IMPLICIT JACOBIAN REGULARIZATION WEIGHTED WITH IMPURITY OF PROBABILITY OUTPUT

Anonymous authors
Paper under double-blind review

ABSTRACT

Gradient descent (GD) plays a crucial role in the success of deep learning, but it is still not fully understood how GD finds minima that generalize well. In many studies, GD has been understood as a gradient flow in the limit of vanishing learning rate. However, this approach has a fundamental limitation in explaining the oscillatory behavior with iterative catapult in a practical finite learning rate regime. To address this limitation, we rather start with strong empirical evidence of the plateau of the sharpness (the top eigenvalue of the Hessian) of the loss function landscape. With this observation, we investigate the Hessian through simple and much lower-dimensional matrices. In particular, to analyze the sharpness, we instead explore the eigenvalue problem for the low-dimensional matrix which is a rank-one modification of a diagonal matrix. The eigendecomposition provides a simple relation between the eigenvalues of the low-dimensional matrix and the impurity of the probability output. We exploit this connection to derive sharpness-impurity-Jacobian relation and to explain how the sharpness influences the learning dynamics and the generalization performance. In particular, we show that GD has implicit regularization effects on the Jacobian norm weighted with the impurity of the probability output.

1 INTRODUCTION

Deep learning has shown to be powerful for many learning tasks in various areas. There has been a lot of work to understand how the learning algorithm leads to this successful training of deep neural networks. Especially, it is crucial to understand the geometric properties of the loss landscape of neural networks and their interaction with the gradient-based optimization methods, such as Stochastic Gradient Descent (SGD), along the training trajectory. It has been studied both from the 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; Fort et al., 2019; Jiang et al., 2020; Barrett & Dherin, 2021; Smith et al., 2021) point of view.

We aim at investigating the Hessian of the training loss (with respect to model parameter) and its top eigenvalue (also called sharpness). The sharpness characterizes the dynamics of neural network training along the optimization trajectory and is correlated with the generalization capability. For example, the sharpness increases in the beginning, and after reaching a certain value, training dynamics becomes unstable, oscillating along the top eigenvector (Jastrzębski et al., 2019; Cohen et al., 2021). Moreover, the rapid increase in the sharpness of the loss landscape in the early phase significantly impacts the final generalization performance (Achille et al., 2019; Jastrzebski et al., 2020; Lewkowycz et al., 2020; Jastrzebski et al., 2021). However, the Hessian of a deep neural network is very high-dimensional which makes it difficult to analyze its eigensystem. Recently, some researchers studied the Hessian by exploiting tools in randomized numerical linear algebra (Sagun et al., 2017; Papyan, 2018; 2019; Ghorbani et al., 2019; Yao et al., 2020) and decomposition of the Hessian (Papyan, 2018; 2019; Fort & Ganguli, 2019).

In this paper, we present a new decomposition of the Hessian using eigendecomposition of low-dimensional matrices. From the eigensystem of the low-dimensional matrix, we can provide a simple and intuitive explanation on the relation between its eigenvalue and the probability output. This

enables us to explain how the sharpness of the loss landscape influences the learning dynamics and the generalization performance.

We summarize the main contributions of the paper as follows:

- We decompose the Hessian with low dimensional matrices, the *logit Hessian* and the logit-weight Jacobian (defined in Definition 1), and investigate the Hessian by the eigendecomposition of the logit Hessian which is a rank-one modification of a diagonal matrix.
- We provide connections between the top eigenvalue of the logit Hessian and the impurity of the probability output.
- We derive a relation between the sharpness, the top eigenvalue of the logit Hessian and the Jacobian. We call it *sharpness-impurity-Jacobian relation*.
- We explain how the sharpness of the loss landscape influences the learning dynamics and the generalization performance. In particular, we find that gradient-based optimizations have implicit effects on penalizing the Jacobian norm (*Implicit Jacobian Regularization*) in a certain phase of training (*Active Regularization Period*).

2 Related Work

We summarize some works on the Hessian, learning dynamics, and generalization of neural networks. In particular, we point out the issue of approximating SGD by a stochastic differential equation (SDE) because a continuous flow cannot capture the oscillatory behavior of discrete updates with iterative catapult, which plays a key role in limiting the sharpness of the loss landscape.

Decomposition of the Hessian Sagun et al. (2016; 2017) empirically found that the eigenvalue spectrum of the Hessian during training is composed of two parts, the bulk which is concentrated around zero and the outliers which are scattered positively away from zero. They showed the bulk depends on the size of the network, and the outliers depend on the data. In particular, the number of outliers matches the number of classes of the data. Further, Papyan (2019) proposed a three-level hierarchical decomposition of the Hessian matrix according to each class, logit coordinate, and example. However, with different decomposition, we analyze the Hessian from another point of view.

SGD as a SDE In many studies, SGD has been understood as a SDE in the limit of vanishing learning rate (Mandt et al., 2017; Li et al., 2017b;a; Smith & Le, 2018; Chaudhari & Soatto, 2018; Jastrzębski et al., 2017; Zhu et al., 2019; Park et al., 2019). However, some theoretical concerns have been raised for such approximations (Yaida, 2019). Moreover, Barrett & Dherin (2021) argued that the SDE analysis in the limit of vanishing learning rate cannot explain the generalization benefits of finite learning rates and they proposed a modified gradient flow for finite learning rates. However, they still consider a continuous gradient flow and thus it has a fundamental limitation in explaining the oscillatory behavior with iterative catapult in a practical learning rate regime (Smith et al., 2021), which will be detailed in the following paragraph.

Oscillatory catapult and the plateau of the sharpness Xing et al. (2018) investigated the roles of learning rate and batch size in SGD dynamics through interpolating the loss landscape between consecutive model parameters during training. They observed SGD explores the parameter space, bouncing between walls of valley-like regions. The large learning rate maintains a high valley height, and a small batch size induces gradient stochasity. They both help exploration through the parameter space with different roles in the training dynamics. Jastrzębski et al. (2019) empirically investigated the evolution of the sharpness (the top eigenvalue of the Hessian) along the whole training trajectory of SGD. They observed initial growth of the sharpness as loss decreases, reaching a maximum sharpness determined by learning rate and batch size, and then it decreases towards the end of training. Due to the initial increase of the sharpness, the SGD step becomes too 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 a solvable training dynamics. They showed that, in their setup with a large learning rate, the loss initially increases while the sharpness decreases, then it converges to a flat minimum. This mechanism is called the 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 nonmonotonically but decreases globally. This behavior of the optimization at the Edge of Stability can be seen as repeated catapult mechanisms. They explicitly marked the limit of the sharpness with $2/\eta$ (η =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 about gradients and Hessians that they are i.i.d isotropic Gaussian with zero mean with varying variance during training. While they focus on building a random model based on the observation, we rather aim to explain the underlying mechanisms.

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

3 BACKGROUND

In this section, we provide some notations, basic equations and definitions for the following sections. Throughout the paper, we use the denominator layout notation for the vector derivatives, i.e., $\nabla_{\boldsymbol{v}}\boldsymbol{u} = \left(\frac{\partial \boldsymbol{u}_j}{\partial \boldsymbol{v}_i}\right)_{ij} \in \mathbb{R}^{v\times u}$ where $\boldsymbol{u} : \mathbb{R}^v \to \mathbb{R}^u$ and $\boldsymbol{v} \in \mathbb{R}^v$. It is also generalized to the cases of scalar, $u = 1$ or $v = 1$.

We consider a problem of learning a C-class classifier which maps an input $\boldsymbol{x} \in \mathcal{X} \subset \mathbb{R}^d$ to a target label $y \in [C]$ where $[C] = \{1, 2, \cdots, C\}$. To this end, we build a parameterized model $f_{\boldsymbol{\theta}} : \mathcal{X} \to \mathcal{Z} \subset \mathbb{R}^C$ with a model parameter $\boldsymbol{\theta} \in \Theta \subset \mathbb{R}^m$ which 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 the input $\boldsymbol{x}$ and the model parameter $\boldsymbol{\theta}$). Then, the logit vector $\boldsymbol{z}$ is given as input to the softmax function to yield a probability vector $\boldsymbol{p} = \mathrm{softmax}(\boldsymbol{z}) \in \Delta^{C-1}$ where $\Delta^{C-1} = \{\boldsymbol{p} \in [0, 1]^C : \boldsymbol{1}^T\boldsymbol{p} = 1, \boldsymbol{p} \geq \boldsymbol{0}\}$. We want the model to match the most probable class $\boldsymbol{c}_1$ to the true label y, where $\boldsymbol{c}(\boldsymbol{x}) \equiv \arg\mathrm{sort}(\boldsymbol{p})$ 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 $l = -\log p$. We use the notations $\|\cdot\|$ for the Euclidean ℓ_2-norm of a vector or for the Euclidean operator norm of a matrix, $\|\cdot\|_\sigma$ for the spectral norm, $\|\cdot\|_F$ for the Frobenius norm, and $\mathrm{tr}(\cdot)$ for the trace of a matrix.

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

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

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

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

$$\nabla^2_{\boldsymbol{z}}l = \nabla_{\boldsymbol{z}}(\nabla_{\boldsymbol{z}}l) = \nabla_{\boldsymbol{z}}(\boldsymbol{p} - \boldsymbol{e}^y) = \mathrm{diag}(\boldsymbol{p}) - \boldsymbol{p}\boldsymbol{p}^T \in \mathbb{R}^{C\times C} \tag{4}$$

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

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

$$\nabla^2_{\boldsymbol{\theta}}l = \nabla_{\boldsymbol{\theta}}\boldsymbol{z}\nabla^2_{\boldsymbol{z}}l\nabla_{\boldsymbol{\theta}}\boldsymbol{z}^T + \sum\nolimits_{j=1}^{C}\nabla^2_{\boldsymbol{\theta}}\boldsymbol{z}_j\nabla_{\boldsymbol{z}_j}l \approx \nabla_{\boldsymbol{\theta}}\boldsymbol{z}\nabla^2_{\boldsymbol{z}}l\nabla_{\boldsymbol{\theta}}\boldsymbol{z}^T \in \mathbb{R}^{m\times m} \tag{5}$$

using a well-known approximation with the Gauss-Newton matrix (see, for example, Schraudolph (2002)). The approximation in Eq (5) has been justified in Sagun et al. (2017); Fort & Ganguli (2019),

especially for the top eigenvalues and eigenvectors. But this approximation sometimes fails in the later phase of training when the top eigenvalues of the Gauss-Newton matrix is not sufficiently isolated from the bulk near 0 (Papyan, 2018). Thus we mainly focus on the early phase of training.

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 $L \equiv \hat{\mathbb{E}}_{\mathcal{D}}[l] \in \mathbb{R}$ which yields $\nabla L = \hat{\mathbb{E}}_{\mathcal{D}}[\nabla l]$ and $\nabla^2 L = \hat{\mathbb{E}}_{\mathcal{D}}[\nabla^2 l]$ where $\hat{\mathbb{E}}_{\mathcal{D}}$ is the expectation over the empirical measure of the training set $\mathcal{D}$. Following from Eq (4) and Eq (5), we define the Hessian matrix $\boldsymbol{H}$ for the total loss and its Gauss-Newton approximation matrix $\boldsymbol{G}$ with the matrices $\boldsymbol{M}$ and $\boldsymbol{J}$ as follows:

Definition 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 defined as follows:*

$$\boldsymbol{M} \equiv \nabla_{\boldsymbol{z}}^2 l = \mathit{diag}(\boldsymbol{p}) - \boldsymbol{p}\boldsymbol{p}^T \in \mathbb{R}^{C \times C} \tag{6}$$

$$\boldsymbol{J} \equiv \nabla_{\boldsymbol{\theta}} \boldsymbol{z} \in \mathbb{R}^{m \times C} \tag{7}$$

$$\boldsymbol{H} \equiv \hat{\mathbb{E}}_{\mathcal{D}}[\nabla_{\boldsymbol{\theta}}^2 l] \approx \hat{\mathbb{E}}_{\mathcal{D}}[\boldsymbol{J}\boldsymbol{M}\boldsymbol{J}^T] \equiv \boldsymbol{G} \in \mathbb{R}^{m \times m} \tag{8}$$

It is interesting to note that while l is dependent on the true label y, the logit Hessian $\boldsymbol{M} = \nabla_{\boldsymbol{z}}^2 l$ is independent of y, and so are $\boldsymbol{J}$, $\boldsymbol{J}\boldsymbol{M}\boldsymbol{J}^T$, and $\boldsymbol{G}$.

4 Decomposition of the Hessian $\boldsymbol{H}$

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., $\boldsymbol{G} = \hat{\mathbb{E}}_{\mathcal{D}}[\boldsymbol{J}\boldsymbol{M}\boldsymbol{J}^T]$. Now, we focus on the logit Hessian matrix $\boldsymbol{M}$ and its eigendecomposition, estimate the top eigenvalue of $\boldsymbol{M}$ with upper/lower bounds, and explore the evolution of the top eigenvalue during training.

4.1 Eigendecomposition of the logit Hessian

The lower-dimensional matrix $\boldsymbol{M} \in \mathbb{R}^{C \times C}$ is simple and fully characterized by only the probability vector $\boldsymbol{p}$ as $\boldsymbol{M} = \mathrm{diag}(\boldsymbol{p}) - \boldsymbol{p}\boldsymbol{p}^T$ in Eq (6), but it turns out to be important for understanding the much higher-dimensional matrix $\boldsymbol{G} \in \mathbb{R}^{m \times m}$ ($C \ll m$). Since $\boldsymbol{M} = \mathrm{diag}(\boldsymbol{p}) - \boldsymbol{p}\boldsymbol{p}^T$ is a rank-one modification of a simple diagonal matrix $\mathrm{diag}(\boldsymbol{p})$, we can obtain the eigenvalues and the eigenvectors from the theory of the rank-one modification of the eigenproblem (see, for example, Bunch et al. (1978); Golub (1973) and (Golub & Van Loan, 2013, Section 8.4.3)). Then, the logit Hessian $\boldsymbol{M}$ can be eigendecomposed as $\boldsymbol{M} = \boldsymbol{Q}\boldsymbol{\Lambda}\boldsymbol{Q}^T = \sum_{i=1}^{C} \lambda^{(i)} \boldsymbol{q}^{(i)} \boldsymbol{q}^{(i)T}$ where $\lambda^{(i)}$ is the i-th largest eigenvalue of $\boldsymbol{M}$ and $\boldsymbol{q}^{(i)}$ is its corresponding normalized eigenvector. For simplicity, 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 $\boldsymbol{p}_{(1)} \geq \boldsymbol{p}_{(2)} \geq \cdots \geq \boldsymbol{p}_{(C)} \geq 0$, because this ordering is related to the eigenvalues $\{\lambda^{(i)}\}_{i=1}^{C}$ as demonstrated in the following theorem. We defer the proof to Appendix A.

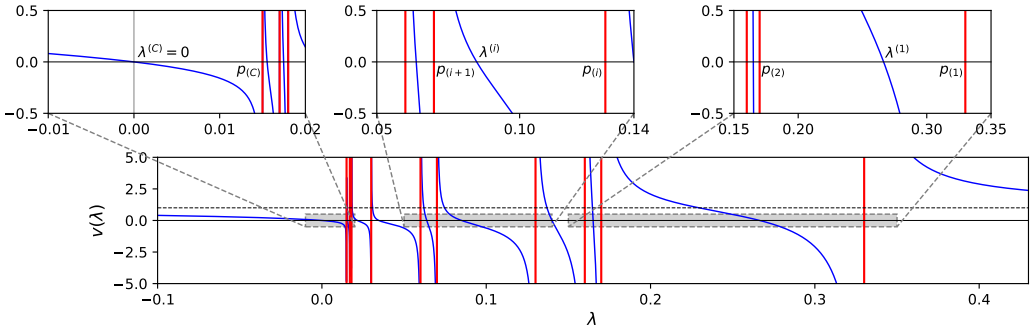

Figure 1: **[Eigenvalues of the logit Hessian] Graph of the secular function $v(\lambda)$ in Eq (9) which has zeros at the eigenvalues $\{\lambda^{(i)}\}_{i=1}^{C}$ of $\boldsymbol{M} = \nabla_{\boldsymbol{z}}^2 l$ (blue curves).** We highlighted the singularities $\lambda = \boldsymbol{p}_{(i)}$ with red vertical lines. It illustrates Theorem 1 (a) and (c).

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

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

$$v(\lambda) = 1 - \sum\nolimits_{i=1}^{C} \frac{\boldsymbol{p}_i^2}{\boldsymbol{p}_i - \lambda} = 0 \tag{9}$$

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

(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 $Gini(q) = 1 - q^2 - (1-q)^2 = 2q(1-q)$ is the Gini impurity for the binary case $(q, 1-q)$.

Figure 1 illustrates the secular function $v(\lambda)$ defined as Eq (9) in Theorem 1 (a). The function $v(\lambda)$ has singularities at the probability values $\{\boldsymbol{p}_{(i)}\}_{i=1}^{C}$ and zeros at the eigenvalues $\{\lambda^{(i)}\}_{i=1}^{C}$, satisfying Theorem 1 (c). Moreover, Theorem 1 (c) and (d) provide the upper/lower bounds on the eigenvalues of $\boldsymbol{M}$. Especially, 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 it *impurity*.

4.2 Evolution of Impurity

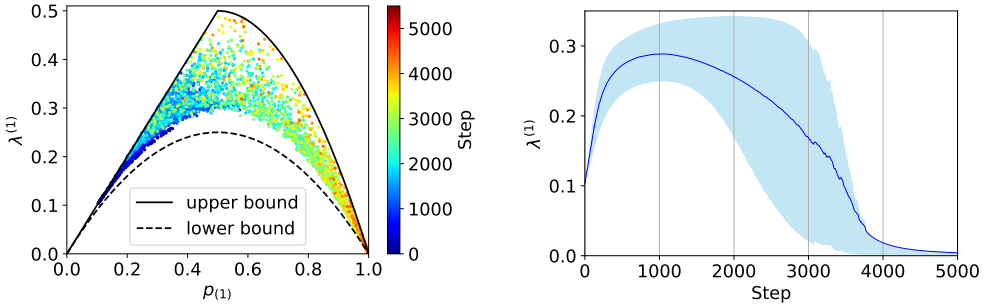

Figure 2: **[Evolution of Impurity] The impurity $\lambda^{(1)}$ increases and then decreases as $\boldsymbol{p}_{(1)}$ increases during training.** Left: The impurity of a typical example is plotted against $\boldsymbol{p}_{(1)}$. We together plot the upper bound $\min\{\boldsymbol{p}_{(1)}, \text{Gini}(\boldsymbol{p}_{(1)})\}$ and the lower bound $\frac{1}{2}\text{Gini}(\boldsymbol{p}_{(1)})$ from Theorem 1 (c) and (d). Right: The impurity is plotted against the training step. Blue curve indicates its mean value and sky-blue area shows the 25-75% quantile range of the impurity for the training data.

We explore the top eigenvalue $\lambda^{(1)}$ of $\boldsymbol{M}$ (also referred to as impurity) during training. Figure 2 demonstrates the evolution of the impurity during training, which increases in the beginning and then decreases in the later phase. We trained a model to zero training loss, and thus, for most 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 1 (c) and (d)), increases at the initial phase of the training because it is 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 almost 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 $\boldsymbol{p}_{(1)}$. We again emphasize that the logit Hessian $\boldsymbol{M}$ is independent of the label y but dependent only on the probability output $\boldsymbol{p}$.

5 IMPLICIT JACOBIAN REGULARIZATION

In this section, from the results of the previous sections, we aim to derive a relation between the sharpness, the impurity and the Jacobian, and answer how the sharpness of the loss landscape influences the learning dynamics and the generalization performance. Detailed experimental settings for each Figure are described in Appendix C.

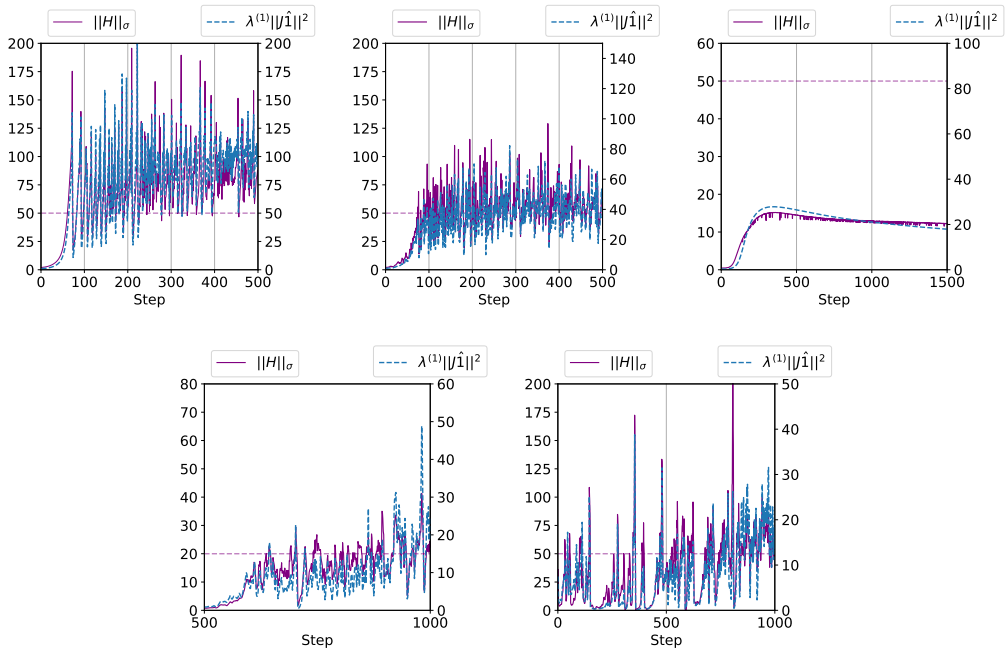

Figure 3: **[Sharpness-Impurity-Jacobian Relation] The sharpness $\|\boldsymbol{H}\|_\sigma$ and $\lambda^{(1)}\|\boldsymbol{J}\hat{\mathbf{1}}\|^2$ behave similarly up to constant, especially in the early phase of training, where $\hat{\mathbf{1}} = \mathbf{1}/\sqrt{C} \in \mathbb{R}^C$.** We empirically validate the sharpness-impurity-Jacobian relation $\|\boldsymbol{H}\|_\sigma \sim \lambda^{(1)}\|\boldsymbol{J}\hat{\mathbf{1}}\|^2$ in various settings (upper left: GD, upper middle: SGD, upper right: MNIST, lower left: CIFAR-100 and lower right: ResNet model). Here, $\lambda^{(1)}$ and $\boldsymbol{J}$ are averaged over the training set $\mathcal{D}$. We highlighted $\|\boldsymbol{H}\|_\sigma = 2/\eta$ with the dashed horizontal line. For MNIST, it is easy to train a model, and thus $\lambda^{(1)}$ decreases before the sharpness reaches the limit. See Appendix E for detailed settings and more figures.

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 $\boldsymbol{J}$. Since 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 follows:

$$\|\boldsymbol{H}\|_\sigma \approx \|\boldsymbol{G}\|_\sigma = \|\hat{\mathbb{E}}_{\mathcal{D}}[\boldsymbol{J}\boldsymbol{M}\boldsymbol{J}^T]\|_\sigma \tag{10}$$

As shown in Eq (10), the impurity $\|\boldsymbol{M}\|_\sigma (= \lambda^{(1)})$ and the square of the norm of the Jacobian $\boldsymbol{J}$ are highly correlated with the sharpness $\|\boldsymbol{H}\|_\sigma$. In particular, $\|\boldsymbol{H}\|_\sigma$ and $\lambda^{(1)}\|\boldsymbol{J}\hat{\mathbf{1}}\|^2$ show surprisingly similar behavior up to constant as demonstrated in Figure 3 where $\hat{\mathbf{1}} = \mathbf{1}/\sqrt{C} \in \mathbb{R}^C$. Here, $\lambda^{(1)}$ and $\boldsymbol{J}$ are averaged over the training set $\mathcal{D}$. We briefly state this sharpness-impurity-Jacobian relation between $\|\boldsymbol{H}\|_\sigma$, $\lambda^{(1)}$, and $\boldsymbol{J}$ as follows:

$$\|\boldsymbol{H}\|_\sigma \sim \lambda^{(1)}\|\boldsymbol{J}\hat{\mathbf{1}}\|^2 \tag{11}$$

The impurity $\lambda^{(1)}$ in the right term acts as an adaptive regularization weight for the regularization on the Jacobian norm as detailed in the following section.

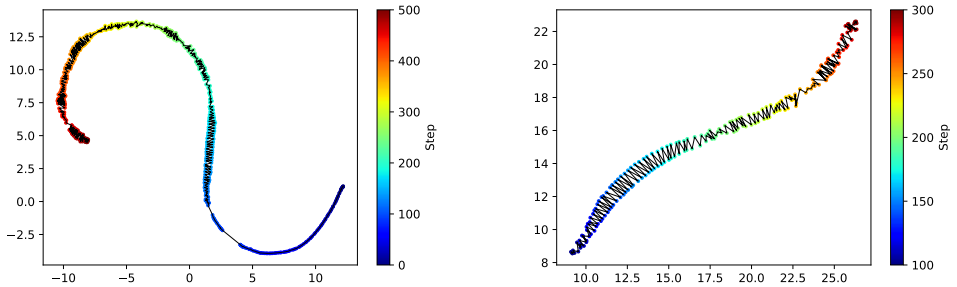

Figure 4: **[Oscillatory catapult] UMAP visualization of the oscillatory catapult in the optimization trajectory $\{\boldsymbol{\theta}^{(t)}\}$ (from blue to red) of full-batch GD (McInnes et al., 2018)**. Left: UMAP of the model parameters trained on CIFAR-10 for the first 500 steps. Right: Zoom-in for the oscillatory steps [100, 300]. After few steps ($\sim$100), it shows the oscillatory behavior with 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 the generalization performance.

Growing Jacobian and sharpness in the early phase of training The weight norm $\|\boldsymbol{\theta}\|$ increases in order to increase the logit norm $\|\boldsymbol{z}\|$ and to minimize the cross-entropy loss during training (Soudry et al., 2018) (see Appendix D for details). This also leads to the increase in the layerwise weight norms and the Jacobian norm. Thus, the increase of the sharpness in the early phase of training can be mainly attributed to the increase of the Jacobian norm (and partially to the top eigenvalue $\lambda^{(1)}$ of $\boldsymbol{M}$). However, as will be discussed in the following paragraphs, the Jacobian norm does not continuously increase throughout training.

Oscillatory catapult and the plateau of the 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 gradient-based optimization. After the sharpness reaching this threshold, the iterate starts to bounce off from one side of the valley to the other, and repeats this (Xing et al., 2018; Jastrzębski et al., 2019). Figure 4 shows this oscillatory behavior with iterative catapult after the sharpness reaching the threshold, using UMAP (McInnes et al., 2018). Due to the catapult, the iterate cannot stay in a sharper area and is catapulted to another area. This mechanism is called the catapult mechanism (Lewkowycz et al., 2020). Figure 3 shows fine-grained patterns in the early phase of training that the sharpness oscillates around the threshold by the two conflicting effects: the Jacobian norm tends to increase the sharpness and the catapult mechanism 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 15 (Top) in Appendix I). For GD with a quadratic loss, the threshold of the sharpness can be easily obtained to be $2/\eta$ ($\eta =$ learing rate), and the threshold value is similar for the cross-entropy loss as shown in Figure 3 (Cohen et al., 2021). The plateau of the sharpness is also observed in the case of SGD, but with a different threshold value depending on the batch size (the smaller the batch size, the lower the plateau) as shown in Figure 6 (Right). 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 by a continuous gradient flow.

Implicit Jacobian Regularization Due to this catapult effect, the Jacobian norm cannot continue to increase. In other words, the gradient-based optimization implicitly penalizes the Jacobian norm since $\|\boldsymbol{J}\hat{\mathbf{1}}\|^2 \sim \|\boldsymbol{H}\|_\sigma/\lambda^{(1)} \lessapprox \frac{2}{\eta\lambda^{(1)}}$. This implicit regularization effect begins after the sharpness reaches the threshold. And the regularization effect diminishes as $\lambda^{(1)}$ decreases (the upper limit $\frac{2}{\eta\lambda^{(1)}}$ is loosened) with increasing $\boldsymbol{p}_{(1)} \geq 0.5$ in the later phase (see Figure 2, 5 and 6) because $\lambda^{(1)}$ acts as a regularization coefficient. This explains why the behavior of the sharpness in the early phase of training seems to impact the final generalization. Moreover, as $\lambda^{(1)}$ decreases to a very small value,

we can observe the decrease in the sharpness. However, the generalization gap keeps increasing, which provides another counter-example that flat minima generalizes poorly (Dinh et al., 2017).

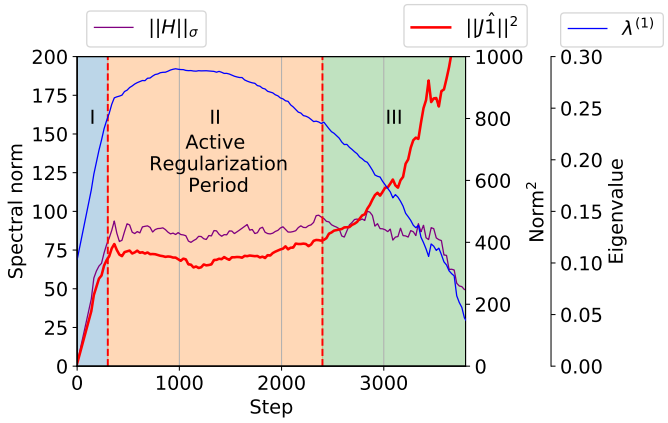

Figure 5: **[Implicit Jacobian Regularization] The step-like structure (red curve) of the three phases of the implicit Jacobian regularization.** It shows the implicit regularization effects on the Jacobian norm in the Active Regularization Period (II).

Figure 5 shows that the gradient-based optimization has implicit regularization effects on the Jacobian norm in a certain period. To be specific, there is a step-like structure of the three phases that the Jacobian norm is (I) initially increasing before the sharpness reaches near the threshold, (II) actively regularized with the plateau of the Jacobian norm, and (III) again increasing as the regularization effect diminishes (as the regularization weight $\lambda^{(1)}$ decreases). We call the second phase *Active Regularization Period*.

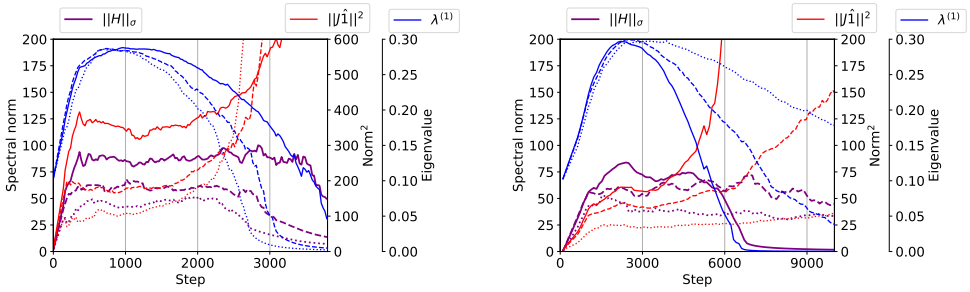

Figure 6: **[Implicit Jacobian Regularization and Training Hyperparameters] The implicit regularization effects on the Jacobian norm vary depending on the hyperparameters used in the training.** The plateau of the sharpness and the Jacobian norm (during the Active Regularization Period), and the n-shaped evolution of the impurity are clearly observed as expected. We used (Left) GD with different learning rates $\eta = 0.04/0.06/0.08$ and (Right) SGD with fixed $\eta = 0.04$ and different batch sizes 512/128/32 (solid/dashed/dotted lines) on CIFAR-10. Training with a large learning rate and a small batch size (dotted line) penalizes the Jacobian norm more strongly with lower limits of the sharpness. Curves are smoothed for visual clarity.

The evolution of the sharpness of the loss landscape is highly affected by the hyperparameters used in training such as learning rate and batch size (Jastrzębski et al., 2019; Lewkowycz et al., 2020; Cohen et al., 2021). As expected, GD with a large learning rate η limits the sharpness with a lower value of $2/\eta$. The large learning rate encourages the stronger implicit regularization on the Jacobian norm. Figure 6 shows, comparing the three red lines (solid/dashed/dotted) with different learning rates (0.04/0.06/0.08) and batch sizes (512/128/32), training with a larger learning rate and a smaller batch size limits the Jacobian norm with a smaller value in the Active Regularization Period. This could explain why training with a large learning rate and a small batch size yields better generalization.

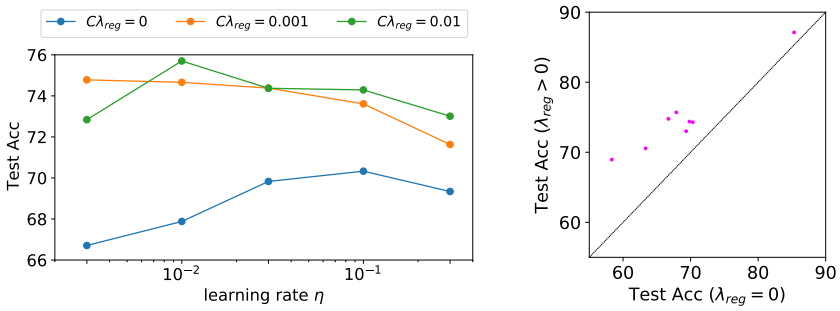

Figure 7: **[Explicit Jacobian Regularization] The explicit Jacobian regularization enhances the test accuracy.** Left: We plot the test accuracy for different learning rates η and regularization coefficients λ_{reg}. The models are trained with batch size of $|\mathcal{B}| = 128$ on CIFAR-10. Right: We compare the test accuracy on CIFAR-10 trained with and without the explicit regularization under different settings.

Explicit Jacobian Regularization To further investigate the effectiveness of the implicit Jacobian regularization, we explicitly regularize the Frobenius norm $\|\boldsymbol{J}\|_F^2$ with the regularization coefficient λ_{reg}. We expect improvements in the generalization performance when introducing the explicit Jacobian regularization. This would support the effectiveness of the implicit Jacobian regularization that it efficiently controls the capacity of the model and limits the update in the overfitting direction of $\boldsymbol{J}\boldsymbol{1} = \nabla_{\boldsymbol{\theta}}(\boldsymbol{1}^T\boldsymbol{z})$ in which it reduces the training loss but not optimizing the margin. To efficiently compute the regularization term, we use an unbiased estimator $\|\boldsymbol{J}\|_F^2 = C\mathbb{E}_{\boldsymbol{u}\sim U(\mathbb{S}^{C-1})}[\|\boldsymbol{J}\boldsymbol{u}\|^2] = C\mathbb{E}_{\boldsymbol{u}\sim U(\mathbb{S}^{C-1})}[\|\nabla_{\boldsymbol{\theta}}(\boldsymbol{u}^T\boldsymbol{z})\|^2]$ where $\boldsymbol{u}$ is randomly drawn from the unit sphere $\mathbb{S}^{C-1}$. Since 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). Figure 7 shows clear improvements in the test accuracy when introducing the explicit Jacobian regularization.

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

$$\text{tr}(\boldsymbol{G}) = \hat{\mathbb{E}}_{\mathcal{D}}[\text{tr}(\boldsymbol{J}\boldsymbol{M}\boldsymbol{J}^T)] = \hat{\mathbb{E}}_{\mathcal{D}}[\sum_{i=1}^{C-1} \lambda^{(i)}\|\boldsymbol{J}\boldsymbol{q}^{(i)}\|^2] \approx \hat{\mathbb{E}}_{\mathcal{D}}[\text{tr}(\boldsymbol{M})\|\boldsymbol{J}\|_F^2]/C \tag{12}$$

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 B for the empirical evidence). Here, the trace of the logit Hessian $\boldsymbol{M}$ can be equivalently written as a C-class Gini impurity: $\text{tr}(\boldsymbol{M}) = \sum_{i=1}^{C}\boldsymbol{p}_i(1-\boldsymbol{p}_i) = 1 - \sum_{i=1}^{C}\boldsymbol{p}_i^2 \equiv \text{Gini}^C(\boldsymbol{p})$ which is $\frac{C-1}{C}$ for the initial uniform distribution and 0 for a one-hot probability. Thus, penalizing $\text{tr}(\boldsymbol{G})$ induces the effects of penalizing $\|\boldsymbol{J}\|_F$, especially in the early phase of training with large $\text{Gini}^C(\boldsymbol{p})$. Thus, as Jastrzebski et al. (2021) argued that Fisher Penalty on the trace of the FIM improves the generalization performance by limiting the memorization, the Jacobian regularization may have similar effects. Moreover, since $\nabla_{\boldsymbol{\theta}}l(\boldsymbol{z},\hat{y}) = \nabla_{\boldsymbol{\theta}}\boldsymbol{z}\nabla_{\boldsymbol{z}}l(\boldsymbol{z},\hat{y}) = \boldsymbol{J}(\boldsymbol{p}-\boldsymbol{e}^{\hat{y}})$, the trace of the FIM they approximately used 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) is $\|\hat{\mathbb{E}}_{(\boldsymbol{x},y)\sim\mathcal{B}}[\boldsymbol{J}(\boldsymbol{p}-\boldsymbol{e}^{y})]\|$ and the explicit Jacobian regularizer is $\|\boldsymbol{J}\|_F^2 = C\mathbb{E}_{\boldsymbol{u}\sim U(\mathcal{S}^{C-1})}\|\hat{\mathbb{E}}_{(\boldsymbol{x},y)\sim\mathcal{B}}[\boldsymbol{J}\boldsymbol{u}]\|^2$. In each case, we emphasize that the Jacobian $\boldsymbol{J}$ plays an important role in the generalization performance.

6 Conclusion

We have investigated the eigensystem of the Hessian through a new decomposition using eigen-decomposition of the low-dimensional logit Hessian. By doing so, we could provide a simple and intuitive explanation on the relation between the gradient-based optimization, the learning dynamics and the generalization performance of neural networks. We hope this research could help answer other intriguing questions on the learning dynamics and generalization.

ETHICS STATEMENT

No ethics statement is needed for this study.

REPRODUCIBILITY STATEMENT

We provide code to reproduce our experiments. We refer the readers to the supplementary material (README.md).

REFERENCES

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

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

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

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

Pratik Chaudhari, Anna Choromanska, Stefano Soatto, Yann LeCun, Carlo Baldassi, Christian Borgs, Jennifer Chayes, Levent Sagun, and Riccardo Zecchina. Entropy-sgd: Biasing gradient descent into wide valleys. *Journal of Statistical Mechanics: Theory and Experiment*, 2019(12):124018, 2019.

Jeremy Cohen, Simran Kaur, Yuanzhi Li, J Zico Kolter, and Ameet Talwalkar. 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.

Laurent Dinh, Razvan Pascanu, Samy Bengio, and Yoshua Bengio. Sharp minima can generalize for deep nets. In *International Conference on Machine Learning*, pp. 1019–1028. PMLR, 2017.

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

Stanislav Fort, Paweł Krzysztof Nowak, Stanislaw Jastrzebski, and Srini Narayanan. Stiffness: A new perspective on generalization in neural networks. *arXiv preprint arXiv:1901.09491*, 2019.

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

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

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

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

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

Guy Gur-Ari, Daniel A Roberts, and Ethan Dyer. Gradient descent happens in a tiny subspace. *arXiv preprint arXiv:1812.04754*, 2018.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

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

Judy Hoffman, Daniel A Roberts, and Sho Yaida. Robust learning with jacobian regularization. *arXiv preprint arXiv:1908.02729*, 2019.

Stanisław Jastrzębski, Zachary Kenton, Devansh Arpit, Nicolas Ballas, Asja Fischer, Yoshua Bengio, and Amos Storkey. Three factors influencing minima in sgd. *arXiv preprint arXiv:1711.04623*, 2017.

Stanislaw Jastrzebski, Maciej Szymczak, Stanislav Fort, Devansh Arpit, Jacek Tabor, Kyunghyun Cho*, and Krzysztof Geras*. 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.

Stanislaw Jastrzebski, Devansh Arpit, Oliver Astrand, Giancarlo B Kerg, Huan Wang, Caiming Xiong, Richard Socher, Kyunghyun Cho, and Krzysztof J Geras. Catastrophic fisher explosion: Early phase fisher matrix impacts generalization. In Marina Meila and Tong Zhang (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.

Stanisław Jastrzębski, Zachary Kenton, Nicolas Ballas, Asja Fischer, Yoshua Bengio, and Amost Storkey. 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.

Yiding Jiang, Behnam Neyshabur, Hossein Mobahi, Dilip Krishnan, and Samy Bengio. Fantastic generalization measures and where to find them. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=SJgIPJBFvH.

Nitish Shirish Keskar, Dheevatsa Mudigere, Jorge Nocedal, Mikhail Smelyanskiy, and Ping Tak Peter Tang. 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.

Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

Aitor Lewkowycz, Yasaman Bahri, Ethan Dyer, Jascha Sohl-Dickstein, and Guy Gur-Ari. The large learning rate phase of deep learning: the catapult mechanism. *arXiv preprint arXiv:2003.02218*, 2020.

Chris Junchi Li, Lei Li, Junyang Qian, and Jian-Guo Liu. Batch size matters: A diffusion approximation framework on nonconvex stochastic gradient descent. *stat*, 1050:22, 2017a.

Qianxiao Li, Cheng Tai, and E Weinan. Stochastic modified equations and adaptive stochastic gradient algorithms. In *International Conference on Machine Learning*, pp. 2101–2110. PMLR, 2017b.

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

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

Leland McInnes, John Healy, Nathaniel Saul, and Lukas Großberger. 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.

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

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

Vardan Papyan. Measurements of three-level hierarchical structure in the outliers in the spectrum of deepnet hessians. In Kamalika Chaudhuri and Ruslan Salakhutdinov (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.

Daniel Park, Jascha Sohl-Dickstein, Quoc Le, and Samuel Smith. 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.

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

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

Levent Sagun, Utku Evci, V Ugur Guney, Yann Dauphin, and Leon Bottou. Empirical analysis of the hessian of over-parametrized neural networks. *arXiv preprint arXiv:1706.04454*, 2017.

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

Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In Yoshua Bengio and Yann LeCun (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.

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

Samuel L Smith, Benoit Dherin, David Barrett, and Soham De. 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.

Daniel Soudry, Elad Hoffer, Mor Shpigel Nacson, Suriya Gunasekar, and Nathan Srebro. The implicit bias of gradient descent on separable data. *The Journal of Machine Learning Research*, 19(1): 2822–2878, 2018.

Huan Wang, Nitish Shirish Keskar, Caiming Xiong, and Richard Socher. Identifying generalization properties in neural networks. *arXiv preprint arXiv:1809.07402*, 2018.

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

Chen Xing, Devansh Arpit, Christos Tsirigotis, and Yoshua Bengio. A walk with sgd. *arXiv preprint arXiv:1802.08770*, 2018.

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

Zhewei Yao, Amir Gholami, Kurt Keutzer, and Michael W Mahoney. Pyhessian: Neural networks through the lens of the hessian. In *2020 IEEE International Conference on Big Data (Big Data)*, pp. 581–590. IEEE, 2020.

Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning (still) requires rethinking generalization. *Communications of the ACM*, 64(3):107–115, 2021.

Zhanxing Zhu, Jingfeng Wu, Bing Yu, Lei Wu, and Jinwen Ma. 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 PROOF OF THEOREM 1

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

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

$$v(\lambda) = 1 - \sum\nolimits_{i=1}^{C} \frac{\boldsymbol{p}_i^2}{\boldsymbol{p}_i - \lambda} = 0 \tag{13}$$

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

(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 $Gini(q) = 1 - q^2 - (1-q)^2 = 2q(1-q)$ is the Gini impurity for the binary case $(q, 1-q)$.

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

$$\begin{aligned}
\phi_{\boldsymbol{M}}(\lambda) &= \det(\text{diag}(\boldsymbol{p}) - \boldsymbol{p}\boldsymbol{p}^T - \lambda\boldsymbol{I}) && (14)\\
&= \det(\text{diag}(\boldsymbol{p}) - \lambda\boldsymbol{I}) \det(\boldsymbol{I} - (\text{diag}(\boldsymbol{p}) - \lambda\boldsymbol{I})^{-1}\boldsymbol{p}\boldsymbol{p}^T) && (15)\\
&= \prod_{i=1}^{C}(\boldsymbol{p}_i - \lambda)\left(1 - \sum_{j=1}^{C}\frac{\boldsymbol{p}_j^2}{\boldsymbol{p}_j - \lambda}\right) && (16)
\end{aligned}$$

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 $\boldsymbol{A} = \text{diag}(\boldsymbol{p}) - \lambda\boldsymbol{I}$, 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:

$$v(\lambda) = 1 - \sum_{i=1}^{C}\frac{\boldsymbol{p}_i^2}{\boldsymbol{p}_i - \lambda} = 0 \tag{17}$$

which implies (a). Note that this result also implies (c) as shown in Figure 1.

Next, to prove (b), put $\boldsymbol{A} \equiv \text{diag}(\boldsymbol{p}) - \lambda\boldsymbol{I}$ 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

$$(\boldsymbol{A} - \boldsymbol{p}\boldsymbol{p}^T)\boldsymbol{q} = (\boldsymbol{A} - \boldsymbol{p}\boldsymbol{p}^T)\boldsymbol{A}^{-1}\boldsymbol{p} = \boldsymbol{p} - \boldsymbol{p}\boldsymbol{p}^T\boldsymbol{A}^{-1}\boldsymbol{p} \tag{18}$$

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

$$\lambda^{(i)}(\boldsymbol{C}) \leq \lambda^{(j)}(\boldsymbol{A}) + \lambda^{(k)}(\boldsymbol{B}) \text{ if } k + j - i = 1 \tag{19}$$

$$\lambda^{(i)}(\boldsymbol{C}) \geq \lambda^{(j)}(\boldsymbol{A}) + \lambda^{(k)}(\boldsymbol{B}) \text{ if } k + j - i = C \tag{20}$$

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

$$\boldsymbol{p}_{(i+1)} \leq \lambda^{(i)}(\boldsymbol{M}) \leq \boldsymbol{p}_{(i)} \text{ for } 1 \leq i \leq C-1 \tag{21}$$

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

$$\lambda^{(1)} \in \bigcup_i \overline{\mathbb{B}}(\boldsymbol{M}_{ii}, \sum_{j \neq i} |\boldsymbol{M}_{ij}|) = \overline{\mathbb{B}}(\boldsymbol{p}_{(1)}(1-\boldsymbol{p}_{(1)}), \boldsymbol{p}_{(1)}(1-\boldsymbol{p}_{(1)})) = [0, 2\boldsymbol{p}_{(1)}(1-\boldsymbol{p}_{(1)})] \quad (22)$$

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 $\boldsymbol{e}^{(1)T} \boldsymbol{M} \boldsymbol{e}^{(1)} = \boldsymbol{M}_{(1)(1)} = \boldsymbol{p}_{(1)}(1-\boldsymbol{p}_{(1)}) \leq \lambda^{(1)}$.

□

B 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 $\boldsymbol{q}^{(i)}$ $(1 \leq i \leq C-1)$ of $\boldsymbol{M}$. The eigenvector

$$\boldsymbol{q}^{(i)} = \alpha \left(\frac{\boldsymbol{p}_j}{\boldsymbol{p}_j - \lambda^{(i)}} \right)_j \tag{23}$$

can be obtained from Theorem 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 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.

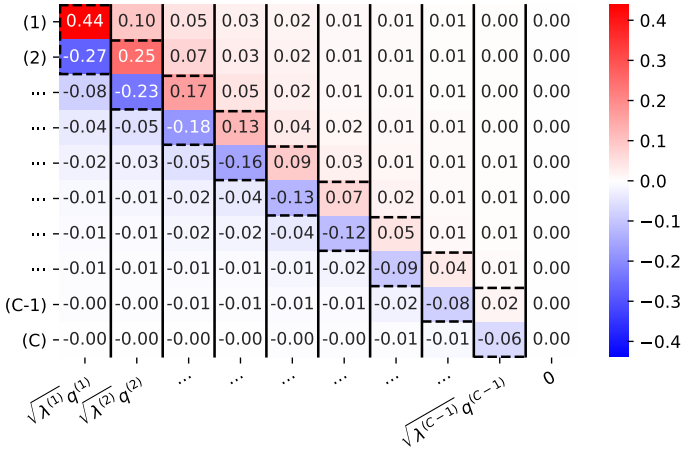

Figure 8: Heatmap of the matrix $\boldsymbol{Q}\boldsymbol{\Lambda}^{1/2} = [\sqrt{\lambda^{(1)}}\boldsymbol{q}^{(1)}; \cdots; \sqrt{\lambda^{(C-1)}}\boldsymbol{q}^{(C-1)}; \boldsymbol{0}]$ averaged over the training set $\mathcal{D}$ where $\boldsymbol{M} = \boldsymbol{Q}\boldsymbol{\Lambda}\boldsymbol{Q}^T$. 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 $0 \leq i \leq C-1$ (see Theorem 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 $1 \leq i \leq C-1$. 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

$$\boldsymbol{J}\boldsymbol{q}^{(i)} = (\boldsymbol{J}\boldsymbol{q}^{(i)})|_{\boldsymbol{\theta}=\boldsymbol{\theta}^{(t)}} = \nabla_{\boldsymbol{\theta}} \left(\boldsymbol{q}^{(i)} (\boldsymbol{\theta}^{(t)})^T \boldsymbol{z}(\boldsymbol{\theta}) \right) |_{\boldsymbol{\theta}=\boldsymbol{\theta}^{(t)}} \in \mathbb{R}^m \tag{24}$$

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.

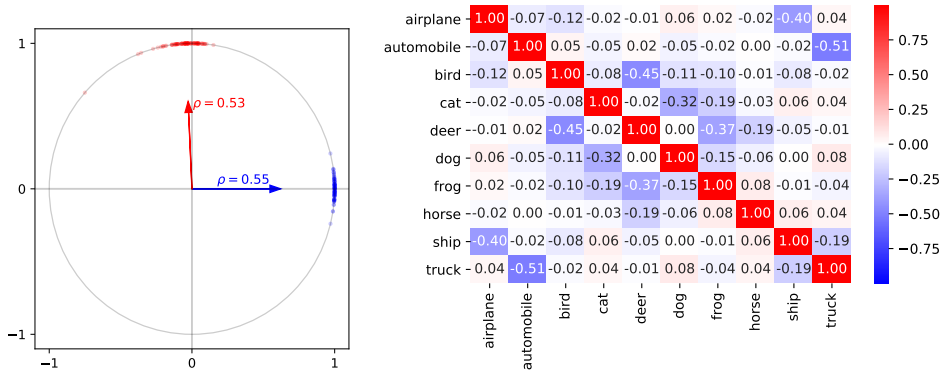

Figure 9: **There are C clusters of $\boldsymbol{m} = \sqrt{\lambda^{(1)}}\boldsymbol{J}\boldsymbol{q}^{(1)}$ according to the most likely class (not the true class). Left: (Within-class similarity)** Directional data of $\boldsymbol{m}$ from $\mathcal{D}_i$ and $\mathcal{D}_j$ for the two classes, dog (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 H for details). Then, we follow a similar approach from (Papyan, 2019) and provide the following equation:

$$\hat{\mathbb{E}}_{\mathcal{D}}[\boldsymbol{m}\boldsymbol{m}^T] = \sum\nolimits_{i=1}^{C} \gamma_i \hat{\mathbb{E}}_{\mathcal{D}_i}[\boldsymbol{m}\boldsymbol{m}^T] = \sum\nolimits_{i=1}^{C} \gamma_i (\overline{\boldsymbol{m}}^i \overline{\boldsymbol{m}}^{iT} + \hat{\mathbb{E}}_{\mathcal{D}_i}[(\boldsymbol{m} - \overline{\boldsymbol{m}}^i)(\boldsymbol{m} - \overline{\boldsymbol{m}}^i)^T]) \quad (25)$$

where $\gamma_i = |\mathcal{D}_i|/|\mathcal{D}|$ and $\overline{\boldsymbol{m}}^i = \hat{\mathbb{E}}_{\mathcal{D}_i}[\boldsymbol{m}]$. Here, the covariance term $\hat{\mathbb{E}}_{\mathcal{D}_i}[(\boldsymbol{m} - \overline{\boldsymbol{m}}^i)(\boldsymbol{m} - \overline{\boldsymbol{m}}^i)^T]$ 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.

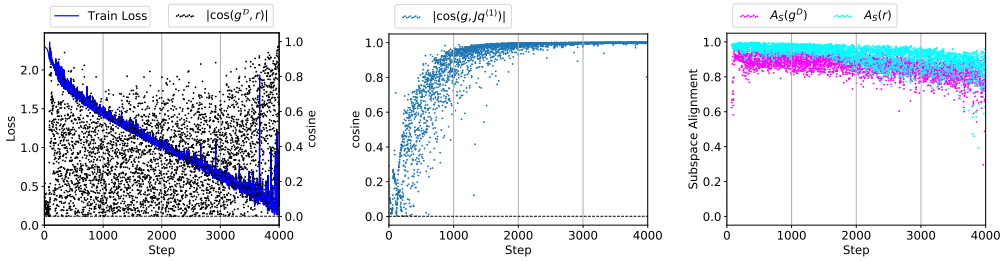

Figure 10: **Left: Total gradient $g^{\mathcal{D}}$ is aligned with the top eigenvector r of the Hessian H at each step during training Jastrzębski et al. (2019); Gur-Ari et al. (2018).** They have large cosine similarities considering that they are very high-dimensional. We highlighted the cosine value 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 2 (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 $\{\overline{m}^i\}_{i=1}^C$.** The subspace alignment measure $A_{\mathcal{S}}$ is defined in Eq (26).

Next, we move on to the subspace $\mathcal{S} \equiv \text{span}(\{\overline{m}^i\}_{i=1}^C)$ spanned by $\{\overline{m}^i\}_{i=1}^C$. As each $m = \sqrt{\lambda^{(1)}} Jq^{(1)}$ is highly aligned with $-g$, it is reasonably expected that the total gradient $g^{\mathcal{D}}$ lies in the subspace $\mathcal{S}$. To measure how much the vector $v \neq 0$ is aligned with the subspace $\mathcal{S}$, we define the cosine similarity between the vector v and its projection $P_{\mathcal{S}}(v)$ onto the subspace $\mathcal{S}$ as follows:

$$A_{\mathcal{S}}(v) \equiv \cos(v, P_{\mathcal{S}}(v)) = \|P_{\mathcal{S}}(v)\| / \|v\| \tag{26}$$

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

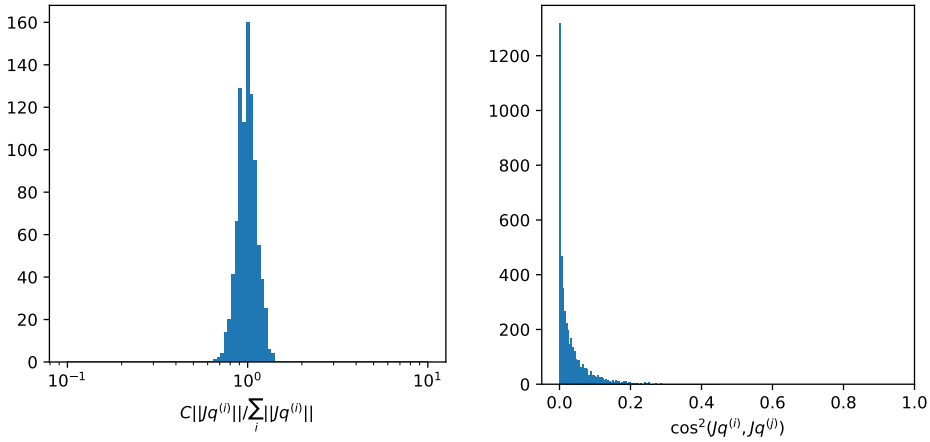

Figure 11: **Histograms of $\frac{\|Jq^{(i)}\|}{\sum_j \|Jq^{(j)}\|/C}$ and $\cos^2(Jq^{(i)}, Jq^{(j)})$.**

C EXPERIMENTAL SETTINGS

C.1 DATA

We use the training set (50k) of CIFAR-10 (Krizhevsky et al. (2009), `https://www.cs.toronto.edu/~kriz/cifar.html`) and its first 5k subset as in Cohen et al. (2021). We also use the training set (60k) of MNIST. They have $C = 10$ number of classes. We also conduct some experiments on the CIFAR-100 dataset with the number of classes $C = 100$. See Figure 20 and 26. We 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 (the only exception is one of the models in Figure 7 (Right) which has the test accuracy of 85.32%)

C.2 NETWORK ARCHITECTURES

We use the following models: VGG-11 (VGG) Simonyan & Zisserman (2015) without batch-normalization, ResNet-20 (ResNet) He et al. (2016) wihtout batch-normalization, a 6-layer CNN (6CNN), two 3-layer fully-connected networks (3FCN-CIFAR and 3FCN-MNIST) with the number of model parameters, $m = 9750922, 268346, 511926, 656810, 199210$, respectively ($m = 9797092$ for VGG on CIFAR-100). In Figure 7, we use another model with $m = 361706$ (SimpleCNN used in Jastrzebski et al. (2021)).

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)).

C.3 HYPERPARAMETERS

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

$$\boldsymbol{\theta}^{(t+1)} = \boldsymbol{\theta}^{(t)} - \eta \boldsymbol{g}^{\mathcal{B}^{(t)}}(\boldsymbol{\theta}^{(t)}) \tag{27}$$

where $\boldsymbol{\theta}^{(t)}$ is the model parameter, $\mathcal{B}^{(t)} \subset \mathcal{D}$ is the training batch at t-th step, $\boldsymbol{g} = \nabla_{\boldsymbol{\theta}} l$, and $\boldsymbol{g}^{\mathcal{B}} = \hat{\mathbb{E}}_{\mathcal{B}}[\boldsymbol{g}]$. We use the simplest form of (S)GD as described in Eq (27) without momentum, weight decay, and learning rate decay (the only exception is one of the models in Figure 7 (Right) which has the test accuracy of 85.32% and it used the momentum of 0.9). We use the learning rates in the rage of $\eta \in [0.001, 0.3]$. For the SGD training, we use the batch sizes $|\mathcal{B}^{(t)}| \in \{32, 64, 128, 512\}$.

C.4 Detailed settings for each Figure

For Figure 2, 3, 9, and 10, we ran the experiments on the CIFAR-10 dataset using 6CNN and trained the model using GD with the learning rate $\eta = 0.04$. See Appendix G for the setting used in Figure 6. We also plot variants of the figures in the main paper for other settings (training steps, data, network architectures, and hyperparameters) in Appendix H-J.

C.5 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 $\mathcal{D}$.

C.6 Power iteration algorithm

Even though we have the secular function $v(\lambda)$ in Eq (9) 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 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 $i \geq n_{\text{max}}$
return $\boldsymbol{v}^T \boldsymbol{M} \boldsymbol{u}$

D 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 6). While the Jacobian norm tends to increase, it stops increasing at a certain point, stays for a few steps, and increases again. We ran the experiments on the CIFAR-10 dataset and 6CNN using GD with learning rate $\eta = 0.04$.

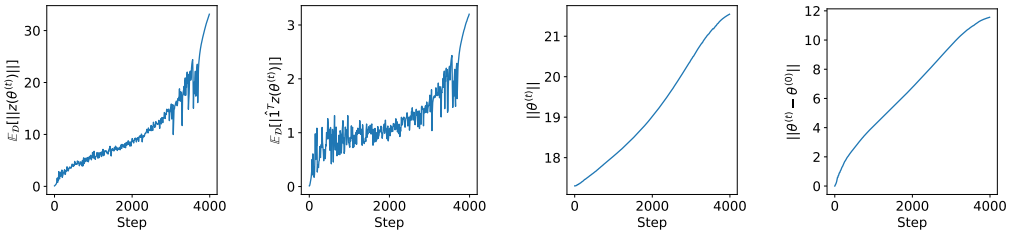

Figure 12: **(Left to Right): the logit norm $\mathbb{E}_{\mathcal{D}}[\|\boldsymbol{z}(\boldsymbol{\theta}^{(t)})\|]$, the absolute value of the logit sum $\mathbb{E}_{\mathcal{D}}[|\hat{1}^T \boldsymbol{z}(\boldsymbol{\theta}^{(t)})|]$, the weight norm $\boldsymbol{\theta}^{(t)}$, the distance from the initial weight $\|\boldsymbol{\theta}^{(t)} - \boldsymbol{\theta}^{(0)}\|$ during training (every 10 steps).** See together with Figure 6 (Left, solid red line) and Figure 5.

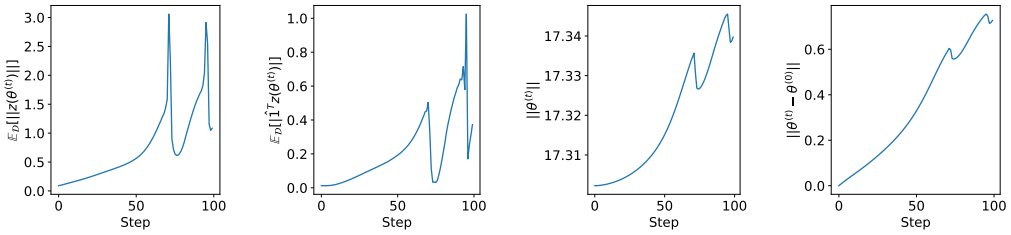

Figure 13: **(Left to Right): the logit norm $\mathbb{E}_{\mathcal{D}}[\|\boldsymbol{z}(\boldsymbol{\theta}^{(t)})\|]$, the absolute value of the logit sum $\mathbb{E}_{\mathcal{D}}[|\hat{1}^T \boldsymbol{z}(\boldsymbol{\theta}^{(t)})|]$, the weight norm $\|\boldsymbol{\theta}^{(t)}\|$, the distance from the initial weight $\|\boldsymbol{\theta}^{(t)} - \boldsymbol{\theta}^{(0)}\|$ in the early phase of training (every step).** See together with Figure 3 (Left).

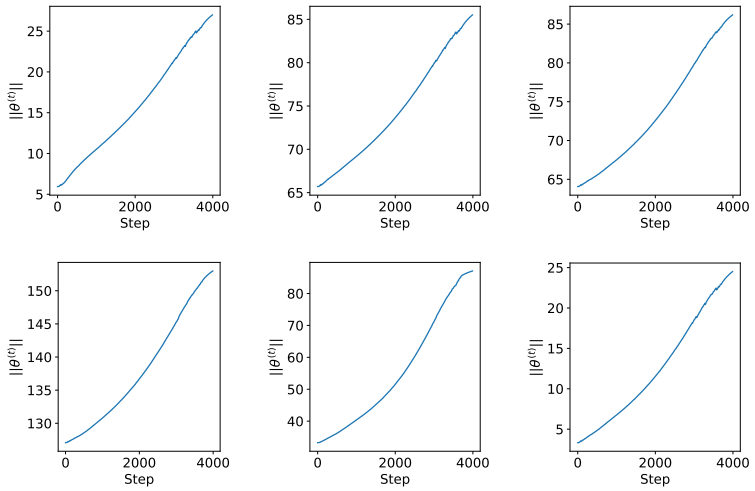

Figure 14: **The layerwise weight norms (6 layers from left to right and from top to bottom) of the 6CNN model in the early phase of training (every 10 step).**

E FIGURE 3 ($\|\boldsymbol{H}\|_\sigma \sim \lambda^{(1)}\|\boldsymbol{J}\hat{\mathbf{1}}\|^2$)

We empirically validate the claim that $\|\boldsymbol{H}\|_\sigma \sim \lambda^{(1)}\|\boldsymbol{J}\hat{\mathbf{1}}\|^2$ 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). Figure 3 and Figure 15 (Middle) are plotted with the equivalent settings (CIFAR-10, 6CNN, and GD with $\eta = 0.04$). At least, $\|\boldsymbol{H}\|_\sigma$ and $\lambda^{(1)}\|\boldsymbol{J}\hat{\mathbf{1}}\|^2$, 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 $\lambda^{(1)}$ decreases before the sharpness reaches the limit.

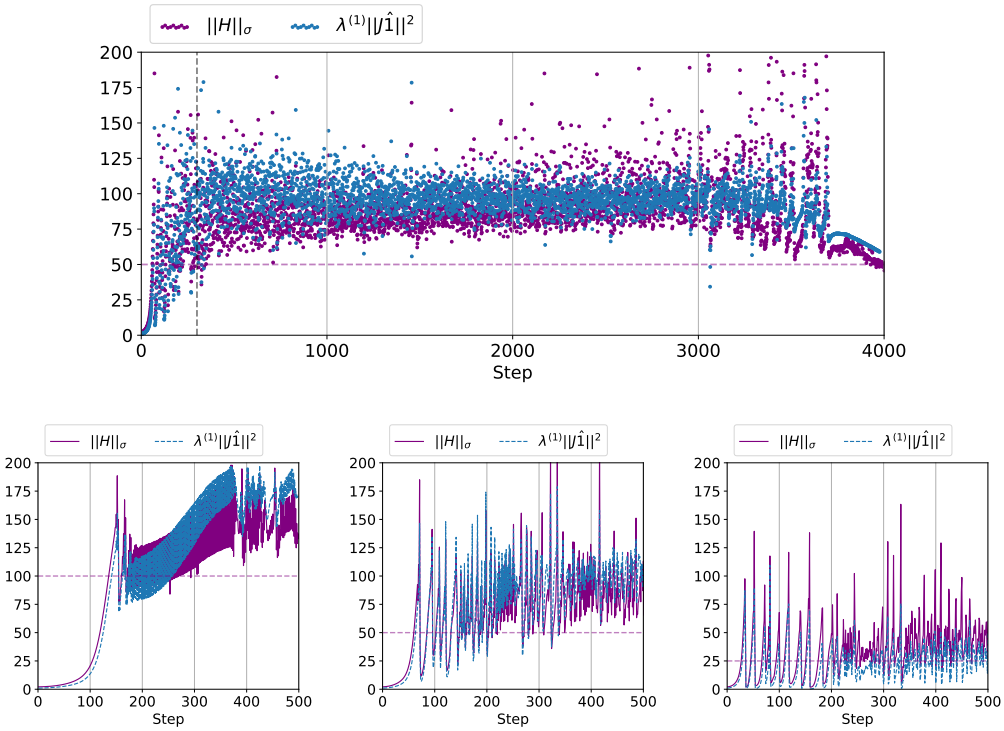

Figure 15: **The relation $\|\boldsymbol{H}\|_\sigma \sim \lambda^{(1)}\|\boldsymbol{J}\hat{\mathbf{1}}\|^2$on the CIFAR-10 dataset and the 6CNN model trained using GD with different learning rates $\eta = 0.02/0.04/0.08$ (from left to right).** Bottom figures are plotted for the early phase of training. Top figure is plotted for $\eta = 0.02$. Curves are plotted for every step. See the Figure 3 caption together.

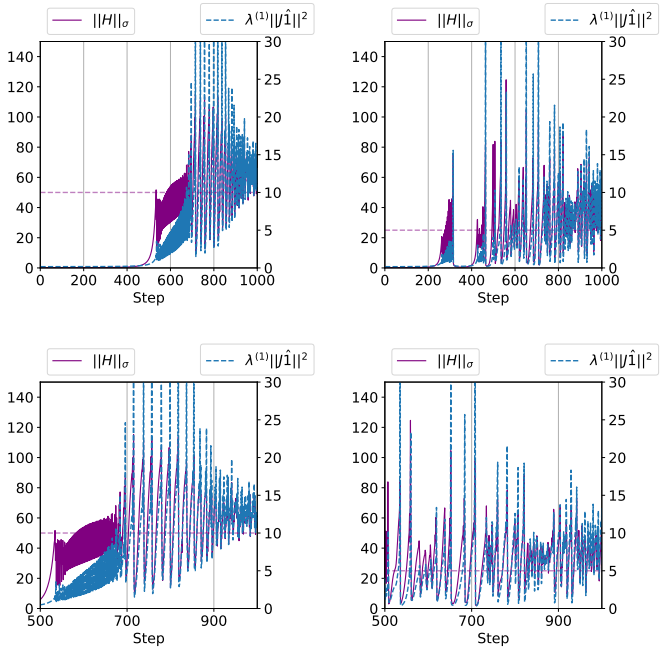

Figure 16: **The relation $\|\boldsymbol{H}\|_\sigma \sim \lambda^{(1)}\|\boldsymbol{J}\hat{\mathbf{1}}\|^2$ on the CIFAR-10 dataset and the VGG model trained using GD with different learning rates $\eta = 0.04/0.08$ (left/right) in the early phase of training.** Curves are plotted for every step (Top: 0-1000 stpes and Bottom: 500-1000 stpes). See the Figure 3 caption together.

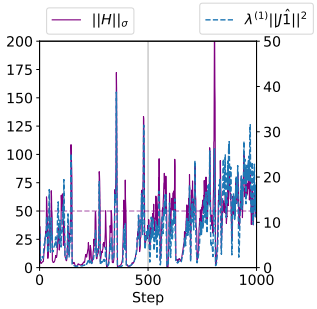

Figure 17: **The relation $\|\boldsymbol{H}\|_\sigma \sim \lambda^{(1)}\|\boldsymbol{J}\hat{\mathbf{1}}\|^2$ on the CIFAR-10 dataset and the ResNet model trained using SGD with learning rate $\eta = 0.04$ and batch sizes $|\mathcal{B}^{(t)}| = 128$ during training (0-1000 steps).** Curves are plotted for every step. See the Figure 3 caption together.

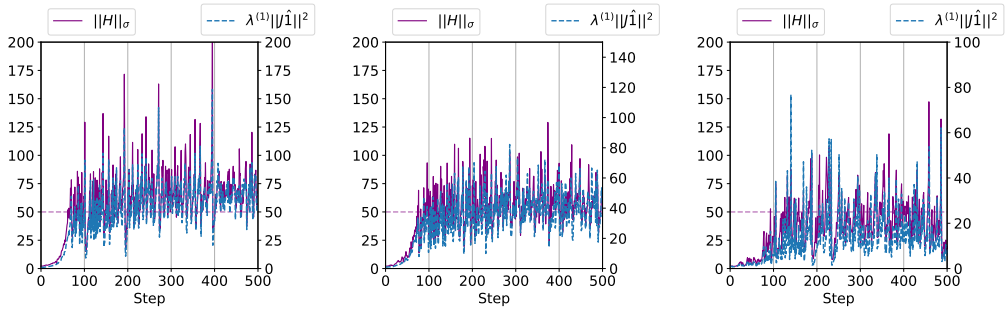

Figure 18: **The relation $\|\boldsymbol{H}\|_\sigma \sim \lambda^{(1)}\|\boldsymbol{J}\hat{\mathbf{1}}\|^2$ on the CIFAR-10 dataset and the 6CNN model trained using SGD with fixed learning rate $\eta = 0.04$ and different batch sizes $|\mathcal{B}^{(t)}| = 512/128/32$ (from left to right) in the initial 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. See the Figure 3 caption together.

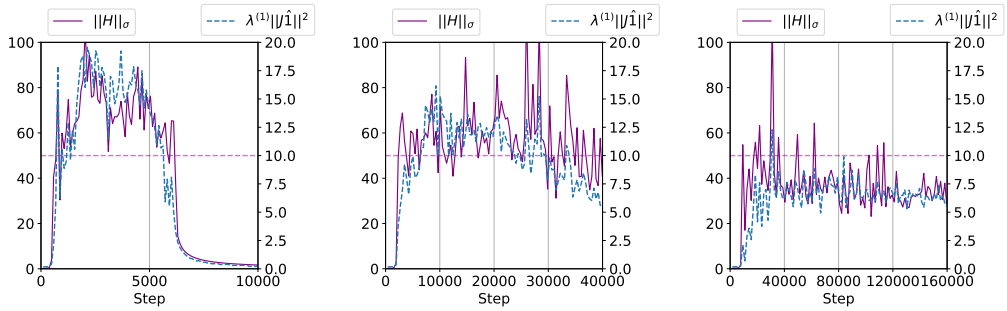

Figure 19: **The relation $\|\boldsymbol{H}\|_\sigma \sim \lambda^{(1)}\|\boldsymbol{J}\hat{\mathbf{1}}\|^2$ on the CIFAR-10 dataset and the VGG model trained using SGD with fixed learning rate $\eta = 0.04$ and different batch sizes $|\mathcal{B}^{(t)}| = 512/128/32$ (from left to right) during training (0-100 epochs).** Curves are plotted for every n steps ($n = 97/388/1552$) where $97 = \lfloor 50000/512 \rfloor$. See the Figure 3 caption together.

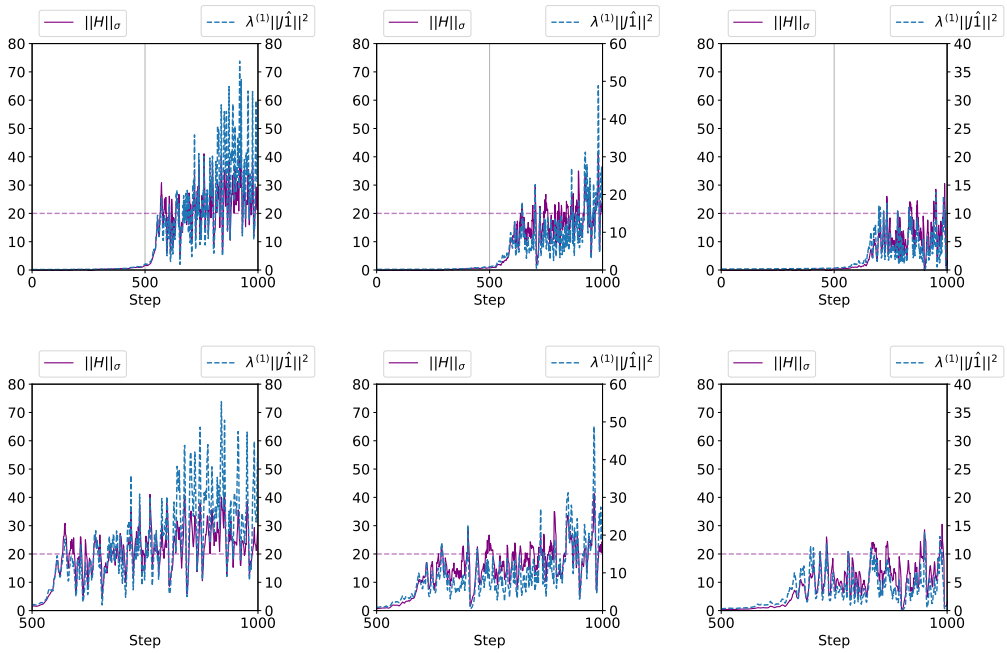

Figure 20: **The relation $\|\boldsymbol{H}\|_\sigma \sim \lambda^{(1)}\|\boldsymbol{J}\hat{\mathbf{1}}\|^2$ on the CIFAR-100 dataset and the VGG model trained using SGD with fixed learning rate $\eta = 0.1$ and different batch sizes $|\mathcal{B}^{(t)}| = 128/64/32$ (from left to right) in the early 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). See the Figure 3 caption together.

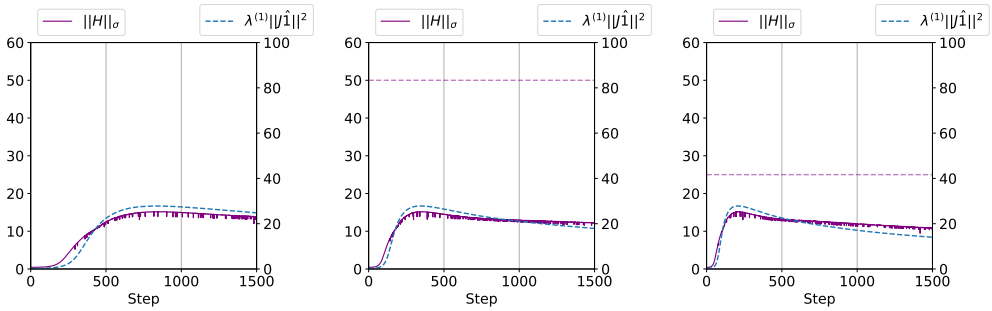

Figure 21: **The relation $\|\boldsymbol{H}\|_\sigma \sim \lambda^{(1)}\|\boldsymbol{J}\hat{\mathbf{1}}\|^2$ on the MNIST dataset and the 3FCN-MNIST model trained using GD with different learning rates $\eta = 0.02/0.04/0.08$ (from left to right) in the early phase of training.** Note that the sharpness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

F FIGURE 4 (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 22 (Bottom).

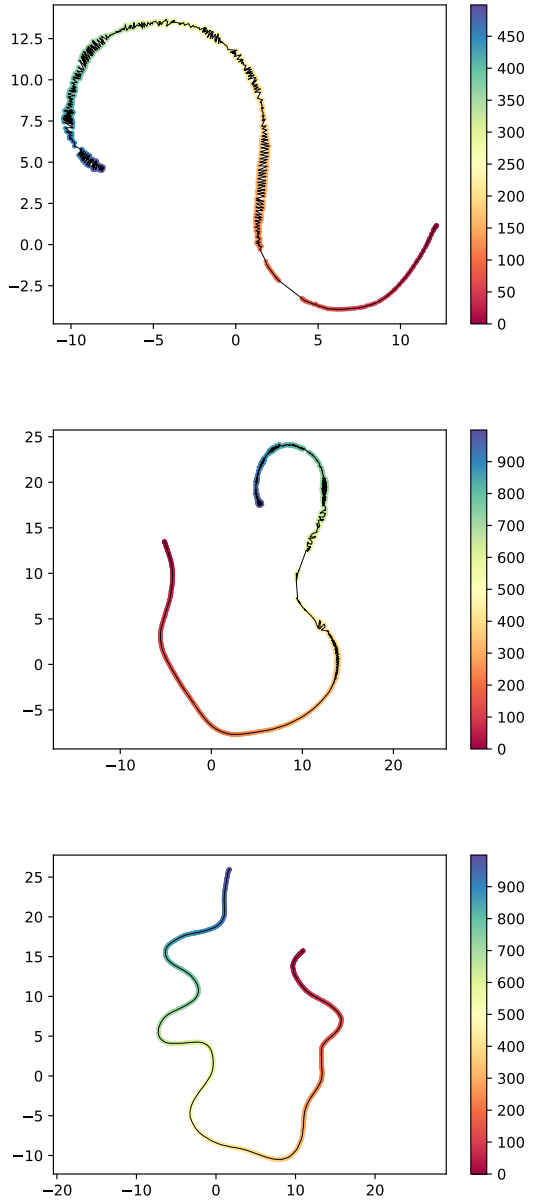

Figure 22: **Visualization of the optimization trajectory using UMAP.** (Top) UMAP on the CIFAR-10 dataset trained with 6CNN for the first 500 steps, (Middle) on the CIFAR-10 dataset trained with VGG for the first 1000 steps, and (Bottom) on the MNIST dataset trained with 3FCN-MNIST for the first 1000 steps (from red to blue).

G FIGURE 6 (IMPLICIT REGULARIZATION ON THE JACOBIAN NORM)

We ran the experiments for Figure 2 on the CIFAR-10 dataset. We used 6CNN and VGG for Figure 2 (Left) and Figure 2 (Right), respectively. Figure 5 illustrates the step-like structure of the three phases of the implicit Jacobian regularization. Note that the implicit regularization effects also exist in the third phase, but the effects gradually become weak as $\lambda^{(1)}$ decreases. 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 implicit regularization on the Jacobian norm begins to activate. We refer the readers to the previous section, Appendix E, for the fine-grained analysis of the early phase of training. We provide some more plots with different settings, as variants of Figure 6. Again, we observe that training with a larger learning rate and a smaller batch size limits the Jacobian norm with a smaller value (dotted red lines) in the Active Regularization Period. Curves are smoothed for visual clarity.

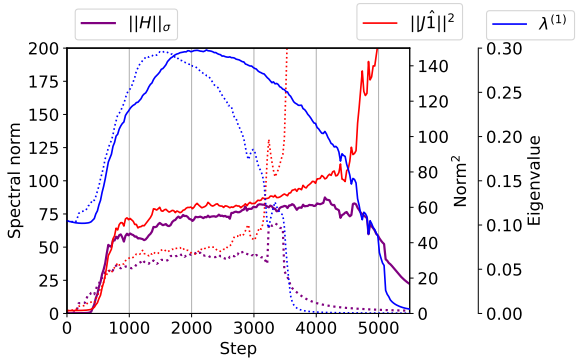

Figure 23: **The evolution of $\|\boldsymbol{H}\|_\sigma$, $\|\boldsymbol{J}\hat{\mathbf{1}}\|^2$, and $\lambda^{(1)}$ on the CIFAR-10 dataset and the VGG model trained using GD with the learning rates $\eta = 0.04/0.08$ (solid/dotted lines).** See the Figure 6 caption together.

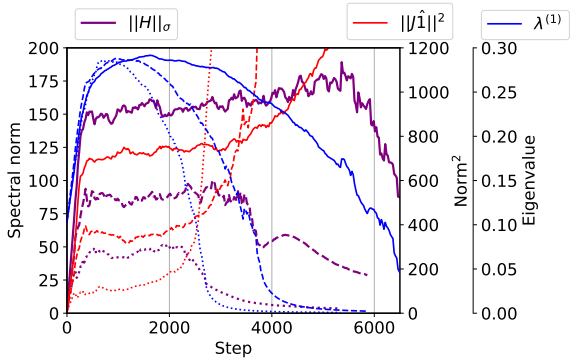

Figure 24: **The evolution of $\|\boldsymbol{H}\|_\sigma$, $\|\boldsymbol{J}\hat{\mathbf{1}}\|^2$, and $\lambda^{(1)}$ on the CIFAR-10 dataset and the 6CNN model trained using GD with the learning rates $\eta = 0.02/0.04/0.08$ (solid/dashed/dotted lines).** See the Figure 6 caption together.

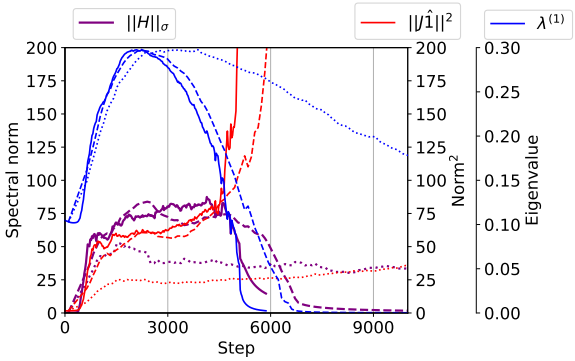

Figure 25: **The evolution of $\|\boldsymbol{H}\|_\sigma$, $\|\boldsymbol{J}\hat{\mathbf{1}}\|^2$, and $\lambda^{(1)}$ on the CIFAR-10 dataset and the VGG model trained using SGD with the fixed learning rate $\eta = 0.04$ and the batch sizes $|\mathcal{B}^{(t)}| = 50000(\textbf{GD})/512/32$ (solid/dashed/dotted lines).** Training with a batch size 512 shows similar evolutions to the GD training. See the Figure 6 caption together.

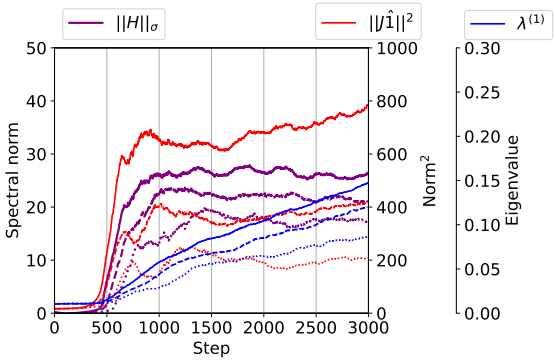

Figure 26: **The evolution of $\|\boldsymbol{H}\|_\sigma$, $\|\boldsymbol{J}\hat{\mathbf{1}}\|^2$, and $\lambda^{(1)}$ on the CIFAR-100 dataset and the VGG model trained using SGD with the fixed learning rate $\eta = 0.1$ and the batch sizes $|\mathcal{B}^{(t)}| = 128/64/32$ (solid/dashed/dotted lines).** See the Figure 6 caption together.

H FIGURE 8 ($\boldsymbol{Q}\boldsymbol{\Lambda}^{1/2}$)

Figure 27 shows the matrix $\boldsymbol{Q}\boldsymbol{\Lambda}^{1/2}$ (Figure 8) for different training steps. Figure 8 and Figure 27 (Top Right) are plotted at the equivalent step ($t = 1000$). Figure 27 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.

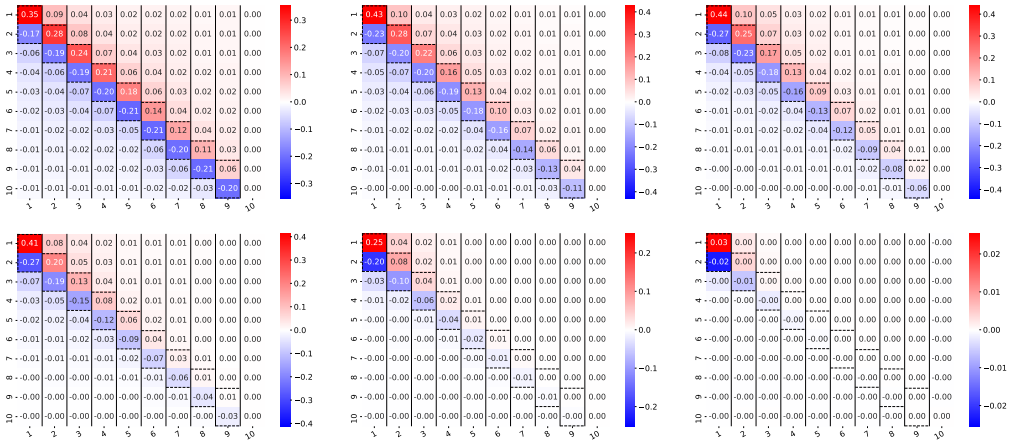

Figure 27: **Evolution of $Q\Lambda^{1/2}$ for some training steps during the training.** They are visualized for 100/500/1000/2000/4000/6000 steps from left to right and from top to bottom.

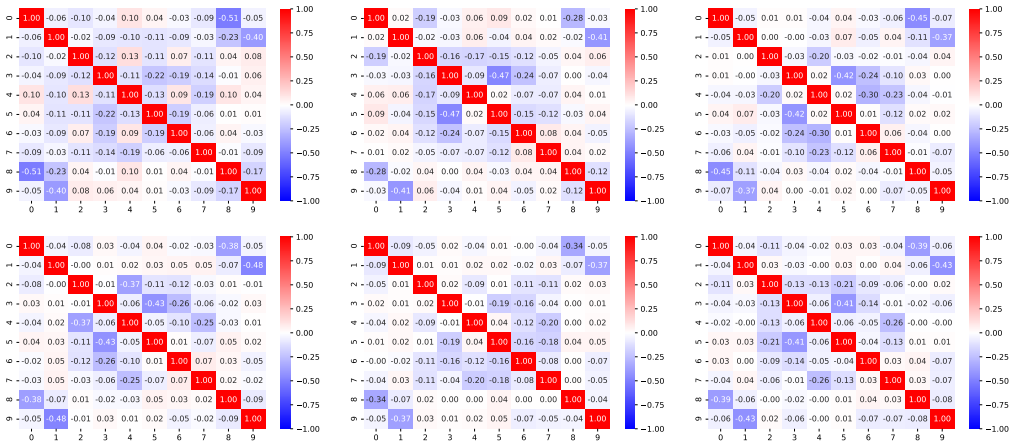

Figure 28: **Cosine similarities of $\{\overline{\boldsymbol{m}}^i\}_{i=1}^C$ during training.** They are visualized for step=500/1000/1500/2000/3000/4000 from left to right and from top to bottom. See the Figure 9 caption.

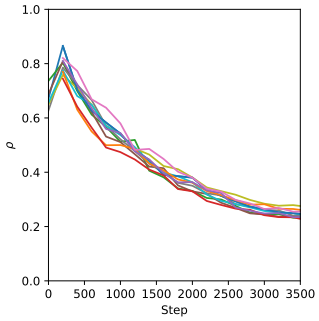

Figure 29: **The Mean Resultant Lengths (MRL) ρ for each $\mathcal{D}_i$.** Note that ρ is not defined for first few steps because some $\mathcal{D}_i$ are empty.

I FIGURE 9 (CLUSTERS OF $\boldsymbol{m}$ AROUND EACH $\overline{\boldsymbol{m}}^i$)

Figure 30 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 28 shows Figure 9 (Right) for different training steps. Figure 29 shows the evolution of the Mean Resultant Length (MRL) ρ in Figure 9 (Left) during training.

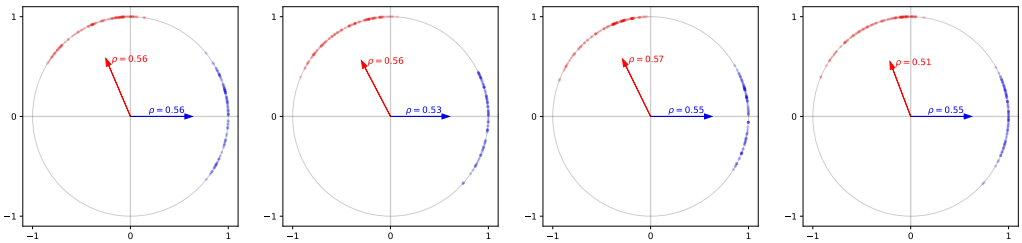

Figure 30: **Directional data of $\boldsymbol{m}$ from $\mathcal{D}_i$ and $\mathcal{D}_j$.** They are visualized for (i, j) = (airplane, ship)/(automobile, truck)/(dog, cat)/(deer, bird) from left to right. See the Figure 9 caption.

J FIGURE 10 (GRADIENT DESCENT HAPPENS MOSTLY IN THE TOP HESSIAN SUBSPACE)

Figure 31 shows similar results with different settings with VGG and learning rate $\eta = 0.08$.

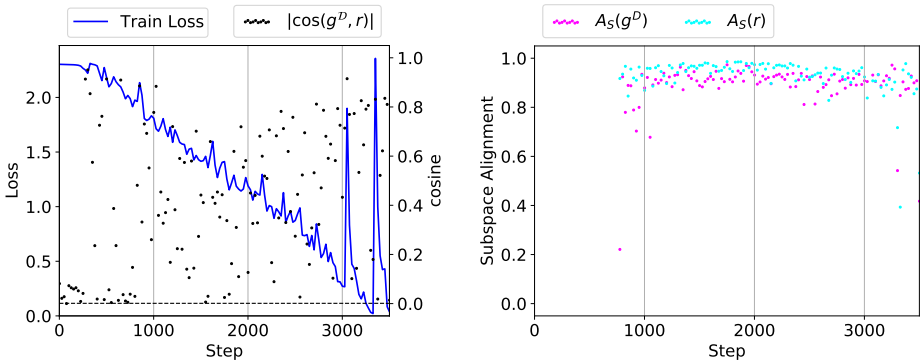

Figure 31: **(Left) Alignment between the two vectors $\boldsymbol{g}^{\mathcal{D}}$ and $\boldsymbol{r}$, and (Right) alignment of $\boldsymbol{g}^{\mathcal{D}}, \boldsymbol{r}$ in the subspace $\mathcal{S} = \text{span}(\{\overline{\boldsymbol{m}}^i\}_{i=1}^C)$ using VGG with $\eta = 0.08$.** See the 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.