Implicit Jacobian Regularization Weighted with Impurity of Probability Output

Sungyoon Lee [1] **Jinseong Park** [2] **Jaewook Lee** [2]

Abstract

The success of deep learning is greatly attributed to stochastic gradient descent (SGD), yet it remains unclear how SGD finds well-generalized models. We demonstrate that SGD has an implicit regularization effect on the logit-weight Jacobian norm of neural networks. This regularization effect is weighted with the *impurity* of the probability output, and thus it is active in a certain phase of training. Moreover, based on these findings, we propose a novel optimization method that explicitly regularizes the Jacobian norm, which leads to similar performance as other state-of-the-art sharpness-aware optimization methods.

1. Introduction

Deep learning has shown to great promise for many learning tasks in various areas. Numerous studies have aimed to understand how learning algorithms lead to the successful training of deep neural networks. In particular, it is crucial to understand the geometric properties of the loss landscape of neural networks and their interaction with gradient-based optimization methods, including stochastic gradient descent (SGD), along the training trajectory. These properties have been studied from the perspectives of both optimization (Gur-Ari et al., 2018; Jastrzębski et al., 2019; Ghorbani et al., 2019; Liu et al., 2020; Lewkowycz et al., 2020; Cohen et al., 2021) and generalization (Hochreiter & Schmidhuber, 1997; Keskar et al., 2017; Dinh et al., 2017; Jastrzębski et al., 2017; Wang et al., 2018; Chaudhari et al., 2019; Jiang et al., 2020; Barrett & Dherin, 2021; Smith et al., 2021).

We investigate the Hessian of the training loss (with respect to the model parameters) and its top eigenvalue (also called *sharpness*). The sharpness characterizes the dynamics of neural network training along the optimization trajectory and appears to be correlated with the generalization capability. To be specific, the sharpness initially increases, and after reaching a certain value, the training dynamics become unstable and the iterate oscillates along the top eigenvector of the Hessian (Jastrzębski et al., 2019; 2020; Cohen et al., 2021). Moreover, the rapid increase in the sharpness during the early phase significantly impacts the final generalization performance (Achille et al., 2019; Golatkar et al., 2019; Jastrzębski et al., 2020; 2021; Lewkowycz et al., 2020; Fort et al., 2020). However, the Hessian of a deep neural network is high-dimensional which makes it difficult to analyze its eigensystem and the sharpness.

In this paper, we investigate the Hessian through two low-dimensional matrices: the *logit Hessian* and the *logit-weight Jacobian* (defined in Definition 3.1). Especially, we analyze the top eigenvalue of the logit Hessian. By doing so, we can provide a simple and intuitive explanation of the relation between the sharpness, the probability output of the classification model, and the Jacobian matrix. This enables us to understand how the sharpness of the loss landscape influences the learning dynamics and the generalization performance. We summarize the main contributions as follows:

- We provide connections between the top eigenvalue of the logit Hessian and the *impurity* of the probability output (Theorem 4.1 in Section 4.1 and 4.2).
- We derive a relation between the sharpness, the top eigenvalue of the logit Hessian, and the Jacobian norm (Theorem 5.1 in Section 5.1), which we call *sharpness-impurity-Jacobian relation*.
- We describe how the sharpness of the loss landscape influences the learning dynamics and the generalization performance (Section 5.2). In particular, we demonstrate that SGD has implicit effects on penalizing the Jacobian norm (*Implicit Jacobian Regularization*) during a certain phase of training (*Active Regularization Period*).
- We also evaluate the *Explicit Jacobian Regularization*, which outperforms state-of-the-art sharpness-aware optimization methods, SAM (Foret et al., 2021) and ASAM (Kwon et al., 2021) (Table 1 in Section 5.3).

[1]Department of Computer Science, Hanyang University [2]Department of Industrial Engineering, Seoul National University. Correspondence to: Sungyoon Lee <sungyoonlee@hanyang.ac.kr>.

Proceedings of the 40^{th} International Conference on Machine Learning, Honolulu, Hawaii, USA. PMLR 202, 2023.

2. Related Work

We summarize some previous studies on the Hessian, learning dynamics, and generalization of neural networks.

Decomposition of the Hessian LeCun et al. (2012); Dauphin et al. (2014); Sagun et al. (2016; 2017) empirically found that the eigenvalue spectrum of the Hessian during training is composed of two parts: the bulk (concentrated around zero) and the outliers (scattered positively away from zero). Further, Papyan (2019; 2020) proposed a three-level hierarchical decomposition of the Hessian according to each label, logit coordinate, and example, using a well-known Gauss-Newton decomposition. Although we also use the Gauss-Newton decomposition, we focus on the top eigenvalue of the logit Hessian for which the predicted class is more important than the true label (detailed in Section 4.2).

Issues with the SDE modeling of SGD In many studies, SGD has been understood as a stochastic differential equation (SDE) in the limit of vanishing learning rate (Mandt et al., 2016; 2017; Hu et al., 2019; Li et al., 2017; 2019a; Smith & Le, 2018; Chaudhari & Soatto, 2018; Jastrzębski et al., 2017; Zhu et al., 2019; Park et al., 2019). However, Yaida (2019, Section 2.3.3) raised some theoretical concerns for such approximations, and Li et al. (2021) derived a sufficient condition for the SDE modeling to fail. Moreover, Barrett & Dherin (2021) argued that the SDE analysis within the limit of vanishing learning rate cannot explain the generalization benefits of finite learning rates, and proposed a modified gradient flow for finite learning rates. However, they still consider a continuous flow, which has a fundamental limitation in explaining the (discrete) oscillatory behavior with iterative catapult in a practical learning rate regime (Smith et al., 2021), as shown in the following paragraph.

Oscillatory catapult and the plateau of sharpness Xing et al. (2018) investigated the roles of the learning rate and batch size in the SGD dynamics. The authors observed that SGD explores the parameter space, bouncing between walls of valley-like regions. A large learning rate maintains a high valley height, and a small batch size induces gradient stochasticity. Both of them help exploring through the parameter space with different roles in the training dynamics. Jastrzębski et al. (2019) empirically investigated the evolution of the sharpness along the whole training trajectory of SGD. They observed the initial growth of the sharpness (dubbed *progressive sharpening* in Cohen et al. (2021)) as the loss decreases, reaching a maximum sharpness determined by learning rate and batch size, which then decreases towards the end of the training. Due to the progressive sharpening, the SGD step becomes excessively large compared to the shape of the loss landscape. This is consistent with the valley-like structure shown in Xing et al. (2018). Lewkowycz et al. (2020) investigated simple theoretical models with solvable training dynamics. They showed that, in their setup with a large learning rate, the loss initially increases whereas the sharpness decreases, converging at a flat minimum. This mechanism is called *catapult mechanism*. Recently, Cohen et al. (2021) found that full-batch GD typically operates in a regime called the *Edge of Stability* where the sharpness can no longer increase and stays near a certain value, and the training loss behaves non-monotonically but decreases globally. This optimization behavior at the Edge of Stability can be seen as repeated catapult mechanisms. The authors explicitly marked the limit of the sharpness with $2/\eta$ where η is a learning rate. To describe the aforementioned evolution of the sharpness, Fort & Ganguli (2019) developed a theoretical model based on a random matrix modelling. To build a simple random model, they introduced assumptions regarding the gradients and Hessians, which are i.i.d. isotropic Gaussian with zero mean and varying variance during training. Whereas they focus on building a random model based on observation, we aim to explain the underlying mechanisms.

Implicit bias of SGD There have been many studies on the implicit bias of SGD (Neyshabur, 2017; Zhang et al., 2021; Soudry et al., 2018; Jastrzębski et al., 2020). Jastrzębski et al. (2021) empirically showed that SGD implicitly penalizes the trace of the Fisher Information Matrix (FIM). They also showed that the trace of FIM explodes during the early phase of training when using a small learning rate, which they call catastrophic Fisher explosion. Barrett & Dherin (2021) and Smith et al. (2021) demonstrated that SGD implicitly penalizes the norm of the total gradient and the non-uniformity of the mini-batch gradients. We demonstrate that the (logit-weight) Jacobian plays an important role in the generalization in each case.

3. Background

In this section, we provide some notations, basic equations, and definitions for the following sections. In this paper, we use the denominator layout notation for the vector derivatives, i.e., $\nabla_{\boldsymbol{v}}\boldsymbol{u} \equiv \left(\frac{\partial \boldsymbol{u}_j}{\partial \boldsymbol{v}_i}\right)_{ij} \in \mathbb{R}^{v\times u}$, where $\boldsymbol{u} : \mathbb{R}^v \to \mathbb{R}^u$ and $\boldsymbol{v} \in \mathbb{R}^v$.

We consider a problem of learning a C-class classifier that maps an input $\boldsymbol{x} \in \mathcal{X} \subset \mathbb{R}^d$ to a target label $y \in [C] = \{1, 2, \cdots, C\}$. To this end, we build a parameterized model $f_{\boldsymbol{\theta}} : \mathcal{X} \to \mathcal{Z} \subset \mathbb{R}^C$ with a model parameter $\boldsymbol{\theta} \in \Theta \subset \mathbb{R}^m$ that outputs a logit vector $\boldsymbol{z} \equiv f_{\boldsymbol{\theta}}(\boldsymbol{x}) \in \mathcal{Z} \subset \mathbb{R}^C$ (we often omit the dependence on $\boldsymbol{x}$ and $\boldsymbol{\theta}$). The logit vector $\boldsymbol{z}$ is then given as an input to the softmax function to yield a probability output $\boldsymbol{p} \equiv \mathrm{softmax}(\boldsymbol{z}) \in \Delta^{C-1}$, where $\Delta^{C-1} \equiv \{\boldsymbol{p} \in [0,1]^C : \mathbf{1}^T\boldsymbol{p} = 1, \boldsymbol{p} \geq \mathbf{0}\}$. We want

the model to match the most probable class $\boldsymbol{c}_1$ to the true label y, where $\boldsymbol{c}(\boldsymbol{x}) \equiv \arg\operatorname{sort}(\boldsymbol{p})$ in descending order. We exchangeably denote the probability value corresponding to the true label y as $p \equiv \boldsymbol{p}_y \in [0, 1]$. The cross-entropy loss, $l = l(\boldsymbol{z}, y) \in \mathbb{R}$, is equivalent to the negative log-likelihood $l \equiv -\log p$. We use the notations $\|\boldsymbol{v}\|$ for the Euclidean norm of a vector $\boldsymbol{v}$ and $\|\boldsymbol{A}\| \equiv \max_{\|\boldsymbol{v}\|=1} \|\boldsymbol{A}\boldsymbol{v}\|$ for the Euclidean operator norm, $\|\boldsymbol{A}\|_F \equiv \sqrt{\sum_{i,j} |\boldsymbol{A}_{ij}|^2}$ for the Frobenius norm of a matrix $\boldsymbol{A}$, and the spectral norm $\|\boldsymbol{S}\|_\sigma$ and the trace $\operatorname{tr}(\boldsymbol{S}) \equiv \sum_i \boldsymbol{S}_{ii}$ of a square matrix $\boldsymbol{S}$.

Starting with a simple computation of the derivatives of the softmax function in (1) (see Appendix C.1), we can easily derive the following equations in order:

$$\nabla_{\boldsymbol{z}} \boldsymbol{p} = \operatorname{diag}(\boldsymbol{p}) - \boldsymbol{p}\boldsymbol{p}^T \in \mathbb{R}^{C\times C} \tag{1}$$

$$\nabla_{\boldsymbol{z}} p = [\nabla_{\boldsymbol{z}} \boldsymbol{p}]_{:,y} = p(\boldsymbol{e}^y - \boldsymbol{p}) \in \mathbb{R}^C \tag{2}$$

$$\nabla_{\boldsymbol{z}} l = \nabla_{\boldsymbol{z}} p \frac{\partial l}{\partial p} = p(\boldsymbol{e}^y - \boldsymbol{p}) \cdot -\frac{1}{p} = \boldsymbol{p} - \boldsymbol{e}^y \in \mathbb{R}^C \tag{3}$$

$$\nabla^2_{\boldsymbol{z}} l = \nabla_{\boldsymbol{z}} (\nabla_{\boldsymbol{z}} l) = \nabla_{\boldsymbol{z}} (\boldsymbol{p} - \boldsymbol{e}^y) = \operatorname{diag}(\boldsymbol{p}) - \boldsymbol{p}\boldsymbol{p}^T \tag{4}$$

where $\operatorname{diag}(\boldsymbol{p}) \equiv (\delta_{ij}\boldsymbol{p}_i)_{ij} \in \mathbb{R}^{C\times C}$ is a diagonal matrix with $\boldsymbol{p}$ as its diagonal entries, and $\boldsymbol{e}^i \equiv (\delta_{ij})_j \in \mathbb{R}^C$ is a one-hot vector with the i-th element as 1.

Next, the Hessian of the loss function l for the given example $\boldsymbol{x}$ with respect to the model parameter can be expressed as follows:

$$\nabla^2_{\boldsymbol{\theta}} l = \nabla_{\boldsymbol{\theta}} \boldsymbol{z} \nabla^2_{\boldsymbol{z}} l \nabla_{\boldsymbol{\theta}} \boldsymbol{z}^T + \sum\nolimits_{c=1}^{C} \nabla^2_{\boldsymbol{\theta}} \boldsymbol{z}_c \nabla_{\boldsymbol{z}_c} l \tag{5}$$

$$\approx \nabla_{\boldsymbol{\theta}} \boldsymbol{z} \nabla^2_{\boldsymbol{z}} l \nabla_{\boldsymbol{\theta}} \boldsymbol{z}^T \in \mathbb{R}^{m\times m} \tag{6}$$

using the well-known Gauss-Newton approximation (see Appendix C.2).

Now, we are ready to consider the training loss for the training set $\mathcal{D}$. We compute the total training loss over $\mathcal{D}$ as $L \equiv \langle l \rangle$ which yields $\nabla L = \langle \nabla l \rangle$ and $\nabla^2 L = \langle \nabla^2 l \rangle$, where $\langle \cdot \rangle$ is the expectation over the empirical measure of the training set $\mathcal{D}$, i.e., $\langle \cdot \rangle \equiv \hat{\mathbb{E}}_{\mathcal{D}}[\cdot]$. We use the notation $\langle \cdot \rangle_{\mathcal{B}}$ when averaging over a mini-batch $\mathcal{B}$, i.e., $\langle \cdot \rangle_{\mathcal{B}} \equiv \hat{\mathbb{E}}_{\mathcal{B}}[\cdot]$. Following from (4) and (6), we define the Hessian $\boldsymbol{H}$ for the total loss and its Gauss-Newton approximation matrix $\boldsymbol{G}$ with matrices $\boldsymbol{M}$ and $\boldsymbol{J}$ as follows:

Definition 3.1. We call $\boldsymbol{M}$ the logit Hessian, $\boldsymbol{J}$ the Jacobian (of the logit function with respect to the model parameter), $\boldsymbol{H}$ the Hessian, and $\boldsymbol{G}$ the Gauss-Newton approximation, which are defined as follows:

$$\boldsymbol{M} \equiv \nabla^2_{\boldsymbol{z}} l = \operatorname{diag}(\boldsymbol{p}) - \boldsymbol{p}\boldsymbol{p}^T \in \mathbb{R}^{C\times C} \tag{7}$$

$$\boldsymbol{J} \equiv \nabla_{\boldsymbol{\theta}} \boldsymbol{z} \in \mathbb{R}^{m\times C} \tag{8}$$

$$\boldsymbol{H} \equiv \langle \nabla^2_{\boldsymbol{\theta}} l \rangle \approx \langle \boldsymbol{J}\boldsymbol{M}\boldsymbol{J}^T \rangle \equiv \boldsymbol{G} \in \mathbb{R}^{m\times m} \tag{9}$$

It is interesting to note that while l is dependent on the true label y, the logit Hessian $\boldsymbol{M} = \nabla^2_{\boldsymbol{z}} l$ is independent of y, as are $\boldsymbol{J}$, $\boldsymbol{J}\boldsymbol{M}\boldsymbol{J}^T$, and $\boldsymbol{G}$. From (9), although we often use the approximation $\|\boldsymbol{H}\|_\sigma \approx \|\boldsymbol{G}\|_\sigma$ as justified by Sagun et al. (2017) and Fort & Ganguli (2019), this approximation occasionally fails during the later phase of training when the top eigenvalues of the Gauss-Newton matrix are insufficiently isolated from the bulk near 0 (Papyan, 2018). Thus, we mainly focus on the early phase of training.

4. Logit Hessian and Impurity

In the previous section, we introduced the Gauss-Newton approximation $\boldsymbol{G}$ of the Hessian $\boldsymbol{H}$, and decomposition of $\boldsymbol{G}$ with the Jacobian $\boldsymbol{J}$ and the logit Hessian $\boldsymbol{M}$, i.e., $\boldsymbol{G} = \langle \boldsymbol{J}\boldsymbol{M}\boldsymbol{J}^T \rangle$. Now, we focus on the logit Hessian $\boldsymbol{M}$ and its eigendecomposition, estimate the top eigenvalue of $\boldsymbol{M}$ with upper/lower bounds (Section 4.1), and explore the evolution of the top eigenvalue during training (Section 4.2).

4.1. Bounds on the Eigenvalues of the Logit Hessian

The lower-dimensional logit Hessian matrix $\boldsymbol{M} \in \mathbb{R}^{C\times C}$ is simple and fully characterized by only the probability vector $\boldsymbol{p}$ as $\boldsymbol{M} = \operatorname{diag}(\boldsymbol{p}) - \boldsymbol{p}\boldsymbol{p}^T$ in (7), however it turns out to be important for understanding the much higher-dimensional matrix $\boldsymbol{G} \in \mathbb{R}^{m\times m}$ ($C \ll m$). Because $\boldsymbol{M} = \operatorname{diag}(\boldsymbol{p}) - \boldsymbol{p}\boldsymbol{p}^T$ is a rank-one modification of a simple diagonal matrix $\operatorname{diag}(\boldsymbol{p})$, we can obtain its eigenvalues $\{\lambda^{(i)}\}_{i=1}^C$ and eigenvectors $\{\boldsymbol{q}^{(i)}\}_{i=1}^C$ from the theory of the rank-one modification of the eigenproblem (see, for example, Bunch et al. (1978); Golub (1973); Golub & Van Loan (2013)) where $\lambda^{(i)} \in \mathbb{R}$ is the i-th largest eigenvalue of $\boldsymbol{M}$ and $\boldsymbol{q}^{(i)} \in \mathbb{R}^C$ is its corresponding eigenvector. We also use the same ordered index of $(i) \in [C]$ with parentheses for the probability output $\boldsymbol{p} \in \Delta^{C-1}$, i.e., $\boldsymbol{c}_i = (i)$ and $\boldsymbol{p}_{(1)} \geq \boldsymbol{p}_{(2)} \geq \cdots \geq \boldsymbol{p}_{(C)}$, because this ordering is related to the eigenvalues $\{\lambda^{(i)}\}_{i=1}^C$ as shown in the following theorem.

Theorem 4.1 (Eigensystem of the logit Hessian $\boldsymbol{M}$)**.** *The eigenvalues $\lambda^{(i)}$ ($\lambda^{(1)} \geq \lambda^{(2)} \geq \cdots \geq \lambda^{(C)}$) and the corresponding eigenvectors $\boldsymbol{q}^{(i)}$ of the logit Hessian $\boldsymbol{M} = \nabla^2_{\boldsymbol{z}} l = \operatorname{diag}(\boldsymbol{p}) - \boldsymbol{p}\boldsymbol{p}^T$ satisfy the following properties:*

(a) The eigenvalue $\lambda^{(i)}$ is the i-th largest solution of the following equation:

$$v(\lambda) = 1 - \sum\nolimits_{i=1}^{C} \boldsymbol{p}_i^2 (\boldsymbol{p}_i - \lambda)^{-1} = 0 \tag{10}$$

(b) The eigenvector $\boldsymbol{q}^{(i)}$ is aligned with the direction of $(\operatorname{diag}(\boldsymbol{p}) - \lambda^{(i)}\boldsymbol{I})^{-1}\boldsymbol{p}$

(c) $\boldsymbol{p}_{(i+1)} \leq \lambda^{(i)} \leq \boldsymbol{p}_{(i)}$ for $1 \leq i \leq C-1$, and $\lambda^{(C)} = 0$

(d) $\frac{1}{2}Gini(\boldsymbol{p}_{(1)}) \leq \lambda^{(1)} \leq Gini(\boldsymbol{p}_{(1)})$*, where* $Gini(q) = 2q(1-q)$ *is the Gini impurity for the binary* $(q, 1-q)$.

We defer the proof to Appendix C.3. In the main text, we mainly focus on investigating the top eigenvalue $\lambda^{(1)}$ of $\boldsymbol{M}$ by utilizing Theorem 4.1 (c) and (d), which provide the upper/lower bounds on $\lambda^{(1)}$ (Theorem 4.1 (a) and (b) are applied in Appendix). To be specific, the top eigenvalue $\lambda^{(1)}$ is bounded by $\frac{1}{2}\text{Gini}(\boldsymbol{p}_{(1)}) \leq \lambda^{(1)} \leq \text{Gini}(\boldsymbol{p}_{(1)})$, and thus we call $\lambda^{(1)}$ the *impurity* (of the probability output).

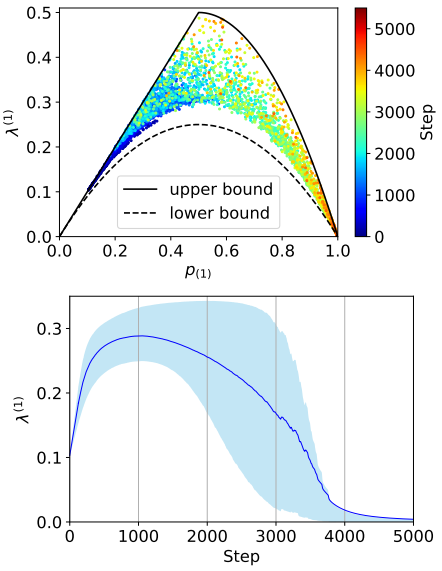

Figure 1: **Impurity $\lambda^{(1)}$ increases and then decreases as $\boldsymbol{p}_{(1)}$ increases during training.** Top: The impurity is plotted against $\boldsymbol{p}_{(1)}$ for a fixed example during training (from blue to red). We plot the upper bound $\min\{\boldsymbol{p}_{(1)}, \text{Gini}(\boldsymbol{p}_{(1)})\}$ (black solid line) and lower bound $\frac{1}{2}\text{Gini}(\boldsymbol{p}_{(1)})$ (black dashed line) from Theorem 4.1 (c) and (d). Bottom: The impurity is plotted against the training step. The blue curve indicates its mean value $\langle\lambda^{(1)}\rangle$ and sky-blue area shows the 25%-75% quantile range for the training data. See Appendix E for detailed settings.

4.2. Evolution of Impurity

We explore the top eigenvalue $\lambda^{(1)}$ of $\boldsymbol{M}$ (also referred to as impurity) during training. Figure 1 demonstrates the n-shaped evolution of the impurity, which increases in the beginning and then decreases in the later phase of training. We trained a model to zero training loss, and thus, for most of the examples, the probability $\boldsymbol{p}_y$ for the true class y eventually becomes the highest probability $\boldsymbol{p}_{(1)}$. As the top probability $\boldsymbol{p}_{(1)}$ increases from $1/C$ to 1 during training, the impurity starts from $\lambda^{(1)} \approx \frac{1}{C} \in [\frac{1}{2}\text{Gini}(\frac{1}{C}), \frac{1}{C}] = [\frac{C-1}{C^2}, \frac{1}{C}]$ (Theorem 4.1 (c) and (d)), and increases at the initial phase of the training, being lower bounded by $\frac{1}{2}\text{Gini}(\boldsymbol{p}_{(1)}) = \boldsymbol{p}_{(1)}(1-\boldsymbol{p}_{(1)})$, which increases for $\boldsymbol{p}_{(1)} \in [0, 0.5]$. Then, $\lambda^{(1)}$ decreases as $\boldsymbol{p}_{(1)}$ becomes larger than 0.5, which leads $\lambda^{(1)}$ to reach nearly 0 at the later phase because it is upper bounded by $\text{Gini}(\boldsymbol{p}_{(1)}) = 2\boldsymbol{p}_{(1)}(1-\boldsymbol{p}_{(1)})$, which decreases for $\boldsymbol{p}_{(1)} \in [0.5, 1]$. Note that Cohen et al. (2021) tried to estimate a similar value, but they use $\boldsymbol{p}_y$, not $\boldsymbol{p}_{(1)}$. Similarly, Papyan (2019) decomposed $\boldsymbol{G}$ into components using the label class information y. Thus, at the beginning of the training, the cluster members are not well-separated according to the true label y. We again emphasize that $\boldsymbol{M}$ is independent of the label y, and the bounds on impurity $\lambda^{(1)}$ are well-described by the probability $\boldsymbol{p}_{(1)}$ of the predicted class.

5. Implicit/Explicit Jacobian Regularization

In this section, based on the results of the previous sections, we aim to derive a relation between the sharpness, the impurity, and the Jacobian (Section 5.1), and answer how the sharpness of the loss landscape influences the learning dynamics and the generalization performance (Section 5.2 and 5.3). Detailed experimental settings for each figure and table are described in Appendix E.

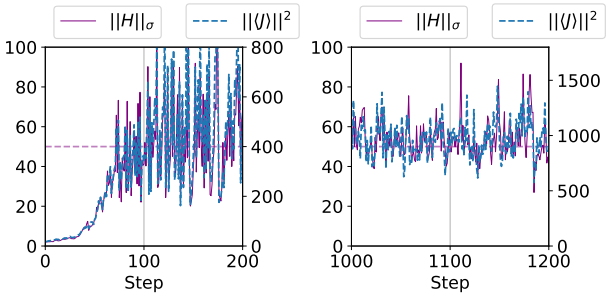

Figure 2: **Sharpness $\|\boldsymbol{H}\|_\sigma$ and Jacobian norm $\|\langle \boldsymbol{J} \rangle\|^2$.** The graphs show similar oscillating behaviors up to a factor $\hat{\lambda}^*$ which is locally constant and slowly changes during training (CIFAR-10, $\eta = 0.04$, $|\mathcal{B}| = 128$; Left: 0-200, Right: 1000-1200 steps). See Appendix I for more details. We highlighted $\|\boldsymbol{H}\|_\sigma = 2/\eta$ with a dashed horizontal line.

5.1. Sharpness-Impurity-Jacobian Relation

We first take a closer look at the sharpness of the loss landscape during training and build a relation between the sharpness $\|\boldsymbol{H}\|_\sigma$, the impurity $\lambda^{(1)}$, and the Jacobian $\boldsymbol{J}$. Because the Gauss-Newton matrix $\boldsymbol{G}$ is known to approximate the true Hessian $\boldsymbol{H}$ well, especially for the top eigenspace (Sagun et al., 2017; Fort & Ganguli, 2019; Papyan, 2019), we can write the sharpness $\|\boldsymbol{H}\|_\sigma$ as $\|\boldsymbol{H}\|_\sigma \approx \|\boldsymbol{G}\|_\sigma = \|\langle \boldsymbol{J}\boldsymbol{M}\boldsymbol{J}^T \rangle\|_\sigma$. This implies that the impurity $\|\boldsymbol{M}\|_\sigma$ and squared Jacobian norm $\|\boldsymbol{J}\|^2$ are highly correlated with the sharpness $\|\boldsymbol{H}\|_\sigma$, as demonstrated in the following theorem. We defer the proof to Appendix C.4.

Theorem 5.1 (Sharpness-Impurity-Jacobian Relation)**.** *For some lower bound* $0 \leq \lambda^* \leq \lambda^{(1)}$ *of the impurity for each* $\boldsymbol{x} \in \mathcal{D}$*, we have* $\|\boldsymbol{G}\|_\sigma = \langle \lambda^* \|\boldsymbol{J}\|^2 \rangle$.

In the next section, we will demonstrate that the sharpness is implicitly upper bounded (Proposition 5.2), and so is the per-example $\lambda^* \|\boldsymbol{J}\|^2$. Therefore, with a large λ^*, the Jacobian norm $\|\boldsymbol{J}\|^2$ is strongly regularized to a small value, i.e., λ^* in $\lambda^* \|\boldsymbol{J}\|^2$ acts as an adaptive regularization weight. Moreover, as the impurity $\lambda^{(1)}$ decreases, so does the lower bound λ^*, and the regularization effect diminishes. Now, because it is computationally inefficient to track $\|\boldsymbol{J}\|^2$ for every $\boldsymbol{x} \in \mathcal{D}$, we instead investigate $\|\langle \boldsymbol{J} \rangle\|^2$. We expect $\|\boldsymbol{H}\|_\sigma = \hat{\lambda}^* \|\langle \boldsymbol{J} \rangle\|^2$ for some $\hat{\lambda}^*$. Figure 2 shows that $\|\boldsymbol{H}\|_\sigma$ and $\|\langle \boldsymbol{J} \rangle\|^2$ have almost identical oscillating behaviors up to a factor $\hat{\lambda}^*$ which is locally constant and slowly changes during training (see Appendix I for further details).

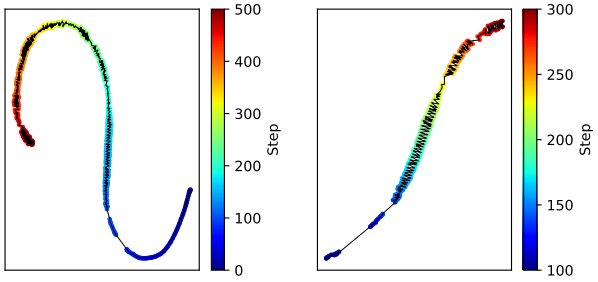

Figure 3: **Oscillatory catapult in the optimization trajectory** $\{\boldsymbol{\theta}^{(t)}\}$ **(from blue to red) of full-batch GD**. Left: UMAP (McInnes et al., 2018) of the model parameters trained on CIFAR-10 for the first 500 steps. Right: Zoom-in of the oscillatory steps [100, 300]. After a few steps ($\sim$100), the sharpness reaches a threshold (see Figure 2) and the iterate shows an oscillatory behavior with an iterative catapult.

5.2. Implicit Jacobian Regularization

Now, we are ready to answer how the sharpness of the loss landscape influences the learning dynamics and generalization performance of neural networks.

Growing Jacobian and progressive sharpening during the early phase of training The weight norm $\|\boldsymbol{\theta}\|$ increases, increasing the logit norm $\|\boldsymbol{z}\|$ and minimizing the cross-entropy loss during training (Soudry et al., 2018) (see Appendix F for details). We hypothesise that this is one of the factors leading to an increase in the layerwise weight norms and the Jacobian norm. Thus, the progressive sharpening can be mainly attributed to the increase of the Jacobian norm (Theorem 5.1). For the MSE loss, Wang et al. (2022) proved the progressive sharpening along with the increase in the (squared) Jacobian norm for a two-layer neural network. We emphasize that the rate of increase in the Jacobian norm, however, varies throughout the training.

Oscillatory catapult and the plateau of sharpness As the sharpness increases in the beginning, the width of the valley of the loss landscape becomes narrower than the discrete step size of the SGD. After the sharpness reaches this threshold, the iterate starts to bounce off from one side of the valley to the other, which is then repeated (Xing et al., 2018; Jastrzębski et al., 2019). Figure 3 shows this oscillatory behavior with an iterative catapult after the sharpness reaches the threshold, using UMAP (McInnes et al., 2018). Owing to the catapult, the iterate cannot remain in a sharper area and returns to stability after the sharpness goes below the threshold (Lewkowycz et al., 2020; Damian et al., 2022). This oscillatory catapult and the plateau of the sharpness are attributed to the *discrete* dynamics of the gradient-based optimization with a finite learning rate and cannot be described through a *continuous* gradient flow. Figure 2 shows fine-grained patterns that the sharpness oscillates up and down around the threshold by the two conflicting effects: the Jacobian norm tends to increase the sharpness, and the self-stabilization reduces it again when the sharpness is over the threshold. Therefore, we can observe the plateau of the sharpness in a coarser scale (see Figure 4). We derive the threshold of the sharpness according to learning rate and batch size in the following proposition.

Proposition 5.2. *For SGD with a quadratic loss, the expected loss decreases when* $\|\boldsymbol{H}\|_\sigma \leq \frac{2\rho_B}{\eta}$*, where* η *is learning rate,* B *is batch size and* $\rho_B \equiv \frac{\|\langle \nabla_{\boldsymbol{\theta}} l \rangle\|^2}{\mathbb{E}_{|\mathcal{B}|=B}[\|\langle \nabla_{\boldsymbol{\theta}} l \rangle_{\mathcal{B}}\|^2]} \leq 1$.

The proof is deferred to Appendix C.5 with further discussion below. We suggest Proposition 5.2 as a modified version of the Edge of Stability proposed in Cohen et al. (2021) ($\rho_B = 1$ for full-batch), which generalizes to mini-batch SGD. As shown in Figure 2, this also approximately holds for the cross-entropy loss. Note that the threshold value $2\rho_B/\eta$ depends on learning rate η and batch size B, which is consistent with the results shown in Figure 5. To be specific, batch gradients $\langle \nabla_{\boldsymbol{\theta}} l \rangle_{\mathcal{B}}$ are more scattered if B is smaller, which leads to a smaller ρ_B. Therefore, a smaller B and a larger η (dotted line) tend to lead to a lower threshold $2\rho_B/\eta$ (see the three purple lines of $\|\boldsymbol{H}\|_\sigma$ for different η and B in Figure 5).

Implicit Jacobian Regularization (IJR) Due to the catapult effect, the rate of increase in the Jacobian norm decreases. In other words, SGD implicitly penalizes the Jacobian norm since $\langle \lambda^* \|\boldsymbol{J}\|^2 \rangle \approx \|\boldsymbol{H}\|_\sigma \leq \frac{2\rho_B}{\eta}$. This implicit Jacobian regularization (IJR) effect begins after the sharpness reaches the threshold. In addition, because the lower bound $\lambda^* \leq \lambda^{(1)}$ acts as a regularization coefficient, the effect diminishes as the impurity $\lambda^{(1)}$ decreases with increasing $\boldsymbol{p}_{(1)} \geq 0.5$ during the later phase (see Figures 1, 4, and 5). This explains why the behavior of the sharpness in the early phase of the training (where $\lambda^{(1)}$ is not small)

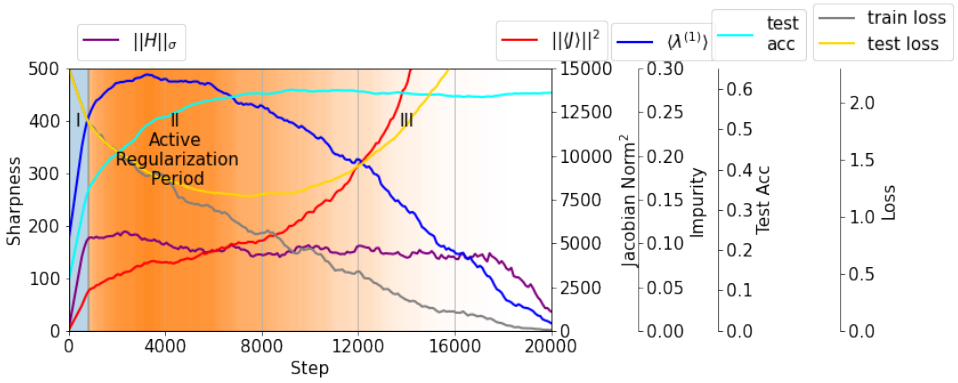

Figure 4: **Three phases of Implicit Jacobian Regularization (IJR).** The Jacobian norm (red) (I) initially rapidly increases before the sharpness (purple) reaches near the threshold, (II) is actively regularized with a gentle slope, and (III) again increases quickly as the regularization effect diminishes (as the regularization weight $\lambda^* \leq \lambda^{(1)}$ decreases) with the slope being gradually steeper. We call phase II the Active Regularization Period (ARP). SGD gradually progresses from phase II to phase III (strong regularization with a high $\langle \lambda^{(1)} \rangle$ indicated with a dark orange color). Thus, we do not explicitly separate these two phases, but we may arbitrarily mark the end of the ARP when $\langle \lambda^{(1)} \rangle$ decreases below a certain value (e.g., 0.25).

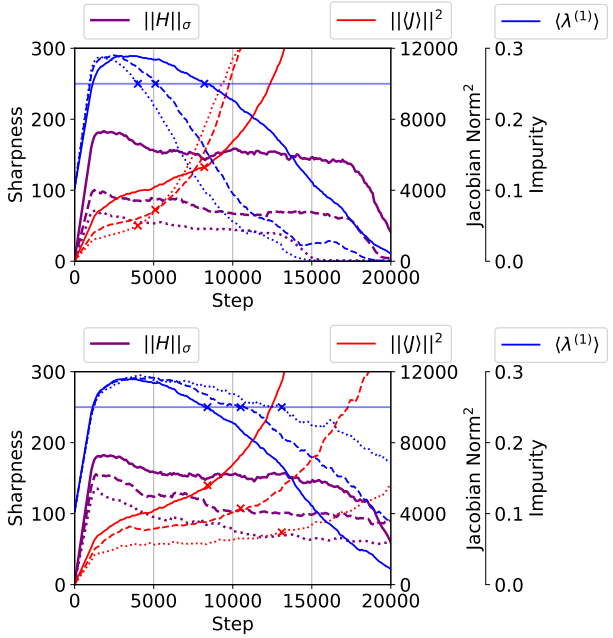

Figure 5: **The effect of IJR varies depending on** (η, B)**.** We used SGD **(solid/dashed/dotted lines)** with fixed $B = 128$ and different η = **0.01/0.02/0.03 (Top)**, and with fixed $\eta = 0.01$ and different B = **128/64/32 (Bottom)** on CIFAR-10. We arbitrarily mark the end of the ARP (where $\langle \lambda^{(1)} \rangle = 0.25$) and the corresponding Jacobian norm with "x". Training with a large learning rate and a small batch size (dotted lines) penalizes the Jacobian norm (red) more strongly with lower limits of the sharpness (purple) in the ARP. The curves are smoothed for visual clarity. See Figure 2 for a finer structure.

seems to highly impact the final generalization (Achille et al., 2019; Golatkar et al., 2019; Jastrzębski et al., 2020; 2021; Lewkowycz et al., 2020; Fort et al., 2020).

Figure 4 shows that SGD has implicit regularization effects on the Jacobian norm in a certain period. There are three phases in which the Jacobian norm (I) initially rapidly increases before the sharpness reaches near the threshold, (II) is actively regularized with a gentle slope, and (III) again increases fast as the regularization effect diminishes (as the implicit regularization weight $\lambda^* \leq \lambda^{(1)}$ decreases) with the slope becoming gradually steeper. We call the second phase *Active Regularization Period (ARP)*.

The evolution of the sharpness is highly affected by learning rate and batch size (Jastrzębski et al., 2019; 2020; Lewkowycz et al., 2020; Cohen et al., 2021). As Figure 5 shows, when comparing the three red lines (solid/dashed/dotted) with different learning rates (0.01/0.02/0.03) and batch sizes (128/64/32), training with a larger learning rate and a smaller batch size (dotted lines) encourages a stronger implicit regularization on the Jacobian norm (see the red "x" marks) with a lower threshold $2\rho_B/\eta$ of the sharpness in the ARP. This could explain why training with a large learning rate and a small batch size yields a better generalization (Keskar et al., 2017; Hoffer et al., 2017; Li et al., 2019b; Jastrzębski et al., 2020).

Jacobian norm and Generalization There has been a line of work on the norm-based capacity control and generalization (Neyshabur et al., 2014; 2015a;b; 2017; 2018; Bartlett et al., 2017; Dziugaite & Roy, 2017; Nagarajan & Kolter, 2019). Similarly, IJR in SGD controls the Jacobian norm and the capacity of the model. Moreover, as the Jacobian norm is directly related to the sharpness (Theorem 5.1),

Table 1: **Explicit Jacobian Regularization (EJR) enhances the test accuracy under various settings.** We report the improvement (ΔAcc.) and Error Reduction Rate (ERR) when trained with EJR, in comparison to the standard SGD. First, we train a simple 6-layer CNN (SimpleCNN) on CIFAR-10 without data augmentation until convergence. Second, we train WRN-28-2 and WRN-28-10 with an efficient variant of EJR and compare it with the state-of-the-art methods, SAM (Foret et al., 2021) and ASAM (Kwon et al., 2021). We use different settings for SimpleCNN and WRN (see Appendix E for details). We report the mean±std of three independent runs. ASAM does not work under the full-batch setting (test accuracy around 20%). More results are found in Appendix Q.

Dataset	Model	Hyperparameters			Test Accuracy				ΔAcc.	ERR
		B	η	λ_{reg}/ρ_{reg}	SGD	SAM	ASAM	EJR	(%p)	(%)
CIFAR-10	SimpleCNN	128	0.003	$\lambda_{reg} = 0.01$	68.92±0.13	74.15±0.01	69.41±0.03	**75.61**±0.03	+6.69	21.53
			0.01		71.08±0.21	74.18±0.08	70.84±0.29	**75.64**±0.02	+4.56	15.77
			0.03		72.16±0.36	74.10±0.13	71.51±0.22	**75.71**±0.10	+3.55	12.75
			0.1		72.59±0.29	74.94±0.29	72.65±0.18	**75.49**±0.05	+2.90	10.58
			0.3		69.17±0.75	71.09±0.06	70.91±0.43	**74.98**±0.64	+5.81	18.85
		50000	0.01	$\lambda_{reg} = 0.001$	69.10±0.05	64.80±0.06	-	**73.99**±0.01	+4.89	15.83
			0.03		69.72±0.07	64.80±0.07	-	**74.19**±0.03	+4.47	14.76
			0.1		69.34±0.39	65.28±0.04	-	**73.88**±0.11	+4.54	14.81
			0.3		64.54±0.39	66.26±0.11	-	**71.86**±0.20	+7.32	20.64
	WRN-28-2	256	0.1	$\rho_{reg} = 2$	95.46±0.03	96.17±0.06	96.16±0.14	**96.25**±0.04	+0.79	17.40
	WRN-28-10	128			96.21±0.03	96.98±0.06	97.13±0.06	**97.21**±0.04	+1.00	26.39
CIFAR-100	WRN-28-2	256	0.1	$\rho_{reg} = 4$	75.32±0.17	78.12±0.31	78.93±0.16	**79.15**±0.39	+3.83	15.52
	WRN-28-10	128			80.72±0.28	83.23±0.17	83.65±0.09	**83.94**±0.10	+3.22	16.70

it can provide connections between the sharpness and the norm-based capacity control. We argue that IJR is one of the main reasons why SGD finds well-generalized minima.

5.3. Explicit Jacobian Regularization (EJR)

To further investigate and boost the effectiveness of IJR, we *explicitly* regularize the Jacobian norm. We expect improvements in the generalization when introducing EJR. This supports the effectiveness of IJR in that it efficiently controls the capacity of the model and helps find better-generalized minima. However, it is computationally hard to back-propagate through the computation graph of the operator norm $\|\langle \boldsymbol{J} \rangle\|^2$ for a practical neural network even with a simple iterative method (see Algorithm 2 in Appendix E). Thus, we instead penalize an upper bound, that is, the Frobenius norm $\|\langle \boldsymbol{J} \rangle\|_F^2 (\geq \|\langle \boldsymbol{J} \rangle\|^2)$, with the regularization coefficient λ_{reg}/C, i.e., we minimize $L + \lambda_{reg}\|\langle \boldsymbol{J} \rangle\|_F^2/C$. The Frobenius regularization term can be efficiently computed with an unbiased estimator $\|\langle \boldsymbol{J} \rangle\|_F^2 = C\mathbb{E}_{\boldsymbol{u}\sim U(\mathbb{S}^{C-1})}[\|\langle \boldsymbol{J} \rangle \boldsymbol{u}\|^2] = C\mathbb{E}_{\boldsymbol{u}\sim U(\mathbb{S}^{C-1})}[\|\nabla_{\boldsymbol{\theta}} \langle \boldsymbol{u}^T \boldsymbol{z} \rangle\|^2]$, where $\boldsymbol{u}$ is randomly drawn from the unit hypersphere $\mathbb{S}^{C-1}$. Because the batch-size is large enough, we efficiently use a single sample $\boldsymbol{u} \sim U(\mathbb{S}^{C-1})$ for each batch, as suggested in Hoffman et al. (2019). Table 1 (SimpleCNN) shows clear improvements in the test accuracy of a simple 6-layer CNN (SimpleCNN) when introducing EJR. We follow a similar setting to Jastrzębski et al. (2021).

Furthermore, we also propose an efficient variant of EJR for larger networks with another regularization coefficient ρ_{reg}, which update the model parameter $\boldsymbol{\theta}^{(t)}$ using two gradient steps like SAM (Foret et al., 2021) as follows:

$$\boldsymbol{\theta}^{(t+1)} = \boldsymbol{\theta}^{(t)} - \eta \nabla_{\boldsymbol{\theta}} L(\hat{\boldsymbol{\theta}}^{(t)}), \tag{11}$$

where

$$\hat{\boldsymbol{\theta}}^{(t)} \equiv \boldsymbol{\theta}^{(t)} + \rho_{reg} \boldsymbol{\delta}^{(t)}/\|\boldsymbol{\delta}^{(t)}\|, \tag{12}$$

$$\boldsymbol{\delta}^{(t)} \equiv \langle \boldsymbol{J} \rangle \boldsymbol{u}^{(t)} = \nabla_{\boldsymbol{\theta}} \langle \boldsymbol{z}^\top \boldsymbol{u}^{(t)} \rangle. \tag{13}$$

Here, $\boldsymbol{J}$ and $\boldsymbol{z}$ are evaluated at $\boldsymbol{\theta}^{(t)}$, and $\boldsymbol{u}^{(t)} \sim U(\mathbb{S}^{C-1})$ is randomly sampled for each step. If we use the gradient ascent step $\boldsymbol{\delta}^{(t)} = \nabla_{\boldsymbol{\theta}} L(\boldsymbol{\theta}^{(t)}) = \langle \boldsymbol{J}(\boldsymbol{p} - \boldsymbol{e}^y) \rangle = \nabla_{\boldsymbol{\theta}} \langle \boldsymbol{z}^\top (\boldsymbol{p} - \boldsymbol{e}^y) \rangle$ instead of (13), we can obtain SAM. We will shortly show that EJR is comparable to SAM, which implies that a random direction $\boldsymbol{u}^{(t)}$ is as good as the specific direction of $\boldsymbol{p} - \boldsymbol{e}^y$, if not better. Note that our perturbation $\boldsymbol{\delta}^{(t)}$ in (13) has a 1/2 chance of increasing the loss ($\boldsymbol{u}$ and $-\boldsymbol{u}$ have the same probability to be sampled). Therefore, we may conclude that SAM is successful not because it solves a minimax problem, but its ascent step is in the column space of the Jacobian, thereby regularizing the Jacobian norm. See Appendix Q for the details.

Figure 6 shows that the efficient version of EJR (dotted line) successfully mitigates overfitting of the model, especially after each learning rate decay. Table 1 (WRN) shows that it outperforms the state-of-the-art sharpness-aware optimization methods such as SAM (Foret et al., 2021) and ASAM (Kwon et al., 2021) on WideResNet (WRN) (Zagoruyko

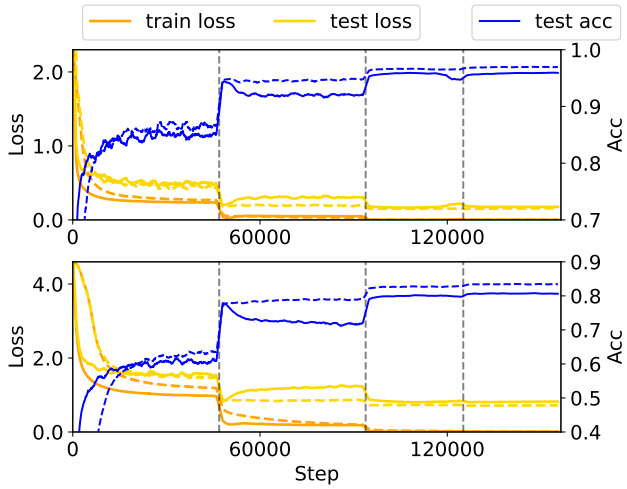

Figure 6: **Efficient EJR (dashed lines) in comparision to SGD (solid lines) for WRN-28-10 on CIFAR-10/CIFAR-100 (Top/Bottom).** EJR effectively mitigates the overfitting, especially after each learning rate decay (undesirable decrease/increase of test accuracy/loss) at steps = $\lceil 50k/128 \rceil \times 400 \times [0.3, 0.6, 0.8]$.

& Komodakis, 2016) with the same computational complexity (we run SGD for twice longer epochs than the others). We follow a similar setting to Kwon et al. (2021). We also evaluate the results of Smith et al. (2021) (regularization on $\|\langle \nabla_{\boldsymbol{\theta}} l \rangle_{\mathcal{B}}\|^2$) and Hoffman et al. (2019) (regularization on $\|\nabla_{\boldsymbol{x}} \boldsymbol{z}\|_F^2$) on WRN, but their performance is about 95.4-95.6/96.2-96.3 (WRN-28-2/WRN-28-10) on CIFAR-10, which is similar to SGD (95.46/96.21), but is not competitive with SAM (96.17/96.98), ASAM (96.16/97.13), and **EJR (96.25/97.21)**. Note that the logit-input Jacobian (Novak et al., 2018; Hoffman et al., 2019) has nothing to do with the implicit bias of SGD in terms of the sharpness and the Edge of Stability, unlike the logit-weight Jacobian.

Connections between the Jacobian and Fisher/Gradient Penalty Our explanation of the implicit bias of SGD may extend to the catastrophic Fisher explosion (Jastrzębski et al., 2021) with $\boldsymbol{G}$ instead of the Fisher Information Matrix (FIM). The trace of $\boldsymbol{G}$ can be written as follows:

$$\begin{aligned} \mathrm{tr}(\boldsymbol{G}) = \langle \mathrm{tr}(\boldsymbol{J}\boldsymbol{M}\boldsymbol{J}^T) \rangle &= \langle \sum_{i=1}^{C-1} \lambda^{(i)} \|\boldsymbol{J}\boldsymbol{q}^{(i)}\|^2 \rangle \\ &\approx \langle \mathrm{tr}(\boldsymbol{M}) \|\boldsymbol{J}\|_F^2 \rangle / C \end{aligned} \tag{14}$$

where we assume $\|\boldsymbol{J}\boldsymbol{q}^{(i)}\|^2 \approx \|\boldsymbol{J}\|_F^2/C$ since $\sum_{i=1}^{C} \|\boldsymbol{J}\boldsymbol{q}^{(i)}\|^2 = \|\boldsymbol{J}\|_F^2$ (see Figure 11 (left) in Appendix D for empirical evidence). Here, the trace of the logit Hessian $\boldsymbol{M}$ can be equivalently written as a C-class Gini impurity, i.e., $\mathrm{tr}(\boldsymbol{M}) = \sum_{i=1}^{C} \boldsymbol{p}_i(1-\boldsymbol{p}_i) = 1 - \sum_{i=1}^{C} \boldsymbol{p}_i^2 \equiv \mathrm{Gini}^C(\boldsymbol{p})$, which is $\frac{C-1}{C}$ for the initial uniform distribution and zero for a one-hot probability. Thus, penalizing $\mathrm{tr}(\boldsymbol{G})$ induces the effect of penalizing $\|\boldsymbol{J}\|_F$, especially in the early phase of training with large C-class impurity $\mathrm{Gini}^C(\boldsymbol{p})$. Thus, as Jastrzębski et al. (2021) argued, Fisher Penalty on the trace of the FIM improves the generalization performance by limiting the memorization, and thus the Jacobian regularization may have similar effects. Moreover, because $\nabla_{\boldsymbol{\theta}} l(\boldsymbol{z}, \hat{y}) = \nabla_{\boldsymbol{\theta}} \boldsymbol{z} \nabla_{\boldsymbol{z}} l(\boldsymbol{z}, \hat{y}) = \boldsymbol{J}(\boldsymbol{p} - \boldsymbol{e}^{\hat{y}})$, the trace of the FIM they approximated is simply $\|\hat{\mathbb{E}}_{\boldsymbol{x}\sim\mathcal{B}} \mathbb{E}_{\hat{y}\sim\boldsymbol{p}} [\nabla_{\boldsymbol{\theta}} l(\boldsymbol{z}, \hat{y})]\|^2 = \|\hat{\mathbb{E}}_{\boldsymbol{x}\sim\mathcal{B}} \mathbb{E}_{\hat{y}\sim\boldsymbol{p}} [\boldsymbol{J}(\boldsymbol{p} - \boldsymbol{e}^{\hat{y}})]\|^2$ with a single sample $\hat{y}$, the gradient norm penalty (Barrett & Dherin, 2021; Smith et al., 2021) is $\|\hat{\mathbb{E}}_{(\boldsymbol{x},y)\sim\mathcal{B}} [\boldsymbol{J}(\boldsymbol{p} - \boldsymbol{e}^{y})]\|^2$, and the EJR regularizer is $\|\boldsymbol{J}\|_F^2 = C\mathbb{E}_{\boldsymbol{u}\sim U(\mathbb{S}^{C-1})} \|\hat{\mathbb{E}}_{(\boldsymbol{x},y)\sim\mathcal{B}} [\boldsymbol{J}\boldsymbol{u}]\|^2$. In each case, the Jacobian $\boldsymbol{J}$ plays an important role in the generalization.

Adversarially robust generalization We also apply EJR to train a model that is robust to adversarial attacks (Szegedy et al., 2013). Adversarially robust training suffers from robust overfitting (Rice et al., 2020) and we expect EJR to help improve robust generalization. We test the effectiveness of EJR with AT (Madry et al., 2018) and TRADES (Zhang et al., 2019). Table 2 shows that the proposed method can obtain better robust generalization against the adversarial attacks. We report the best results with the regularization parameter tuning. We follow a similar setting to Pang et al. (2020). It outperforms the baselines (Madry et al., 2018; Zhang et al., 2019) in terms of the robust accuracy against AA (Croce & Hein, 2020). See Appendix R for details.

Table 2: **Adversarial training with efficient EJR on CIFAR-10 compared with AT (Madry et al., 2018) and TRADES (Zhang et al., 2019).** We report standard accuracy **(Std)** and robust accuracy ($\epsilon = 8/255$) against **PGD-20** and AutoAttack (**AA**) (Croce & Hein, 2020).

Method	PreAct ResNet-18			WRN-34-10		
	Std	PGD-20	AA	Std	PGD-20	AA
AT	82.45	52.85	48.76	86.99	52.20	49.83
AT-EJR	82.32	53.58	**49.14**	86.85	57.82	**53.73**
TRADES	82.34	52.83	49.06	83.62	57.08	53.29
TRADES-EJR	81.56	53.05	**49.56**	83.97	57.48	**53.85**

6. Conclusion

We investigated the Hessian using the Jacobian and the top eigenvalue of the logit Hessian. By doing so, we provided a simple and intuitive explanation on the relation between the sharpness of the loss landscape, the learning dynamics of the gradient-based optimization methods, and the generalization performance of neural networks. We hope this research can help answer other intriguing questions regarding SGD.

Acknowledgements

S. Lee was supported by the research fund of Hanyang University (HY-202300000000552). This work was also partly supported by the Institute of Information & communications Technology Planning & Evaluation (IITP) (No.RS-2023-002206284, Artificial intelligence for prediction of structure-based protein interaction reflecting physicochemical principles and No.2022-0-00984, Development of Artificial Intelligence Technology for Personalized Plug-and-Play Explanation and Verification of Explanation) and by the National Research Foundation of Korea (NRF) (No. RS-2023-00244896 and No. 2019R1A2C2002358) grant funded by the Korean government (MSIT).

References

Achille, A., Rovere, M., and Soatto, S. Critical learning periods in deep networks. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=BkeStsCcKQ.

Barrett, D. and Dherin, B. Implicit gradient regularization. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=3q5IqUrkcF.

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

Bunch, J. R., Nielsen, C. P., and Sorensen, D. C. Rank-one modification of the symmetric eigenproblem. *Numerische Mathematik*, 31(1):31–48, 1978.

Chaudhari, P. and Soatto, S. Stochastic gradient descent performs variational inference, converges to limit cycles for deep networks. In *International Conference on Learning Representations*, 2018.

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

Cohen, J., Kaur, S., Li, Y., Kolter, J. Z., and Talwalkar, A. Gradient descent on neural networks typically occurs at the edge of stability. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=jh-rTtvkGeM.

Croce, F. and Hein, M. Reliable evaluation of adversarial robustness with an ensemble of diverse parameter-free attacks. In *International conference on machine learning*, pp. 2206–2216. PMLR, 2020.

Damian, A., Nichani, E., and Lee, J. D. Self-stabilization: The implicit bias of gradient descent at the edge of stability. *arXiv preprint arXiv:2209.15594*, 2022.

Dauphin, Y. N., Pascanu, R., Gulcehre, C., Cho, K., Ganguli, S., and Bengio, Y. Identifying and attacking the saddle point problem in high-dimensional non-convex optimization. *Advances in neural information processing systems*, 27, 2014.

Dinh, L., Pascanu, R., Bengio, S., and Bengio, Y. Sharp minima can generalize for deep nets. In *International Conference on Machine Learning*, pp. 1019–1028. PMLR, 2017.

Dziugaite, G. K. and 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.

Foret, P., Kleiner, A., Mobahi, H., and Neyshabur, B. Sharpness-aware minimization for efficiently improving generalization. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=6Tm1mposlrM.

Fort, S. and Ganguli, S. Emergent properties of the local geometry of neural loss landscapes. *arXiv preprint arXiv:1910.05929*, 2019.

Fort, S., Dziugaite, G. K., Paul, M., Kharaghani, S., Roy, D. M., and Ganguli, S. Deep learning versus kernel learning: an empirical study of loss landscape geometry and the time evolution of the neural tangent kernel. *Advances in Neural Information Processing Systems*, 33: 5850–5861, 2020.

Fulton, W. Eigenvalues, invariant factors, highest weights, and schubert calculus. *Bulletin of the American Mathematical Society*, 37(3):209–249, 2000.

Gershgorin, S. A. Uber die abgrenzung der eigenwerte einer matrix. *Известия Российской академии наук. Серия математическая*, 7(6): 749–754, 1931.

Ghorbani, B., Krishnan, S., and Xiao, Y. An investigation into neural net optimization via hessian eigenvalue density. In *International Conference on Machine Learning*, pp. 2232–2241. PMLR, 2019.

Golatkar, A. S., Achille, A., and Soatto, S. Time matters in regularizing deep networks: Weight decay and data augmentation affect early learning dynamics, matter little near convergence. *Advances in Neural Information Processing Systems*, 32, 2019.

Golub, G. H. Some modified matrix eigenvalue problems. *Siam Review*, 15(2):318–334, 1973.

Golub, G. H. and Van Loan, C. F. *Matrix computations*, volume 3. JHU press, 2013.

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

Gur-Ari, G., Roberts, D. A., and Dyer, E. Gradient descent happens in a tiny subspace. *arXiv preprint arXiv:1812.04754*, 2018.

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

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

Hoffer, E., Hubara, I., and Soudry, D. Train longer, generalize better: closing the generalization gap in large batch training of neural networks. *Advances in neural information processing systems*, 30, 2017.

Hoffman, J., Roberts, D. A., and Yaida, S. Robust learning with jacobian regularization. *arXiv preprint arXiv:1908.02729*, 2019.

Hu, W., Li, C. J., Li, L., and Liu, J.-G. On the diffusion approximation of nonconvex stochastic gradient descent. *Annals of Mathematical Sciences and Applications*, 4(1), 2019.

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

Jastrzębski, S., Kenton, Z., Ballas, N., Fischer, A., Bengio, Y., and Storkey, A. On the relation between the sharpest directions of DNN loss and the SGD step length. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=SkgEaj05t7.

Jastrzębski, S., Szymczak, M., Fort, S., Arpit, D., Tabor, J., Cho*, K., and Geras*, K. The break-even point on optimization trajectories of deep neural networks. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=r1g87C4KwB.

Jastrzębski, S., Arpit, D., Astrand, O., Kerg, G. B., Wang, H., Xiong, C., Socher, R., Cho, K., and Geras, K. J. Catastrophic fisher explosion: Early phase fisher matrix impacts generalization. In Meila, M. and Zhang, T. (eds.), *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pp. 4772–4784. PMLR, 18–24 Jul 2021. URL https://proceedings.mlr.press/v139/jastrzebski21a.html.

Jiang, Y., Neyshabur, B., Mobahi, H., Krishnan, D., and Bengio, S. Fantastic generalization measures and where to find them. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=SJgIPJBFvH.

Keskar, N. S., Mudigere, D., Nocedal, J., Smelyanskiy, M., and Tang, P. T. P. On large-batch training for deep learning: Generalization gap and sharp minima. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017. URL https://openreview.net/forum?id=H1oyRlYgg.

Krizhevsky, A. and Hinton, G. Learning multiple layers of features from tiny images. *Master's thesis, Department of Computer Science, University of Toronto*, 2009.

Kwon, J., Kim, J., Park, H., and Choi, I. K. Asam: Adaptive sharpness-aware minimization for scale-invariant learning of deep neural networks, 2021.

LeCun, Y. A., Bottou, L., Orr, G. B., and Müller, K.-R. Efficient backprop. In *Neural networks: Tricks of the trade*, pp. 9–48. Springer, 2012.

Lee, S. and Jang, C. A new characterization of the edge of stability based on a sharpness measure aware of batch gradient distribution. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=bH-kCY6LdKg.

Lewkowycz, A., Bahri, Y., Dyer, E., Sohl-Dickstein, J., and Gur-Ari, G. The large learning rate phase of deep learning: the catapult mechanism. *arXiv preprint arXiv:2003.02218*, 2020.

Li, C. J., Li, L., Qian, J., and Liu, J.-G. Batch size matters: A diffusion approximation framework on nonconvex stochastic gradient descent. *stat*, 1050:22, 2017.

Li, Q., Tai, C., and Weinan, E. Stochastic modified equations and dynamics of stochastic gradient algorithms i: Mathematical foundations. *The Journal of Machine Learning Research*, 20(1):1474–1520, 2019a.

Li, Y., Wei, C., and Ma, T. Towards explaining the regularization effect of initial large learning rate in training neural networks. *Advances in Neural Information Processing Systems*, 32, 2019b.

Li, Z., Malladi, S., and Arora, S. On the validity of modeling sgd with stochastic differential equations (sdes). *Advances in Neural Information Processing Systems*, 34, 2021.

Liu, C., Zhu, L., and Belkin, M. Toward a theory of optimization for over-parameterized systems of non-linear equations: the lessons of deep learning. *arXiv preprint arXiv:2003.00307*, 2020.

Liu, H., Simonyan, K., and Yang, Y. DARTS: Differentiable architecture search. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=S1eYHoC5FX.

Loshchilov, I. and Hutter, F. Sgdr: Stochastic gradient descent with warm restarts. *arXiv preprint arXiv:1608.03983*, 2016.

Madry, A., Makelov, A., Schmidt, L., Tsipras, D., and Vladu, A. Towards deep learning models resistant to adversarial attacks. In *International Conference on Learning Representations*, 2018.

Mandt, S., Hoffman, M., and Blei, D. A variational analysis of stochastic gradient algorithms. In *International conference on machine learning*, pp. 354–363. PMLR, 2016.

Mandt, S., Hoffman, M. D., and Blei, D. M. Stochastic gradient descent as approximate bayesian inference. *Journal of Machine Learning Research*, 18:1–35, 2017.

McInnes, L., Healy, J., Saul, N., and Großberger, L. Umap: Uniform manifold approximation and projection. *Journal of Open Source Software*, 3(29):861, 2018. doi: 10.21105/joss.00861. URL https://doi.org/10.21105/joss.00861.

Müller, R., Kornblith, S., and Hinton, G. E. When does label smoothing help? *Advances in neural information processing systems*, 32, 2019.

Nagarajan, V. and Kolter, J. Z. Generalization in deep networks: The role of distance from initialization. *arXiv preprint arXiv:1901.01672*, 2019.

Neyshabur, B. Implicit regularization in deep learning. *arXiv preprint arXiv:1709.01953*, 2017.

Neyshabur, B., Tomioka, R., and Srebro, N. In search of the real inductive bias: On the role of implicit regularization in deep learning. *arXiv preprint arXiv:1412.6614*, 2014.

Neyshabur, B., Salakhutdinov, R., and Srebro, N. Path-sgd: Path-normalized optimization in deep neural networks. *arXiv preprint arXiv:1506.02617*, 2015a.

Neyshabur, B., Tomioka, R., and Srebro, N. Norm-based capacity control in neural networks. In *Conference on Learning Theory*, pp. 1376–1401. PMLR, 2015b.

Neyshabur, B., Bhojanapalli, S., and Srebro, N. A pac-bayesian approach to spectrally-normalized margin bounds for neural networks. *arXiv preprint arXiv:1707.09564*, 2017.

Neyshabur, B., Li, Z., Bhojanapalli, S., LeCun, Y., and Srebro, N. Towards understanding the role of over-parametrization in generalization of neural networks. *arXiv preprint arXiv:1805.12076*, 2018.

Novak, R., Bahri, Y., Abolafia, D. A., Pennington, J., and Sohl-Dickstein, J. Sensitivity and generalization in neural networks: an empirical study. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=HJC2SzZCW.

Pang, T., Yang, X., Dong, Y., Su, H., and Zhu, J. Bag of tricks for adversarial training. In *International Conference on Learning Representations*, 2020.

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

Papyan, V. Measurements of three-level hierarchical structure in the outliers in the spectrum of deepnet hessians. In Chaudhuri, K. and Salakhutdinov, R. (eds.), *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 5012–5021. PMLR, 09–15 Jun 2019. URL https://proceedings.mlr.press/v97/papyan19a.html.

Papyan, V. Traces of class/cross-class structure pervade deep learning spectra. *Journal of Machine Learning Research*, 21(252):1–64, 2020.

Park, D., Sohl-Dickstein, J., Le, Q., and Smith, S. The effect of network width on stochastic gradient descent and generalization: an empirical study. In *International Conference on Machine Learning*, pp. 5042–5051. PMLR, 2019.

Rice, L., Wong, E., and Kolter, Z. Overfitting in adversarially robust deep learning. In *International Conference on Machine Learning*, pp. 8093–8104. PMLR, 2020.

Robbins, H. and Monro, S. A stochastic approximation method. *The annals of mathematical statistics*, pp. 400–407, 1951.

Sagun, L., Bottou, L., and LeCun, Y. Eigenvalues of the hessian in deep learning: Singularity and beyond. *arXiv preprint arXiv:1611.07476*, 2016.

Sagun, L., Evci, U., Guney, V. U., Dauphin, Y., and Bottou, L. Empirical analysis of the hessian of over-parametrized neural networks. *arXiv preprint arXiv:1706.04454*, 2017.

Schraudolph, N. N. Fast curvature matrix-vector products for second-order gradient descent. *Neural computation*, 14(7):1723–1738, 2002.

Simonyan, K. and Zisserman, A. Very deep convolutional networks for large-scale image recognition. In Bengio, Y. and LeCun, Y. (eds.), *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015. URL http://arxiv.org/abs/1409.1556.

Smith, S. L. and Le, Q. V. A bayesian perspective on generalization and stochastic gradient descent. In *International Conference on Learning Representations*, 2018.

Smith, S. L., Dherin, B., Barrett, D., and De, S. On the origin of implicit regularization in stochastic gradient descent. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=rq_Qr0c1Hyo.

Soudry, D., Hoffer, E., Nacson, M. S., Gunasekar, S., and Srebro, N. The implicit bias of gradient descent on separable data. *The Journal of Machine Learning Research*, 19(1):2822–2878, 2018.

Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I., and Fergus, R. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*, 2013.

Wang, H., Keskar, N. S., Xiong, C., and Socher, R. Identifying generalization properties in neural networks. *arXiv preprint arXiv:1809.07402*, 2018.

Wang, Z., Li, Z., and Li, J. Analyzing sharpness along GD trajectory: Progressive sharpening and edge of stability. In Oh, A. H., Agarwal, A., Belgrave, D., and Cho, K. (eds.), *Advances in Neural Information Processing Systems*, 2022. URL https://openreview.net/forum?id=thgItcQrJ4y.

Weyl, H. Das asymptotische verteilungsgesetz der eigenwerte linearer partieller differentialgleichungen (mit einer anwendung auf die theorie der hohlraumstrahlung). *Mathematische Annalen*, 71(4):441–479, 1912.

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

Yaida, S. Fluctuation-dissipation relations for stochastic gradient descent. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=SkNksoRctQ.

Yao, Z., Gholami, A., Keutzer, K., and 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.

Zagoruyko, S. and Komodakis, N. Wide residual networks. *arXiv preprint arXiv:1605.07146*, 2016.

Zhang, C., Bengio, S., Hardt, M., Recht, B., and Vinyals, O. Understanding deep learning (still) requires rethinking generalization. *Communications of the ACM*, 64(3):107–115, 2021.

Zhang, H., Yu, Y., Jiao, J., Xing, E., El Ghaoui, L., and Jordan, M. Theoretically principled trade-off between robustness and accuracy. In *International Conference on Machine Learning*, pp. 7472–7482, 2019.

Zhu, Z., Wu, J., Yu, B., Wu, L., and Ma, J. The anisotropic noise in stochastic gradient descent: Its behavior of escaping from sharp minima and regularization effects. In *International Conference on Machine Learning*, pp. 7654–7663. PMLR, 2019.

A. Notations

We summarize the notations for a quick reference.

$\boldsymbol{x} \in \mathcal{X} \subset \mathbb{R}^d$	input
$y \in [C] = \{1, 2, \cdots, C\}$	the corresponding label
$f_{\boldsymbol{\theta}} : \mathcal{X} \to \mathcal{Z} \subset \mathbb{R}^C$	parameterized model
$\boldsymbol{\theta} \in \Theta \subset \mathbb{R}^m$	model parameter
$\boldsymbol{z} = f_{\boldsymbol{\theta}}(x) \in \mathcal{Z} \subset \mathbb{R}^C$	logit vector
$\boldsymbol{p} = \mathrm{softmax}(\boldsymbol{z}) \in \Delta^{C-1}$	probability output
$\Delta^{C-1} = \{\boldsymbol{p} \in [0,1]^C : \mathbf{1}^T \boldsymbol{p} = 1, \boldsymbol{p} \geq \mathbf{0}\}$	probability simplex
$p = \boldsymbol{p}_y$	probability value
$\boldsymbol{c} = \arg\mathrm{sort}(\boldsymbol{p})$	class output
$\ell = \ell(\boldsymbol{z}, y) \in \mathbb{R}$	cross-entropy loss
$\|\boldsymbol{v}\| = \sqrt{\sum_i \boldsymbol{v}_i^2}$	Euclidean norm of a vector $\boldsymbol{v}$
$\|\boldsymbol{A}\| = \max_{\|\boldsymbol{v}\|=1} \|\boldsymbol{A}\boldsymbol{v}\|$	Euclidean operator norm of a matrix $\boldsymbol{A}$
$\|\boldsymbol{A}\|_F = \sqrt{\sum_{i,j} \lvert\boldsymbol{A}_{ij}\rvert^2}$	Frobenius norm of a matrix $\boldsymbol{A}$
$\|\boldsymbol{S}\|_\sigma$	spectral norm of a square matrix $\boldsymbol{S}$
$\mathrm{tr}(\boldsymbol{S}) = \sum_i \boldsymbol{S}_i i$	trace of a square matrix $\boldsymbol{S}$
$\mathrm{diag}(\boldsymbol{v}) = (\delta_{ij}\boldsymbol{v}_i)_{ij}$	diagonal matrix with $\boldsymbol{v}$ as its diagonal entries
$\boldsymbol{e}^i = (\delta_{ij})_j$	one-hot vector with the i-th element as 1
$\mathcal{D}$	training set
$L = \langle \ell \rangle$	total training loss
$\langle \cdot \rangle = \langle \cdot \rangle_{\mathcal{D}} = \hat{\mathbb{E}}_{\mathcal{D}}[\cdot]$	expectation over $\mathcal{D}$
$\langle \cdot \rangle_{\mathcal{B}} = \hat{\mathbb{E}}_{\mathcal{B}}[\cdot]$	expectation over $\mathcal{B}$
$\mathcal{B}$	mini-batch
$\boldsymbol{H} = \langle \nabla^2_{\boldsymbol{\theta}} \ell \rangle \in \mathbb{R}^{m \times m}$	Hessian
$\boldsymbol{M} = \nabla^2_{\boldsymbol{z}} \ell \in \mathbb{R}^{C \times C}$	logit Hessian
$\boldsymbol{J} = \nabla_{\boldsymbol{\theta}} \boldsymbol{z} \in \mathbb{R}^{m \times C}$	(logit-weight) Jacobian
$\boldsymbol{G} = \langle \boldsymbol{J}\boldsymbol{M}\boldsymbol{J}^T \rangle \in \mathbb{R}^{m \times m}$	Gauss-Newton approximation
$\lambda^{(i)}$	the i-th largest eigenvalues of $\boldsymbol{M}$
$\boldsymbol{q}^{(i)}$	the corresponding eigenvector
$(i) = \boldsymbol{c}_i \in [C]$	the ordered index
$\mathrm{Gini}(q) = 2q(1-q)$	(binary) Gini impurity
$\mathrm{Gini}^C(\boldsymbol{p}) = 1 - \sum_{i=1}^C \boldsymbol{p}_i^2$	(C-ary) Gini impurity
$\eta > 0$	learning rate
$B = \lvert\mathcal{B}\rvert$	batch size

B. Summary

The concepts in the paper, weight Hessisan $\boldsymbol{H}$, logit Hessian $\boldsymbol{M}$, impurity $\lambda^{(1)} \equiv \|\boldsymbol{M}\|_\sigma$, sharpness $\|\boldsymbol{H}\|_\sigma$, logit Jacobian $\boldsymbol{J}$ and Gauss-Newton matrix $\boldsymbol{G}$, connect with each other as follows:

$$\|\boldsymbol{H}\|_\sigma \overset{\text{(i)}}{\approx} \|\boldsymbol{G}\|_\sigma \overset{\text{(ii)}}{=} \langle \lambda^* \|\boldsymbol{J}\|^2 \rangle \overset{\text{(iii)}}{\leq} \langle \lambda^{(1)} \|\boldsymbol{J}\|^2 \rangle \overset{\text{(iv)}}{=} \langle \|\boldsymbol{M}\|_\sigma \|\boldsymbol{J}\|^2 \rangle,$$

where (i), (ii), (iii), (iv) come from the Gauss-Newton approximation, Theorem 5.1, $\lambda^* \leq \lambda^{(1)}$, and $\lambda^{(1)} \equiv \|\boldsymbol{M}\|_\sigma$, respectively.

C. Proofs

C.1. Proof of (1)

$$\nabla_{\boldsymbol{z}} \boldsymbol{p} = \operatorname{diag}(\boldsymbol{p}) - \boldsymbol{p}\boldsymbol{p}^T \in \mathbb{R}^{C \times C} \tag{1}$$

Proof. By definition of the softmax function,

$$\boldsymbol{p}_j = [\operatorname{softmax}(\boldsymbol{z})]_j = \frac{\exp(\boldsymbol{z}_j)}{\sum_k \exp(\boldsymbol{z}_k)} = \exp(\boldsymbol{z}_j) s^{-1} \tag{15}$$

where $s = \sum_k \exp(\boldsymbol{z}_k)$, we have

$$\nabla_{\boldsymbol{z}_i} \boldsymbol{p}_j = \begin{cases} -\exp(\boldsymbol{z}_j) s^{-2} \exp(\boldsymbol{z}_i) = -\boldsymbol{p}_i \boldsymbol{p}_j, & \text{if } i \neq j \\ -\exp(\boldsymbol{z}_i) s^{-2} \exp(\boldsymbol{z}_i) + \exp(\boldsymbol{z}_i) s^{-1} = -\boldsymbol{p}_i^2 + \boldsymbol{p}_i, & \text{if } i = j \end{cases} \tag{16}$$

which leads to $\nabla_{\boldsymbol{z}} \boldsymbol{p} = (\nabla_{\boldsymbol{z}_i} \boldsymbol{p}_j)_{ij} = -\boldsymbol{p}\boldsymbol{p}^T + \operatorname{diag}(\boldsymbol{p})$. □

C.2. Proof of (6)

We provide a self-contained proof of the first part (6) (see, for example, Schraudolph (2002) for more).

$$\nabla^2_{\boldsymbol{\theta}} l = \nabla_{\boldsymbol{\theta}} \boldsymbol{z} \nabla^2_{\boldsymbol{z}} l \nabla_{\boldsymbol{\theta}} \boldsymbol{z}^T + \sum\nolimits_{c=1}^{C} \nabla^2_{\boldsymbol{\theta}} \boldsymbol{z}_c \nabla_{\boldsymbol{z}_c} l \tag{6}$$

Proof. We apply the chain rule to obtain the following equations:

$$(\nabla^2_{\boldsymbol{\theta}} l)_{i,j} = \frac{\partial}{\partial \boldsymbol{\theta}_j} \frac{\partial l}{\partial \boldsymbol{\theta}_i} = \frac{\partial}{\partial \boldsymbol{\theta}_j} \left(\sum_{c=1}^{C} \frac{\partial l}{\partial \boldsymbol{z}_c} \frac{\partial \boldsymbol{z}_c}{\partial \boldsymbol{\theta}_i} \right) \tag{17}$$

$$= \sum_{c=1}^{C} \frac{\partial}{\partial \boldsymbol{\theta}_j} \left(\frac{\partial l}{\partial \boldsymbol{z}_c} \frac{\partial \boldsymbol{z}_c}{\partial \boldsymbol{\theta}_i} \right) \tag{18}$$

$$= \sum_{c=1}^{C} \frac{\partial}{\partial \boldsymbol{\theta}_j} \left(\frac{\partial l}{\partial \boldsymbol{z}_c} \right) \frac{\partial \boldsymbol{z}_c}{\partial \boldsymbol{\theta}_i} + \sum_{c=1}^{C} \frac{\partial l}{\partial \boldsymbol{z}_c} \frac{\partial^2 \boldsymbol{z}_c}{\partial \boldsymbol{\theta}_j \partial \boldsymbol{\theta}_i} \tag{19}$$

$$= \sum_{c=1}^{C} \left(\sum_{k=1}^{C} \frac{\partial \boldsymbol{z}_c}{\partial \boldsymbol{\theta}_j} \frac{\partial^2 l}{\partial \boldsymbol{z}_c \partial \boldsymbol{z}_k} \right) \frac{\partial \boldsymbol{z}_c}{\partial \boldsymbol{\theta}_i} + \sum_{c=1}^{C} \frac{\partial^2 \boldsymbol{z}_c}{\partial \boldsymbol{\theta}_j \partial \boldsymbol{\theta}_i} \frac{\partial l}{\partial \boldsymbol{z}_c} \tag{20}$$

which leads to the conclusion. □

Remark C.1. Assuming $\nabla_{\boldsymbol{z}_c} l$ are uncorrelated with $\nabla_{\boldsymbol{\theta}} \boldsymbol{z}_c{}^2$, we can approximate $\nabla^2_{\boldsymbol{\theta}} l \approx \nabla_{\boldsymbol{\theta}} \boldsymbol{z}^T \nabla^2_{\boldsymbol{z}} l \nabla_{\boldsymbol{\theta}} \boldsymbol{z}$ (Sagun et al., 2017; Fort & Ganguli, 2019).

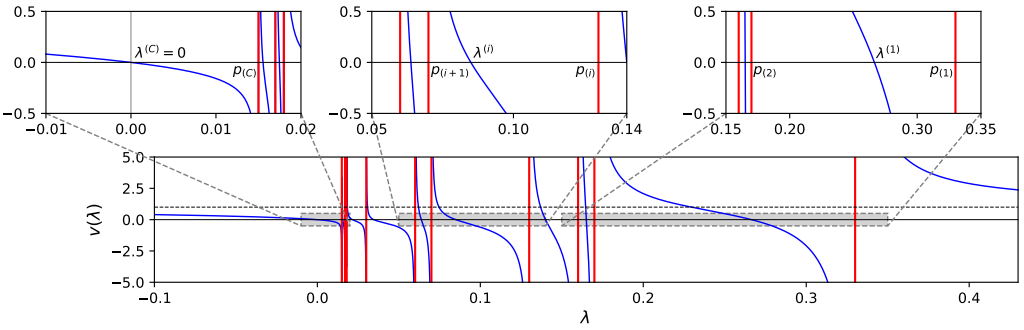

Figure 7: **Graph of secular function $v(\lambda)$ (blue curves) in (10) for some $\boldsymbol{p}$ having zeros at the eigenvalues $\{\lambda^{(i)}\}_{i=1}^{C}$ of $\boldsymbol{M}=\nabla_{\boldsymbol{z}}^2 l$.** We highlighted the singularities $\lambda=\boldsymbol{p}_{(i)}$ with red vertical lines. The figure illustrates Theorem 4.1 (a) and (c).

C.3. Proof of Theorem 4.1

Theorem 4.1 (restated). The eigenvalues $\lambda^{(i)}$ $(\lambda^{(1)}\geq\lambda^{(2)}\geq\cdots\geq\lambda^{(C)})$ and the corresponding eigenvectors $\boldsymbol{q}^{(i)}$ of the logit Hessian $\boldsymbol{M}=\nabla_{\boldsymbol{z}}^2 l=\mathrm{diag}(\boldsymbol{p})-\boldsymbol{p}\boldsymbol{p}^T$ satisfy the following properties:

(a) The eigenvalue $\lambda^{(i)}$ is the i-th largest solution of the following equation:

$$v(\lambda)=1-\sum_{i=1}^{C}\frac{\boldsymbol{p}_i^2}{\boldsymbol{p}_i-\lambda}=0 \tag{21}$$

(b) The eigenvector $\boldsymbol{q}^{(i)}$ is aligned with the direction of $(\mathrm{diag}(\boldsymbol{p})-\lambda^{(i)}\boldsymbol{I})^{-1}\boldsymbol{p}$

(c) $\boldsymbol{p}_{(i+1)}\leq\lambda^{(i)}\leq\boldsymbol{p}_{(i)}$ for $1\leq i\leq C-1$, and $\lambda^{(C)}=0$

(d) $\frac{1}{2}\mathrm{Gini}(\boldsymbol{p}_{(1)})\leq\lambda^{(1)}\leq\mathrm{Gini}(\boldsymbol{p}_{(1)})$ where $\mathrm{Gini}(q)=2q(1-q)$ is the Gini impurity for the binary case $(q,1-q)$.

Proof. The eigenvalues $\lambda^{(i)}$ of $\boldsymbol{M}=\mathrm{diag}(\boldsymbol{p})-\boldsymbol{p}\boldsymbol{p}^T$ are the zeros of the following characteristic polynomial:

$$\begin{aligned}\phi_{\boldsymbol{M}}(\lambda)&=\det(\mathrm{diag}(\boldsymbol{p})-\boldsymbol{p}\boldsymbol{p}^T-\lambda\boldsymbol{I}) &(22)\\ &=\det(\mathrm{diag}(\boldsymbol{p})-\lambda\boldsymbol{I})\det(\boldsymbol{I}-(\mathrm{diag}(\boldsymbol{p})-\lambda\boldsymbol{I})^{-1}\boldsymbol{p}\boldsymbol{p}^T) &(23)\\ &=\prod_{i=1}^{C}(\boldsymbol{p}_i-\lambda)\left(1-\sum_{j=1}^{C}\frac{\boldsymbol{p}_j^2}{\boldsymbol{p}_j-\lambda}\right) &(24)\end{aligned}$$

where the second equality follows from $\boldsymbol{A}-\boldsymbol{p}\boldsymbol{p}^T=\boldsymbol{A}(\boldsymbol{I}-\boldsymbol{A}^{-1}\boldsymbol{p}\boldsymbol{p}^T)$ with the matrix $\boldsymbol{A}=\mathrm{diag}(\boldsymbol{p})-\lambda\boldsymbol{I}$, and the third inequality holds because $\det(\boldsymbol{I}+\boldsymbol{u}\boldsymbol{v}^T)=1+\boldsymbol{u}^T\boldsymbol{v}$ for vectors $\boldsymbol{u}$ and $\boldsymbol{v}$. Then it is equivalent to solving the following equation:

$$v(\lambda)=1-\sum_{i=1}^{C}\frac{\boldsymbol{p}_i^2}{\boldsymbol{p}_i-\lambda}=0 \tag{25}$$

which implies (a). Note that this result also implies (c).

Next, to prove (b), put $\boldsymbol{A}\equiv\mathrm{diag}(\boldsymbol{p})-\lambda\boldsymbol{I}$ and $\boldsymbol{q}\equiv\boldsymbol{A}^{-1}\boldsymbol{p}$. Then it is required to show that $(\boldsymbol{M}-\lambda\boldsymbol{I})\boldsymbol{q}=(\boldsymbol{A}-\boldsymbol{p}\boldsymbol{p}^T)\boldsymbol{q}=\boldsymbol{0}$ for the eigenvalues $\lambda=\lambda^{(i)}$. We have

$$(\boldsymbol{A}-\boldsymbol{p}\boldsymbol{p}^T)\boldsymbol{q}=(\boldsymbol{A}-\boldsymbol{p}\boldsymbol{p}^T)\boldsymbol{A}^{-1}\boldsymbol{p}=\boldsymbol{p}-\boldsymbol{p}\boldsymbol{p}^T\boldsymbol{A}^{-1}\boldsymbol{p} \tag{26}$$

Here, $(\boldsymbol{p}\boldsymbol{p}^T\boldsymbol{A}^{-1}\boldsymbol{p})_i=\sum_{j,k}\boldsymbol{p}_i\boldsymbol{p}_j\boldsymbol{A}_{jk}^{-1}\boldsymbol{p}_k=\sum_{j,k}\boldsymbol{p}_i\boldsymbol{p}_j\delta_{jk}(\boldsymbol{p}_j-\lambda)^{-1}\boldsymbol{p}_k=\sum_k\boldsymbol{p}_i\boldsymbol{p}_k(\boldsymbol{p}_k-\lambda)^{-1}\boldsymbol{p}_k=\boldsymbol{p}_i\sum_k\boldsymbol{p}_k^2/(\boldsymbol{p}_k-\lambda)=\boldsymbol{p}_i$. The last equality holds for the eigenvalues $\lambda=\lambda^{(i)}$ which follows from (a).

Now, we want to prove the statement (c). Since

$$\lambda^{(i)}(\boldsymbol{C}) \leq \lambda^{(j)}(\boldsymbol{A}) + \lambda^{(k)}(\boldsymbol{B}) \text{ if } k+j-i=1 \tag{27}$$

$$\lambda^{(i)}(\boldsymbol{C}) \geq \lambda^{(j)}(\boldsymbol{A}) + \lambda^{(k)}(\boldsymbol{B}) \text{ if } k+j-i=C \tag{28}$$

for $\boldsymbol{C} = \boldsymbol{A} + \boldsymbol{B} \in \mathbb{R}^{C\times C}$ where $\lambda^{(i)}(\boldsymbol{D})$ is the i-th largest eigenvalue of a matrix $\boldsymbol{D}$ (Weyl, 1912; Fulton, 2000), we can get $\lambda^{(i)}(\boldsymbol{C}) \leq \lambda^{(i)}(\boldsymbol{A}) + \lambda^{(1)}(\boldsymbol{B})$ and $\lambda^{(i)}(\boldsymbol{C}) \geq \lambda^{(i+1)}(\boldsymbol{A}) + \lambda^{(C-1)}(\boldsymbol{B})$. Thus, for $\boldsymbol{A} = \text{diag}(\boldsymbol{p})$ and $\boldsymbol{B} = -\boldsymbol{p}\boldsymbol{p}^T$, we can get

$$\boldsymbol{p}_{(i+1)} \leq \lambda^{(i)}(\boldsymbol{M}) \leq \boldsymbol{p}_{(i)} \text{ for } 1 \leq i \leq C-1 \tag{29}$$

since $\lambda^{(i)}(\boldsymbol{A}) = \boldsymbol{p}_{(i)}$, $\lambda^{(i+1)}(\boldsymbol{A}) = \boldsymbol{p}_{(i+1)}$ and $\lambda^{(1)}(\boldsymbol{B}) = \lambda^{(C-1)}(\boldsymbol{B}) = 0$. Moreover, since $\boldsymbol{M}\boldsymbol{1} = \boldsymbol{p} - \boldsymbol{p}\boldsymbol{p}^T\boldsymbol{1} = \boldsymbol{p} - \boldsymbol{p}\sum_i \boldsymbol{p}_i = \boldsymbol{0}$, the smallest eigenvalue is $\lambda^{(C)} = 0$.

Lastly, we prove the statement (d). From the Gershgorin circle theorem (Gershgorin, 1931), we have

$$\lambda^{(1)} \in \bigcup_i \overline{\mathbb{B}}(\boldsymbol{M}_{ii}, \sum_{j\neq i} |\boldsymbol{M}_{ij}|) = \overline{\mathbb{B}}(\boldsymbol{p}_{(1)}(1-\boldsymbol{p}_{(1)}), \boldsymbol{p}_{(1)}(1-\boldsymbol{p}_{(1)})) = [0, 2\boldsymbol{p}_{(1)}(1-\boldsymbol{p}_{(1)})] \tag{30}$$

which implies $\lambda^{(1)} \leq 2\boldsymbol{p}_{(1)}(1-\boldsymbol{p}_{(1)})$. Note that $\boldsymbol{p}_{(1)}(1-\boldsymbol{p}_{(1)}) \geq \boldsymbol{p}_{(i)}(1-\boldsymbol{p}_{(i)})$ since $g(t) = t(1-t)$ is increasing for $0 \leq t \leq 0.5$. In detail, if $\boldsymbol{p}_{(1)} \geq 0.5$, since $\boldsymbol{p}_{(i)} \leq 1 - \boldsymbol{p}_{(1)} \leq 0.5$, we have $g(\boldsymbol{p}_{(i)}) \leq g(1-\boldsymbol{p}_{(1)}) = g(\boldsymbol{p}_{(1)})$. Otherwise ($\boldsymbol{p}_{(1)} < 0.5$), since $\boldsymbol{p}_{(i)} \leq \boldsymbol{p}_{(1)}$, it leads to the same inequality $g(\boldsymbol{p}_{(i)}) \leq g(\boldsymbol{p}_{(1)})$. With the Rayleigh principle, we can express the largest eigenvalue as $\lambda^{(1)} = \max_{\|\boldsymbol{u}\|_2=1} \boldsymbol{u}^T\boldsymbol{M}\boldsymbol{u}$, and thus $\boldsymbol{e}^{(1)T}\boldsymbol{M}\boldsymbol{e}^{(1)} = \boldsymbol{M}_{(1)(1)} = \boldsymbol{p}_{(1)}(1-\boldsymbol{p}_{(1)}) \leq \lambda^{(1)}$. □

C.4. Proof of Theorem 5.1

Theorem 5.1 (restated). For some lower bound $0 \leq \lambda^* \leq \lambda^{(1)}$ of the impurity for each $\boldsymbol{x} \in \mathcal{D}$, we have

$$\|\boldsymbol{G}\|_\sigma = \langle \lambda^* \|\boldsymbol{J}\|^2 \rangle. \tag{31}$$

Proof. We start with the Rayleigh principle:

$$\|\langle \boldsymbol{J}\boldsymbol{M}\boldsymbol{J}^T \rangle\|_\sigma = \max_{\|\boldsymbol{q}\|=1} \boldsymbol{q}^T \langle \boldsymbol{J}\boldsymbol{M}\boldsymbol{J}^T \rangle \boldsymbol{q} = \max_{\|\boldsymbol{q}\|=1} \langle \boldsymbol{q}^T \boldsymbol{J}\boldsymbol{M}\boldsymbol{J}^T \boldsymbol{q} \rangle \tag{32}$$

Since $\boldsymbol{M} = \sum_i \lambda^{(i)} \boldsymbol{q}^{(i)} \boldsymbol{q}^{(i)T}$, we can continue by putting $\boldsymbol{v} = \boldsymbol{J}^T\boldsymbol{q}$, and then

$$(32) = \max_{\|\boldsymbol{q}\|=1} \langle \boldsymbol{v}^T \boldsymbol{M} \boldsymbol{v} \rangle = \max_{\|\boldsymbol{q}\|=1} \langle \sum_i \lambda^{(i)} (\boldsymbol{q}^{(i)T} \boldsymbol{v})^2 \rangle \tag{33}$$

Then by putting $\tilde{\lambda} = \sum_i \gamma^{(i)} \lambda^{(i)}$ with $\gamma^{(i)} = (\boldsymbol{q}^{(i)T}\boldsymbol{v})^2 / \sum_i (\boldsymbol{q}^{(i)T}\boldsymbol{v})^2 \geq 0$ ($\sum_i \gamma_i = 1$),

$$(33) = \max_{\|\boldsymbol{q}\|=1} \langle \tilde{\lambda} \sum_i (\boldsymbol{q}^{(i)T} \boldsymbol{v})^2 \rangle = \max_{\|\boldsymbol{q}\|=1} \langle \tilde{\lambda} \sum_i (\boldsymbol{q}^{(i)T} \boldsymbol{J}^T \boldsymbol{q})^2 \rangle \tag{34}$$

$$= \max_{\|\boldsymbol{q}\|=1} \langle \tilde{\lambda} \sum_i (\boldsymbol{q}^{(i)T} (\boldsymbol{J}^T \boldsymbol{q}))^2 \rangle \tag{35}$$

Since $\{\boldsymbol{q}^{(i)}\}_{i=1}^C$ is an orthonormal basis of $\mathbb{R}^C$ (eigenvectors of a symmetric matrix $\boldsymbol{M}$), we have the following by putting $\lambda^* = \frac{\|\boldsymbol{J}^T\boldsymbol{q}^*\|^2}{\|\boldsymbol{J}\|^2}\tilde{\lambda}$ with $\boldsymbol{q}^* = \arg\max_{\|\boldsymbol{q}\|=1} \langle \tilde{\lambda} \|\boldsymbol{J}^T\boldsymbol{q}\|^2 \rangle$,

$$(35) = \max_{\|\boldsymbol{q}\|=1} \langle \tilde{\lambda} \|\boldsymbol{J}^T\boldsymbol{q}\|^2 \rangle = \langle \tilde{\lambda} \|\boldsymbol{J}^T\boldsymbol{q}^*\|^2 \rangle = \langle \lambda^* \|\boldsymbol{J}\|^2 \rangle \tag{36}$$

Because of the definition of λ^* and $\tilde{\lambda}$ with $\|\boldsymbol{q}^*\| = 1$ and $\sum_i \gamma^{(i)} = 1$ ($\gamma^{(i)} \geq 0$), we have

$$\lambda^* \leq \tilde{\lambda} \leq \lambda^{(1)}. \tag{37}$$

□

C.5. Proof of Proposition 5.2

Proposition 5.2 (restated)**.** *For SGD with a quadratic loss, the expected loss decreases when* $\|\boldsymbol{H}\|_\sigma \leq \frac{2\rho_B}{\eta}$*, where* η *is learning rate,* B *is batch size and* $\rho_B \equiv \frac{\|\langle\nabla_{\boldsymbol{\theta}} l\rangle\|^2}{\mathbb{E}_{|\mathcal{B}|=B}[\|\langle\nabla_{\boldsymbol{\theta}} l\rangle_{\mathcal{B}}\|^2]} \leq 1$.

Proof. Put $\boldsymbol{g} = \nabla_{\boldsymbol{\theta}} l(\boldsymbol{\theta})$. With SGD update $\boldsymbol{\theta}' = \boldsymbol{\theta} - \eta\langle\boldsymbol{g}\rangle_{\mathcal{B}}$ and $L' = L(\boldsymbol{\theta}')$, we can obtain

$$L' = L - \eta\nabla L^T\langle\boldsymbol{g}\rangle_{\mathcal{B}} + \frac{\eta^2}{2}\langle\boldsymbol{g}\rangle_{\mathcal{B}}^T\boldsymbol{H}\langle\boldsymbol{g}\rangle_{\mathcal{B}} \tag{38}$$

Therefore, the expected loss at the next step is

$$\mathbb{E}[L'] = L - \eta\nabla L^T\mathbb{E}[\langle\boldsymbol{g}\rangle_{\mathcal{B}}] + \frac{\eta^2}{2}\mathbb{E}[\langle\boldsymbol{g}\rangle_{\mathcal{B}}^T\boldsymbol{H}\langle\boldsymbol{g}\rangle_{\mathcal{B}}] \tag{39}$$

$$\leq L - \eta\|\nabla L\|^2 + \frac{\eta^2}{2}\mathbb{E}[\|\langle\boldsymbol{g}\rangle_{\mathcal{B}}\|^2]\|\boldsymbol{H}\|_\sigma \tag{40}$$

When $\|\boldsymbol{H}\|_\sigma \leq \frac{2}{\eta}\frac{\|\nabla L\|^2}{\mathbb{E}[\|\langle\boldsymbol{g}\rangle_{\mathcal{B}}\|^2]} = \frac{2\rho_B}{\eta}$, i.e., $-\eta\|\nabla L\|^2 + \frac{\eta^2}{2}\mathbb{E}[\|\langle\boldsymbol{g}\rangle_{\mathcal{B}}\|^2\|\boldsymbol{H}\|_\sigma \leq 0$, the expected loss decreases, and the iterate stays within the quadratic basin and does not diverge. □

Remark C.2. A necessary and sufficient condition for the expected loss to decrease is that $-\eta\|\nabla L\|^2 + \frac{\eta^2}{2}\mathbb{E}[\langle\boldsymbol{g}\rangle_{\mathcal{B}}^T\boldsymbol{H}\langle\boldsymbol{g}\rangle_{\mathcal{B}}] \leq 0$ from (39), i.e., $\frac{\mathbb{E}[\langle\boldsymbol{g}\rangle_{\mathcal{B}}^T\boldsymbol{H}\langle\boldsymbol{g}\rangle_{\mathcal{B}}]}{\mathbb{E}[\|\langle\boldsymbol{g}\rangle_{\mathcal{B}}\|^2]} \leq \frac{2\rho_B}{\eta}$. Thus, when the batch gradients are aligned with the sharpest direction, the condition is equivalent to $\|\boldsymbol{H}\|_\sigma \leq \frac{2\rho_B}{\eta}$.

Remark C.3. We can qualitatively analyze the norm of the batch gradient $\langle\boldsymbol{g}\rangle_{\mathcal{B}}$ and the ratio ρ_B with respect to the batch size B. First, if $B = N = |\mathcal{D}|$ (full-batch), then $\rho_B = 1$ by definition. Second, if B is large enough, then $\rho_B \approx 1$ since $\mathbb{E}_{|\mathcal{B}|=B}[\|\langle\boldsymbol{g}\rangle_{\mathcal{B}}\|^2] \approx \|\langle\boldsymbol{g}\rangle\|^2$ (cf. $B = 512$ in Figure 24). Now, we consider the third case of $B \ll N$. Put $\boldsymbol{g}_{\mathcal{B}} = \langle\boldsymbol{g}\rangle_{\mathcal{B}}$ and $\boldsymbol{g}_{\mathcal{D}} = \langle\boldsymbol{g}\rangle$. Then, we have

$$\mathbb{E}[\|\boldsymbol{g}_{\mathcal{B}} - \boldsymbol{g}_{\mathcal{D}}\|^2] = \mathbb{E}[(\boldsymbol{g}_{\mathcal{B}} - \boldsymbol{g}_{\mathcal{D}})^T(\boldsymbol{g}_{\mathcal{B}} - \boldsymbol{g}_{\mathcal{D}})] = \mathbb{E}[\boldsymbol{g}_{\mathcal{B}}^T\boldsymbol{g}_{\mathcal{B}} - \boldsymbol{g}_{\mathcal{B}}^T\boldsymbol{g}_{\mathcal{D}} - \boldsymbol{g}_{\mathcal{D}}^T\boldsymbol{g}_{\mathcal{B}} + \boldsymbol{g}_{\mathcal{D}}^T\boldsymbol{g}_{\mathcal{D}}] = \mathbb{E}[\|\boldsymbol{g}_{\mathcal{B}}\|^2] - \|\boldsymbol{g}_{\mathcal{D}}\|^2 \tag{41}$$

and

$$\mathbb{E}[\|\boldsymbol{g}_{\mathcal{B}} - \boldsymbol{g}_{\mathcal{D}}\|^2] = \frac{1}{B}\frac{N-B}{N-1}\mathbb{E}[\|\boldsymbol{g} - \boldsymbol{g}_{\mathcal{D}}\|^2] \tag{42}$$

(see Appendix A in Smith et al. (2021) for the detailed proof). Therefore,

$$\frac{\mathbb{E}[\|\langle\boldsymbol{g}\rangle_{\mathcal{B}}\|^2]}{\|\langle\boldsymbol{g}\rangle\|^2} = \frac{\mathbb{E}[\|\boldsymbol{g}_{\mathcal{B}}\|^2]}{\|\boldsymbol{g}_{\mathcal{D}}\|^2} = \frac{\|\boldsymbol{g}_{\mathcal{D}}\|^2 + \mathbb{E}[\|\boldsymbol{g}_{\mathcal{B}} - \boldsymbol{g}_{\mathcal{D}}\|^2]}{\|\boldsymbol{g}_{\mathcal{D}}\|^2} = 1 + \frac{\mathbb{E}[\|\boldsymbol{g}_{\mathcal{B}} - \boldsymbol{g}_{\mathcal{D}}\|^2]}{\|\boldsymbol{g}_{\mathcal{D}}\|^2} \tag{43}$$

$$= 1 + \frac{1}{B}\frac{N-B}{N-1}\frac{\mathbb{E}[\|\boldsymbol{g} - \boldsymbol{g}_{\mathcal{D}}\|^2]}{\|\boldsymbol{g}_{\mathcal{D}}\|^2} = 1 + \frac{1}{B}A = \frac{A+B}{B} \geq 1 \tag{44}$$

which becomes larger as B decreases ($B \ll N$) where $A = \frac{N-B}{N-1}\frac{\mathbb{E}[\|\boldsymbol{g}-\boldsymbol{g}_{\mathcal{D}}\|^2]}{\|\boldsymbol{g}_{\mathcal{D}}\|^2} > 0$. In other words, the smaller the batch size B, the smaller the ratio $\rho_B = \frac{\|\langle\boldsymbol{g}\rangle\|^2}{\mathbb{E}[\|\langle\boldsymbol{g}\rangle_{\mathcal{B}}\|^2]} = \frac{B}{A+B}$ Thus, if the batch gradients are diverse such that $\frac{\mathbb{E}[\|\boldsymbol{g}-\boldsymbol{g}_{\mathcal{D}}\|^2]}{\|\boldsymbol{g}_{\mathcal{D}}\|^2} \gg B$ and $A \gg B$, then we have $\rho_B \propto B$ which leads to the sharpness $\propto \frac{B}{\eta}$, a similar result with the linear scaling rule (Goyal et al., 2017). We refer the readers to Lee & Jang (2023) for further detailed analysis.

D. Gradient descent in the top Hessian subspace

Gur-Ari et al. (2018) showed that the gradient of the loss quickly converges to a tiny subspace spanned by a few top eigenvectors of the Hessian after a short training. Then, the top Hessian subspace does not evolve much, which implies gradient descent happens in a tiny subspace. However, the underlying mechanism has not been fully understood.

Direction of $\boldsymbol{q}^{(i)}$ and two salient elements We investigate the direction of the eigenvector $\boldsymbol{q}^{(i)}$ $(1 \le i \le C-1)$ of $\boldsymbol{M}$. The eigenvector

$$\boldsymbol{q}^{(i)} = \alpha \left(\frac{\boldsymbol{p}_j}{\boldsymbol{p}_j - \lambda^{(i)}} \right)_j \tag{45}$$

can be obtained from Theorem 4.1 (b) for some $\alpha > 0$. Here, the magnitude of the denominator $|\boldsymbol{p}_j - \lambda^{(i)}|$ is small for the two indices $j = (i), (i+1)$, and is large for the others. This is because the eigenvalue $\lambda^{(i)}$ lies between $\boldsymbol{p}_{(i+1)}$ and $\boldsymbol{p}_{(i)}$ (Theorem 4.1 (c)). Therefore, the eigenvector $\boldsymbol{q}^{(i)}$ has a relatively large positive value in $\boldsymbol{q}^{(i)}_{(i)}$ and a large negative value in $\boldsymbol{q}^{(i)}_{(i+1)}$ compared to the other components.

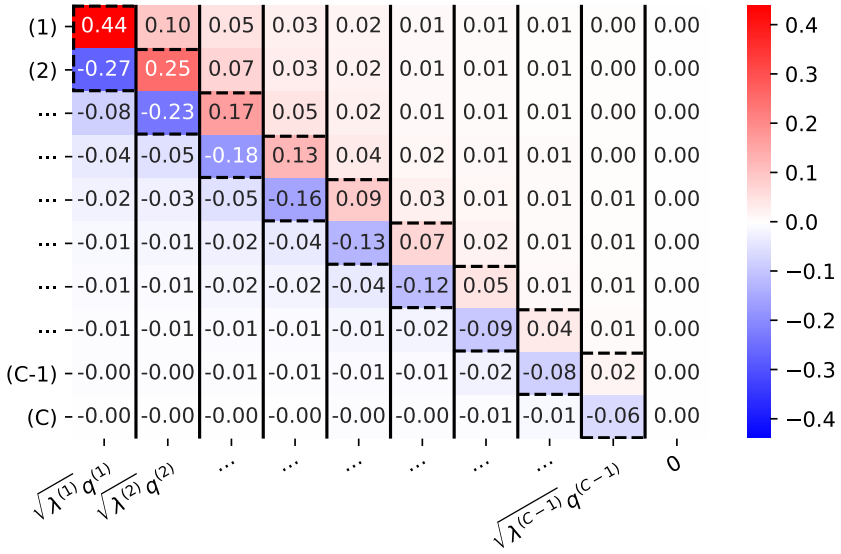

Figure 8: Heatmap of the matrix $\boldsymbol{Q}\boldsymbol{\Lambda}^{1/2} = [\sqrt{\lambda^{(1)}}\boldsymbol{q}^{(1)}; \cdots; \sqrt{\lambda^{(C-1)}}\boldsymbol{q}^{(C-1)}; \boldsymbol{0}]$ averaged over the training set $\mathcal{D}$ where $\boldsymbol{M} = \boldsymbol{Q}\boldsymbol{\Lambda}\boldsymbol{Q}^T$. Each column of $\boldsymbol{Q}\boldsymbol{\Lambda}^{1/2}$ visualizes the color-encoded direction of $\boldsymbol{q}^{(i)}$ multiplied by $\sqrt{\lambda^{(i)}}$. We highlighted the elements $\boldsymbol{q}^{(i)}_{(i)}$ and $\boldsymbol{q}^{(i)}_{(i+1)}$ with the dashed boxes for $0 \le i \le C-1$ (see Theorem 4.1 (b)).

Figure 8 shows the directions of $\boldsymbol{q}^{(i)}$ with the heatmap of the matrix $\boldsymbol{Q}\boldsymbol{\Lambda}^{1/2}$ where $\boldsymbol{Q}\boldsymbol{\Lambda}^{1/2} = [\sqrt{\lambda^{(1)}}\boldsymbol{q}^{(1)}; \cdots; \sqrt{\lambda^{(C-1)}}\boldsymbol{q}^{(C-1)}; \boldsymbol{0}] \in \mathbb{R}^{C \times C}$ for $\boldsymbol{M} = \boldsymbol{Q}\boldsymbol{\Lambda}\boldsymbol{Q}^T$. As expected, considering each column of $\boldsymbol{Q}\boldsymbol{\Lambda}^{1/2}$, the eigenvector $\boldsymbol{q}^{(i)}$ is colored in red $(+)$ at $\boldsymbol{q}^{(i)}_{(i)}$ and in blue $(-)$ at $\boldsymbol{q}^{(i)}_{(i+1)}$ for $1 \le i \le C-1$. The two salient elements are highlighted with the dashed boxes.

Direction of $\boldsymbol{J}\boldsymbol{q}^{(i)}$ and margin maximization In light of the previous discussion, the direction of

$$\boldsymbol{J}\boldsymbol{q}^{(i)} = (\boldsymbol{J}\boldsymbol{q}^{(i)})|_{\boldsymbol{\theta}=\boldsymbol{\theta}^{(t)}} = \nabla_{\boldsymbol{\theta}} \left(\boldsymbol{q}^{(i)}(\boldsymbol{\theta}^{(t)})^T \boldsymbol{z}(\boldsymbol{\theta}) \right) \Big|_{\boldsymbol{\theta}=\boldsymbol{\theta}^{(t)}} \in \mathbb{R}^m \tag{46}$$

is approximately a direction maximizing $\boldsymbol{q}^{(i)}_{(i)}\boldsymbol{z}_{(i)} + \boldsymbol{q}^{(i)}_{(i+1)}\boldsymbol{z}_{(i+1)}$ at the current parameter $\boldsymbol{\theta}^{(t)} \in \Theta$ because the other terms are relatively small. In other words, it tends to maximize the margin $\boldsymbol{z}_{(i)} - \boldsymbol{z}_{(i+1)}$ in the logit space $\mathcal{Z}$ between the two classes (i) and $(i+1)$ (see Figure 8). In particular, $\boldsymbol{J}\boldsymbol{q}^{(1)}$ is approximately a direction that maximizes the margin between the most likely class and the second most likely class.

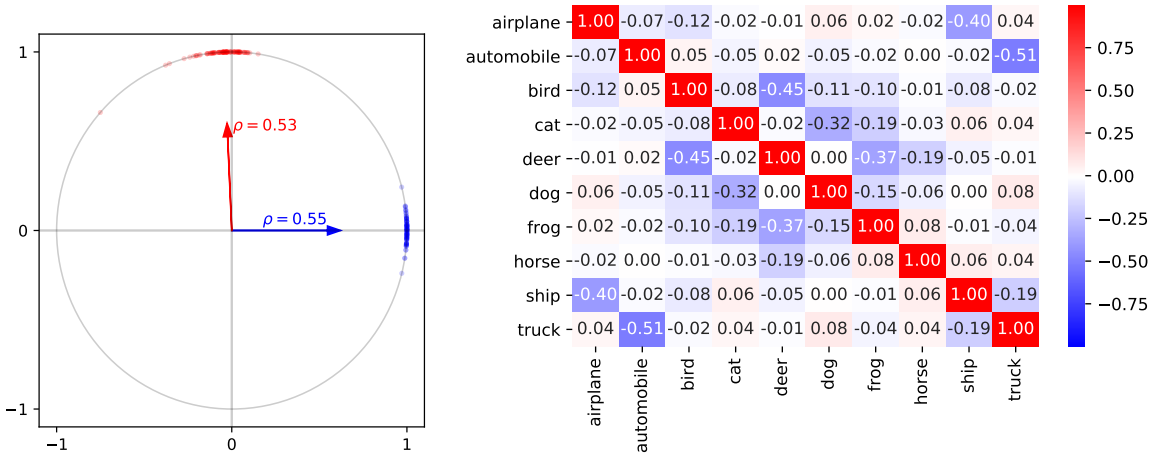

Figure 9: **There are C clusters of $\boldsymbol{m} = \sqrt{\lambda^{(1)}}\boldsymbol{J}\boldsymbol{q}^{(1)}$ according to the most likely class (not the true class). Left: (Within-class similarity)** Directional data of $\boldsymbol{m}$ from $\mathcal{D}_i$ and $\mathcal{D}_j$ for the two classes, dog (blue) and automobile (red). They are projected onto the 2D-plane spanned by the two mean vectors indicated with the arrows. We highlight the MRL ρ for each class. The directional data of $\boldsymbol{m}$ are concentrated within the class but separated from each other. **Right: (Between-class dissimilarity)** Cosine similarities between each pair of $\{\overline{\boldsymbol{m}}^i\}$. They are mostly orthogonal, but some pairs are even negatively aligned, for example, automobile and truck. This is because the examples predicted to be automobile mostly have the second most probable class as truck.

Clustering of $\boldsymbol{J}\boldsymbol{q}^{(1)}$ and the most probable class We first define following subsets of the training set according to the most probable class (and the second most one): $\mathcal{D}_i = \{\boldsymbol{x} \in \mathcal{D} : \boldsymbol{c}_1(\boldsymbol{x}) = i\} \subset \mathcal{D}$ and $\mathcal{D}_{ij} = \{\boldsymbol{x} \in \mathcal{D} : \boldsymbol{c}_1(\boldsymbol{x}) = i, \boldsymbol{c}_2(\boldsymbol{x}) = j\} \subset \mathcal{D}$ for $i \neq j \in [C]$. Note that $\mathcal{D}_i = \bigcup_{j \neq i} \mathcal{D}_{ij}$. Given two examples from $\mathcal{D}_{ij}$, their $\boldsymbol{J}\boldsymbol{q}^{(1)}$ are expected to be highly aligned to each other. This is because the direction of $\boldsymbol{J}\boldsymbol{q}^{(1)}$ is approximately a direction of maximizing the margin and of learning the features to discriminate the class i from the class j. Moreover, two examples from $\mathcal{D}_i$ also have highly-aligned $\boldsymbol{J}\boldsymbol{q}^{(1)}$. Figure 9 (Left) shows the concentration of the directional data of $\boldsymbol{J}\boldsymbol{q}^{(1)}$ from $\mathcal{D}_i$. We also compute the mean resultant length (MRL) to measure the concentration. The MRL ρ of the directional variable $V \in \mathbb{S}^{m-1} \equiv \{\boldsymbol{v} \in \mathbb{R}^m : \|\boldsymbol{v}\| = 1\}$ defined as $\rho \equiv \|\mathbb{E}[V]\| \in [0, 1]$ indicates how V is distributed (the higher, the more concentrated).

Now, we focus on $\boldsymbol{m} \equiv \sqrt{\lambda^{(1)}}\boldsymbol{J}\boldsymbol{q}^{(1)}$ as the other $\lambda^{(i)}$-terms are dominated by the $\lambda^{(1)}$-term after a few epochs (see Appendix M for details). Then, we follow a similar approach from Papyan (2019) and provide the following equation:

$$\langle \boldsymbol{m}\boldsymbol{m}^T \rangle = \sum_{i=1}^{C} \gamma_i \langle \boldsymbol{m}\boldsymbol{m}^T \rangle_{\mathcal{D}_i} = \sum_{i=1}^{C} \gamma_i (\overline{\boldsymbol{m}}^i \overline{\boldsymbol{m}}^{iT} + \langle (\boldsymbol{m} - \overline{\boldsymbol{m}}^i)(\boldsymbol{m} - \overline{\boldsymbol{m}}^i)^T \rangle_{\mathcal{D}_i}) \tag{47}$$

where $\gamma_i = |\mathcal{D}_i|/|\mathcal{D}|$ and $\overline{\boldsymbol{m}}^i = \langle \boldsymbol{m} \rangle_{\mathcal{D}_i}$. Here, the covariance term $\langle (\boldsymbol{m} - \overline{\boldsymbol{m}}^i)(\boldsymbol{m} - \overline{\boldsymbol{m}}^i)^T \rangle_{\mathcal{D}_i}$ is weak as $\boldsymbol{m}$ is concentrated within $\mathcal{D}_i$, and thus we can roughly approximate $\boldsymbol{G}$ with $\sum_{i=1}^{C} \gamma_i \overline{\boldsymbol{m}}^i \overline{\boldsymbol{m}}^{iT}$. This implies that the top eigensubspace of the Hessian highly overlaps with the at most C-dimensional subspace spanned by $\{\overline{\boldsymbol{m}}^i\}_{i=1}^C$. Figure 9 (Right) demonstrates that the mean vectors $\{\overline{\boldsymbol{m}}^i\}_{i=1}^C$ are well separated from each other. This also implies the outliers in the Hessian spectrum (Sagun et al., 2016; 2017).

Why gradient descent happens mostly in the top Hessian subspace? Given input $\boldsymbol{x}$, after the model becomes to correctly predict the true label y, the gradient descent direction $-\boldsymbol{g} = \boldsymbol{J}(\boldsymbol{e}^y - \boldsymbol{p})$ used in the training tends to be highly aligned with $\boldsymbol{J}\boldsymbol{q}^{(1)}$. This is because $\boldsymbol{e}^y - \boldsymbol{p}$ and $\boldsymbol{q}^{(1)}$ both have similar direction. They have positive values $1 - \boldsymbol{p}_y$ and $\boldsymbol{q}^{(1)}_{(1)}$ in $y(= (1))$-th element, negative value $-\boldsymbol{p}_i$ and $\boldsymbol{q}^{(1)}_i$ in the others, and especially large negative value for the second most probable class $i = (2)$. Figure 10 (Middle) shows the cosine similarity between the gradient descent direction $-\boldsymbol{g}$ and $\boldsymbol{J}\boldsymbol{q}^{(1)}$. As expected, they are highly aligned with the cosine similarity near 1 as the two vectors $\boldsymbol{e}^y - \boldsymbol{p}$ and $\boldsymbol{q}^{(1)}$ become more aligned to each other.

Next, we move on to the subspace $\mathcal{S} \equiv \text{span}(\{\overline{\boldsymbol{m}}^i\}_{i=1}^C)$ spanned by $\{\overline{\boldsymbol{m}}^i\}_{i=1}^C$. As each $\boldsymbol{m} = \sqrt{\lambda^{(1)}}\boldsymbol{J}\boldsymbol{q}^{(1)}$ is highly aligned with $-\boldsymbol{g}$, it is reasonably expected that the total gradient $\boldsymbol{g}^{\mathcal{D}}$ lies in the subspace $\mathcal{S}$. To measure how much the vector $\boldsymbol{v} \neq \boldsymbol{0}$

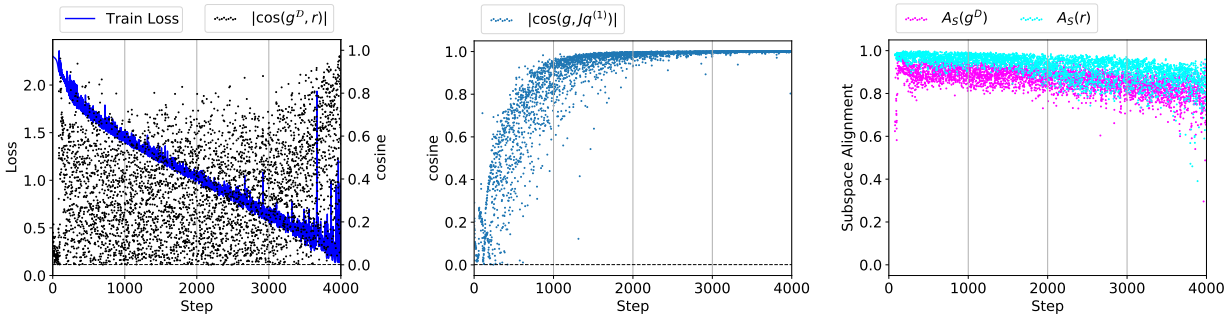

Figure 10: **Left: Total gradient $g^{\mathcal{D}} = \langle g \rangle$ is aligned with the top eigenvector r of the Hessian H at each step during training (Jastrzębski et al., 2019; Gur-Ari et al., 2018).** They have large cosine similarities considering that they are very high-dimensional. We highlighted the cosine value for random m-dimensional vectors in Θ with the dashed horizontal line (about 1e-3). **Middle: $Jq^{(1)}$ (or m) is highly aligned with the gradient g for given example at each step during training.** They have cosine similarities near 1 as the model becomes to correctly predict the true label. See Figure 1 (Right) together. **Right: Total gradient $g^{\mathcal{D}}$ and the top eigenvector r of the Hessian H mostly lie in the at most C-dimensional subspace $\mathcal{S}$ spanned by $\{\overline{\boldsymbol{m}}^i\}_{i=1}^C$.** The subspace alignment measure $A_{\mathcal{S}}$ is defined in (48).

is aligned with the subspace $\mathcal{S}$, we define the cosine similarity between the vector $\boldsymbol{v}$ and its projection $P_{\mathcal{S}}(\boldsymbol{v})$ onto the subspace $\mathcal{S}$ as follows:

$$A_{\mathcal{S}}(\boldsymbol{v}) \equiv \cos(\boldsymbol{v}, P_{\mathcal{S}}(\boldsymbol{v})) = \|P_{\mathcal{S}}(\boldsymbol{v})\|/\|\boldsymbol{v}\| \tag{48}$$

In particular, $A_{\mathcal{S}}(\boldsymbol{v}) = 0$ means $\boldsymbol{v} \in \mathcal{S}^{\perp}$ and $A_{\mathcal{S}}(\boldsymbol{v}) = 1$ means $\boldsymbol{v} \in \mathcal{S}$. Figure 10 (Right) shows the high alignment of the total gradient $\boldsymbol{g}^{\mathcal{D}}$ in the subspace $\mathcal{S} \equiv \text{span}(\{\overline{\boldsymbol{m}}^i\}_{i=1}^C)$ although the subspace $\mathcal{S}$ is of dimension at most $C \ll m$. Moreover, since $\boldsymbol{G}$ can be roughly approximated by $\sum_{i=1}^C \gamma^i \overline{\boldsymbol{m}}^i \overline{\boldsymbol{m}}^{iT}$, the top eigenvector $\boldsymbol{r}$ of the Hessian $\boldsymbol{H}$ is also highly aligned with the subspace $\mathcal{S}$.

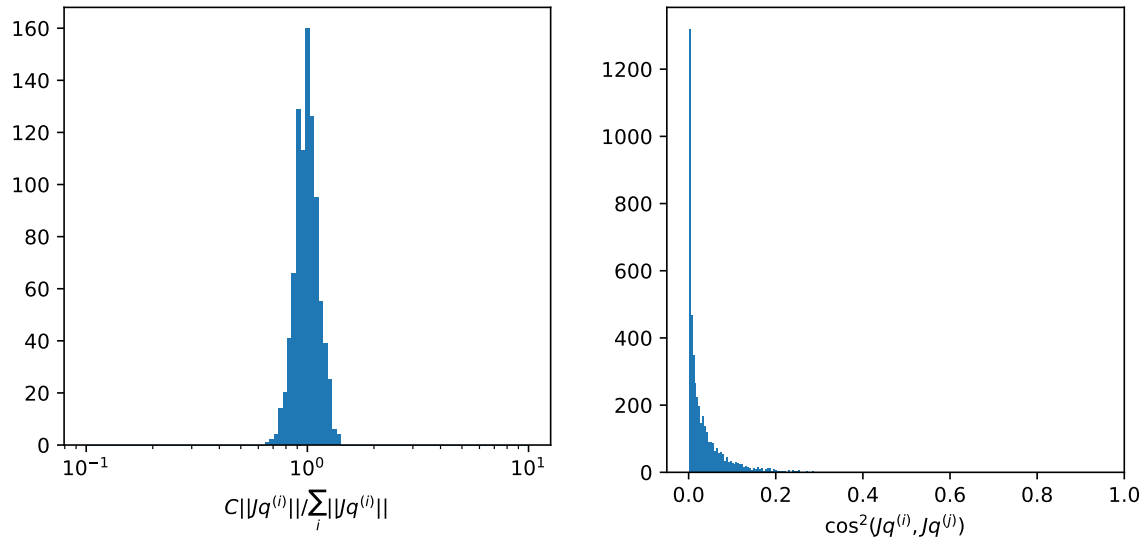

Figure 11: **Histograms of $\frac{\|\boldsymbol{J}\boldsymbol{q}^{(i)}\|}{\sum_j \|\boldsymbol{J}\boldsymbol{q}^{(j)}\|/C}$ and $\cos^2(\boldsymbol{J}\boldsymbol{q}^{(i)}, \boldsymbol{J}\boldsymbol{q}^{(j)})$.**

E. Experimental settings

We summarize the experimental settings for each Figure and Table in the main text in Table 3 and 4.

E.1. Data

We use the CIFAR-10 dataset ((Krizhevsky & Hinton, 2009), https://www.cs.toronto.edu/~kriz/cifar.html) and the MNIST dataset which have $C = 10$ number of classes. We also conduct some experiments on the CIFAR-100 dataset with the number of classes $C = 100$. We sometimes do not use the data augmentation for training (1) not to introduce the randomness in the training loss and (2) to allow the training loss to converge to a small value.

E.2. Network architectures

We use the following models: VGG-11 (VGG) (Simonyan & Zisserman, 2015) without batch-normalization, VGG for CIFAR-100 (VGG-CIFAR-100), ResNet-20 (ResNet) (He et al., 2016) wihtout batch-normalization, a 6-layer CNN (6CNN), SimpleCNN used in Jastrzębski et al. (2021) (SimpleCNN) two 3-layer fully-connected networks (3FCN-CIFAR and 3FCN-MNIST), and two WRNs (WRN-28-2/WRN-28-10) for CIFAR-10 and CIFAR-100 with the number of model parameters, $m = 9750922, 9797092, 268346, 511926, 361706, 656810, 199210, 1467610/36479194, 1479220/36536884$, respectively.

We use a modified version of the implementation of VGG-11 from https://github.com/chengyangfu/pytorch-vgg-cifar10/blob/master/vgg.py without the dropout layers and ResNet-20 from https://github.com/locuslab/edge-of-stability/blob/github/src/resnet_cifar.py. We change the last linear layer for the CIFAR-100 dataset. The 6CNN model can be expressed in the Pytorch code as follows:

```
nn.Sequential(
    nn.Conv2d(3, 32, 3, stride=1, padding=1, bias=False)
    nn.ReLU(),
    nn.Conv2d(32, 32, 4, stride=2, padding=1, bias=False)
    nn.ReLU(),
    nn.Conv2d(32, 64, 3, stride=1, padding=1, bias=False)
    nn.ReLU(),
    nn.Conv2d(64, 64, 4, stride=2, padding=1, bias=False)
    nn.ReLU(),
    nn.Flatten(),
    nn.Linear(4096, 100, bias=True),
    nn.ReLU(),
    nn.Linear(100, 10, bias=True),
)
```

and the 3FCN architecture is as follows:

```
nn.Sequential(
    nn.Flatten(),
    nn.Linear(n, 200, bias=True),
    nn.ReLU(),
    nn.Linear(200, 200, bias=True),
    nn.ReLU(),
    nn.Linear(200, 10, bias=True),
)
```

where `n=784` for 3FCN-MNIST, and `n=3072` for 3FCN-CIFAR (the same one used in Cohen et al. (2021)).

E.3. Hyperparameters

SGD (Robbins & Monro, 1951) with the learning rate η can be expressed as follows:

$$\boldsymbol{\theta}^{(t+1)} = \boldsymbol{\theta}^{(t)} - \eta \langle \boldsymbol{g}(\boldsymbol{\theta}^{(t)}) \rangle_{\mathcal{B}^{(t)}} \tag{49}$$

where $\boldsymbol{\theta}^{(t)}$ is the model parameter, $\mathcal{B}^{(t)} \subset \mathcal{D}$ is the training batch at t-th step, and $\boldsymbol{g}(\boldsymbol{\theta}) = \nabla_{\boldsymbol{\theta}} l$. For a simple CNN, we use the simplest form of SGD as described in (49) without momentum, weight decay, and learning rate decay. However, for a larger network, we use the momentum of 0.9, the weight decay of 0.0005, and two types of learning rate schedule, the multi-step scheduler with the learning rate decay of 0.2 in [30%, 60%, 80%] of the whole training epochs and the cosine annealing (Loshchilov & Hutter, 2016).

E.4. Hessian

When computing the top eigenvalue $\|\boldsymbol{H}\|_\sigma$ and the corresponding eigenvector $\boldsymbol{r}$ of the Hessian $\boldsymbol{H}$, we use the tool developed in PyHessian ((Yao et al., 2020), `https://github.com/amirgholami/PyHessian`, MIT License) based on power iteration method using a small subset (5-25%) of training dataset $\mathcal{D}$.

E.5. Power iteration algorithm

Even though we have the secular function $v(\lambda)$ in (10) and the algorithms for computing the eigenvectors (Bunch et al., 1978), we use the power iteration in Algorithm 1 to get the top eigenvalue $\lambda^{(1)}$ of the logit Hessian $\boldsymbol{M} \in \mathbb{R}^{C\times C}$ since we can run the algorithm for a mini-batch in parallel. Then, we can compute the corresponding top eigenvector $\boldsymbol{q}^{(1)}$ from Theorem 4.1 (b). To compute the second largest eigenvalue, we apply the power iteration to $\boldsymbol{M'} \equiv \boldsymbol{M} - \lambda^{(1)}\boldsymbol{q}^{(1)}\boldsymbol{q}^{(1)T}$ instead of $\boldsymbol{M}$ after computing $\lambda^{(1)}$ and $\boldsymbol{q}^{(1)}$.

Algorithm 1 Power iteration

input matrix $\boldsymbol{M}$, maximum iteration $n_{\max}$, tolerance bound ϵ
output the spectral norm $\lambda^{(1)} = \|\boldsymbol{M}\|_\sigma$ of the matrix $\boldsymbol{M}$
  Initialize $\boldsymbol{u} \in \mathbb{R}^C$ with a random vector.
  $i \leftarrow 0$
  repeat
    $\boldsymbol{v} \leftarrow \boldsymbol{M}\boldsymbol{u}/\|\boldsymbol{M}\boldsymbol{u}\|$
    $\boldsymbol{u} \leftarrow \boldsymbol{M}^T\boldsymbol{v}/\|\boldsymbol{M}^T\boldsymbol{v}\|$
    $i \leftarrow i + 1$
  until it converges within the tolerance bound ϵ or $i \geq n_{\max}$
  return $\boldsymbol{v}^T\boldsymbol{M}\boldsymbol{u}$

We also compute the operator norm of the Jacobian $\|\langle \boldsymbol{J}\rangle\|$ with the power iteration as in Algorithm 2. It requires$(C+1)$-times scalar function differentiations with respect to $\boldsymbol{\theta}$ for each iteration.

Algorithm 2 Power iteration for the Jacobian

input logit function $\boldsymbol{z}_{\boldsymbol{\theta}}$ (not matrix $\langle \boldsymbol{J}\rangle$), maximum iteration $n_{\max}$, tolerance bound ϵ
output the operator norm $\|\langle \boldsymbol{J}\rangle\|$ of the Jacobian
  Initialize $\boldsymbol{u} \in \mathbb{R}^C$ with a random vector.
  $i \leftarrow 0$
  repeat
    Compute $\langle \boldsymbol{J}\rangle\boldsymbol{u} = \nabla_{\boldsymbol{\theta}}\langle \boldsymbol{u}^T\boldsymbol{z}\rangle$ ($1\times$ scalar function differentiation)
    $\boldsymbol{v} \leftarrow \langle \boldsymbol{J}\rangle\boldsymbol{u}/\|\langle \boldsymbol{J}\rangle\boldsymbol{u}\|$
    Compute $\langle \boldsymbol{J}\rangle^T\boldsymbol{v} = (\boldsymbol{v}^T\langle \boldsymbol{J}\rangle)^T = [\boldsymbol{v}^T\nabla_{\boldsymbol{\theta}}\langle \boldsymbol{z}_1\rangle, \cdots, \boldsymbol{v}^T\nabla_{\boldsymbol{\theta}}\langle \boldsymbol{z}_C\rangle]^T$ ($C\times$ scalar function differentiations)
    $\boldsymbol{u} \leftarrow \langle \boldsymbol{J}\rangle^T\boldsymbol{v}/\|\langle \boldsymbol{J}\rangle^T\boldsymbol{v}\|$
    $i \leftarrow i + 1$
  until it converges within the tolerance bound ϵ or $i \geq n_{\max}$
  return $\boldsymbol{v}^T(\langle \boldsymbol{J}\rangle\boldsymbol{u})$

Table 3: Experimental settings for each Figure and Table in the main text (see Table 4 together)

Figure/Table	Dataset	Model	Batch Size	Initial lr	Others
Figure 1	CIFAR-10	6CNN	50000	0.04	Setting 1
Figure 2	CIFAR-10	6CNN	128	0.04	Setting 1
Figure 3	CIFAR-10	6CNN	50000	0.04	Setting 1
Figure 4	CIFAR-10	6CNN	128	0.01	Setting 1
Figure 5	CIFAR-10	6CNN	-	-	Setting 1
Figure 6 (Left)	CIFAR-10	WRN-28-10	128	0.1	Setting 2
Figure 6 (Right)	CIFAR-100	WRN-28-10	128	0.1	Setting 2
Table 1	CIFAR-10	SimpleCNN	-	-	Setting 1
Table 1[a]	-	WRN-28-10/WRN-28-2[b]	128/256	0.1	Setting 3
Table 2	CIFAR-10	PreAct ResNet-18/WRN-34-10[c]	128	0.1	Setting 4

[a] For SAM, we use $\rho = 0.05/0.1$ for CIFAR-10/100.
For ASAM, we use $\rho = 0.5/1$ and $\eta = 0.01$ for CIFAR-10/100.
For EJR, we use $\rho = 2/4$ for CIFAR-10/100.
[b] For WRN-28-2, we use a different setting from Kwon et al. (2021) that $B = 256$ and epochs of 1600 (not $B = 128$ and epochs of 200).
[b] For PreAct ResNet-18, we train the model for 110 epochs.
For WRN-34-10, we train the model for 120 epochs.

Table 4: Additional experimental settings for each Figure and Table in the main text (see Table 3 together)

Settings	Data Aug.	Label Smoothing (Müller et al., 2019)	Epochs	lr scheduler	Momentum	Weight Decay	Description
Setting 1	None	None	until convergence (< 100k)	constant	None	None	Simple SGD
Setting 2	Basic[a]	0.1	400	multi-step decay[b]	0.9	0.0005	used in SAM implementation[c]
Setting 3	Basic	0.1	200/1600	cosine annealing	0.9	0.0005	used in ASAM (Kwon et al., 2021)
Setting 4	Basic	None	110/120	multi-step decay[d]	0.9	0.0005	used in Pang et al. (2020)[e,f]

[a] random crop (with 4-pixel padding) and random horizontal flip
[b] lr decay of 0.2 at $[0.3, 0.6, 0.8]\times$epochs
[c] https://github.com/davda54/sam
[d] lr decay of 0.1 at $[100, 105]$ for 110 epochs
lr decay of 0.1 at $[100, 110]$ for 120 epochs
[e] BN mode when crafting adversarial examples for training: train for AT and eval for TRADES
[f] ReLU activation

F. The tendency of the Jacobian norm to increase

The weight norm $\|\boldsymbol{\theta}^{(t)}\|$ increases (Figure 12 (the third one)) in order to increase the logit norm $\|\boldsymbol{z}(\boldsymbol{\theta}^{(t)})\|$ (Figure 12 (the leftmost figure)) and to minimize the cross-entropy loss during training (Soudry et al., 2018). This also leads to the increase in the layerwise weight norms (Figure 14) and the Jacobian norm (Figure 5). We ran the experiments on the CIFAR-10 dataset and 6CNN with learning rate $\eta = 0.04$ and batch size 50000 (full-batch).

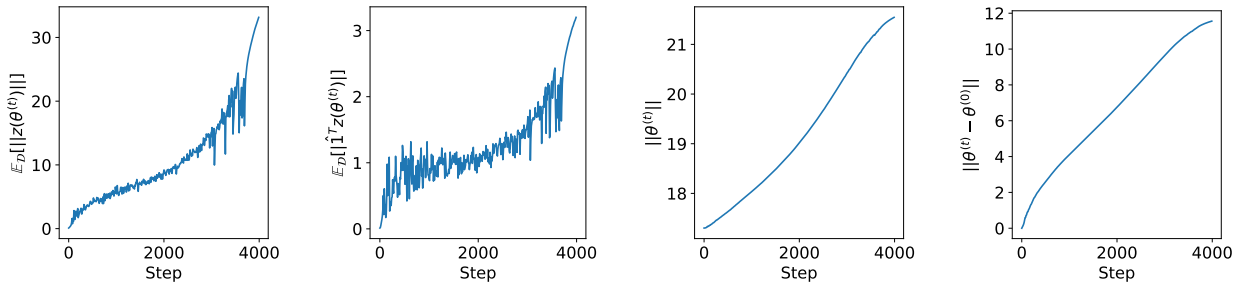

Figure 12: **(Left to Right): the logit norm $\langle\|\boldsymbol{z}(\boldsymbol{\theta}^{(t)})\|\rangle$, the absolute value of the logit sum $\langle|\hat{1}^T\boldsymbol{z}(\boldsymbol{\theta}^{(t)})|\rangle$, the weight norm $\boldsymbol{\theta}^{(t)}$, the distance from the initial weight $\|\boldsymbol{\theta}^{(t)} - \boldsymbol{\theta}^{(0)}\|$ during training (every 10 steps).** See together with Figure 5 (Left, solid red line) and Figure 4.

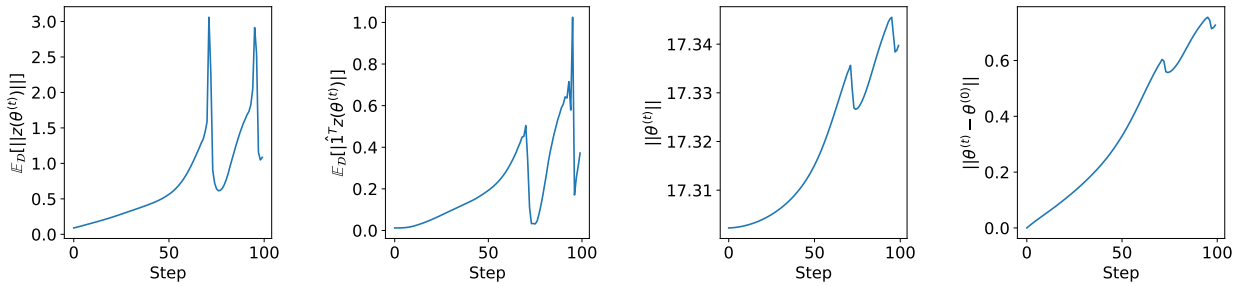

Figure 13: **(Left to Right): the logit norm $\langle\|\boldsymbol{z}(\boldsymbol{\theta}^{(t)})\|\rangle$, the absolute value of the logit sum $\langle|\hat{1}^T\boldsymbol{z}(\boldsymbol{\theta}^{(t)})|\rangle$, the weight norm $\|\boldsymbol{\theta}^{(t)}\|$, the distance from the initial weight $\|\boldsymbol{\theta}^{(t)} - \boldsymbol{\theta}^{(0)}\|$ in the early phase of training (every step).**

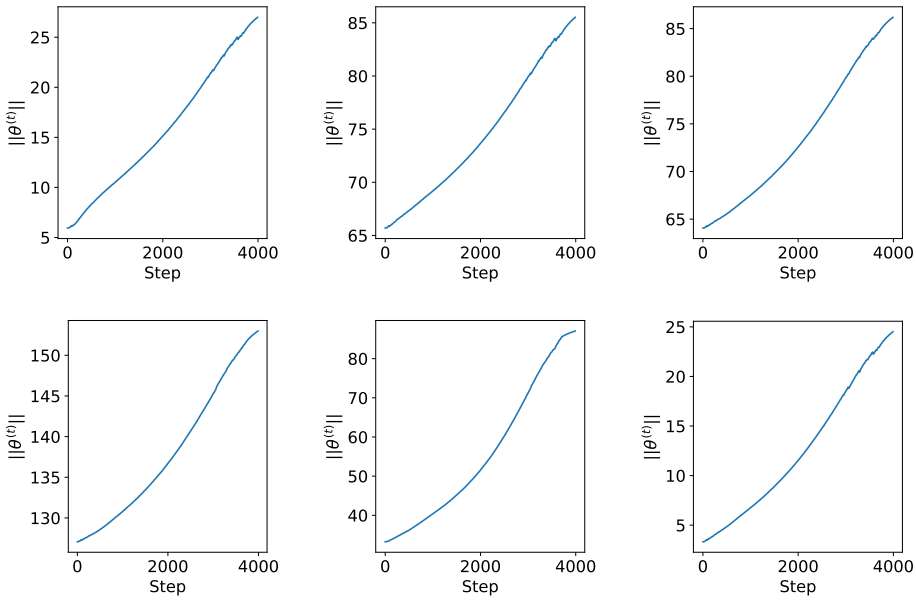

Figure 14: **The layerwise weight norms (6 layers from left to right and from top to bottom) of the 6CNN model in the early phase of training (every 10 step).**

G. $\|\boldsymbol{H}\|_\sigma \propto \langle\lambda^{(1)}\rangle\|\langle\boldsymbol{J}\rangle\hat{\mathbf{1}}\|^2$

Surprisingly, we empirically observed that $\|\boldsymbol{H}\|_\sigma \propto \langle\lambda^{(1)}\rangle\|\langle\boldsymbol{J}\rangle\hat{\mathbf{1}}\|^2$ during training where $\hat{\mathbf{1}} = \mathbf{1}/\sqrt{C} \in \mathbb{R}^C$. Since it requires further theoretical grounding, we left it as future work. We observe this relation for a variety of learning rates (Figure 15), network architectures (Figure 16 and 17), batch sizes (Figure 18 and 19), and datasets (Figure 20 and 21). At least, $\|\boldsymbol{H}\|_\sigma$ and $\langle\lambda^{(1)}\rangle\|\langle\boldsymbol{J}\rangle\hat{\mathbf{1}}\|^2$, they increase and decrease together. We emphasize that Figure 21 shows the case when the sharpness did not reach the limit $2/\eta$. This is because the learning rate is relatively low and it is easy to train a model for the MNIST dataset, and thus $\langle\lambda^{(1)}\rangle$ decreases before the sharpness reaches the limit.

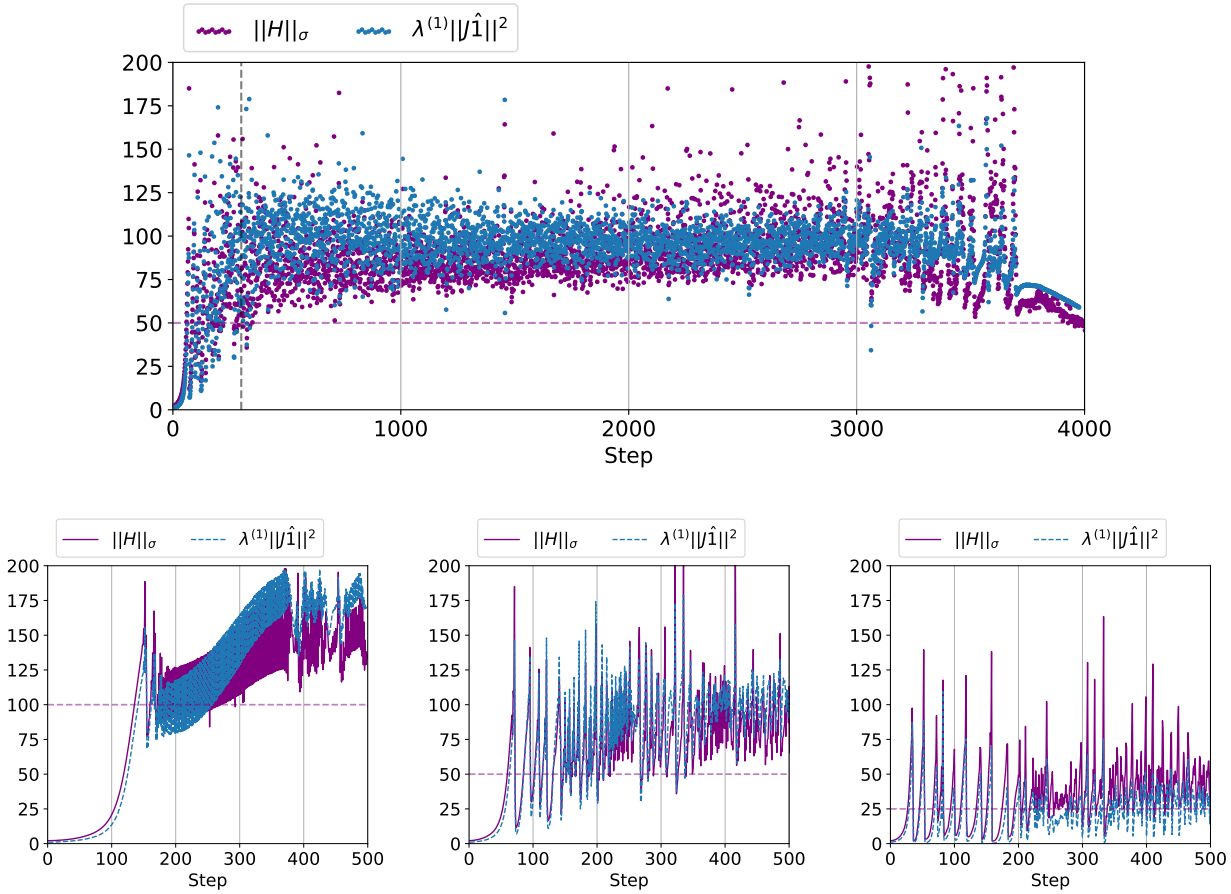

Figure 15: **The relation $\|\boldsymbol{H}\|_\sigma \propto \langle\lambda^{(1)}\rangle\|\langle\boldsymbol{J}\rangle\hat{\mathbf{1}}\|^2$ on the CIFAR-10 dataset and the 6CNN model trained using full-batch GD with different learning rates $\eta = 0.02/0.04/0.08$ (from left to right).** Bottom figures are plotted for the early phase of training. Top figure is plotted for $\eta = 0.02$. Curves are plotted for every step.

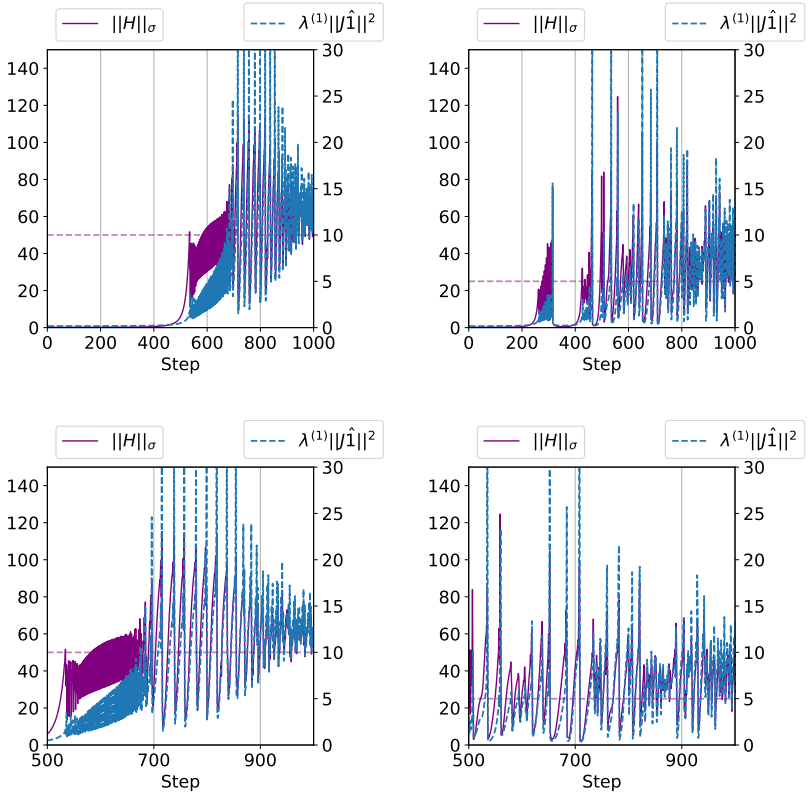

Figure 16: **The relation $\|\boldsymbol{H}\|_\sigma \propto \langle\lambda^{(1)}\rangle\|\langle\boldsymbol{J}\rangle\hat{\mathbf{1}}\|^2$ on the CIFAR-10 dataset and the VGG model trained using full-batch GD with different learning rates $\eta = 0.04/0.08$ (left/right) in the early phase of training.** Curves are plotted for every step (Top: 0-1000 stpes and Bottom: 500-1000 stpes).

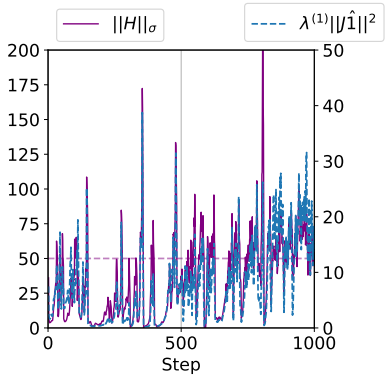

Figure 17: **The relation $\|\boldsymbol{H}\|_\sigma \propto \langle\lambda^{(1)}\rangle\|\langle\boldsymbol{J}\rangle\hat{\mathbf{1}}\|^2$ on the CIFAR-10 dataset and the ResNet model trained using SGD with learning rate $\eta = 0.04$ and batch sizes $|\mathcal{B}^{(t)}| = 128$ during training (0-1000 steps).** Curves are plotted for every step.

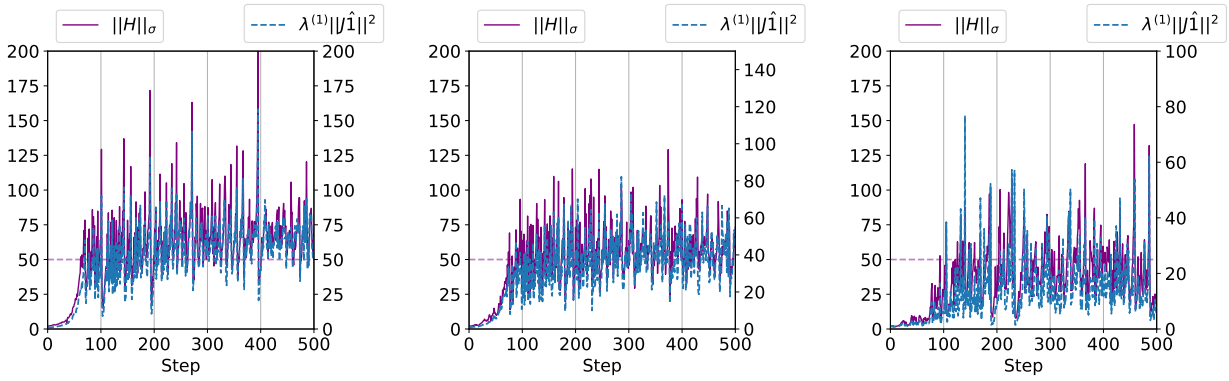

Figure 18: **The relation $\|\boldsymbol{H}\|_\sigma \propto \langle\lambda^{(1)}\rangle\|\langle\boldsymbol{J}\rangle\hat{\mathbf{1}}\|^2$ on the CIFAR-10 dataset and the 6CNN model trained using SGD with fixed learning rate $\eta = 0.04$ and different batch sizes $|\mathcal{B}^{(t)}| = 512/128/32$ (from left to right) in the initial phase (0-500 steps).** Note that the proportionality constant may change according to the batch size (the smaller the batch size, the larger the proportionality constant). Curves are plotted for every step.

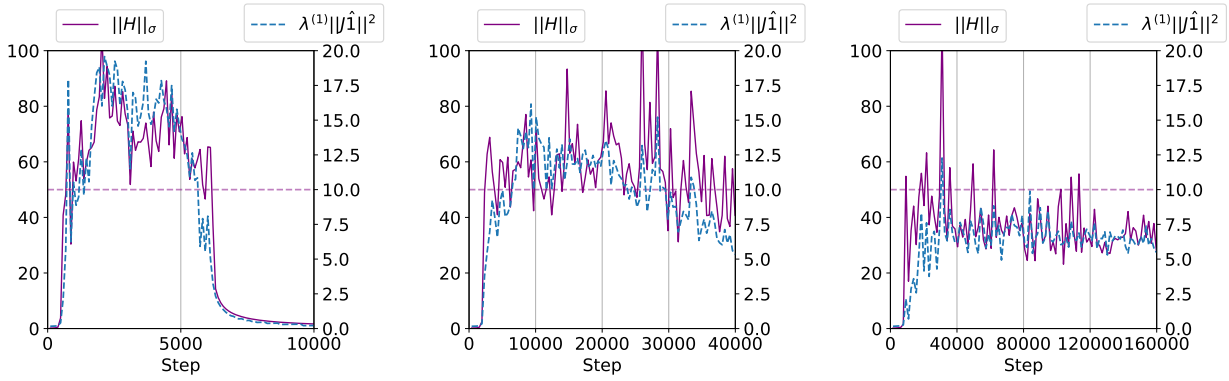

Figure 19: **The relation $\|\boldsymbol{H}\|_\sigma \propto \langle\lambda^{(1)}\rangle\|\langle\boldsymbol{J}\rangle\hat{\mathbf{1}}\|^2$ on the CIFAR-10 dataset and the VGG model trained using SGD with fixed learning rate $\eta = 0.04$ and different batch sizes $|\mathcal{B}^{(t)}| = 512/128/32$ (from left to right) during training (0-100 epochs).** Curves are plotted for every n steps ($n = 97/388/1552$) where $97 = \lfloor 50000/512 \rfloor$.

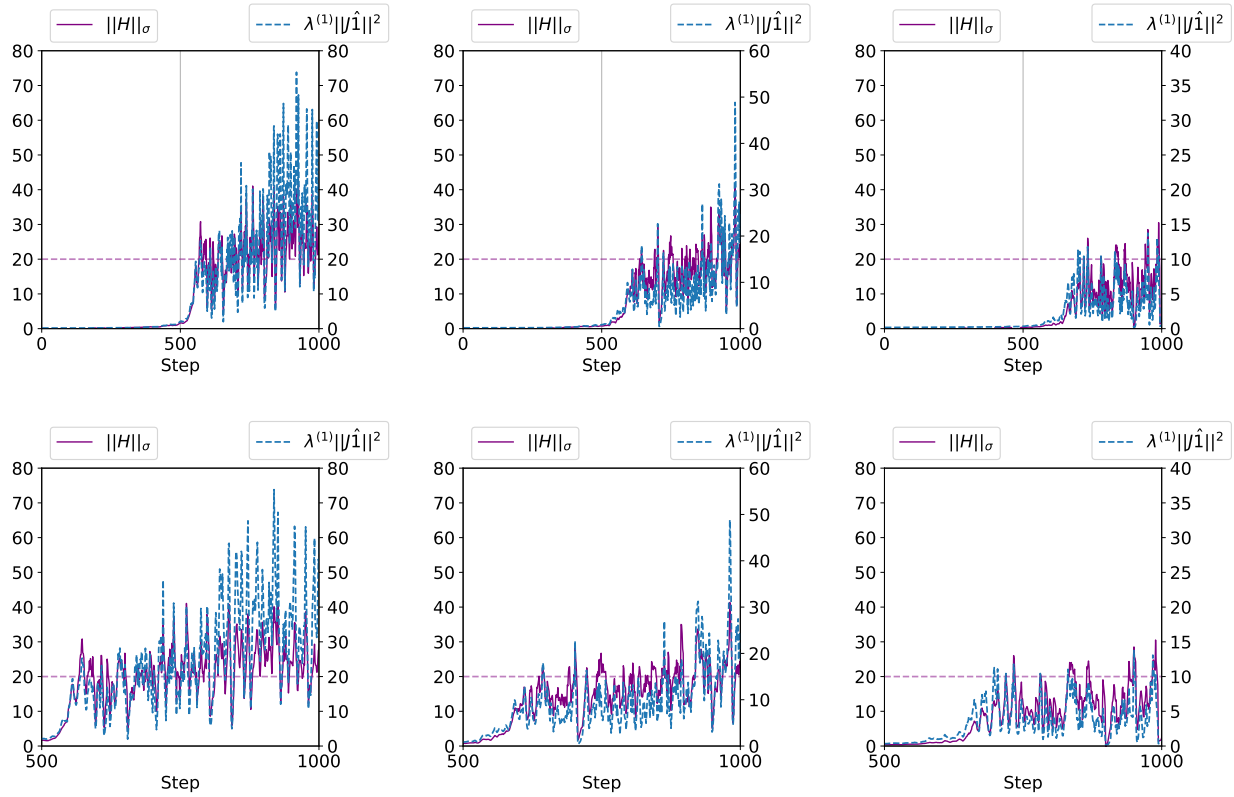

Figure 20: **The relation $\|\boldsymbol{H}\|_\sigma \propto \langle\lambda^{(1)}\rangle\|\langle\boldsymbol{J}\rangle\hat{\mathbf{1}}\|^2$ on the CIFAR-100 dataset and the VGG model trained using SGD with fixed learning rate $\eta = 0.1$ and different batch sizes $|\mathcal{B}^{(t)}| = 128/64/32$ (from left to right) in the early phase of training.** Note that the proportionality constant may change according to the batch size (the smaller the batch size, the larger the proportionality constant). Curves are plotted for every step (Top: 0-1000 stpes and Bottom: 500-1000 stpes).

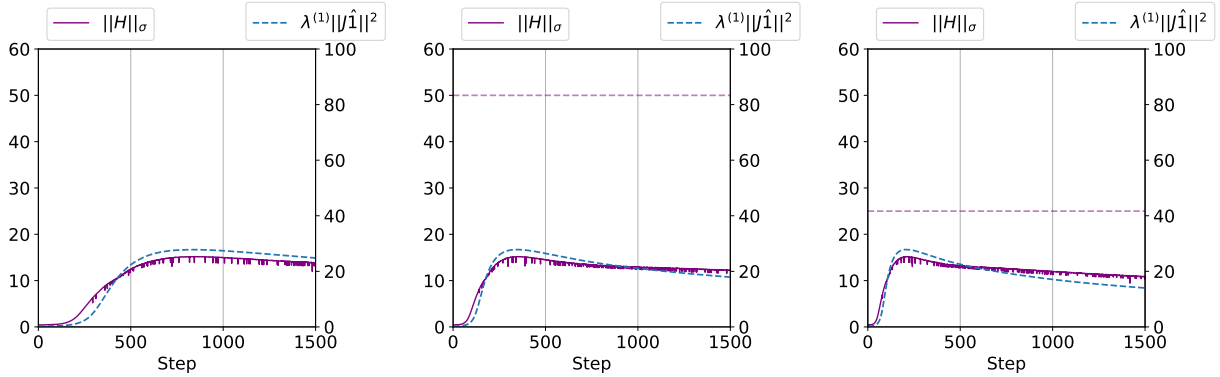

Figure 21: **The relation $\|\boldsymbol{H}\|_\sigma \propto \langle\lambda^{(1)}\rangle\|\langle\boldsymbol{J}\rangle\hat{\mathbf{1}}\|^2$ on the MNIST dataset and the 3FCN-MNIST model trained using full-batch GD with different learning rates $\eta = 0.02/0.04/0.08$ (from left to right) in the early phase of training.** Note that the sharpness did not reach the limit $2/\eta$ (the dashed horizontal line). Curves are plotted for every step.

H. Implicit Regularization on $\|\langle \boldsymbol{J} \rangle \hat{\mathbf{1}}\|^2$

We further investigate the relation $\|\boldsymbol{H}\|_\sigma \propto \langle \lambda^{(1)} \rangle \|\langle \boldsymbol{J} \rangle \hat{\mathbf{1}}\|^2$ in the previous section and its effect on $\|\langle \boldsymbol{J} \rangle \hat{\mathbf{1}}\|^2$. For the early phase of training, it is hard to see the initial rapid growth of the sharpness in this smoothed curves and when exactly the regularization begins to activate. We refer the readers to the previous section, Appendix G, for the fine-grained analysis of the early phase of training. We provide plots with different settings. Again, we observe that training with a larger learning rate and a smaller batch size limits $\|\langle \boldsymbol{J} \rangle \hat{\mathbf{1}}\|^2$ with a smaller value (dotted red lines) in the Active Regularization Period. Curves are smoothed for visual clarity.

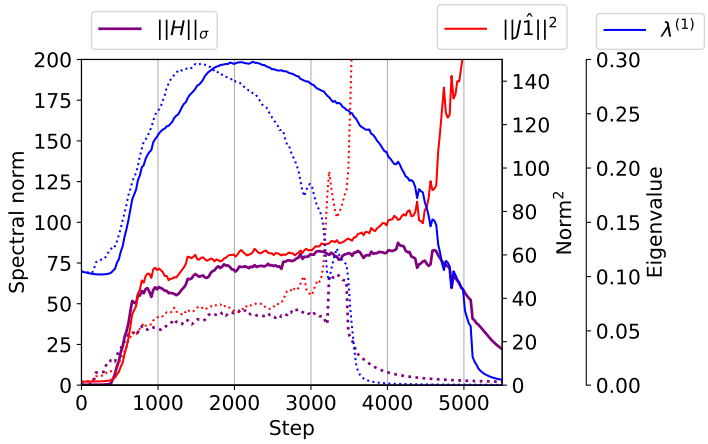

Figure 22: **The evolution of $\|\boldsymbol{H}\|_\sigma$, $\|\langle \boldsymbol{J} \rangle \hat{\mathbf{1}}\|^2$, and $\langle \lambda^{(1)} \rangle$ on the CIFAR-10 dataset and the VGG model trained using full-batch GD with the learning rates $\eta = 0.04/0.08$ (solid/dotted lines).** See the Figure 5 caption together.

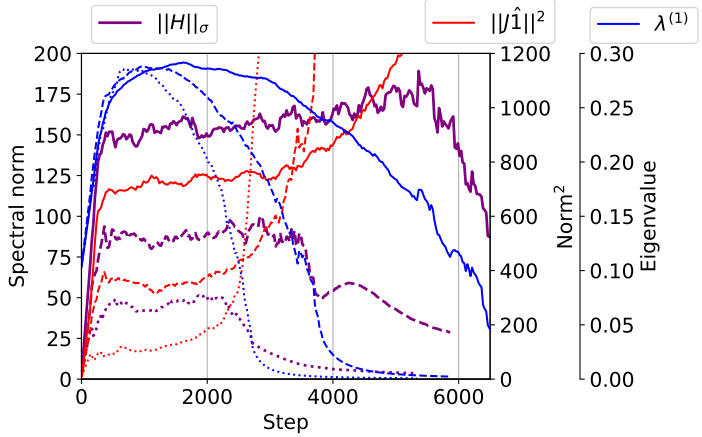

Figure 23: **The evolution of $\|\boldsymbol{H}\|_\sigma$, $\|\langle \boldsymbol{J} \rangle \hat{\mathbf{1}}\|^2$, and $\langle \lambda^{(1)} \rangle$ on the CIFAR-10 dataset and the 6CNN model trained using full-batch GD with the learning rates $\eta = 0.02/0.04/0.08$ (solid/dashed/dotted lines).** See the Figure 5 caption together.

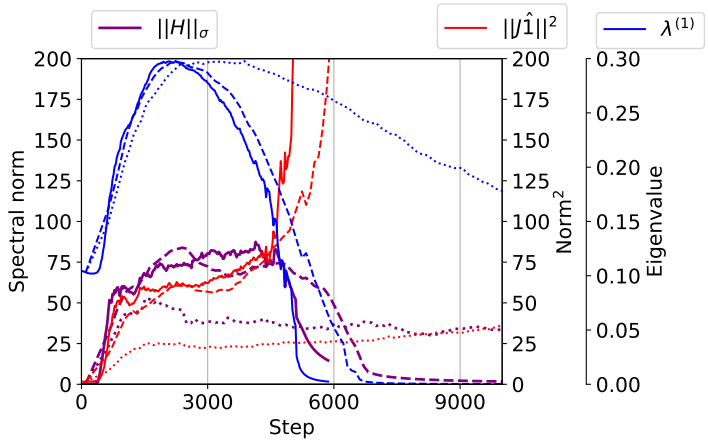

Figure 24: **The evolution of $\|\boldsymbol{H}\|_\sigma$, $\|\langle \boldsymbol{J} \rangle \hat{\mathbf{1}}\|^2$, and $\langle \lambda^{(1)} \rangle$ on the CIFAR-10 dataset and the VGG model trained using SGD with the fixed learning rate $\eta = 0.04$ and the batch sizes $|\mathcal{B}^{(t)}| = 50000(\textbf{GD})/512/32$ (solid/dashed/dotted lines).** Training with a batch size 512 shows similar evolutions to the GD training. See the Figure 5 caption together.

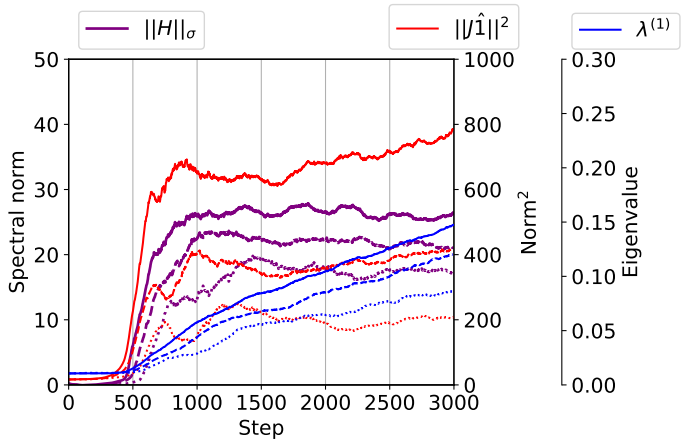

Figure 25: **The evolution of $\|\boldsymbol{H}\|_\sigma$, $\|\langle \boldsymbol{J} \rangle \hat{\mathbf{1}}\|^2$, and $\langle \lambda^{(1)} \rangle$ on the CIFAR-100 dataset and the VGG model trained using SGD with the fixed learning rate $\eta = 0.1$ and the batch sizes $|\mathcal{B}^{(t)}| = 128/64/32$ (solid/dashed/dotted lines).** See the Figure 5 caption together.

I. Figure 2 (Sharpness and Jacobian norm)

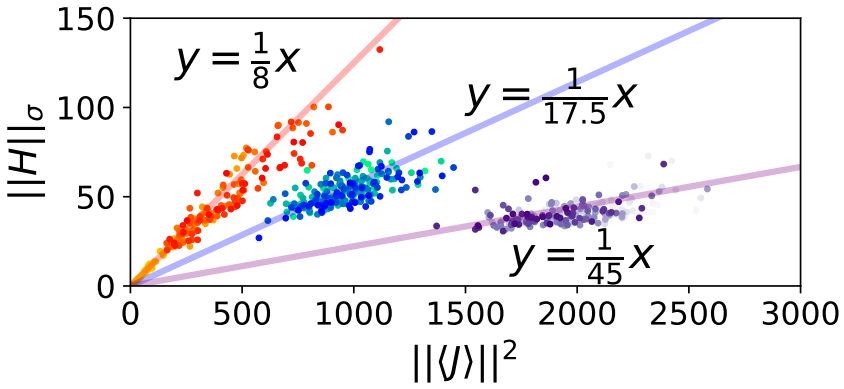

Figure 26: The sharpness $\|\boldsymbol{H}\|_\sigma$ and the Jacobian norm $\|\langle \boldsymbol{J} \rangle\|^2$ show similar behavior up to a factor $\hat{\lambda}^*$ which is locally constant and slowly changes during training (CIFAR-10, $\eta = 0.04$, $B = 128$). $\hat{\lambda}^* \approx 1/8, 1/17.5, 1/45$ (red/blue/purple) for 0-200/1000-1200/4000-4200 steps

J. Figure 3 (Visualization of the optimization trajectory)

We use UMAP (McInnes et al., 2018) to visualize the optimization trajectory $\{\boldsymbol{\theta}^{(t)}\}_{t\in[T]} \subset \Theta \subset \mathbb{R}^m$ in a 2D space. As GD enters into the Edge of Stability (Cohen et al., 2021), it oscillates in a direction nearly orthogonal to its global descent direction (Xing et al., 2018). Note that GD may not enter the Edge of Stability as shown in Figure 27 (Bottom).

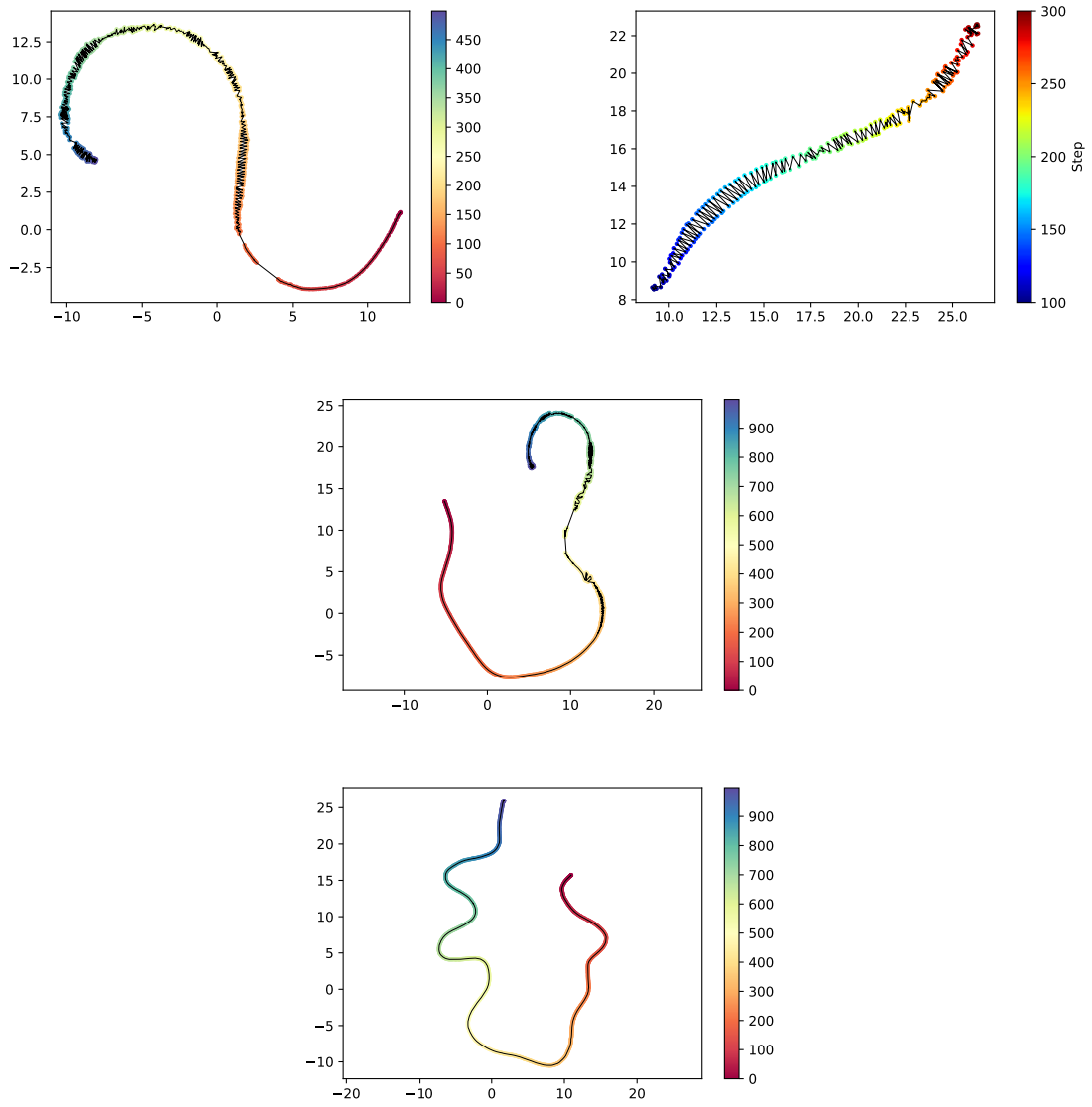

Figure 27: **Visualization of the optimization trajectory using UMAP.** UMAP on the CIFAR-10 dataset trained with 6CNN for the first 500 steps (Top Left) and for 100-300 steps (Top Right), on the CIFAR-10 dataset trained with VGG for the first 1000 steps (Middle), and on the MNIST dataset trained with 3FCN-MNIST for the first 1000 steps (Bottom) from red to blue.

K. Figure 4 (Active Regularization Period)

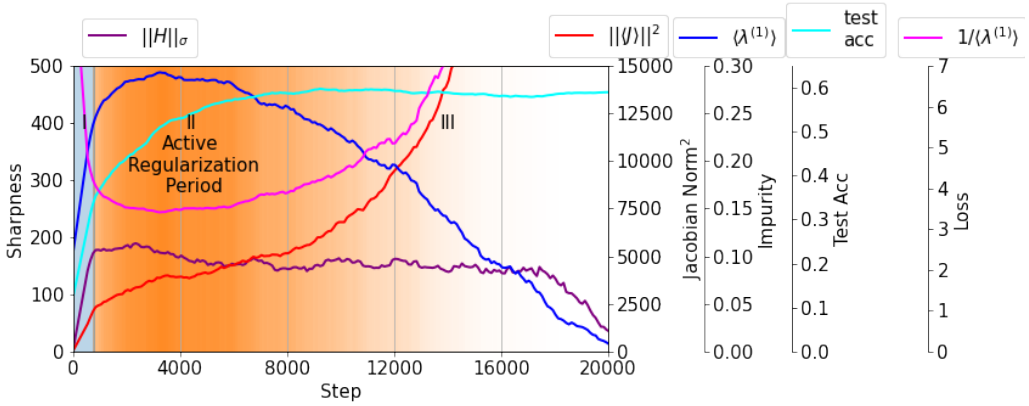

Figure 28: **Three phases of Implicit Jacobian Regularization (IJR).** Figure 4 with $1/\langle\lambda^{(1)}\rangle$. As $1/\langle\lambda^{(1)}\rangle$ increases, the Jacobian norm increases faster in phase III.

L. Figure 6 (Efficient EJR)

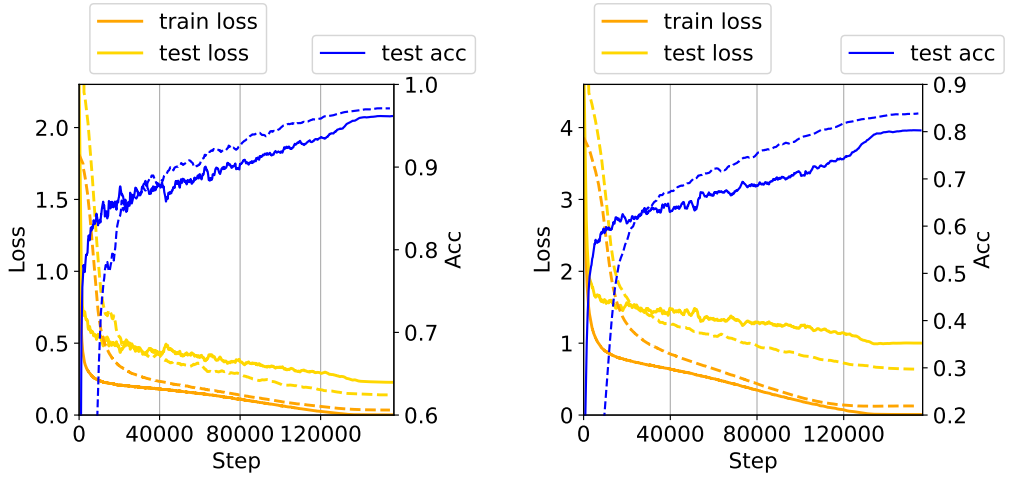

Figure 29: **The effectiveness of efficient EJR (dashed lines) compared to SGD (solid lines) trained with Setting 3 in Table 4 (cosine annealing) for WRN-28-10 on CIFAR-10 (Left) and CIFAR-100 (Right).** Unlike Figure 6, we train the model with EJR for 200 epochs and with SGD for 400 epochs and thus we plot with different x-axis to match this difference ($\times 2$ for EJR).

M. Figure 8 ($Q\Lambda^{1/2}$)

Figure 30 shows the matrix $\boldsymbol{Q}\boldsymbol{\Lambda}^{1/2}$ (Figure 8) for different training steps. Figure 8 and Figure 30 (Top Right) are plotted at the equivalent step ($t = 1000$). Figure 30 demonstrates that $\lambda^{(1)}$ becomes more dominant than the others as training progresses. The argument that there are two salient elements in each $\boldsymbol{q}^{(i)}$ in its (i)- and $(i+1)$-th elements is empirically shown to be valid throughout the training.

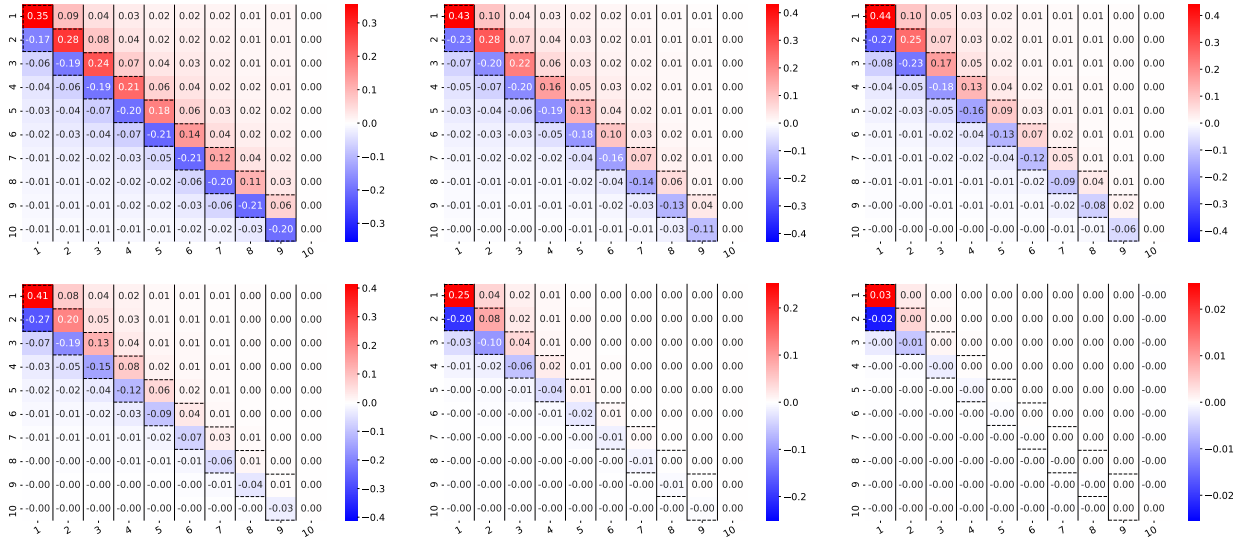

Figure 30: **Evolution of $\boldsymbol{Q}\boldsymbol{\Lambda}^{1/2}$ for some training steps during the training.** They are visualized for 100/500/1000/2000/4000/6000 steps from left to right and from top to bottom.

N. Figure 9 (Clusters of m around each $\overline{\boldsymbol{m}}^i$)

Figure 31 shows Figure 9 (Left) for some different class pairs (i, j) with negative cosine similarity, i.e., $\cos(\overline{\boldsymbol{m}}^i, \overline{\boldsymbol{m}}^j) < 0$. We use the model trained for $t = 1000$ steps. Figure 32 shows Figure 9 (Right) for different training steps. Figure 33 shows the evolution of the Mean Resultant Length (MRL) ρ in Figure 9 (Left) during training.

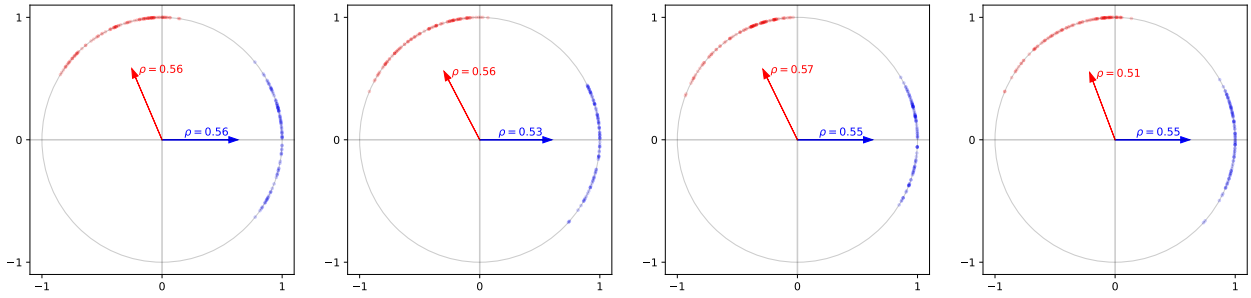

Figure 31: **Directional data of m from $\mathcal{D}_i$ and $\mathcal{D}_j$.** They are visualized for (i, j) = (airplane, ship)/(automobile, truck)/(dog, cat)/(deer, bird) from left to right. See the Figure 9 caption.

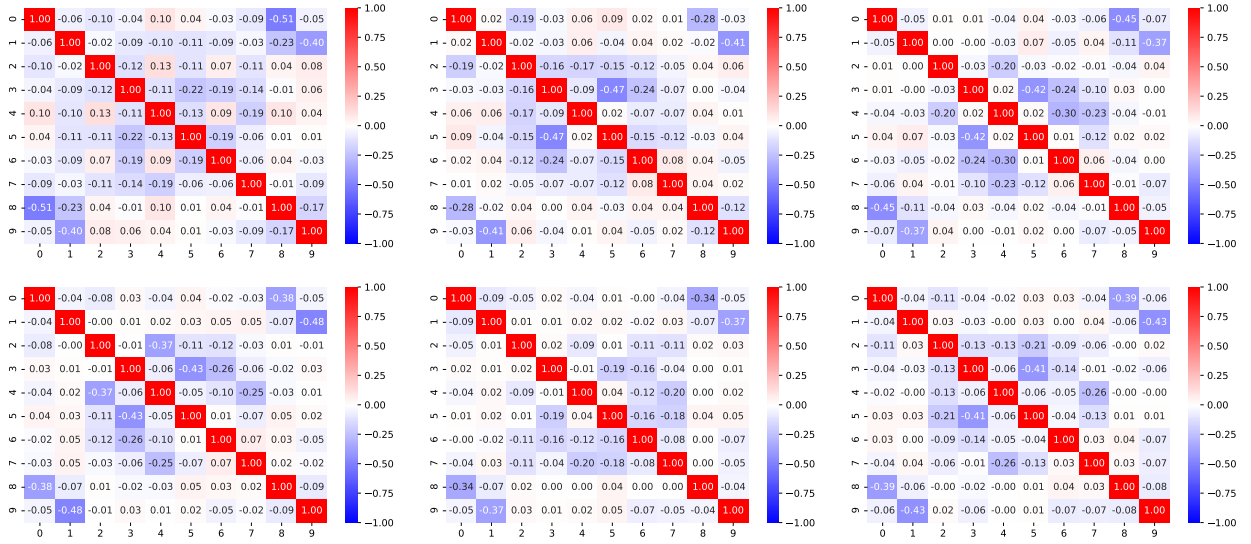

Figure 32: **Cosine similarities of $\{\overline{\boldsymbol{m}}^i\}_{i=1}^C$ during training.** They are visualized for step=500/1000/1500/2000/3000/4000 from left to right and from top to bottom. See the Figure 9 caption.

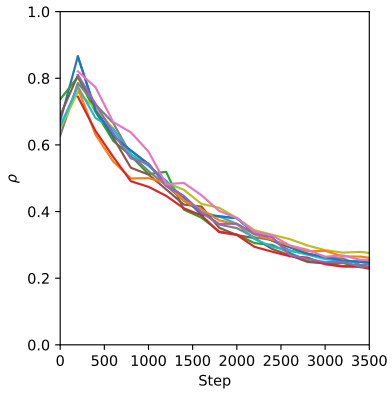

Figure 33: **The Mean Resultant Lengths (MRL) ρ for each $\mathcal{D}_i$.** Note that ρ is not defined for first few steps because some $\mathcal{D}_i$ are empty.

O. Figure 10 (gradient descent happens mostly in the top Hessian subspace)

Figure 34 shows similar results with different settings with VGG and learning rate $\eta = 0.08$.

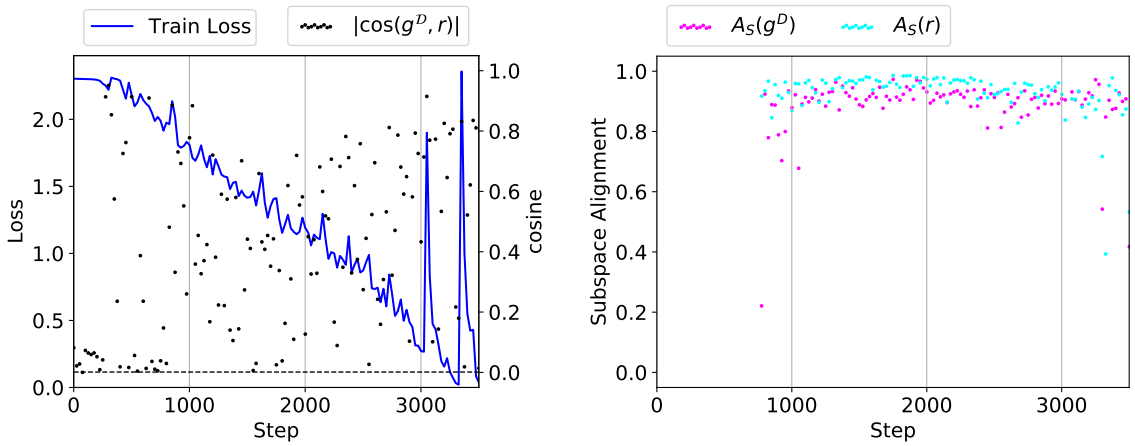

Figure 34: **(Left) Alignment between the two vectors $g^{\mathcal{D}}$ and r, and (Right) alignment of $g^{\mathcal{D}}$, r in the subspace $\mathcal{S} =$ span$(\{\overline{m}^i\}_{i=1}^C)$ using VGG with $\eta = 0.08$.** See the Figure 10 caption. They are plotted for every 25 steps. Note that $A_{\mathcal{S}}$ is not defined for first few steps (about 0-800 steps) because some $\mathcal{D}_i$ are empty.

P. Analysis of the MSE loss

In the main text, we focus on the cross-entropy loss. Here, we briefly analyze the MSE loss, $l = \frac{1}{2}\|\boldsymbol{z} - \boldsymbol{e}^y\|^2$. Then, we have $\boldsymbol{M} = \nabla_{\boldsymbol{z}}^2 l = \boldsymbol{I}$, $\lambda^{(1)} = \|\boldsymbol{M}\|_\sigma = 1$ and $\boldsymbol{G} = \langle \boldsymbol{J}\boldsymbol{J}^T \rangle$. It leads to the same conclusion as in Theorem 5.1:

$$\|\boldsymbol{G}\|_\sigma = \|\langle \boldsymbol{J}\boldsymbol{J}^T \rangle\|_\sigma \leq \langle \|\boldsymbol{J}\boldsymbol{J}^T\|_\sigma \rangle = \langle \|\boldsymbol{J}\|^2 \rangle \tag{50}$$

We empirically observed that $\|\boldsymbol{H}\|_\sigma \propto \|\langle \boldsymbol{J} \rangle\|^2$ as shown in Figure 35.

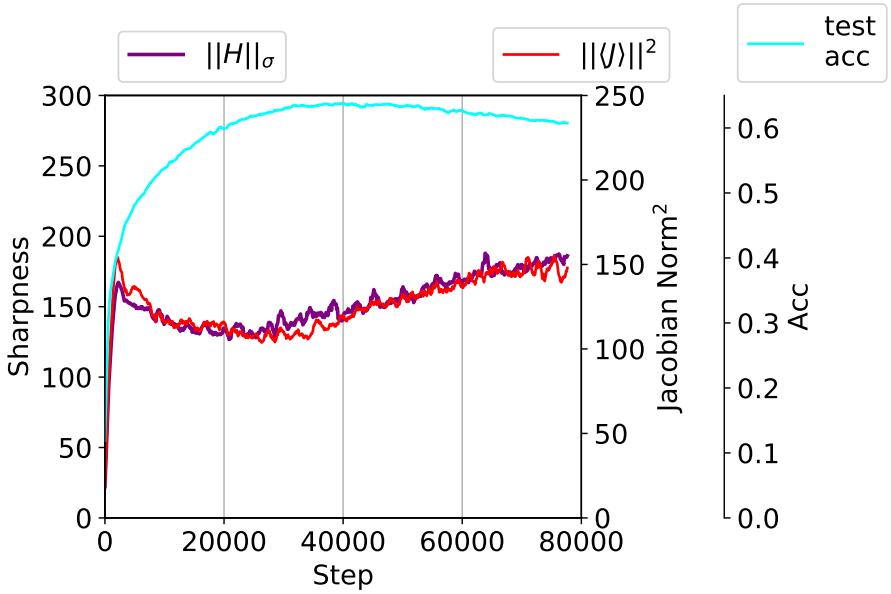

Figure 35: The sharpness $\|\boldsymbol{H}\|_\sigma$ and the Jacobian norm $\|\langle \boldsymbol{J} \rangle\|^2$ during training with the MSE loss

Q. Details of Explicit Jacobian Regularization (EJR)

We first propose a simple form of EJR with the regularized loss as follows:

$$\tilde{L}(\boldsymbol{\theta}) = L(\boldsymbol{\theta}) + \lambda_{reg}\|\langle \boldsymbol{J} \rangle\|_F^2/C \tag{51}$$

and update the model parameter as

$$\boldsymbol{\theta}^{(t+1)} = \boldsymbol{\theta}^{(t)} - \eta(\nabla_{\boldsymbol{\theta}} L(\boldsymbol{\theta}^{(t)}) + \lambda_{reg}\nabla_{\boldsymbol{\theta}}\|\langle \boldsymbol{J} \rangle\|_F^2/C) \tag{52}$$

Efficient EJR However, this requires to build a computational graph for $\|\langle \boldsymbol{J} \rangle\|_F^2$ which is inefficient for a large network (e.g. WRN-28-10). To this end, we propose an efficient variant of EJR. SVD of the Jacobian is $\langle \boldsymbol{J} \rangle = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T$ where $\boldsymbol{U} \in \mathbb{R}^{m\times m}, \boldsymbol{\Sigma} \in \mathbb{R}^{m\times C}, \boldsymbol{V} \in \mathbb{R}^{C\times C}$ where

$$\boldsymbol{\Sigma}_{i,j} = \begin{cases} \sigma_i & \text{if } i = j \in [C] \\ 0 & \text{if } i \neq j \end{cases}$$

and $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_C \geq 0$. Then, we have $\|\langle \boldsymbol{J} \rangle\| = \sigma_1$ and $\|\langle \boldsymbol{J} \rangle\|_F^2 = \sum_i \sigma_i^2$. We propose to update the model parameter as follows:

$$\boldsymbol{\theta}^{(t+1)} = \boldsymbol{\theta}^{(t)} - \eta\nabla_{\boldsymbol{\theta}} L(\hat{\boldsymbol{\theta}}^{(t)}) \tag{53}$$

where

$$\hat{\boldsymbol{\theta}} = \boldsymbol{\theta} + \tilde{\rho}_{reg}\langle \boldsymbol{J} \rangle \boldsymbol{u} \tag{54}$$

and $\tilde{\rho}_{reg} = \rho_{reg}/\|\langle \boldsymbol{J} \rangle \boldsymbol{u}\|$ with $\boldsymbol{u} \sim \mathbb{S}^{C-1}$ as in Foret et al. (2021); Liu et al. (2019). This perturbation in $\hat{\boldsymbol{\theta}}$ is highly aligned with the principal left-singular vectors of the Jacobian and the principal eigenspace of the Hessian. Therefore, the update of (53) leads to minimize the largest singular values σ_i of the Jacobian and the sharpness as well. As shown in Table 1, this method outperforms the state-of-the-art sharpness-aware optimization methods like SAM (Foret et al., 2021) and ASAM (Kwon et al., 2021) which can be attributed to the principal eigensubspace awareness.

Efficient EJR.v2 We propose another variant using $\hat{L}$ instead of L in (53) as follows:

$$\hat{L} = L + \mu_{reg}\boldsymbol{u}^T\langle \boldsymbol{z} \rangle \tag{55}$$

We can approximate $\hat{L}$ as follows:

$$\begin{aligned}
\hat{L}(\hat{\boldsymbol{\theta}}) &\approx \hat{L}(\boldsymbol{\theta}) + (\nabla_{\boldsymbol{\theta}}\hat{L}(\boldsymbol{\theta}))^T(\tilde{\rho}_{reg}\langle \boldsymbol{J} \rangle \boldsymbol{u}) && (56)\\
&= \hat{L}(\boldsymbol{\theta}) + (\nabla_{\boldsymbol{\theta}} L(\boldsymbol{\theta}))^T(\tilde{\rho}_{reg}\langle \boldsymbol{J} \rangle \boldsymbol{u}) + (\nabla_{\boldsymbol{\theta}}\mu_{reg}\boldsymbol{u}^T\langle \boldsymbol{z} \rangle)^T(\tilde{\rho}_{reg}\langle \boldsymbol{J} \rangle \boldsymbol{u}) && (57)\\
&= \hat{L}(\boldsymbol{\theta}) + (\nabla_{\boldsymbol{\theta}} L(\boldsymbol{\theta}))^T(\tilde{\rho}_{reg}\langle \boldsymbol{J} \rangle \boldsymbol{u}) + (\mu_{reg}\langle \boldsymbol{J} \rangle \boldsymbol{u})^T(\tilde{\rho}_{reg}\langle \boldsymbol{J} \rangle \boldsymbol{u}) && (58)\\
&= \hat{L}(\boldsymbol{\theta}) + (\nabla_{\boldsymbol{\theta}} L(\boldsymbol{\theta}))^T(\tilde{\rho}_{reg}\langle \boldsymbol{J} \rangle \boldsymbol{u}) + \mu_{reg}\rho_{reg}\|\langle \boldsymbol{J} \rangle \boldsymbol{u}\| && (59)\\
&\approx \hat{L}(\boldsymbol{\theta}) + (\nabla_{\boldsymbol{\theta}} L(\boldsymbol{\theta}))^T(\tilde{\rho}_{reg}\langle \boldsymbol{J} \rangle \boldsymbol{u}) + \mu_{reg}\rho_{reg}\|\langle \boldsymbol{J} \rangle\|_F/\sqrt{C} && (60)
\end{aligned}$$

We used the first-order Taylor expansion of $\hat{L}(\boldsymbol{\theta} + \hat{\rho}_{reg}\langle \boldsymbol{J} \rangle \boldsymbol{u})$ in (56). We expect additional effect of minimizing $\mu_{reg}\boldsymbol{u}^T\boldsymbol{z} + \mu_{reg}\rho_{reg}\|\langle \boldsymbol{J} \rangle\|_F/\sqrt{C} \approx \mu_{reg}\rho_{reg}\|\langle \boldsymbol{J} \rangle\|_F/\sqrt{C}$ compared to the first variant.

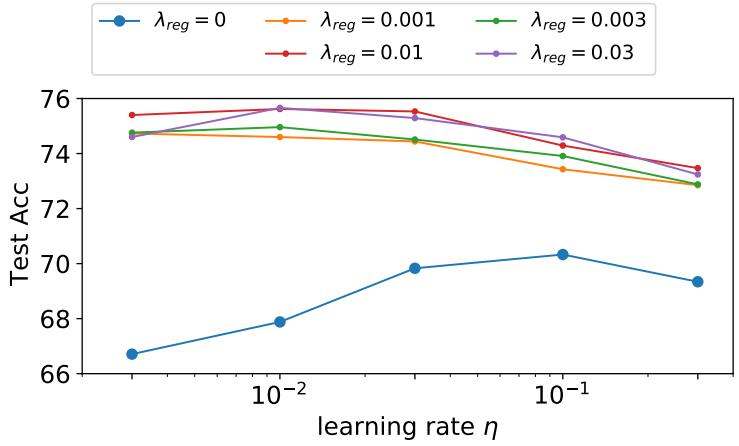

Figure 36: We plot the test accuracy for different learning rates η and regularization coefficients λ_{reg}. The models are trained with batch size of $|\mathcal{B}| = 128$ on CIFAR-10.

Table 5: **Effectiveness of Efficient EJR**. We report improvement (**ΔAcc.**) and Error Reduction Rate (**ERR**) on CIFAR-10 when trained with EJR (**+EJR**), compared to the standard training (**Baseline**). We used Setting 1 and Setting 2 in Table 4 for SimpleCNN and WRN-28-10, respectively.

Dataset	Network Architecture	Batch Size	lr	Reg. param.	Test Accuracy		ΔAcc.	ERR
					Baseline	+EJR	(%p)	(%)
CIFAR-10	SimpleCNN	128	0.003	$\lambda_{reg} = 0.01$	66.71	**75.40**	+8.69	26.10
			0.01	$\lambda_{reg} = 0.03$	67.88	**75.66**	+7.78	24.22
			0.03	$\lambda_{reg} = 0.01$	69.83	**75.53**	+5.70	18.89
			0.1	$\lambda_{reg} = 0.03$	70.33	**74.59**	+4.26	14.36
			0.3	$\lambda_{reg} = 0.01$	69.34	**73.47**	+4.13	13.47
		50000 (full-batch)	0.01	$\lambda_{reg} = 0.001$	66.81	**74.43**	+7.62	22.96
			0.03		67.72	**74.31**	+6.59	20.42
			0.1		67.53	**73.69**	+6.16	18.97
			0.3		61.08	**72.15**	+11.07	28.44
	WRN-28-10 (200 epochs)	128	0.1	$\rho_{reg} = 0.5$	95.93±0.15	**96.44**	+0.51	12.72
				$\rho_{reg} = 1$		**96.57**	+0.64	15.72
				$\rho_{reg} = 2$		**96.62**	+0.69	16.95
				$\rho_{reg} = 3$		**96.30**	+0.37	9.09
	WRN-28-10 (400 epochs)	128	0.1	$\rho_{reg} = 0.5$	96.10±0.05	**96.65**	+0.55	14.10
				$\rho_{reg} = 1$		**96.78**	+0.68	17.44
				$\rho_{reg} = 2$		**97.07**	+0.97	24.87
				$\rho_{reg} = 3$		**96.79**	+0.69	17.69
CIFAR-100	WRN-28-10 (200 epochs)	128	0.1	$\rho_{reg} = 0.5$	80.29±0.25	**80.42**	+0.13	0.66
				$\rho_{reg} = 1$		**81.11**	+0.82	4.16
				$\rho_{reg} = 2$		**81.50**	+1.21	6.14
				$\rho_{reg} = 3$		**82.51**	+2.22	11.26
				$\rho_{reg} = 4$		**82.65**	+2.36	11.97
				$\rho_{reg} = 5$		**82.31**	+2.02	10.25
				$\rho_{reg} = 6$		**82.03**	+1.74	8.83
	WRN-28-10 (400 epochs)	128	0.1	$\rho_{reg} = 1$	80.69±0.21	**82.55**	+1.86	9.63
				$\rho_{reg} = 2$		**82.51**	+1.82	9.42
				$\rho_{reg} = 3$		**82.84**	+2.15	11.13
				$\rho_{reg} = 4$		**83.35**	+2.66	13.78
				$\rho_{reg} = 5$		**83.73**	+3.04	15.74
				$\rho_{reg} = 6$		**83.16**	+2.47	12.79
				$\rho_{reg} = 7$		**83.12**	+2.43	12.58
				$\rho_{reg} = 8$		**81.16**	+0.47	2.43

Table 6: **Effectiveness of Efficient EJR.v2**. We report improvement (Δ**Acc.**) and Error Reduction Rate (**ERR**) on CIFAR-10 when trained with the second variant of EJR (**+EJR.v2**), compared to the standard training (**Baseline**). We used Setting 2 in Table 4

Dataset	**Network Architecture**	**Batch Size**	**lr**	**Reg. param.**	**Test Accuracy**		**ΔAcc. (%p)**	**ERR (%)**
					Baseline	**+EJR.v2**		
CIFAR-10	WRN-28-10 (400 epochs)	128	0.1	$\rho_{reg} = 2, \mu_{reg} = 0.001$	$96.10_{\pm 0.05}$	**97.28**	+1.18	30.26
				$\rho_{reg} = 2, \mu_{reg} = 0.003$		**97.32**	+1.23	31.28
				$\rho_{reg} = 2, \mu_{reg} = 0.01$		**97.33**	+1.23	31.54
				$\rho_{reg} = 2, \mu_{reg} = 0.03$		**97.38**	+1.28	32.82
				$\rho_{reg} = 2, \mu_{reg} = 0.1$		**97.17**	+1.07	27.44
				$\rho_{reg} = 1, \mu_{reg} = 0.01$		**97.07**	+0.97	24.87
				$\rho_{reg} = 1, \mu_{reg} = 0.02$		**97.34**	+1.24	31.79
				$\rho_{reg} = 1, \mu_{reg} = 0.06$		**97.33**	+1.23	31.54
				$\rho_{reg} = 1, \mu_{reg} = 0.1$		**97.26**	+1.16	29.74

R. Adversarial Robustness

We also apply EJR to train a model that is robust to adversarial attacks (Szegedy et al., 2013). Adversarially robust training suffers from robust overfitting (Rice et al., 2020) and we expect EJR to help to improve robust generalization. We test the effectiveness of EJR with AT (Madry et al., 2018) and TRADES (Zhang et al., 2019). Table 2 shows that the proposed method can obtain better robust generalization against the adversarial attacks. We report the best result with the regularization parameter tuning. We follow a similar setting to Pang et al. (2020).

We tune the regularization parameter ρ_{reg} as in Table 7 to improve the results and to report the best results in Table 2. However, we did not perform an exhaustive investigation due to limited computational resources, so there is still room for further improvement.

Table 7: **Adversarial training with efficient EJR on CIFAR-10 compared with AT (Madry et al., 2018) and TRADES (Zhang et al., 2019).** We report **Standard Accuracy** and robust accuracy ($\epsilon = 8/255$) against **PGD-20** and AutoAttack (**AA**) (Croce & Hein, 2020).

Model	Method	Epoch	ρ_{reg}	β_{TRADES}	Standard	PGD-20	AA
PreAct ResNet-18	AT Baseline	110	-	-	82.45	52.85	48.76
	AT-EJR	110	0.05	-	82.32	53.58	**49.14**
		110	0.1	-	81.45	53.61	48.77
	TRADES Baseline	110	-	6	82.34	52.83	49.06
		110	-	10	79.20	53.19	49.07
	TRADES-EJR	110	0.1	6	81.56	53.05	**49.56**
		110	0.1	10	80.43	53.44	49.00
		110	0.2	6	81.87	53.19	49.37
		110	0.2	10	80.68	53.37	49.21
WRN-34-10	AT Baseline	120	-	-	86.99	52.20	49.83
	AT-EJR	120	0.05	-	87.16	55.89	52.70
		120	0.1	-	86.85	57.82	**53.73**
	TRADES Baseline	120	-	6	85.23	55.68	52.46
		120	-	10	83.62	57.08	53.29
	TRADES-EJR	120	0.02	6	85.77	56.97	53.62
		120	0.02	10	83.61	57.08	53.73
		120	0.1	6	85.91	56.34	53.18
		120	0.1	10	83.97	57.48	**53.85**