tag{3}$$

2.2 An analysis of the eigensubspace of the FIM

This section discusses the characteristics of the eigensubspace of FIM of neural networks. In the analysis of the eigenspectra of deep neural network Hessians of [37], the FIM in (2) has been decomposed as follows:

$$F(\theta) = \frac{1}{N} \sum_{i=1}^{N} J_i^\top \left(\mathrm{diag}(p_i) - p_i p_i^\top \right) J_i = \frac{1}{N} \sum_{i=1}^{N} J_i^\top M_i^\top M_i J_i, \tag{4}$$

where $M_i = \mathrm{diag}(\sqrt{p_i}) \left(I - \mathbb{1} p_i^\top \right) \in \mathbb{R}^{C \times C}$, $\sqrt{p_i} = (\sqrt{(p_i)_1}, \ldots, \sqrt{(p_i)_C})^\top \in \mathbb{R}^C$, and $\mathbb{1} = (1, \ldots, 1)^\top \in \mathbb{R}^C$ is a vector whose elements are all one.

According to the decomposition, the FIM can be thought of as a non-centralized second moment of the (modified) logit gradients, i.e., each row of $M_i J_i \in \mathbb{R}^{C \times m}$ whose j-th row vector is represented as

$$(M_i J_i)_{j:}^\top = \sqrt{(p_i)_j} \cdot \left((J_{i,j})^\top - \sum_{j'=1}^{C} (p_i)_{j'} (J_{i,j'})^\top \right) \in \mathbb{R}^m, \tag{5}$$

where $J_{i,j} \in \mathbb{R}^{1 \times m}$ is the j-th row of J_i. It has been observed that these logit gradients form a kind of hierarchical structure after sufficient training. In the observation from [36], the gradients of the $c' \neq c$-th logit calculated from the data in class c are gathered to form a cluster. A matrix obtained from averaging the outer products of the averaged logit gradients belonging to each cluster is then attributed to the outliers of the eigenspectra, and the outliers of the eigenvalues appear as much as the number of classes. That is, the principal eigenvalues of the FIM (and possibly the eigenvectors) can be closely connected with distinguishing each class c from the rest of the $c' \neq c$ classes (as can be implied from averaging (5) for $j \neq c$ and by assuming $(p_i)_c \approx 1$). This becomes more evident in Appendix A, which assumes the prediction is balanced as $(p_i)_{c'} = \epsilon$ for all $c' \neq c$ with $\epsilon \ll 1$.

In order to more intuitively understand the characteristics of the eigensubspace of FIM implied from the above analysis, we provide a classification example for two-dimensional synthetic data generated from mixtures of three Gaussians in Figure 1. We trained a three-layer fully connected neural network with 70-dimensional hidden units (the number of parameters = 5,393).

Figure 1 (a) shows the top twenty eigenvalues of the FIM of the trained neural network. Note that there are few outliers among the eigenvalues in the entire 5,393-dimensional parameter space, with the remainder being almost 0. The eigenvalues indicate the rate of change of the probability density (modeled by the neural network) measured by the KL-divergence for a unit change of parameter values along the corresponding eigenvectors. The existence of a few outliers means that the density change mainly occurs in only a few specific directions and that the amount of change in probability density can vary significantly according to the direction of change in parameter values.

This fact can also be observed in decision boundaries obtained from neural networks with the parameter values perturbed along some eigenvectors shown in Figure 1 (b-d). When perturbing the values along the principal eigenvectors, a significant change appears in the decision boundary, increasing or decreasing the margin of certain classes. On the other hand, for eigenvectors with small eigenvalues, there is almost no change in the decision boundary according to the same level of change in parameter values (in the Euclidean sense).

These examples imply that it would be more meaningful to reflect the different influences on the probabilistic model according to the change direction when we measure the quantities concerning changes in the parameter values, e.g., the sharpness of the loss landscape. By reflecting on these implications and using the unit change in probability distribution modeled by neural networks as our 'ruler' to measure the sharpness, that is, applying the information geometry, the following section presents a sharpness measure that does not rely on how the model is parametrized.

3 An information geometric sharpness measure

Reflecting the above analysis, we define an information geometric sharpness (IGS) measure as follows:

$$\mathrm{IGS}(\theta) = \frac{1}{N} \sum_{i=1}^{N} \left(\frac{\partial l_i}{\partial \theta} \right) F(\theta)^{\dagger} \left(\frac{\partial l_i}{\partial \theta} \right)^{\top}, \tag{6}$$

where $F(\theta)^{\dagger} \in \mathbb{R}^{m \times m}$ is the pseudo-inverse of $F(\theta)$ and $l_i = l(y_i, f(x_i; \theta))$ is the cross-entropy loss evaluated at the i-th data (x_i, y_i).

Our measure evaluates the squared norm of the loss gradients calculated at each data point using the (pseudo-inverse of) FIM and averages them for all data points.[4] Since $\left(\frac{\partial l_i}{\partial \theta}\right)^{\top} \in \mathbb{R}^m$ belongs to the span of the eigenvectors of $F(\theta)$ with non-zero eigenvalues, our sharpness measure is well-defined in the sense that there are not any components of $\left(\frac{\partial l_i}{\partial \theta}\right)$ neglected in measuring its information geometric norm (we elaborate on this in Appendix B).

Our sharpness measure in (6) is an intrinsic quantity, i.e., coordinate-invariant (in differential geometric terms). To see why, observe that under a local coordinate transformation (or a reparametrization) $\phi : \mathbb{R}^m \to \mathbb{R}^m$, $\theta \mapsto \theta' = \phi(\theta)$, i.e., the model is now represented with respect to a different parameter θ' as $\tilde{f}(\cdot; \theta') = f(\cdot; \phi^{-1}(\theta')) = f(\cdot; \theta)$, F and $\left(\frac{\partial l_i}{\partial \theta}\right)$ transform according to the following rules: (i) $F \mapsto F' = \Phi^{-\top} F \Phi^{-1}$, where $\Phi = \frac{\partial \phi}{\partial \theta} \in \mathbb{R}^{m \times m}$; (ii) $\left(\frac{\partial l_i}{\partial \theta}\right) \mapsto \left(\frac{\partial l_i}{\partial \theta'}\right) = \left(\frac{\partial l_i}{\partial \theta}\right) \Phi^{-1}$, where it can be verified that the $\left(\frac{\partial l_i}{\partial \theta}\right) F(\theta)^{\dagger} \left(\frac{\partial l_i}{\partial \theta}\right)^{\top}$ remains the same. Since the above measure gives the same value regardless of which parametrization (e.g., θ or θ') the statistical model is parametrized, it is free from the reparametrization-variant issues raised in [8].

We can also define other reparametrization-invariant measures by measuring the information geometric norm of the gradients of the entire loss or the mini-batch losses. For mini-batches of size b, the corresponding measure can be defined as follows:

$$\mathrm{IGS}_b(\theta) = \mathbb{E}_{p(B)} \left[\left(\frac{1}{b} \sum_{x_i \in B} \frac{\partial l_i}{\partial \theta} \right) F(\theta)^{\dagger} \left(\frac{1}{b} \sum_{x_i \in B} \frac{\partial l_i}{\partial \theta} \right)^{\top} \right], \tag{7}$$

where $\mathbb{E}_{p(B)}[\cdot]$ denotes the expectation with respect to some distribution $p(B)$ of mini-batches $B \subset \{x_1, \ldots, x_N\}$ of size $|B| = b$. We discuss the relation between (6) and (7) in Appendix C. Note

[4] The reason for defining the sharpness measure as in (6) will be evident in Section 3.3 where we develop the relationship between our measure and the generalization.

that this definition can be linked to the concept of m-sharpness, which measures the sharpness of the loss landscape by averaging the sharpness of mini-batch losses and shows a better correlation to the generalization as the batch size reduces [12].

3.1 Transformation invariance

Neural networks composed of activation functions such as ReLU and leaky ReLU possess scale-invariant properties. In the case of layer-wise scaling, for consecutive layers of linear transforms and ReLU, e.g., $f(x; \{W_1, \ldots, W_L\}) = W_L \cdot \mathrm{ReLU}(W_{L-1} \cdots \mathrm{ReLU}(W_1 x))$, where $W_l \in \mathbb{R}^{d_l \times d_{l-1}}$ for $l = 1, \ldots, L$ with $d_0 = D$ and $d_L = C$, if each weight W_l is multiplied by a constant $c_l > 0$ and the constants satisfy $\prod_{l=1}^{L} c_l = 1$, there is no change in the outputs of the neural network under the same input values. A similar concept includes the node-wise scaling, multiplying weights entering a node (or a hidden variable in neural networks) by a positive constant and dividing weights out of the node by the same constant.

These kinds of weight scaling can be viewed as a transformation (or a mapping between identical parameter spaces) that ensures that the neural network model maintains the same probability density function, i.e., remains equivalent. If a transformation is differentiable and locally invertible without changing the probability density function that the neural network parameter models for all parameters on a given neighborhood $U \subseteq \mathbb{R}^m$ of θ, our measures defined in (6) and (7) become invariant with respect to such a transformation. This fact can be expressed as the following proposition:

Proposition 3.1. *Suppose there exist open subsets $U, V \subseteq \mathbb{R}^m$ and an invertible and locally differentiable transformation $g : U \to V$ that satisfies $f(x; \theta) = f(x; g(\theta))$ for all $x \in \mathbb{R}^D$ and $\theta \in U$ with $g(U) \subseteq V$. The information geometric sharpness measures in* (6) *and* (7) *are invariant to such a transformation, i.e.,* $\mathrm{IGS}(\theta) = \mathrm{IGS}(g(\theta))$ *(or* $\mathrm{IGS}_b(\theta) = \mathrm{IGS}_b(g(\theta))$*) for all $\theta \in U$.*

The proof of Proposition 3.1 is provided in Appendix D.1.

Since the layer-wise and node-wise scalings considered for neural networks with ReLU satisfy the assumptions of Proposition 3.1, our measures in (6) and (7) are invariant to such scalings. Proposition 3.1 is more general than the scale-invariance, and Appendix D.2 provides examples of transformations other than parameter scalings that satisfy the assumptions in the proposition.

3.2 A comparison to some previous sharpness measures

After [8] pointed out the critical scale- and reparametrization-variant issues in the sharpness measures considered in [17, 22, 49], various scale-invariant sharpness measures have been proposed that suit to neural networks with activation functions satisfying the non-negative homogeneity conditions (i.e., $\sigma(a \cdot x) = a \cdot \sigma(x)$ for $a > 0$) such as ReLU [40, 39, 46, 26]. The measure presented in [40] is based on the geometry of the quotient manifold obtained from an equivalence class of neural networks, establishing the scale-invariance depending on the considered equivalence types. The work in [39] considers a layer-wise flatness measure for neural networks and additionally explores the conditions for the measure to explain the generalization well. Also, a scale-invariant sharpness measure based on the PAC-Bayes theory is developed in [46]. The Fisher-Rao norm is proposed as a capacity measure for neural networks in [26] and defined as $\theta^\top F(\theta)\theta$, which is scale-invariant.

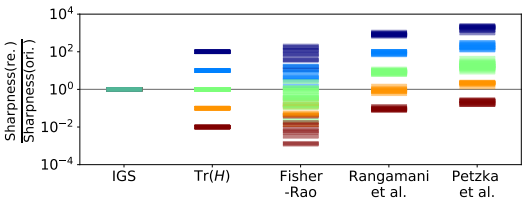


Figure 2: The ratio of sharpness measures evaluated for the reparametrized models to those for the original models ($\frac{\text{Sharpness (re.)}}{\text{Sharpness (ori.)}}$). Note that the y-axis is in the log scale. For the nonlinear reparametrization, we use $\eta = g(\theta) = (|\theta - \hat{\theta}|^2 + b)^a(\theta - \hat{\theta}) + \hat{\eta}$ considered in [8] with several choices of (a, b), which are represented by different colors.

However, the above measures do not satisfy the invariance to general nonlinear reparametrizations, hence solving the problem raised by [8] only partially. That is, for a nonlinear reparametrization $\theta \mapsto \eta = g(\theta)$, the measures evaluated on the reparametrized model $\tilde{f}(\cdot; \eta) = f(\cdot; g^{-1}(\eta)) = f(\cdot; \theta)$ (with respect to η) can arbitrarily vary from those evaluated on the original model $f(\cdot; \theta)$ (with respect to θ). In Figure 2, we empirically demonstrate the existence of this problem in conventional sharpness

measures such as the trace of the Hessian, the measures of [26], [40], and [39] (respectively denoted as $\mathrm{Tr}(H)$, Fisher-Rao, Rangamani et al., and Petzka et al. on the x-axis of the figure; see Appendix G.1 for details). We can observe that the magnitudes of these measures can vary significantly.

Compared with the previous measures, only our measure (denoted as IGS on the x-axis of Figure 2) is invariant to the considered reparametrizations. These experiments empirically demonstrate that our measure satisfies the reparametrization-invariance, hence providing a solution that properly resolves the issues posed by [8].

3.3 Connections to the generalization

3.3.1 Connections to Takeuchi's information criterion (TIC)

Our measures can be linked to generalization in different aspects. When we train parametric models via the maximum likelihood estimation (this is equivalent to minimizing the negative log-likelihood of (1)), the obtained maximum likelihood estimate is asymptotically unbiased, guaranteed theoretically by the asymptotic normality. However, the log-likelihood value (or the negative loss) evaluated at the obtained estimate does not enjoy such a property.

The well-known Akaike's information criterion (AIC) in the model selection literature calculates this bias under the assumption that the parametric model contains the actual data distribution [1]. Takeuchi's information criterion (TIC) is a generalized form of the AIC by considering the case of misspecified models, i.e., the parametric models do not contain the actual data distribution [44, 9, 45].

The bias $b(\hat{\theta})$ of the log-likelihood value evaluated at the maximum likelihood estimate is defined as follows:

$$b(\hat{\theta}) = \mathbb{E}_{\mathcal{D}}\left[\frac{1}{|\mathcal{D}|}\sum_{(x_i,y_i)\in\mathcal{D}} \log p(x_i, y_i; \hat{\theta}(\mathcal{D})) - \mathbb{E}_{p(x,y)}\left[\log p(x, y; \hat{\theta}(\mathcal{D}))\right]\right], \tag{8}$$

where $\mathcal{D} = \{(x_1, y_1), \ldots, (x_N, y_N)\}$ is the set of N training data with $(x_i, y_i) \sim p(x, y)$ and $\hat{\theta}(\mathcal{D})$ denotes the maximum likelihood estimate obtained from using $\mathcal{D}$ [9]. Since our loss is the negative log-likelihood, the first and second terms inside $\mathbb{E}_{\mathcal{D}}[\cdot]$ correspond to the negative training and test losses, respectively. Consequently, the bias in (8) becomes the expectation of the parametric model's generalization gap (i.e., test loss – training loss).

TIC calculates this bias under asymptotic assumptions, i.e., in the limit $N \to \infty$, as follows [9]:

$$\mathrm{TIC} = \lim_{N\to\infty} b(\hat{\theta}) = \frac{1}{N}\mathrm{Tr}(H(\theta_0)^{-1}C(\theta_0)), \tag{9}$$

where $\theta_0 \in \mathbb{R}^m$ is a local maximum of the expected log-likelihood, $H(\theta_0) \in \mathbb{R}^{m\times m}$ is the Hessian of the (expected) loss, and $C(\theta_0) = \mathbb{E}_{p(x,y)}\left[\left(\frac{\partial l(y,f(x;\theta))}{\partial\theta}\right)^\top \left(\frac{\partial l(y,f(x;\theta))}{\partial\theta}\right)\right]\Big|_{\theta=\theta_0} \in \mathbb{R}^{m\times m}$ is the non-centered covariance of the loss gradients. Here both $H(\theta_0)$ and $C(\theta_0)$ are evaluated using the data generating distribution $p(x, y)$.

The formulation in (9) is very similar to our information geometric sharpness measure when evaluated at a local minimum $\hat{\theta} \in \mathbb{R}^m$ of the loss in (1). Compared to TIC, our measure in (6) contains the FIM instead of the Hessian, and the expectation for $C(\theta)$ is taken with respect to the empirical distribution rather than the true distribution. After a model is sufficiently trained, the Hessian can be approximated to the FIM [36, 28], indicating that our sharpness measure is closely related to TIC.

It has been observed that TIC predicts the generalization gap of neural network models well when the expectation with respect to $p(x, y)$ is approximated by a finite sum of the integrands over the test data [45]. These findings are also confirmed by experiments using our measures. In Figure 3, compared to the other sharpness measures (labeled the same as in Figure 2), we can observe that our measure correlates better with the generalization gap. The experimental details are provided in Appendix G.2.

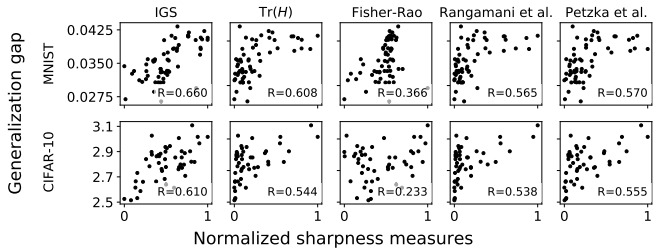


Figure 3: Plots for various sharpness measures (normalized to be in [0,1]) vs. the generalization gap. The top and bottom rows are obtained using MNIST and CIFAR-10 data sets, respectively. In each subplot, we provide the correlation coefficient between the corresponding measure and the generalization gap.

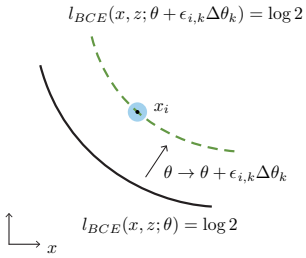


Figure 4: Change in the decision boundary according to the perturbation in parameter values $\theta \to \theta + \epsilon_{i,k}\Delta\theta_k$.

3.3.2 Connections to the margin

Our measure can also be linked to generalization in terms of margin. Unlike the usual definition of the margin as the distance from the decision boundary to its closest data, we devise a similar concept developed in the parametric model space rather than the input space.

Given a model parameter value, if we perturb the value so that the model's decision boundary passes through the location of a nearby data as shown in Figure 4, the difference between the initial and the perturbed parameter values would correspond to a sort of margin defined in the model parameter space. Since the parameter space dimension is high, we can perturb the model parameter value in various directions. Considering the perturbations of parameter values along the eigenvectors of the FIM results in an interesting connection between this margin and our sharpness measure. We now formalize this relationship for the binary classification case, while that for the multi-class case is derived in Appendix E.

We define the binary cross-entropy loss as $l_i(\theta) = l_{BCE}(x_i, z_i; \theta) = -z_i \log p(x_i; \theta) - (1 - z_i)\log(1 - p(x_i; \theta))$, where $z_i \in \{0, 1\}$ denotes the binary class label for an input $x_i \in \mathbb{R}^D$ and $p(x_i; \theta)$ denotes the model's prediction on the probability that x_i belongs to class one. The condition for x_i to lie on the decision boundary is $p(x_i) = \frac{1}{2}$, which is equal to $l_i(\theta) = \log 2$ regardless of the z_i value.

Suppose we perturb the neural network parameter value from θ to $\theta + \epsilon_{i,k}\Delta\theta_k$, where $\Delta\theta_k \in \mathbb{R}^m$ is the k-th eigenvector of the FIM associated with the k-th eigenvalue λ_k and $\epsilon_{i,k}$ is a scalar. For the decision boundary of the perturbed model to lie on x_i, the scalar $\epsilon_{i,k}$ should satisfy $\log 2 = l_i(\theta + \epsilon_{i,k}\Delta\theta_k) \approx l_i(\theta) + \epsilon_{i,k}\left(\frac{\partial l_i}{\partial \theta}\right)\Delta\theta_k$, where we apply the first-order Taylor expansion by assuming small $\epsilon_{i,k}$. The scalar $\epsilon_{i,k}$ can then be approximated as $\epsilon_{i,k} \approx \frac{\log 2 - l_i(\theta)}{\left(\frac{\partial l_i}{\partial \theta}\right)\Delta\theta_k}$.

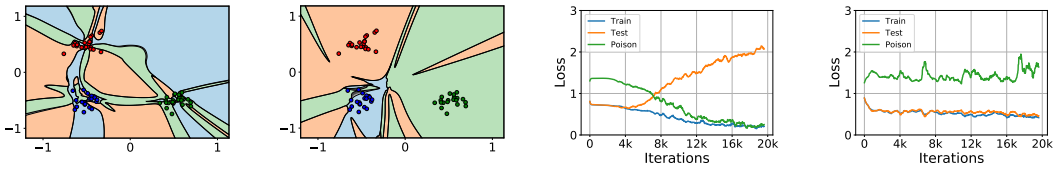
A sort of margin can be derived in an information geometric sense by computing the squared norm of the perturbation vector $\epsilon_{i,k}\Delta\theta_k$ using the inner product based on the FIM as follows:

$$\epsilon_{i,k}^2 \Delta\theta_k^\top F(\theta)\Delta\theta_k = \epsilon_{i,k}^2 \lambda_k \approx \lambda_k \left(\frac{\log 2 - l_i(\theta)}{\left(\frac{\partial l_i}{\partial \theta}\right)\Delta\theta_k}\right)^2. \tag{10}$$

The summation over $k = 1, \ldots, m'$ (with m' as the largest k with non-zero λ_k) of the reciprocal of these squared norms (hence weighing more on small $\epsilon_{i,k}^2$ values which would reduce the error from approximations) can then be related to our sharpness measure as follows:

$$\sum_{k=1}^{m'} \frac{1}{\epsilon_{i,k}^2 \lambda_k} \approx \sum_{k=1}^{m'} \frac{1}{\lambda_k}\left(\frac{\left(\frac{\partial l_i}{\partial \theta}\right)\Delta\theta_k}{\log 2 - l_i(\theta)}\right)^2 = \frac{\left(\frac{\partial l_i}{\partial \theta}\right) F(\theta)^\dagger \left(\frac{\partial l_i}{\partial \theta}\right)^\top}{(\log 2 - l_i(\theta))^2}, \tag{11}$$

where we have used $F(\theta)^\dagger = \sum_{k=1}^{m'} \frac{1}{\lambda_k}\Delta\theta_k\Delta\theta_k^\top$ in deriving the last identity. Note that our sharpness measure at each data (x_i, y_i) appears in the numerator of (11). This indicates that the smaller our sharpness measure, the larger the squared norms ($\epsilon_{i,k}^2\lambda_k$) in the LHS, which means that the margin becomes larger in an information geometric sense.



(a) Decision boundary from SGD

(b) Decision boundary from SGD with regularization

(c) Loss values from SGD

(d) Loss values from SGD with regularization

Figure 5: A synthetic three-class classification example trained using SGD with and without regularization under the effect of poison data. The loss values are smoothed for better visualization.

4 Using the sharpness measure as a regularizer to train neural networks

We now apply our sharpness measure as a regularizer to train neural networks to see if regularizing the measure can improve the generalization performance. The corresponding loss function is written as

$$\mathcal{L}(\theta) = \frac{1}{N}\sum_{i=1}^{N} l_i(\theta) + \rho \cdot \mathrm{IGS}(\theta), \tag{12}$$

where ρ is a coefficient for the regularization term. We consider a toy example and then consider image classification tasks involving MNIST and CIFAR-10/100 data sets. We also provide some tractable and possibly efficient ways to regularize our sharpness measure. All the experiments have been performed using the PyTorch library [38].

4.1 A toy example

We first apply our regularizer to a toy example. In [18], they have trained neural networks on poison data with the wrong labels to hamper the generalization performance and make the loss landscape extremely sharp while fitting all the training data. We follow this experimental setting and check whether the obtained solution can avoid the undesirable generalization performance and loss landscapes when applying our regularizer (calculated on the training data). The experimental details, as well as how the IGS is approximated for this example, are provided in Appendix G.3.

The experimental results are shown in Figure 5. The results from SGD without regularization show very irregular decision boundaries with tiny margins, whereas relatively soft boundaries with larger margins appear when our regularizer is used, especially around the training data. Applying our regularizer also shows a lower generalization gap, i.e., the difference between the training and test losses, than the SGD without regularization.

4.2 MNIST and CIFAR-10/100

To check the effect of the regularizer in more realistic settings, we apply our regularizer to the classification of MNIST and CIFAR-10/100 data sets. For this experiment, we regularize the mini-batch IGS defined in (7). For an efficient implementation of our regularizer, the following approximation on perturbed loss functions with a perturbation $\delta = \rho F(\theta)^{\dagger}\left(\frac{\partial l}{\partial \theta}\right)^{\top} \in \mathbb{R}^m$ (with a small $\rho > 0$) is useful:

$$l(\theta+\delta) \;\approx\; l(\theta) + \left(\frac{\partial l}{\partial \theta}\right)\delta \;\approx\; l(\theta) + \rho \cdot \left(\frac{\partial l}{\partial \theta}\right) F(\theta)^{\dagger}\left(\frac{\partial l}{\partial \theta}\right)^{\top}. \tag{13}$$

Note that for $l = \frac{1}{b}\sum_{x_i \in B} l_i$ with a mini-batch B of size $|B| = b$, minimizing (13) becomes minimizing a stochastic version of (12) (with $\mathrm{IGS}_b(\theta)$), where the stochasticity comes from sampling a mini-batch B. Therefore we minimize (13) in our experiments with ρ as a tunable hyperparameter. To obtain δ, we resort to an EKFAC-based approximation for the natural gradients [13] as detailed in Appendix G.4.1.

A three-layer fully connected neural network (3FCN) is used for the experiments using MNIST data, and various convolutional neural networks such as VGG [42], ResNet [15], and WideResNet [50] are used for CIFAR-10/100 experiments, with data augmentation, cosine annealing [27], and label

Table 1: Averages and standard errors of the test classification accuracies for SGD, GR, SAM, ASAM, and SGD with our regularization method on MNIST, CIFAR-10, and CIFAR-100 data sets.

Data set	Model	SGD	GR	SAM	ASAM	Ours
MNIST	3FCN	98.13 ± 0.05	98.20 ± 0.01	98.62 ± 0.01	**98.65** ± 0.07	**98.65** ± 0.03
CIFAR-10	VGG11-BN	92.75 ± 0.11	93.13 ± 0.14	93.61 ± 0.08	93.78 ± 0.13	**93.81** ± 0.06
	ResNet-56	94.08 ± 0.17	94.40 ± 0.23	95.22 ± 0.11	**95.36** ± 0.06	94.96 ± 0.01
	WRN-16-8	95.98 ± 0.03	96.15 ± 0.10	96.70 ± 0.14	**97.03** ± 0.06	96.57 ± 0.10
CIFAR-100	VGG11-BN	72.25 ± 0.17	73.05 ± 0.32	73.69 ± 0.20	73.61 ± 0.18	**74.24** ± 0.22
	ResNet-56	73.14 ± 0.10	74.67 ± 0.25	75.90 ± 0.25	75.87 ± 0.27	**76.07** ± 0.11
	WRN-16-8	80.55 ± 0.33	81.13 ± 0.06	82.20 ± 0.11	82.38 ± 0.09	**82.44** ± 0.29

smoothing [30] methods additionally applied. For comparison, we consider the gradient regularization (GR) method [4, 43], sharpness-aware minimization (SAM) method [12], and the adaptive SAM (ASAM) method [24], an extension of SAM to involve the scale-invariance. The detailed experimental settings are explained in Appendix G.4.2.

We provide the averages and standard errors of the test classification accuracies obtained from three runs of each method in Table 1. As can be seen from the table, one can confirm that the generalization performance of SGD is significantly improved with our regularizer. Furthermore, our method shows better performance than the GR method and a comparable performance improvement with the SAM and ASAM methods.

5 Conclusion

Various empirical results observed from running deep learning algorithms such as SGD have suggested that minima with a flatter loss landscape tend to show better generalization performance. However, the previous definitions of sharpness/flatness have faced a severe dependence on reparametrizations or parameter scalings that do not change the model output and the generalization performance [8].

In this paper, based on an information geometric analysis of the neural network parameter space, we have proposed an information geometric sharpness measure which is reparametrization- and scale-invariant, making it free from the issues posed by [8]. We have also discussed the connection of the measure to generalization. In addition, a regularizer that reduces the suggested sharpness measure is proposed, showing a significant improvement in generalization performance under practical settings.

Unlike many other sharpness measures, the Fisher information matrix (FIM) plays a crucial role in our measure. Since it can be computationally demanding to calculate the FIM and our sharpness measure for large-scale neural networks, more efficient evaluation methods should be developed to apply our measure to such networks. Concerning the regularization of our measure during training, exploring better ways to reduce the time complexity and increase generalization performance is left for future work. Furthermore, analyzing the generalization properties of models obtained from deep learning algorithms such as SGD or natural gradient descent [2, 28] using our measure would be another intriguing future work.

Acknowledgments and Disclosure of Funding

C. Jang and Y.-K. Noh were supported by IITP Artificial Intelligence Graduate School Program for Hanyang University funded by MSIT (Grant No. 2020-0-01373). S. Lee was supported by a KIAS Individual Grant (AP083601) via the Center for AI and Natural Sciences at Korea Institute for Advanced Study. F. C. Park was supported in part by SRRC NRF grant 2016R1A5A1938472, IITP-MSIT grant 2022-0-00480 (Training and Inference Methods for Goal-Oriented AI Agents), SNU-AIIS, SNU-IAMD, and the SNU Institute for Engineering Research. Y.-K. Noh was partly supported by NRF/MSIT (Grant No. 2018R1A5A7059549, 2021M3E5D2A01019545) and IITP/MSIT (Grant No. IITP-2021-0-02068).

References

[1] Akaike, H.: A new look at the statistical model identification. IEEE transactions on automatic control **19**(6), 716–723 (1974)

[2] Amari, S.I.: Natural gradient works efficiently in learning. Neural computation **10**(2), 251–276 (1998)

[3] Amari, S.i.: Information geometry and its applications, vol. 194. Springer (2016)

[4] Barrett, D.G., Dherin, B.: Implicit gradient regularization. arXiv preprint arXiv:2009.11162 (2020)

[5] Bartlett, P.L., Foster, D.J., Telgarsky, M.J.: Spectrally-normalized margin bounds for neural networks. Advances in Neural Information Processing Systems **30**, 6240–6249 (2017)

[6] Boothby, W.M.: An introduction to differentiable manifolds and Riemannian geometry, Revised, vol. 120. Gulf Professional Publishing (2003)

[7] Chaudhari, P., Choromanska, A., Soatto, S., LeCun, Y., Baldassi, C., Borgs, C., Chayes, J., Sagun, L., Zecchina, R.: Entropy-sgd: Biasing gradient descent into wide valleys. Journal of Statistical Mechanics: Theory and Experiment **2019**(12), 124018 (2019)

[8] Dinh, L., Pascanu, R., Bengio, S., Bengio, Y.: Sharp minima can generalize for deep nets. In: International Conference on Machine Learning, pp. 1019–1028. PMLR (2017)

[9] Dixon, M., Ward, T.: Takeuchi's information criteria as a form of regularization. arXiv preprint arXiv:1803.04947 (2018)

[10] Dziugaite, G.K., Roy, D.M.: Computing nonvacuous generalization bounds for deep (stochastic) neural networks with many more parameters than training data. arXiv preprint arXiv:1703.11008 (2017)

[11] Fomenko, A., Novikov, S., Dubrovin, B.: Modern Geometry-Methods and Applications, Part I: The Geometry of Surfaces, Transformation Groups, and Fields. Springer, Berlin (1992)

[12] Foret, P., Kleiner, A., Mobahi, H., Neyshabur, B.: Sharpness-aware minimization for efficiently improving generalization. arXiv preprint arXiv:2010.01412 (2020)

[13] George, T., Laurent, C., Bouthillier, X., Ballas, N., Vincent, P.: Fast approximate natural gradient descent in a kronecker factored eigenbasis. Advances in Neural Information Processing Systems **31** (2018)

[14] Ghorbani, B., Krishnan, S., Xiao, Y.: An investigation into neural net optimization via hessian eigenvalue density. In: International Conference on Machine Learning, pp. 2232–2241. PMLR (2019)

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

[16] Hinton, G.E., Van Camp, D.: Keeping the neural networks simple by minimizing the description length of the weights. In: Proceedings of the sixth annual conference on Computational learning theory, pp. 5–13 (1993)

[17] Hochreiter, S., Schmidhuber, J.: Flat minima. Neural computation **9**(1), 1–42 (1997)

[18] Huang, W.R., Emam, Z., Goldblum, M., Fowl, L., Terry, J.K., Huang, F., Goldstein, T.: Understanding generalization through visualizations (2020)

[19] Izmailov, P., Podoprikhin, D., Garipov, T., Vetrov, D., Wilson, A.G.: Averaging weights leads to wider optima and better generalization. In: 34th Conference on Uncertainty in Artificial Intelligence 2018, UAI 2018, pp. 876–885. Association For Uncertainty in Artificial Intelligence (AUAI) (2018)

[20] Jastrzebski, S., Arpit, D., Astrand, O., Kerg, G.B., Wang, H., Xiong, C., Socher, R., Cho, K., Geras, K.J.: Catastrophic fisher explosion: Early phase fisher matrix impacts generalization. In: International Conference on Machine Learning, pp. 4772–4784. PMLR (2021)

[21] Jiang, Y., Neyshabur, B., Mobahi, H., Krishnan, D., Bengio, S.: Fantastic generalization measures and where to find them. In: International Conference on Learning Representations (2019)

[22] Keskar, N.S., Nocedal, J., Tang, P.T.P., Mudigere, D., Smelyanskiy, M.: On large-batch training for deep learning: Generalization gap and sharp minima. In: 5th International Conference on Learning Representations, ICLR 2017 (2017)

[23] Kunstner, F., Hennig, P., Balles, L.: Limitations of the empirical fisher approximation for natural gradient descent. Advances in neural information processing systems **32** (2019)

[24] Kwon, J., Kim, J., Park, H., Choi, I.K.: Asam: Adaptive sharpness-aware minimization for scale-invariant learning of deep neural networks. arXiv preprint arXiv:2102.11600 (2021)

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

[26] Liang, T., Poggio, T., Rakhlin, A., Stokes, J.: Fisher-rao metric, geometry, and complexity of neural networks. In: The 22nd International Conference on Artificial Intelligence and Statistics, pp. 888–896. PMLR (2019)

[27] Loshchilov, I., Hutter, F.: Sgdr: Stochastic gradient descent with warm restarts. arXiv preprint arXiv:1608.03983 (2016)

[28] Martens, J.: New insights and perspectives on the natural gradient method. Journal of Machine Learning Research **21**, 1–76 (2020)

[29] Martens, J., Grosse, R.: Optimizing neural networks with kronecker-factored approximate curvature. arXiv preprint arXiv:1503.05671 (2015)

[30] Müller, R., Kornblith, S., Hinton, G.: When does label smoothing help? arXiv preprint arXiv:1906.02629 (2019)

[31] Nair, V., Hinton, G.E.: Rectified linear units improve restricted boltzmann machines. In: Icml (2010)

[32] Neyshabur, B., Bhojanapalli, S., Mcallester, D., Srebro, N.: Exploring generalization in deep learning. Advances in Neural Information Processing Systems **30**, 5947–5956 (2017)

[33] Neyshabur, B., Salakhutdinov, R., Srebro, N.: Path-sgd: path-normalized optimization in deep neural networks. In: Proceedings of the 28th International Conference on Neural Information Processing Systems-Volume 2, pp. 2422–2430 (2015)

[34] Neyshabur, B., Tomioka, R., Srebro, N.: Norm-based capacity control in neural networks. In: Conference on Learning Theory, pp. 1376–1401. PMLR (2015)

[35] Nielsen, F.: An elementary introduction to information geometry. Entropy **22**(10), 1100 (2020)

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

[37] Papyan, V.: Measurements of three-level hierarchical structure in the outliers in the spectrum of deepnet hessians. In: International Conference on Machine Learning, pp. 5012–5021. PMLR (2019)

[38] Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., Lin, Z., Desmaison, A., Antiga, L., Lerer, A.: Automatic differentiation in pytorch (2017)

[39] Petzka, H., Kamp, M., Adilova, L., Sminchisescu, C., Boley, M.: Relative flatness and generalization. Advances in Neural Information Processing Systems **34** (2021)

[40] Rangamani, A., Nguyen, N.H., Kumar, A., Phan, D., Chin, S.P., Tran, T.D.: A scale invariant measure of flatness for deep network minima. In: ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pp. 1680–1684. IEEE (2021)

[41] Sagun, L., Evci, U., Guney, V.U., Dauphin, Y., Bottou, L.: Empirical analysis of the hessian of over-parametrized neural networks. arXiv preprint arXiv:1706.04454 (2017)

[42] Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556 (2014)

[43] Smith, S.L., Dherin, B., Barrett, D.G., De, S.: On the origin of implicit regularization in stochastic gradient descent. arXiv preprint arXiv:2101.12176 (2021)

[44] Takeuchi, K.: The distribution of information statistics and the criterion of goodness of fit of models. Mathematical Science **153**, 12–18 (1976)

[45] Thomas, V., Pedregosa, F., Merriënboer, B., Manzagol, P.A., Bengio, Y., Le Roux, N.: On the interplay between noise and curvature and its effect on optimization and generalization. In: International Conference on Artificial Intelligence and Statistics, pp. 3503–3513. PMLR (2020)

[46] Tsuzuku, Y., Sato, I., Sugiyama, M.: Normalized flat minima: Exploring scale invariant definition of flat minima for neural networks using pac-bayesian analysis. In: International Conference on Machine Learning, pp. 9636–9647. PMLR (2020)

[47] Wu, J., Hu, W., Xiong, H., Huan, J., Braverman, V., Zhu, Z.: On the noisy gradient descent that generalizes as sgd. In: International Conference on Machine Learning, pp. 10367–10376. PMLR (2020)

[48] Yao, Z., Gholami, A., Keutzer, K., Mahoney, M.W.: Pyhessian: Neural networks through the lens of the hessian. In: 2020 IEEE International Conference on Big Data (Big Data), pp. 581–590. IEEE (2020)

[49] Yao, Z., Gholami, A., Lei, Q., Keutzer, K., Mahoney, M.W.: Hessian-based analysis of large batch training and robustness to adversaries. Advances in Neural Information Processing Systems **31** (2018)

[50] Zagoruyko, S., Komodakis, N.: Wide residual networks. arXiv preprint arXiv:1605.07146 (2016)

Checklist

1. For all authors...

(a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes] The need to consider information geometry in measuring sharpness is discussed in Section 2; the definition of our sharpness measure, its reparametrization-invariance, and its connection to generalization are explained in Section 3; and the experiments to use our measure as a regularizer in neural network training are performed in Section 4.

(b) Did you describe the limitations of your work? [Yes] See Section 5.

(c) Did you discuss any potential negative societal impacts of your work? [N/A]

(d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes] We have read the ethics review guidelines and checked that our paper conforms to them.

2. If you are including theoretical results...

(a) Did you state the full set of assumptions of all theoretical results? [Yes] See Section 3.1.

(b) Did you include complete proofs of all theoretical results? [Yes] See Appendix D.1.

3. If you ran experiments...

(a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] See supplemental materials.

(b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] See Appendix G.

(c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes] See Tables 1 and 2.

(d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] See Appendix G.4.2.

4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...

(a) If your work uses existing assets, did you cite the creators? [Yes] See Appendix G.

(b) Did you mention the license of the assets? [Yes] See Appendix G.

(c) Did you include any new assets either in the supplemental material or as a URL? [Yes] See supplemental materials.

(d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]

(e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]

5. If you used crowdsourcing or conducted research with human subjects...

(a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]

(b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]

(c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]

Appendix

A The Fisher information matrix and class margins

As mentioned in Section 2.2 of the manuscript, [37] suggested a hierarchy of the logit gradients based on the following decomposition of the Fisher information matrix:

$$\frac{1}{N}\sum_{i=1}^{N} J_i^\top \left(\mathrm{diag}(p_i) - p_i p_i^\top\right) J_i = \frac{1}{N}\sum_{i=1}^{N} J_i^\top M_i^\top M_i J_i, \tag{14}$$

where $M_i = \mathrm{diag}(\sqrt{p_i})\left(I - \mathbb{1}p_i^\top\right) \in \mathbb{R}^{C\times C}$, $\sqrt{p_i} = (\sqrt{(p_i)_1}, \ldots, \sqrt{(p_i)_C}) \in \mathbb{R}^C$, and $\mathbb{1} = (1, \ldots, 1) \in \mathbb{R}^C$ is a vector whose elements are all one.

According to the proposed hierarchy, the dominant component of the FIM attributing the outliers of the eigenspectra turns out to be

$$G = (C-1)\sum_{c=1}^{C} \frac{N_c}{N} \delta_c \delta_c^\top, \tag{15}$$

where N_c is the number of data belonging to class c. Here the vector $\delta_c \in \mathbb{R}^m$ is defined as

$$\delta_c = \frac{1}{N_c \cdot (C-1)} \sum_{c' \neq c} \sum_{i \in I_c} \delta_{i,c'}, \tag{16}$$

where $\delta_{i,j} \in \mathbb{R}^m$ is the j-th row vector of the matrix $M_i J_i \in \mathbb{R}^{C \times m}$ (also defined in (5)) and $I_c = \{i \in \{1, \dots, N\} | y_i = c\}$ is the set of indices for data in class c.

The dominant component of FIM in (15) implies that the FIM can be related to the difference between the c-th component and $c' \neq c$-th components in the logit derivatives. This relation can be seen more evidently using some assumptions on the predicted probability p_i.

Assuming the prediction for the classes $c' \neq c$ is balanced, i.e., $(p_i)_{c'}$, the c'-th component of p_i for $c' \neq c$, are all equal to a constant $\epsilon \ll 1$, we can approximate $\delta_{i,c'}$ and δ_c as follows:

$$\delta_{i,c'} = \sqrt{\epsilon} \cdot (J_{i,c'} - J_{i,c})^\top + o(\sqrt{\epsilon}), \tag{17}$$

$$\delta_c = \sqrt{\epsilon} \cdot \left(\frac{1}{N_c} \sum_{i \in I_c} \left(\frac{1}{C-1} \sum_{c' \neq c} J_{i,c'} - J_{i,c} \right)^\top \right) + o(\sqrt{\epsilon}), \tag{18}$$

where $J_{i,j} \in \mathbb{R}^{1 \times m}$ denotes the j-th row of the logit derivative $J_i \in \mathbb{R}^{C \times m}$. In this case, δ_c can be interpreted as the gradient of the differences in the averaged logit values that can discriminate class c from the other classes $c' \neq c$. Since the dominant component of the FIM is insisted to be the G term in (15), we conjecture that the eigensubspace of the FIM can be related to the class margins as long as the hierarchy from [37] is valid. Similar derivations can be found in [36].

B On the well-definedness of IGS

From (2), the Fisher information matrix is written as

$$F(\theta) = \frac{1}{N} \sum_{i=1}^{N} J_i^\top \left(\text{diag}(p_i) - p_i p_i^\top \right) J_i, \tag{19}$$

where $J_i = \left(\frac{\partial f}{\partial \theta}(x) \right) \Big|_{x = x_i} \in \mathbb{R}^{C \times m}$ and $p_i = (p(1|x_i;\theta), \dots, p(C|x_i;\theta)) \in \mathbb{R}^C$ is the model's prediction on the probability that x_i belongs to each class.

The FIM is positive semi-definite when the number of parameters is very large, e.g., for the over-parametrized models with $m \gg N$. Therefore, we need to verify if our definition of information geometric measure in (6) involving the pseudo-inverse of $F(\theta)$ is well-defined in the sense that there are not any components of $\left(\frac{\partial l_i}{\partial \theta} \right)$ neglected in measuring the information geometric norm by (6). For this purpose, we characterize the null space of $F(\theta)$ and compare it to that of the loss derivative $\left(\frac{\partial l_i}{\partial \theta} \right) \in \mathbb{R}^{1 \times m}$.

From (19) the null space of $F(\theta)$, $\text{Null}(F(\theta)) = N_1 \cup N_2$, where N_1 and N_2 are defined as

$$N_1 = \{\dot{\theta} \in \mathbb{R}^m \mid J_i \dot{\theta} = 0, \quad \forall i \in \{1, \dots, N\}\}, \tag{20}$$

$$N_2 = \{\dot{\theta} \in \mathbb{R}^m \mid J_i \dot{\theta} = c \cdot (1, \dots, 1)^\top, \quad c \in \mathbb{R}, \quad \forall i \in \{1, \dots, N\}\}. \tag{21}$$

Note that the condition for N_2 comes from the fact that the matrix $\left(\text{diag}(p_i) - p_i p_i^\top \right) \in \mathbb{R}^{C \times C}$ in (19) possesses the vector $(1, \dots, 1)^\top \in \mathbb{R}^C$ as its eigenvector with the eigenvalue of zero.

For $\dot{\theta} \in \text{Null}(F(\theta))$, we can easily show that $\left(\frac{\partial l_i}{\partial \theta} \right) \dot{\theta} = 0$ for all $i = 1, \dots, N$, since $\left(\frac{\partial l_i}{\partial \theta} \right) = -(e_{y_i}^\top - \sum_j (p_i)_j e_j^\top) J_i \dot{\theta} = 0$ for both the cases of $\dot{\theta} \in N_1$ and $\dot{\theta} \in N_2$. Therefore, $\text{Null}(F(\theta)) \subset \text{Null}(\left(\frac{\partial l_i}{\partial \theta} \right))$ holds, and we can say that the vector $\left(\frac{\partial l_i}{\partial \theta} \right)^\top \in \mathbb{R}^m$ belongs to the span of the eigenvectors of $F(\theta)$ with non-zero eigenvalues for all $i = 1, \dots, N$. Hence our information geometric measure in (6) does not neglect any components in $\left(\frac{\partial l_i}{\partial \theta} \right)$.

C On the relationship between IGS and mini-batch IGS

To obtain the relationship between IGS in (6) and mini-batch IGS in (7), we express them as follows:

$$\mathrm{IGS}(\theta) = \frac{1}{N}\sum_{i=1}^{N}\left(\frac{\partial l_i}{\partial \theta}\right) F(\theta)^{\dagger}\left(\frac{\partial l_i}{\partial \theta}\right)^{\top} = \mathrm{Tr}\left(F(\theta)^{\dagger}S(\theta)\right), \tag{22}$$

$$\mathrm{IGS}_b(\theta) = \mathbb{E}_B\left[\left(\frac{1}{b}\sum_{x_i\in B}\frac{\partial l_i}{\partial \theta}\right) F(\theta)^{\dagger}\left(\frac{1}{b}\sum_{x_i\in B}\frac{\partial l_i}{\partial \theta}\right)^{\top}\right] = \mathrm{Tr}\left(F(\theta)^{\dagger}S_b(\theta)\right), \tag{23}$$

where $S(\theta) = \frac{1}{N}\sum_{i=1}^{N}\left(\frac{\partial l_i}{\partial \theta}\right)^{\top}\left(\frac{\partial l_i}{\partial \theta}\right) \in \mathbb{R}^{m\times m}$ is the second moment of the (individual) loss gradients and $S_b(\theta) = \mathbb{E}_B\left[\left(\frac{1}{b}\sum_{x_i\in B}\frac{\partial l_i}{\partial \theta}\right)^{\top}\left(\frac{1}{b}\sum_{x_i\in B}\frac{\partial l_i}{\partial \theta}\right)\right] \in \mathbb{R}^{m\times m}$ is that of the mini-batch loss gradients with a batch size of b.

For our purpose, the relationship between the covariance of loss gradients and that of mini-batch loss gradients is useful. The relationship is established in [47] as $C_b(\theta) = \frac{\gamma_{b,N}}{b}C(\theta)$, where $C(\theta), C_b(\theta) \in \mathbb{R}^{m\times m}$ are respectively the covariance of the (individual) loss gradients and mini-batch loss gradients (with a batch size of b), and $\gamma_{b,N} = \frac{N-b}{N-1}$ if mini-batches are sampled without replacement and 1 if sampled with replacement. From the relations between the covariances and second moments given as $C(\theta) = S(\theta) - S_N(\theta)$ and $C_b(\theta) = S_b(\theta) - S_N(\theta)$ with $S_N(\theta) = \left(\frac{1}{N}\sum_{i=1}^{N}\frac{\partial l_i}{\partial \theta}\right)^{\top}\left(\frac{1}{N}\sum_{i=1}^{N}\frac{\partial l_i}{\partial \theta}\right) \in \mathbb{R}^{m\times m}$, we can relate $S(\theta)$ and $S_b(\theta)$ as follows:

$$S_b(\theta) = C_b(\theta) + S_N(\theta) = \frac{\gamma_{b,N}}{b}C(\theta) + S_N(\theta) \tag{24}$$

$$= \frac{\gamma_{b,N}}{b}S(\theta) + (1 - \frac{\gamma_{b,N}}{b})S_N(\theta). \tag{25}$$

By applying (25) to (23), we can obtain the relationship between IGS and mini-batch IGS as follows:

$$\mathrm{IGS}_b(\theta) = \frac{\gamma_{b,N}}{b}\mathrm{Tr}\left(F(\theta)^{\dagger}S(\theta)\right) + (1 - \frac{\gamma_{b,N}}{b})\mathrm{Tr}\left(F(\theta)^{\dagger}S_N(\theta)\right) \tag{26}$$

$$= \frac{\gamma_{b,N}}{b}\mathrm{IGS}(\theta) + (1 - \frac{\gamma_{b,N}}{b})\mathrm{IGS}_N(\theta). \tag{27}$$

By defining other information geometric sharpness measures as $\text{IGS-C}(\theta) \equiv \mathrm{IGS}(\theta) - \mathrm{IGS}_N(\theta)$ and $\mathrm{IGS}_b\text{-C}(\theta) \equiv \mathrm{IGS}_b(\theta) - \mathrm{IGS}_N(\theta)$, which can be considered as replacing the second moments with covariances in (22) and (23), the relationship between them becomes as follows:

$$\mathrm{IGS}_b\text{-C}(\theta) = \frac{\gamma_{b,N}}{b}\text{IGS-C}(\theta). \tag{28}$$

D On the transformation invariance of IGS

D.1 Proof of Proposition 3.1

Proof. To prove $\mathrm{IGS}(\theta) = \mathrm{IGS}(g(\theta))$ (or $\mathrm{IGS}_b(\theta) = \mathrm{IGS}_b(g(\theta))$) for all $\theta \in U$, we have to show whether the following equality holds for any $\theta \in U$ and $i, j \in \{1, \ldots, N\}$:

$$\left(\frac{\partial l_i}{\partial \theta}\right) F(\theta)^{\dagger}\left(\frac{\partial l_j}{\partial \theta}\right)^{\top} = \left(\frac{\partial l_i}{\partial g(\theta)}\right) F(g(\theta))^{\dagger}\left(\frac{\partial l_j}{\partial g(\theta)}\right)^{\top}. \tag{29}$$

By differentiating $f(x;\theta) = f(x;g(\theta))$ with respect to θ, we get $\left(\frac{\partial f}{\partial \theta}\right) = \left(\frac{\partial f}{\partial g(\theta)}\right)\left(\frac{\partial g(\theta)}{\partial \theta}\right)$, where $\left(\frac{\partial g(\theta)}{\partial \theta}\right) \in \mathbb{R}^{m\times m}$ is invertible by the assumption. Applying this identity to the definition of the Fisher information matrix gives the transformation rule of $F(\theta) = \left(\frac{\partial g(\theta)}{\partial \theta}\right)^{\top} F(g(\theta))\left(\frac{\partial g(\theta)}{\partial \theta}\right)$. Furthermore, from the chain rule, $\left(\frac{\partial l_i}{\partial \theta}\right) = \left(\frac{\partial l_i}{\partial g(\theta)}\right)\left(\frac{\partial g(\theta)}{\partial \theta}\right)$. Combining these two identities and using the fact that $\left(\frac{\partial g(\theta)}{\partial \theta}\right)$ is invertible and $(AMA^{\top})^{\dagger} = A^{-\top}M^{\dagger}A^{-1}$ for an invertible matrix $A \in \mathbb{R}^{m\times m}$ and any $M \in \mathbb{R}^{m\times m}$ give the desired (29). □

D.2 Other transformation examples

We provide several examples of transformations that satisfy the assumption of Proposition 3.1 as follows:

- For the fully-connected neural networks, the network function remains the same if we apply identical permutations for the i-th layer weights row-wisely and the $i+1$-th layer weights column-wisely. (For convolutional neural networks, applying permutations channel-wisely would maintain the network function the same.)
- For the fully-connected neural networks, when we apply odd functions (i.e., $\phi(-u) = -\phi(u)$) as an activation function, the network function remains the same if we apply identical sign change to some rows of the i-th layer weights and corresponding columns of the $i+1$-th layer weights.
- As a bit more complicated nonlinear example, consider a neural network with two-dimensional input $x = (x_1, x_2)$, three-dimensional hidden layer, and one-dimensional output. Consider $s(x) = x^2$ as the (element-wise) nonlinear activation function to apply. Denote by $W^{(1)} \in \mathbb{R}^{3\times 2}$ the input-to-hidden weights and by $W^{(2)} \in \mathbb{R}^{1\times 3}$ the hidden-to-output weights. The network function then becomes

$$f(x; W^{(1)}, W^{(2)}) = W^{(2)} \cdot s(W^{(1)}x) = \sum_{i=1}^{3} W_{1i}^{(2)} (W_{i1}^{(1)} x_1 + W_{i2}^{(1)} x_2)^2,$$

where W_{ij} is the (i,j) entry of W.

Set another network with the same architecture as

$$\tilde{f}(x; \tilde{W}^{(1)}, \tilde{W}^{(2)}) = \tilde{W}^{(2)} \cdot s(\tilde{W}^{(1)}x) = \sum_{i=1}^{3} \tilde{W}_{1i}^{(2)} (\tilde{W}_{i1}^{(1)} x_1 + \tilde{W}_{i2}^{(1)} x_2)^2,$$

using weights $\tilde{W}^{(1)} \in \mathbb{R}^{3\times 2}, \tilde{W}^{(2)} \in \mathbb{R}^{1\times 3}$.

The condition for the two networks modeling identical function can be expressed as follows:

$$\sum_{i=1}^{3} W_{1i}^{(2)} (W_{i1}^{(1)} x_1 + W_{i2}^{(1)} x_2)^2 = \sum_{i=1}^{3} \tilde{W}_{1i}^{(2)} (\tilde{W}_{i1}^{(1)} x_1 + \tilde{W}_{i2}^{(1)} x_2)^2 \text{ for all } x \in \mathbb{R}^2.$$

If we rearrange the above for each coefficient of x_1^2, x_1x_2, x_2^2, we obtain

$$\sum_{i=1}^{3} W_{1i}^{(2)} (W_{i1}^{(1)})^2 = \sum_{i=1}^{3} \tilde{W}_{1i}^{(2)} (\tilde{W}_{i1}^{(1)})^2, \tag{30}$$

$$\sum_{i=1}^{3} W_{1i}^{(2)} W_{i1}^{(1)} W_{i2}^{(1)} = \sum_{i=1}^{3} \tilde{W}_{1i}^{(2)} \tilde{W}_{i1}^{(1)} \tilde{W}_{i2}^{(1)}, \tag{31}$$

$$\sum_{i=1}^{3} W_{1i}^{(2)} (W_{i2}^{(1)})^2 = \sum_{i=1}^{3} \tilde{W}_{1i}^{(2)} (\tilde{W}_{i2}^{(1)})^2. \tag{32}$$

For the set of $W^{(1)}, W^{(2)}, \tilde{W}^{(1)}, \tilde{W}^{(2)}$ that satisfy above equations, the two network functions $f(x; W^{(1)}, W^{(2)})$ and $\tilde{f}(x; \tilde{W}^{(1)}, \tilde{W}^{(2)})$ become identical.

Consider finding $\tilde{W}^{(2)}$ for given $W^{(1)}, W^{(2)}$, and $\tilde{W}^{(1)} = \phi_1(W^{(1)})$ for a bijective nonlinear function $\phi : \mathbb{R}^{3\times 2} \to \mathbb{R}^{3\times 2}$. Since there are three equations, i.e., (30), (31), and (32), and three variables of $(\tilde{W}_{11}^{(2)}, \tilde{W}_{12}^{(2)}, \tilde{W}_{13}^{(2)})$ in $\tilde{W}^{(2)}$, we can find the solution of $\tilde{W}^{(2)}$ and represent it as $\tilde{W}^{(2)} = \phi_2(W^{(1)}, W^{(2)})$.

We can then figure out that two different weight configurations can model identical network functions, with a nonlinear relationship between each other given as $\tilde{W}^{(1)} = \phi_1(W^{(1)})$ and $\tilde{W}^{(2)} = \phi_2(W^{(1)}, W^{(2)})$.

Even if the input or output dimension increases, if the hidden variable dimension increases appropriately, we can find mappings between weights similarly that can model identical neural network functions.

E An information geometric margin for multi-class problems

An information geometric margin can also be formulated in the multi-class classification case similarly as done in Section 3.3. For a data point (x_i, y_i) of class $y_i = c$, we consider the decision boundary between class c and class c' for all $c' \neq c$. The condition for a point $x \in \mathbb{R}^D$ to lie on this decision boundary between classes c and c' is given as $f_c(x;\theta) = f_{c'}(x;\theta)$, where $f_j(\cdot;\theta) : \mathbb{R}^D \to \mathbb{R}$ is the j-th logit function (or the j-th output of the neural network). Again, we consider perturbing the parameter values along the eigenvectors of the Fisher information matrix to make the data (x_i, y_i) lie on this decision boundary.

Suppose we perturb the parameter value from θ to $\theta + \epsilon_{i,k,c'}\Delta\theta_k$, where $\Delta\theta_k$ is the k-th eigenvector (associated with the k-th eigenvalue λ_k) and $\epsilon_{i,k,c'}$ is a scalar. To make the data (x_i, y_i) lie on the decision boundary, the perturbation step size $\epsilon_{i,k,c'}$ should satisfy the following:

$$f_{i,c}(\theta + \epsilon_{i,k,c'}\Delta\theta_k) = f_{i,c'}(\theta + \epsilon_{i,k,c'}\Delta\theta_k), \tag{33}$$

where $f_{i,j}(\theta) \in \mathbb{R}$ is the $f_j(x_i;\theta)$.

From the first-order Taylor expansion assuming small $\epsilon_{i,k,c'}$, the following approximations hold:

$$f_{i,c}(\theta) + \epsilon_{i,k,c'} J_{i,c}\Delta\theta_k \approx f_{i,c'}(\theta) + \epsilon_{i,k,c'} J_{i,c'}\Delta\theta_k, \tag{34}$$

$$\epsilon_{i,k,c'} \approx \frac{f_{i,c}(\theta) - f_{i,c'}(\theta)}{(J_{i,c'} - J_{i,c})\,\Delta\theta_k}, \tag{35}$$

$$\frac{f_{i,c}(\theta) - f_{i,c'}(\theta)}{\epsilon_{i,k,c'}} \approx (J_{i,c'} - J_{i,c})\,\Delta\theta_k, \tag{36}$$

where $J_{i,j} = \left(\frac{\partial f_{i,j}}{\partial\theta}\right) \in \mathbb{R}^{1\times m}$. Note that $\epsilon_{i,k,c'}$ can have different signs for different c's according to the sign of the $(J_{i,c'} - J_{i,c})\,\Delta\theta_k$ term.

Considering the weighted sum of (36) with the weights of the probability $p_{i,c'}$, i.e., the probability of x_i being class c' predicted from the model, we can approximate the directional derivative of the cross-entropy loss using $\epsilon_{i,k,c'}$ as follows:

$$\sum_{c'\neq c} p_{i,c'}\left(\frac{f_{i,c}(\theta) - f_{i,c'}(\theta)}{\epsilon_{i,k,c'}}\right) \approx \left(\sum_{c'\neq c} p_{i,c'} J_{i,c'} - (1 - p_{i,c})J_{i,c}\right)\Delta\theta_k = \left(\frac{\partial l_i}{\partial\theta}\right)\Delta\theta_k, \tag{37}$$

where we have used $\left(\frac{\partial l_i}{\partial\theta}\right) = -(e_{y_i}^\top - \sum_j p_{i,j} e_j^\top)J_i \in \mathbb{R}^{1\times m}$.

Our sharpness measure can then be related to the (signed) norm of the perturbation vector $\sqrt{\lambda_k}\epsilon_{i,k,c'}$ as follows:

$$\left(\frac{\partial l_i}{\partial\theta}\right) F^\dagger(\theta) \left(\frac{\partial l_i}{\partial\theta}\right)^\top = \sum_{k=1}^{m'} \frac{1}{\lambda_k}\left(\left(\frac{\partial l_i}{\partial\theta}\right)\Delta\theta_k\right)^2 \approx \sum_{k=1}^{m'}\left(\sum_{c'\neq c} p_{i,c'}\left(\frac{f_{i,c}(\theta) - f_{i,c'}(\theta)}{\sqrt{\lambda_k}\epsilon_{i,k,c'}}\right)\right)^2, \tag{38}$$

where the last approximation comes from applying (37). Therefore as in the binary classification case, we can observe that the smaller our sharpness measure, the larger the magnitude of the margin, i.e., the (signed) norm $\sqrt{\lambda_k}\epsilon_{i,k,c'}$ of the perturbation vector, in the information geometric sense.

F Adaptability of IGS to the square loss

Our measure can be defined for the square loss by considering the log-likelihood for a Gaussian, i.e., $l_i(\theta) = l(y_i, f(x_i;\theta)) = -\log p(y_i|x_i;\theta) = \frac{1}{2}(y_i - f(x_i;\theta))^2 + \text{const.}$ in (1) for $p(y|x;\theta) = C\exp(-\frac{1}{2}(y - f(x;\theta))^2)$ with C as a normalization constant. The corresponding reparametrization- and scale-invariance properties, as well as the connections with generalization in terms of TIC and margin, can all be seen as well.

More specifically, the invariance properties are satisfied by definition. Since we do not assume any specific form of the likelihood during the derivation of TIC, the TIC can be well-defined for the

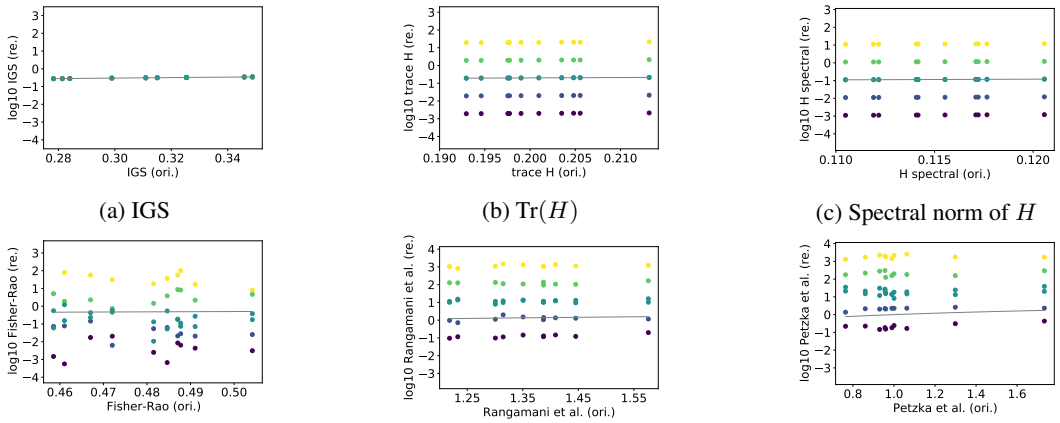
square loss [9], and the link between our measure and the TIC comes again from the fact that the FIM and the Hessian are approximately equal when the neural network is sufficiently trained for the square loss [28]. In terms of our definition of margin, if we let $l_i = \frac{1}{2}(z_i - f(x_i;\theta))^2$ ($z_i \in \{0,1\}$) and set the condition for data lying on the decision boundary as $f(x;\theta) = \frac{1}{2}$, we can show the connection of the measure to the margin without much difficulty according to a discussion similar to Section 3.3.2.

G Experimental details

G.1 Reparametrization invariance

We examine the reparametrization-invariance of the IGS and compare it to other sharpness measures such as the trace of the Hessian, the spectral norm (or the top eigenvalue) of the Hessian, the Fisher-Rao norm [26], the Rangamani et al.'s measure [40], and the Petzka et al.'s measure [39].[5]

We consider a two-layer fully connected neural network with 20-dimensional hidden units and ReLU activation functions for this experiment. The neural network is trained on two-dimensional synthetic data generated from mixtures of three Gaussians via SGD with a learning rate of 0.1 and a batch size of 40 until the training loss gets lower than 5e-3. After training ten neural networks initialized with different random seeds, we obtain reparametrized neural networks for each of them by considering an element-wise nonlinear reparametrization $\eta = g(\theta) = (|\theta - \hat{\theta}|^2 + b)^a(\theta - \hat{\theta}) + \hat{\eta}$, considered in Figure 5 in Section 5 of [8]. We consider several pairs of (a, b) for $a \in \{-0.5, 0.5\}$ and $b \in \{0.01, 0.1, 1\}$ in defining $g(\theta)$, with $\hat{\theta}$ set as the obtained parameter and $\hat{\eta}$ sampled from the standard normal distribution. The above sharpness measures are then calculated for both the original models (with respect to θ) and their reparametrized ones $\tilde{f}(\cdot;\eta) = f(\cdot;g^{-1}(\eta)) = f(\cdot;\theta)$ (with respect to η).



(a) IGS (b) Tr(H) (c) Spectral norm of H

(d) Fisher-Rao norm (e) Rangamani et al.'s measure (f) Petzka et al.'s measure

Figure 6: Sharpness measures evaluated for the original models vs. the logarithm (to base 10) of sharpness measures for the reparametrized models. We consider nonlinear reparametrizations of $\eta = g(\theta) = (|\theta - \hat{\theta}|^2 + b)^a(\theta - \hat{\theta}) + \hat{\eta}$ considered in [8] with several choices of (a, b), which are represented by different colors. The solid lines ($y = \log_{10}(x)$) denote the case of reparametrization-invariance.

In Figure 6, we plot the sharpness measures calculated for the original models on the x-axis and the logarithm of those for the reparametrized ones on the y-axis. Points with the same x-axis values correspond to reparametrizations from the identical original model with different choices of (a, b). The solid lines ($y = \log_{10}(x)$) denote the case in which the sharpness measures evaluated for the original model and the reparameterized model are the same. From the figure, we can observe that only the IGS is invariant to the considered reparametrizations, while the other measures including the scale-invariant ones can be arbitrarily varied in their magnitudes significantly. These figures are

[5] To evaluate the latter two sharpness measures we use the codes in the following URL: https://github.com/kampmichael/RelativeFlatnessAndGeneralization (Apache License 2.0).

merged into a single figure in Figure 2 of the manuscript, reporting the ratio of the sharpness measures evaluated for the reparameterized models to those evaluated for the original models.

G.2 Correlations of the IGS and other sharpness measures to the generalization gap

To investigate the correlations of the IGS and other sharpness measures to the generalization performances, we experiment using LeNet-5 [25] trained on MNIST and CIFAR-10 data sets via SGD. To obtain models possessing different generalization performances for MNIST data sets, we train the models via SGD with learning rates in $\{0.002, 0.005, 0.01, 0.02, 0.05, 0.1\}$ and batch sizes in $\{64, 128, 256, 512, 1024\}$, and the training is terminated when the training loss becomes lower than 0.01. For CIFAR-10 data sets, we use learning rates in $\{0.001, 0.002, 0.005, 0.01, 0.02, 0.05\}$ and the identical batch sizes to the above, and the training is terminated when the training loss gets lower than 0.1. All the experiments are performed for two different random seeds, and only the models with loss converged in a specified number of iterations are considered.

For the trained models, we compute the IGS and other sharpness measures such as the trace of the Hessian, the Fisher-Rao norm [26], the Rangamani et al.'s measure [40], and the Petzka et al.'s measure [39] similarly as explained in Appendix G.1. (When calculating our measure, we consider the expectation taken with respect to the data distribution (similarly to [45]), which is approximated by a finite sum of the integrands over the test data.) In Figure 3, the sharpness values are normalized to be in [0,1]. We can observe that the IGS has better correlation with the generalization gap, i.e., the difference between the train and test losses, than other sharpness measures in the figure.

In calculating the IGS, a crucial difference to the experiments in [45] is that we use LeNet-5, of which the number of parameters exceeds 60,000. Therefore, it is difficult to compute the exact values of IGS for the obtained models. Here we briefly explain how the IGS is evaluated in our experiments.

We first obtain the top-m' eigenvalues/eigenvectors of the FIM numerically via the power iteration, using subsampled data points $\{x_1, \ldots, x_{N_s}\}$ (with $N_s = 1,024$ in our experiments) from the data set as explained in Appendix G.5.1. The dimension m' of eigensubspace to consider (for the pseudo-inverse of the approximated FIM) is chosen to be the least number that makes the sum of top-m' eigenvalues exceed 95% of $\mathrm{Tr}(F(\theta))$.

To further reduce the computational costs, the mini-batch IGS in (7) is computed from mini-batch gradients (with a batch size of 128) obtained by iterating for an epoch as follows:

$$\mathrm{IGS}_b(\theta) \approx \frac{1}{|\mathcal{B}|} \sum_{B \in \mathcal{B}} \sum_{k=1}^{m'} \frac{1}{\lambda_k} \left(\left(\frac{1}{b} \sum_{x_i \in B} \frac{\partial l_i}{\partial \theta} \right) \Delta\theta_k \right)^2 ,$$

where $\mathcal{B}$ is the considered set of mini-batches. We then convert the mini-batch IGS to IGS using (27). We perform these procedures for five different subsamples to approximate the FIM, and average the obtained IGS values.

G.3 Regularization experiments – a toy example

G.3.1 A finite difference-based approximation of IGS

Consider the following eigendecomposition of the Fisher information matrix

$$F(\theta) = \sum_{k=1}^{m'} \lambda_k \Delta\theta_k \Delta\theta_k^\top , \tag{39}$$

where λ_k, $\Delta\theta_k$ are respectively the k-th eigenvalue and eigenvector of $F(\theta)$, and m' is the largest index of non-zero λ_k. The sharpness measure is then approximated as follows:

$$\text{IGS}(\theta) = \frac{1}{N}\sum_{i=1}^{N}\left(\frac{\partial l_i}{\partial \theta}\right)\left(\sum_{k=1}^{m'}\frac{1}{\lambda_k}\Delta\theta_k\Delta\theta_k^\top\right)\left(\frac{\partial l_i}{\partial \theta}\right)^\top \tag{40}$$

$$= \frac{1}{N}\sum_{i=1}^{N}\sum_{k=1}^{m'}\left(\frac{1}{\sqrt{\lambda_k}}\left(\frac{\partial l_i}{\partial \theta}\right)\Delta\theta_k\right)^2 \tag{41}$$

$$\approx \frac{1}{N\cdot\alpha^2}\sum_{i=1}^{N}\mathbb{E}_\epsilon\left[\left(l_i(\theta+\alpha\sum_{k=1}^{m'}\frac{\epsilon_k}{\sqrt{\lambda_k}}\Delta\theta_k) - l_i(\theta)\right)^2\right], \tag{42}$$

where $\mathbb{E}_\epsilon[\cdot]$ denotes the expectation with respect to an m'-dimensional vector $\epsilon = (\epsilon_1, \ldots, \epsilon_{m'}) \sim N(0, I)$ in (42), and the first-order approximation is used to derive (42) with $\alpha > 0$ as a small scalar.

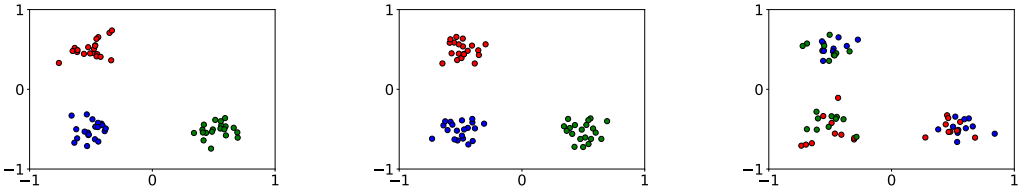
To avoid possible numerical instabilities stemming from the varying scales of the eigenvalues, we can consider following regularizer using a fixed amount of perturbation from θ in the first $l_i(\cdot)$-term in (42):

$$\text{IGS}(\theta) \approx \frac{1}{N\cdot\alpha^2}\sum_{i=1}^{N}\mathbb{E}_\epsilon\left[\|v_\epsilon\|^2\left(l_i(\theta+\alpha\frac{v_\epsilon}{\|v_\epsilon\|}) - l_i(\theta)\right)^2\right], \tag{43}$$

where $v_\epsilon \equiv \sum_{k=1}^{m'}\frac{\epsilon_k}{\sqrt{\lambda_k+\eta}}\Delta\theta_k$ is a perturbation vector with $\eta \geq 0$ included for numerical stability.

G.3.2 Data sets, models, and hyperparameters

For the toy examples in Section 4.1, we consider classifying two-dimensional synthetic data generated from mixtures of three Gaussians. As shown in Figure 7, we assign distinct labels to each Gaussian for training data. Test data are labeled analogously to the training data. For poison data, we assign labels different from training and test data to each Gaussian by randomly choosing labels for each data among the other two labels. We sample 60 data for each training, test, and poison data.



(a) Training data (b) Test data (c) Poison data

Figure 7: Data used in the experiments in Section 4.1.

A six-layer fully connected neural network with 100-dimensional hidden units and hyperbolic tangent (Tanh) activation functions is used for the experiments. We use a learning rate of 0.1 and a batch size of 30 for both SGD with and without regularization. The hyperparameters for regularization are set as $\alpha = 0.5$, $\tilde{\rho} = \rho/\alpha^2 = 2.0$, and $\eta = 0.01$. We use the top three eigenvalues and eigenvectors (i.e., $m' = 3$ for v_ϵ in (43)) obtained from the PyHessian library [48][6] to approximate the regularization term as in (43) and update them every 20 iterations. We refer the reader to Appendix G.5.1 for additional details of the eigendecomposition.

For a better interpretation of the loss trajectory, we have smoothed the trajectory by averaging 20 consecutive loss values in Figures 5 (c) and (d).

[6]URL: https://github.com/amirgholami/PyHessian (MIT license).

G.4 Regularization experiments – MNIST and CIFAR-10/100

G.4.1 Approximations of the natural gradients

As mentioned in Section 4.2, it is required to approximate the natural gradient $F(\theta)^{\dagger}\left(\frac{\partial l}{\partial \theta}\right)^{\top} \in \mathbb{R}^m$ in (13) to regularize the mini-batch IGS. In this experiment, we consider some approximations based on the eigenvalue-corrected Kronecker factorization (EKFAC) method [13][7] to efficiently calculate the FIM and its (pseudo-)inverse. (We provide a brief explanation of the EKFAC method in Appendix G.5.2.) We devise three different methods for the approximation; the methods differ in detail from each other in terms of which labels are used to approximate the FIM or to calculate $\left(\frac{\partial l}{\partial \theta}\right)$ in the EKFAC method. The methods are classified as follows:

- When approximating $F(\theta)$ in [20], they use sampled labels according to the probabilistic model that the neural network models with current parameter values. We follow this idea and use sampled labels to approximate $F(\theta)$ and the original labels for $\left(\frac{\partial l}{\partial \theta}\right)$ in our first method denoted as **Ours 1** in Table 2.
- The second method uses original labels for both $F(\theta)$ and $\left(\frac{\partial l}{\partial \theta}\right)$ and is denoted as **Ours 2** in Table 2. This method can be considered as utilizing the empirical Fisher discussed in [23].
- The third method uses sampled labels for both $F(\theta)$ and $\left(\frac{\partial l}{\partial \theta}\right)$ and is denoted as **Ours 3** in Table 2. (The results from this method are reported in Table 1, and the method is denoted as **Ours** in the table.)

Table 2 reports the averages and standard errors of the test classification accuracies obtained from three runs of all of our regularization methods as well as SGD, GR, SAM, and ASAM methods. All of our methods show superior performance to SGD, and the third method (Ours 3) shows the best performance among them for most of the considered settings. The experimental details are provided in the next section.

Table 2: Averages and standard errors of the test classification accuracies for SGD, GR, SAM, ASAM, and SGD with our regularization methods on MNIST, CIFAR-10, and CIFAR-100 data sets.

Data set	Model	SGD	GR	SAM	ASAM	Ours 1	Ours 2	Ours 3
MNIST	3FCN	98.13± 0.05	98.20± 0.01	98.62 ± 0.01	**98.65** ± 0.07	98.52 ± 0.05	98.56 ± 0.02	**98.65** ± 0.03
CIFAR-10	VGG11-BN	92.75 ± 0.11	93.13 ± 0.14	93.61 ± 0.08	93.78 ± 0.13	93.67 ± 0.10	93.74 ± 0.09	**93.81** ± 0.06
	ResNet-20	92.68 ± 0.20	93.05 ± 0.21	93.46 ± 0.11	**93.65** ± 0.20	93.42 ± 0.11	93.31 ± 0.03	93.40 ± 0.16
	ResNet-56	94.08 ± 0.17	94.40 ± 0.23	95.22 ± 0.11	**95.36** ± 0.06	95.08 ± 0.26	95.05 ± 0.08	94.96 ± 0.01
	WRN-16-4	95.37 ± 0.16	95.43 ± 0.11	96.19 ± 0.12	**96.37** ± 0.18	95.98 ± 0.04	96.04 ± 0.04	96.01 ± 0.10
	WRN-16-8	95.98 ± 0.03	96.15 ± 0.10	96.70 ± 0.14	**97.03** ± 0.06	96.67 ± 0.08	96.64 ± 0.02	96.57 ± 0.10
CIFAR-100	VGG11-BN	72.25 ± 0.17	73.05 ± 0.32	73.69 ± 0.20	73.61 ± 0.18	73.69 ± 0.16	73.64 ± 0.39	**74.24** ± 0.22
	ResNet-20	69.01 ± 0.54	70.00 ± 0.27	70.57 ± 0.30	**70.82** ± 0.23	70.75 ± 0.14	70.40 ± 0.58	70.63 ± 0.26
	ResNet-56	73.14 ± 0.10	74.67 ± 0.25	75.90 ± 0.25	75.87 ± 0.27	75.98 ± 0.25	75.91 ± 0.13	**76.07** ± 0.11
	WRN-16-4	77.18 ± 0.36	78.02 ± 0.21	79.64 ± 0.13	**79.97** ± 0.11	79.70 ± 0.10	79.57 ± 0.13	**79.97** ± 0.07
	WRN-16-8	80.55 ± 0.33	81.13 ± 0.06	82.20 ± 0.11	82.38 ± 0.09	81.95 ± 0.03	81.92 ± 0.42	**82.44** ± 0.29

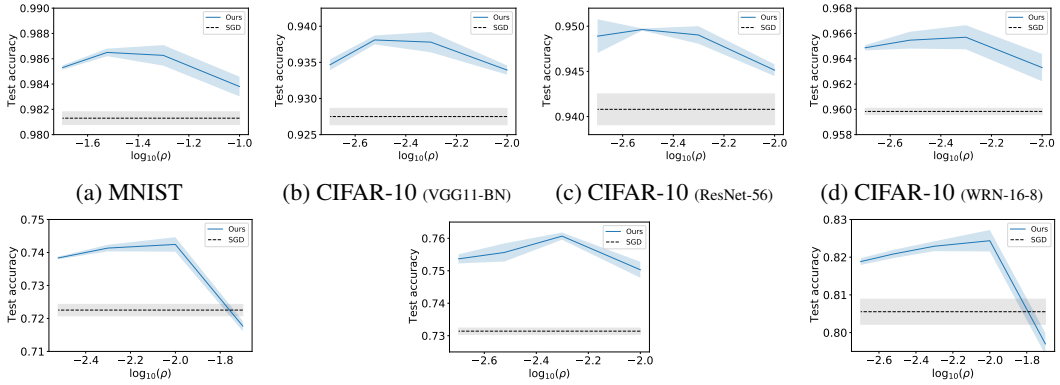
G.4.2 Models and hyperparameters

For the experiments using MNIST data, a three-layer fully connected neural network with 200-dimensional hidden units and ReLU activation functions is used. The neural network is trained with a learning rate of 0.1 and a batch size of 128 for 200 epochs.

For CIFAR-10/100 experiments, convolutional neural networks such as VGG [42], ResNet [15], and WideResNet [50] are used. The neural networks are trained with a batch size of 128 and a weight decay coefficient of 5e-4 for 200 epochs. We additionally apply data augmentation and label smoothing [30] methods, and the learning rates are scheduled according to the cosine annealing method [27] with the maximum learning rate of 0.1 and 10 epochs to warm up.

In approximating the natural gradient using the EKFAC method [13], we set the (Kronecker-factored) eigenbasis update frequency of 100 and 500 for the MNIST and CIFAR-10/100 experiments, respectively. We also apply the running average option in correcting the scaling of natural gradients. The

[7] URL: https://github.com/Thrandis/EKFAC-pytorch (MIT License).



(a) MNIST (b) CIFAR-10 (VGG11-BN) (c) CIFAR-10 (ResNet-56) (d) CIFAR-10 (WRN-16-8)

(e) CIFAR-100 (VGG11-BN) (f) CIFAR-100 (ResNet-56) (g) CIFAR-100 (WRN-16-8)

Figure 8: Test accuracies according to varying ρ.

hyperparameters for our regularization method are chosen among $\rho \in \{0.01, 0.02, 0.03, 0.05, 0.1\}$ for the MNIST experiments and $\rho \in \{0.001, 0.002, 0.003, 0.005, 0.01, 0.02\}$ for the CIFAR-10/100 experiments. To provide the sensitivity to the hyperparameter ρ of our method (the thrid one proposed in Appendix G.4.1), we have depicted the results under various choices of ρ in Figure 8. We consider the regularization coefficient $\rho \in \{5e-4, 0.001, 0.002, 0.003, 0.005, 0.01, 0.02, 0.03, 0.05\}$ for the squared gradient norm in the GR method [43]. We consider $\rho \in \{0.05, 0.1, 0.2, 0.3, 0.5, 1.0\}$ for the SAM method [12], and consider $\rho \in \{0.3, 0.5, 1.0, 2.0, 3.0, 5.0\}$ and $\eta = 0.01$ for the ASAM method [24], where the hyperparameters ρ and η are defined as in [12, 24].[8] We report the results obtained from hyperparameters showing the best test accuracy in Tables 1 and 2, where the results are averaged over trials from three different random seeds. Due to the space constraints, in Table 1 of the manuscript, we report only the results from VGG11-BN, ResNet-56, and WRN-16-8 for CIFAR-10/100 experiments and only the third method among the methods introduced in Appendix G.4.1.

Under our experimental settings, elapsed times per epoch (in seconds) during training for SGD/GR/SAM/ASAM/Ours are 1.68/3.50/3.60/3.85/5.70 in MNIST experiments. For CIFAR-10/100 experiments, those are 5.95/24.7/14.8/16.1/40.1 for VGG11-BN, 25.7/98.3/63.7/68.3/66.9 for ResNet-56, and 24.2/145/47.8/48.0/80.7 for WRN-16-8. Most of the experiments have been performed using NVIDIA Tesla V100 GPUs.

Note that in our regularization method, the process of finding the natural gradient in $\delta = \rho F(\theta)^{\dagger} \left(\frac{\partial l}{\partial \theta}\right)^{\top}$ in (13) is added once every iteration compared to vanilla SGD. Therefore, additional time complexity depends on the details of methods to obtain the natural gradients, i.e., the EKFAC method (see Appendix G.5.2).

G.5 Details for the eigendecomposition of the FIM

Several approximations have been used in our experiments for the eigendecomposition (required to get the pseudo-inverse) of the FIM. The computations in Sections 3.3.1 and 4.1 are based on the power iteration, and those in Section 4.2 are based on the EKFAC method. In this section, we explain some details of these approximations.

G.5.1 The power iteration

In Section 4.1, we utilize the functions of the PyHessian library [48], which are based on the power iteration (see Algorithm 2 in [48]), to obtain the top few eigenvalues/eigenvectors of the Hessian, i.e., λ_k and $\Delta\theta_k$ in (39). (Note that we can consider the eigenvalues/eigenvectors of the Hessian here since we can approximate the Hessian to the FIM.)

In Section 3.3.1 and Appendix G.2, to obtain the eigenvalues/eigenvectors of the FIM we first subsample N_s data points $\{x_1, \ldots, x_{N_s}\}$ from the data set. We then calculate the Jacobian of

[8] URLs: https://github.com/SamsungLabs/ASAM (Apache License 2.0), https://github.com/davda54/sam (MIT license).

the loss for the sampled labels with respect to the parameters, i.e., $\tilde{J} = \frac{\partial \tilde{l}}{\partial \theta} \in \mathbb{R}^{N_s \times m}$, where $\tilde{l} = (l(\tilde{y}_1, f(x_1;\theta)), \ldots, l(\tilde{y}_{N_s}, f(x_{N_s};\theta))) \in \mathbb{R}^{N_s}$ and the labels $\tilde{y}_i$ are sampled according to the probabilistic model $p(y|x_i;\theta)$ that the neural network models for $i = 1, \ldots, N_s$ as discussed in [20]. The FIM can then be approximated as $F(\theta) \approx \frac{1}{N_s}\tilde{J}^\top \tilde{J}$, and its top-$m'$ eigenvalues/eigenvectors are numerically obtained via the power iteration in Algorithm 1. Note that the time complexity of the multiplication $F(\theta)v \approx \frac{1}{N_s}\tilde{J}^\top(\tilde{J}v)$ in line 5 of Algorithm 1 is $O(N_s m^2)$, where $v \in \mathbb{R}^m$ is an approximation of the eigenvector during the iteration.

Algorithm 1 Power Iteration for the Computation of Top Eigenvalues/Eigenvectors of the FIM

Input: The Jacobian of the loss for sampled labels with respect to parameters $\tilde{J} = \frac{\partial \tilde{l}}{\partial \theta} \in \mathbb{R}^{N_s \times m}$, the dimension of top eigenvalues/eigenvectors to compute m', the number of iterations N_{iter}.
Initialize: The top eigenvalue vector $\lambda = 0 \in \mathbb{R}^{m'}$, the top eigenvector matrix $V = 0 \in \mathbb{R}^{m \times m'}$.
Iteration:

1: **for** $k = 1, \ldots, m'$ **do**
2: Draw a random vector $v \in \mathbb{R}^m$ from $N(0, I)$.
3: **for** $i = 1, \ldots, N_{\text{iter}}$ **do**
4: Project v to the orthogonal complement of V and normalize, $v = \frac{(I - VV^\top)v}{\|(I - VV^\top)v\|}$.
5: Compute $F(\theta)v \approx \frac{1}{N_s}\tilde{J}^\top(\tilde{J}v)$ and reset v, $v = F(\theta)v$.
6: **end for**
7: Project v to the orthogonal complement of V and normalize, $v = \frac{(I - VV^\top)v}{\|(I - VV^\top)v\|}$.
8: Update the top eigenvector matrix V, set $V_k = v \in \mathbb{R}^m$, where V_k is the k-th column of V.
9: Update the top eigenvalue vector λ, set $\lambda_k = v^\top F(\theta) v$, where λ_k is the k-th element of λ.
10: **end for**

Output: The top eigenvalues/eigenvectors $\lambda \in \mathbb{R}^{m'}, V \in \mathbb{R}^{m \times m'}$.

In implementing the power iteration, we handle matrices of size $N_s \times m$ with m as the neural network parameter size. Since memory usage is proportional to the network size, additional approximations may be needed to handle larger neural networks.

G.5.2 The EKFAC method

The Kronecker factorization (KFAC) method is developed to efficiently implement the natural gradient descent algorithm [29]. The method approximates the FIM as layer-wise block-diagonal and efficiently performs eigendecomposition via Kronecker factorization for each block. The eigenvalue-corrected KFAC (EKFAC) method additionally performs an eigenvalue correction step to make the decomposition based on the Kronecker-factored eigenbasis closer to the FIM [13]. (Compared to the power iteration, these methods are based on stronger approximations and may be more inaccurate, but we can apply them to larger neural networks.)

The time complexity of the EKFAC method (for obtaining $\delta = \rho F(\theta)^\dagger \left(\frac{\partial l}{\partial \theta}\right)^\top$ in (13)) becomes the addition of that for computing gradient $\left(\frac{\partial l}{\partial \theta}\right)^\top$, that for solving layer-wise eigenvalue problems to obtain $F(\theta)^\dagger$, and that for the multiplication of $F(\theta)^\dagger$ with $\left(\frac{\partial l}{\partial \theta}\right)^\top$. Due to the Kronecker factorization, the time complexity for solving the eigenvalue problems per layer becomes $O(d^3)$, where d denotes the typical layer input and output node sizes. (Note that in the EKFAC method, the eigenvalue problem is usually solved per several tens or hundreds of iteration steps.) After that, $F(\theta)^\dagger$ is not obtained explicitly but considered in the decomposed form, i.e., $F(\theta)^\dagger = US^{-1}U^\top$ for the eigendecomposition of $F(\theta) = USU^\top$. The multiplication of $F(\theta)^\dagger$ with $\left(\frac{\partial l}{\partial \theta}\right)^\top$ is done layer-wisely and efficiently by using the mixed Kronecker matrix-vector product with complexity $O(d^3)$ per layer [13].

The final complexity for the EKFAC method is $O(bLd^2) + O(Ld^3)$, where $O(bLd^2)$ is that for computing gradient (hence that for SGD), L is the number of layers, and b is the mini-batch size. For a more detailed analysis of the coefficients of each term, we refer the reader to Section 8 of [29].