Journal of Machine Learning Research 25 (2024) 1-44 Submitted 4/22; Revised 9/24; Published 11/24

# Learning Discretized Neural Networks under Ricci Flow

**Jun Chen**[1,2] JUNC@ZJU.EDU.CN
**Hanwen Chen**[1] CHENHANWEN@ZJU.EDU.CN
**Mengmeng Wang**[1] MENGMENGWANG@ZJU.EDU.CN
**Guang Dai**[3] GUANG.GDAI@GMAIL.COM
**Ivor W. Tsang**[4,5,6] IVOR.TSANG@GMAIL.COM
**Yong Liu**[1*] YONGLIU@IIPC.ZJU.EDU.CN

[1] *Institute of Cyber-Systems and Control, Zhejiang University, China*
[2] *School of Computer Science and Technology, Zhejiang Normal University, China*
[3] *SGIT AI Lab, State Grid Corporation of China, China*
[4] *Centre for Frontier Artificial Intelligence Research, Agency for Science, Technology and Research (A*STAR), Singapore*
[5] *Institute of High Performance Computing, Agency for Science, Technology and Research (A*STAR), Singapore*
[6] *College of Computing and Data Science, Nanyang Technological University, Singapore*

**Editor:** Aurelien Garivier

## Abstract

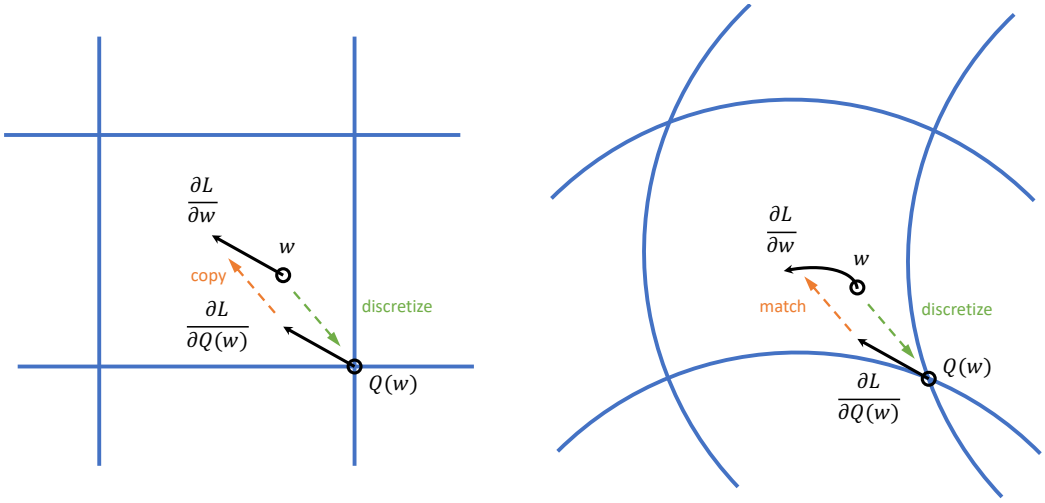
In this paper, we study Discretized Neural Networks (DNNs) composed of low-precision weights and activations, which suffer from either infinite or zero gradients due to the non-differentiable discrete function during training. Most training-based DNNs in such scenarios employ the standard Straight-Through Estimator (STE) to approximate the gradient w.r.t. discrete values. However, the use of STE introduces the problem of gradient mismatch, arising from perturbations in the approximated gradient. To address this problem, this paper reveals that this mismatch can be interpreted as a metric perturbation in a Riemannian manifold, viewed through the lens of duality theory. Building on information geometry, we construct the Linearly Nearly Euclidean (LNE) manifold for DNNs, providing a background for addressing perturbations. By introducing a partial differential equation on metrics, i.e., the Ricci flow, we establish the dynamical stability and convergence of the LNE metric with the $L^2$-norm perturbation. In contrast to previous perturbation theories with convergence rates in fractional powers, the metric perturbation under the Ricci flow exhibits exponential decay in the LNE manifold. Experimental results across various datasets demonstrate that our method achieves superior and more stable performance for DNNs compared to other representative training-based methods.

**Keywords:** discretized neural networks, gradient perturbation, information geometry, ricci flow, riemannian manifold

*. Corresponding author.

©2024 Jun Chen, Hanwen Chen, Mengmeng Wang, Guang Dai, Ivor W. Tsang and Yong Liu.

License: CC-BY 4.0, see https://creativecommons.org/licenses/by/4.0/. Attribution requirements are provided at http://jmlr.org/papers/v25/22-0444.html.




(a) Gradient propagation with STE

(b) Gradient propagation with metrics

Figure 1: Comparison of STE and our method. We denote the arrows and points as gradients and weights, respectively. In particular, when a point falls on the grid point, it means that the weight is discretized at this time. In the forward pass, the continuous weight $\boldsymbol{w}$ is mapped to a discrete weight $Q(\boldsymbol{w})$ via a discrete function. In the backward pass, the gradient is propagated from $\partial L/\partial Q(\boldsymbol{w})$ to $\partial L/\partial \boldsymbol{w}$. (a) The STE simply copies the gradient, i.e., $\partial L/\partial \boldsymbol{w} = \partial L/\partial Q(\boldsymbol{w})$. (b) Our method matches the gradient by introducing the proper metric $g_{\boldsymbol{w}}$, i.e., $\partial L/\partial \boldsymbol{w} = g_{\boldsymbol{w}}^{-1}\partial L/\partial Q(\boldsymbol{w})$ in a Riemannian manifold.

## 1. Introduction

Discretized Neural Networks (DNNs) (Courbariaux et al., 2016; Li et al., 2016; Zhu et al., 2016) have been proven to be efficient in computing, significantly reducing computational complexity, storage space, power consumption, and resources (Chen et al., 2021). Considering a discretized neural network that can be well-trained, the gradient w.r.t. the continuous weight[1] $\boldsymbol{w}$ propagating through a discrete function $Q(\cdot)$, i.e., $\frac{\partial L}{\partial \boldsymbol{w}} = \frac{\partial L}{\partial Q(\boldsymbol{w})}\frac{\partial Q(\boldsymbol{w})}{\partial \boldsymbol{w}}$, suffers from either infinite or zero derivatives because the derivative $\partial Q(\boldsymbol{w})/\partial \boldsymbol{w}$ is not calculable. In the backward pass, one can obtain the gradient $\partial L/\partial Q(\boldsymbol{w})$, but must update the continuous weight $\boldsymbol{w}$ using the gradient $\partial L/\partial \boldsymbol{w}$. Since the gradient $\partial L/\partial \boldsymbol{w}$ can not be obtained explicitly, the derivative $\partial Q(\boldsymbol{w})/\partial \boldsymbol{w}$ serves as a bridge to calculate $\partial L/\partial \boldsymbol{w}$ through the standard chain rule.

In order to address the problem of either infinite or zero gradients caused by the non-differentiable discrete function, Hinton (2012) first proposed the concept of Straight-Through Estimator (STE). This estimator directly equates $\partial L/\partial \boldsymbol{w}$ and $\partial L/\partial Q(\boldsymbol{w})$ in back-

1. In this paper, the continuous weight is relative to the neural network (its data type is full-precision). And the discretized weight is relative to the discretized neural network (its data type is low-precision).

propagation as if the derivative $\partial Q(\boldsymbol{w})/\partial \boldsymbol{w}$ had been the identity function. Furthermore, the rigorous definition of STE was developed by Bengio et al. (2013). This definition can be summarized as: the gradient w.r.t. the discretized weight can be approximated by the gradient w.r.t. the continuous weight with clipping, as shown in Figure 1(a). Subsequently, Courbariaux et al. (2016) applied STE to binarized neural networks and provided an approximated gradient as follows:

$$\begin{aligned} &\frac{\partial L}{\partial \boldsymbol{w}} = \frac{\partial L}{\partial \operatorname{sign}(\boldsymbol{w})}\mathbb{I}(\boldsymbol{w}), \\ &\text{where } \mathbb{I}(w_i) := \begin{cases} 1 & \text{if } |w_i| \leq 1 \\ 0 & \text{otherwise} \end{cases} \text{ and } \operatorname{sign}(w_i) := \begin{cases} +1 & \text{if } w_i \geq 0 \\ -1 & \text{otherwise} \end{cases}. \end{aligned} \tag{1}$$

Clearly, $\mathbb{I}(\cdot)$ is the indicator function, and $\operatorname{sign}(\cdot)$ is the binary function. Note that the discrete function $Q(\cdot)$ will degenerate to the binary function $\operatorname{sign}(\cdot)$ in binarized neural networks. In this context, $\partial L/\partial \operatorname{sign}(\boldsymbol{w})$ represents the gradient w.r.t. the binarized weight in binarized neural networks. STE had been successfully implemented in the training of binarized neural networks, and it was further extended to ternary neural networks (Li et al., 2016) and arbitrary bit-width discretized neural networks (Zhou et al., 2016).

In contrast, Non-STE methods encompass all techniques that do not rely on STE, such as those proposed by Hou et al. (2016), Bai et al. (2018), and Leng et al. (2018). However, the learning process of Non-STE methods is heavily dependent on hyper-parameters (Chen et al., 2019), such as weight partition portion in each iteration (Zhou et al., 2017) and penalty setting in tuning (Leng et al., 2018). Consequently, STE methods are widely adopted in DNNs due to their simplicity and versatility.

Nevertheless, the introduction of STE into DNNs inevitably leads to the problem of *gradient mismatch*: the gradient w.r.t. the continuous weight is not strictly equal to the gradient w.r.t. the discretized weight (Chen et al., 2019), compromising the training stability of DNNs (Cai et al., 2017; Liu et al., 2018; Qin et al., 2020). Furthermore, the formula of STE indicates that this problem can be alleviated by modifying the gradient.

Zhou et al. (2016) firstly proposed to transform the weight $\boldsymbol{w}$ into the new one $\tilde{\boldsymbol{w}}$ via

$$\tilde{\boldsymbol{w}} = \frac{\tanh(\boldsymbol{w})}{\max(|\tanh(\boldsymbol{w})|)}.$$

By discretizing the new weight $\tilde{\boldsymbol{w}}$, the STE then acts on $\tilde{\boldsymbol{w}}$. During back-propagation, the gradient can be further computed as follows

$$\frac{\partial L}{\partial \boldsymbol{w}} = \frac{\partial L}{\partial Q(\tilde{\boldsymbol{w}})}\frac{1-\tanh^2(\boldsymbol{w})}{\max(|\tanh(\boldsymbol{w})|)}.$$

The authors aim to manually redefine the indicator function $\mathbb{I}(\boldsymbol{w})$ as $\frac{1-\tanh^2(\boldsymbol{w})}{\max(|\tanh(\boldsymbol{w})|)}$. This modification is motivated by the fact that the function $\frac{1-\tanh^2(\boldsymbol{w})}{\max(|\tanh(\boldsymbol{w})|)}$ facilitates a smooth transition, thereby preventing abrupt clipping of the indicator function near $\pm 1$. It is remarkable that Chen et al. (2019) proposed to learn $\partial L/\partial \boldsymbol{w}$ by a neural network, e.g., fully-connected layers or LSTM (Sak et al., 2014). Their specific approach is to use neural

networks as a shared meta quantizer $M_\psi$ parameterized by $\psi$ across layers to replace the gradient via:

$$\frac{\partial L}{\partial \boldsymbol{w}} = M_\psi \left( \frac{\partial L}{\partial Q(\boldsymbol{w})}, \overline{\boldsymbol{w}} \right) \frac{\partial \overline{\boldsymbol{w}}}{\partial \boldsymbol{w}},$$

where $\overline{\boldsymbol{w}}$ is the weight from the meta quantizer. With the input of the gradient $\partial L/\partial Q(\boldsymbol{w})$, the meta quantizer outputs a new gradient to match $\partial L/\partial \boldsymbol{w}$ by updating the weight $\overline{\boldsymbol{w}}$ in the training process. Recently, Ajanthan et al. (2021) formulated the binarization of neural networks as a constrained optimization problem by introducing a mirror descent framework (Nemirovsky and Yudin, 1983). This method performs gradient descent in the dual space (unconstrained space) with gradients computed in the primal space (discrete space). Specifically, by mapping the primal variable $\boldsymbol{w}$ into the dual variable $\tilde{\boldsymbol{w}} = \tanh(\beta_k \boldsymbol{w})$, the gradient can be expressed as

$$\frac{\partial L}{\partial \boldsymbol{w}} = \frac{\partial L}{\partial \tilde{\boldsymbol{w}}} \left(1 - \tanh^2(\beta_k \boldsymbol{w})\right).$$

As the hyper-parameter $\beta_k$ approaches infinity, $\tilde{\boldsymbol{w}}$ gradually converges to $\text{sign}(\boldsymbol{w})$ until the corresponding neural network is fully binarized with an adaptive mirror map.

However, the method proposed by Zhou et al. (2016) only avoided abrupt clipping of $\mathbb{I}(\boldsymbol{w})$ by using $\frac{1-\tanh^2(\boldsymbol{w})}{\max(|\tanh(\boldsymbol{w})|)}$, which does not fundamentally alleviate the gradient mismatch in essence. Subsequently, while Chen et al. (2019) suggested automatically matching the gradient by learning a new neural network (a meta quantizer), it introduces additional errors in the gradient propagation due to extra weights from the meta quantizer, thereby intensifying the problem of gradient mismatch. Furthermore, Ajanthan et al. (2021) bypassed the problem of gradient mismatch by directly calculating the derivative $\partial \tilde{\boldsymbol{w}}/\partial \boldsymbol{w} = \left(1 - \tanh^2(\beta_k \boldsymbol{w})\right)$, implying that this method does not maintain discrete weights during training. Consequently, the problem of gradient mismatch still remains to be solved.

### 1.1 Contributions

In this study, we address the gradient mismatch between $\partial L/\partial \boldsymbol{w}$ and $\partial L/\partial Q(\boldsymbol{w})$, treating it as a perturbation phenomenon between these two gradients. By introducing the framework of Riemannian geometry in Figure 1(b), we further regard the gradient mismatch as a metric perturbation in a Riemannian manifold (Section 2.2) through the lens of duality theory (Amari and Nagaoka, 2000). As a partial differential equation on metrics, the Ricci flow (Sheridan and Rubinstein, 2006), is introduced, the metric perturbation can be exponentially decayed in theory, providing a solution to the problem of gradient mismatch. The main contributions of this paper are summarized in the following four aspects:

- We propose the LNE manifold endowed with the LNE metric, which is a special form of Ricci-flat metrics in essence. According to the information geometry (Amari, 2016), we construct LNE manifolds for neural networks, providing a background for dealing with perturbations.
- We reveal the stability of LNE manifolds under the Ricci-DeTurck flow with the $L^2$-norm perturbation on the basis of the connection between the Ricci-DeTurck flow and the Ricci flow. In this way, any Ricci flow starting close to the LNE metric exists

for all time and converges to the LNE metric. This stands in contrast to previous perturbation theories, where the convergence rate is in fractional powers. Instead, the metric perturbation under the Ricci flow exhibits exponential decay in the LNE manifold, providing theoretical support for effectively solving the problem of gradient mismatch.

- Based on the appealing characteristics of LNE manifolds under Ricci flow, a novel DNNs with the acceptable complexity, i.e., Ricci Flow Discretized Neural Network (RF-DNN) is developed. In practice, we calculate the Ricci curvature in such a way that the selection of coordinate systems is related to the input transformations of neural networks. In essence, the discrete Ricci flow is employed to overcome the problem of gradient mismatch.

- The experiments are implemented on several classification benchmark datasets and network structures. Experimental results demonstrate the effectiveness of RF-DNN compared with other representative training-based methods.

### 1.2 Overall Organization

The paper is organized as follows. In Section 2, we introduce the motivation and Ricci flow. Section 3 deduces the corresponding LNE manifold for neural networks based on the geometric structure measured by the LNE divergence. The stability of LNE manifolds under the Ricci-DeTurck is proved in Section 4. In Section 5, we calculate the approximated gradient in the LNE manifold to avoid solving the inverse of the LNE metric. Section 6 presents how to introduce discrete Ricci flow into DNNs and yields the corresponding algorithm. The experimental results and ablation studies for RF-DNNs are presented in Section 7. Section 8 concludes the entire paper. Proofs are provided in the Appendices.

The Ricci flow on Ricci-flat metrics is known in the literature to be stable for $C^0$ perturbations in the $L^\infty$-norm (Section 2.4). Based on a Bregman divergence (Bregman, 1967), the LNE metric, a special form of Ricci-flat metrics, is introduced in neural networks via the LNE divergence (Theorem 10). The stability of LNE manifolds under the Ricci-DeTurck flow is then proved (Corollary 29 and Theorem 15). A discretization of the Ricci flow is therefore proposed, leading to a practical algorithm (RF-DNNs, Algorithm 2).

## 2. Motivation and Formulation

### 2.1 Background

To establish the foundation for our study throughout the paper, we begin with the basic background for feed-forward DNNs, drawing from the work by Martens and Grosse (2015). Important notations are listed in Appendix B.

A neural network can be regarded as a function that transforms the input $\boldsymbol{a}_0$ into the output $\boldsymbol{a}_l$ through a series of $l$ layers. For the $i$-th layer ($i \in \{1, 2, \ldots, l\}$), we denote $\boldsymbol{W}_i$ as the weight matrix, $\boldsymbol{s}_i$ as the vector of these weighted sum, and $\boldsymbol{a}_i$ as the vector of output (also known as the activation). Each layer receives vectors of a weighted sum of the input from the previous layer and calculates their output through a nonlinear function. Note that we ignore the bias vector for brevity.

For a DNN, the introduction of a discrete function $Q(\cdot)$ is necessary to discretize the weight matrix $\boldsymbol{W}_i$ and the activation vector $\boldsymbol{a}_i$. We denote the discretized weight matrix as $\hat{\boldsymbol{W}}_i = Q(\boldsymbol{W}_i)$ and the discretized activation vector as $\hat{\boldsymbol{a}}_i = Q(\boldsymbol{a}_i)$. Then, the feed-forward of DNNs at each layer is given as follows:

$$\begin{aligned} \boldsymbol{s}_i &= \hat{\boldsymbol{W}}_i \hat{\boldsymbol{a}}_{i-1} \\ \boldsymbol{a}_i &= f(\boldsymbol{s}_i) \\ \hat{\boldsymbol{a}}_i &= Q(\boldsymbol{a}_i) \end{aligned} \tag{2}$$

where $f$ is a nonlinear (activation) function. The vectorized weights in each layer, before and after discretization, are denoted as $\boldsymbol{w}$ and $\hat{\boldsymbol{w}}$, respectively. Additionally, we define the discretized parameter vector as $\hat{\boldsymbol{\xi}} = \left[\mathrm{vec}\left(\hat{\boldsymbol{W}}_1\right)^\top, \mathrm{vec}\left(\hat{\boldsymbol{W}}_2\right)^\top, \ldots, \mathrm{vec}\left(\hat{\boldsymbol{W}}_l\right)^\top\right]^\top$, which consists of all of the network's parameters concatenated together, where $\mathrm{vec}(\cdot)$ is the operator that vectorizes a matrix by stacking their columns together. Similarly, the parameter vector is defined as $\boldsymbol{\xi} = \left[\mathrm{vec}\left(\boldsymbol{W}_1\right)^\top, \mathrm{vec}\left(\boldsymbol{W}_2\right)^\top, \ldots, \mathrm{vec}\left(\boldsymbol{W}_l\right)^\top\right]^\top$. Details regarding the back-propagation of DNNs are provided in later sections.

### 2.2 Motivation

We consider that the source of the gradient mismatch lies in a perturbation phenomenon between $\partial L/\partial \boldsymbol{w}$ and $\partial L/\partial Q(\boldsymbol{w})$ in terms of linear operators, expressed as:

$$\frac{\partial L}{\partial \boldsymbol{w}} = \frac{\partial L}{\partial Q(\boldsymbol{w})} + \mathcal{P}\left(\frac{\partial L}{\partial Q(\boldsymbol{w})}\right), \tag{3}$$

where the perturbation function $\mathcal{P}$ takes the gradient $\partial L/\partial Q(\boldsymbol{w})$ as input, with $\mathcal{P}\left(\partial L/\partial Q(\boldsymbol{w})\right)$ being much smaller than $\partial L/\partial Q(\boldsymbol{w})$. In general, $\mathcal{P}\left(\partial L/\partial Q(\boldsymbol{w})\right)$ can be expressed as $\mathcal{P}\left(\partial L/\partial Q(\boldsymbol{w})\right) = o\left(\partial L/\partial Q(\boldsymbol{w})\right)$. If the perturbation term $\mathcal{P}\left(\partial L/\partial Q(\boldsymbol{w})\right)$ can be significantly eliminated or decayed, an elegant solution to the gradient mismatch arises. Within the framework of perturbation theory in linear spaces (Kato, 2013), the rate of convergence for perturbations is typically expressed in fractional powers.

Inspired by the mirror descent framework[2], one can map the parameter from the primal space to the dual space, and subsequently calculate the gradient in the dual space. Naturally[3], when the Riemannian metric structure is introduced by means of information geometry, the gradient mismatch is conclusively viewed as a metric perturbation in a Riemannian manifold. Specifically, we rewrite the gradient $\partial L/\partial \boldsymbol{w}$ in Euclidean space as the gradient $\tilde{\partial} L/\tilde{\partial} \boldsymbol{w}$ in a Riemannian manifold. For the sake of simplicity, we use "$\tilde{\partial}$" to denote the derivative in a Riemannian manifold and "$\partial$" to denote the derivative in Euclidean space.

2. Mirror descent induces non-Euclidean structure by solving iterative optimization problems using different proximity functions. This algorithm is introduced by Nemirovsky and Yudin (1983), and analyzed by Beck and Teboulle (2003).
3. Natural gradient descent selects the steepest descent along a Riemannian manifold by multiplying the standard gradient by the inverse of the metric tensor (Amari, 1998).It is worth mentioning that mirror descent and natural gradient descent are proven to be equivalent (Raskutti and Mukherjee, 2015), implying that mirror descent represents the steepest descent direction along the Riemannian manifold corresponding to the choice of Bregman divergence.

The difference between these two gradients is governed by the inverse of the corresponding metric tensor. The problem of gradient mismatch can be further expressed as:

$$\frac{\tilde{\partial} L}{\tilde{\partial} \boldsymbol{w}} = g_{\boldsymbol{w}}^{-1} \frac{\partial L}{\partial Q(\boldsymbol{w})}, \tag{4}$$

where the perturbation item is implied in the metric $g_{\boldsymbol{w}}$, with the term $\frac{\partial L}{\partial Q(\boldsymbol{w})}$ representing the gradient $\partial L(\boldsymbol{w})$ as defined in Definition 1. Then the metric perturbation emerges, and the perturbation at this time is referred to the deviation from the original metric. In this way, we present the generalization of STE in a Riemannian manifold, which will degenerate into the standard STE when the Riemannian metric $g$ returns to the Euclidean metric $\delta$.

**Definition 1** *(Amari, 1998) The steepest descent direction of $L(\boldsymbol{w})$ in a Riemannnian manifold, i.e., the* ***natural gradient descent****, is given by*

$$\tilde{\partial} L(\boldsymbol{w}) = g_{\boldsymbol{w}}^{-1} \partial L(\boldsymbol{w}),$$

*where $g^{-1} = (g^{ij})$ is the inverse of the metric $g = (g_{ij})$ and $\partial L(\boldsymbol{w})$ is the gradient:*

$$\partial L(\boldsymbol{w}) = \left[ \frac{\partial L(\boldsymbol{w})}{\partial w_1}, \cdots, \frac{\partial L(\boldsymbol{w})}{\partial w_n} \right]^\top.$$

Subsequently, a key question arises: What kind of manifolds do we need to construct to naturally and effectively handle metric perturbations? Or, what makes a manifold "good" in the presence of perturbations? In practice, general relativity gives an excellent example in nature of dealing with small gravitational perturbations within the framework of a Riemannian manifold (Wald, 2010). To address the approximation in scenarios where gravity is "weak", the spacetime metric is nearly flat at this time in the context of general relativity. This approximation is sufficient for most cases, except for phenomena involving gravitational collapse and the large-scale structure of the universe. Assuming that the deviation $\gamma_{ij}$ of the actual spacetime metric $g_{ij} = \eta_{ij} + \gamma_{ij}$ from a flat metric $\eta_{ij}$ is "small", the linearized gravity is introduced to approximate the gravity in general relativity[4]. In this context, "smallness" is defined such that the components of $\gamma_{ij}$ are much smaller than 1 in the global inertial coordinate system of $\eta_{ij}$. Such a linearly nearly flat metric greatly simplifies the calculation of "weak" gravity, and manifolds constructed with such metrics are considered sufficient for approximating the manifold with perturbations.

Similarly, in this paper, we define the Linearly Nearly Euclidean (LNE) metric by regarding the Euclidean metric $\delta_{ij}$ as the flat metric $\eta_{ij}$. This metric plays a crucial role in handling metric perturbations in the background of LNE manifolds for DNNs. Motivated

4. Firstly, when analyzing flat spacetime corresponding to "zero" gravity, the flat metric $\eta_{ij}$ can be employed. Secondly, when dealing with nearly flat spacetime indicative of "weak" gravity, one can use the nearly flat metric $g_{ij} = \eta_{ij} + \gamma_{ij}$ for analysis. In this case, $g_{ij}$ and $\eta_{ij}$ are very close, allowing the first-order Taylor expansion of this linearized form to yield sufficiently accurate results. For instance, this metric form can accurately analyze the gravity produced by celestial bodies like the Earth or the Sun. Thirdly, when faced with "strong" gravity resulting from the large-scale structure of the universe, the linearized metric is no longer applicable due to the curved nature of spacetime.

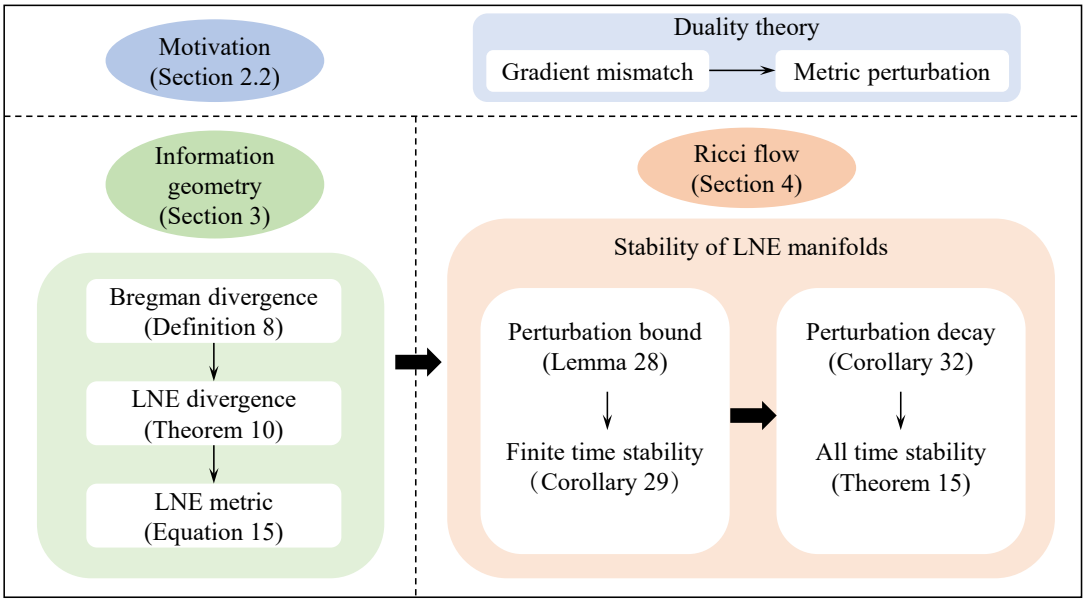



Figure 2: The overview of the theoretical ideas.

by the natural gradient descent connecting a neural network with the Riemannian metric[5], LNE metrics can be mathematically constructed in neural networks. To achieve this, our method involves introducing a convex function to derive the LNE divergence with the assistance of Bregman divergence (Bregman, 1967). The transition from a convex function to the LNE divergence operates within the mirror descent framework. Subsequently, the step from the LNE divergence to the LNE metric incorporates the concept of information geometry. Consequently, the LNE metric emerges in the gradient of the LNE manifold, similar to Definition 1. Finally, with the constructed manifold for DNNs in place, the remaining problem is how to efficiently decay the metric perturbation. This is achieved by employing a geometric tool, i.e., Ricci flow.

In addition, a series of proofs about stability illustrates that the Ricci flow can decay the metric perturbation in the cases of Ricci-flat metrics. Therefore, as long as we can prove that a small perturbation of the LNE metric under the Ricci flow decays, the metric perturbation can be alleviated, providing a theoretical solution for the problem of gradient mismatch in the training of DNNs. In contrast to previous perturbation theories, where the convergence rate is in fractional powers, the metric perturbation under the Ricci flow can

5. In the natural gradient descent, the Riemannian metric is expressed in the form of Fisher information matrix, i.e., $g = \begin{bmatrix} E\left[\mathrm{vec}\left(\frac{\partial L}{\partial \boldsymbol{W}_1}\right)\mathrm{vec}\left(\frac{\partial L}{\partial \boldsymbol{W}_1}\right)^\top\right] & \cdots & E\left[\mathrm{vec}\left(\frac{\partial L}{\partial \boldsymbol{W}_1}\right)\mathrm{vec}\left(\frac{\partial L}{\partial \boldsymbol{W}_l}\right)^\top\right] \\ \vdots & \ddots & \vdots \\ E\left[\mathrm{vec}\left(\frac{\partial L}{\partial \boldsymbol{W}_l}\right)\mathrm{vec}\left(\frac{\partial L}{\partial \boldsymbol{W}_1}\right)^\top\right] & \cdots & E\left[\mathrm{vec}\left(\frac{\partial L}{\partial \boldsymbol{W}_l}\right)\mathrm{vec}\left(\frac{\partial L}{\partial \boldsymbol{W}_l}\right)^\top\right] \end{bmatrix}$, where Fisher information matrix is associated with the weights from a neural network (Martens and Grosse, 2015).

be exponentially decayed in the LNE manifold. Figure 2 give an overview of the theoretical ideas to facilitate sorting out the solution steps.

### 2.3 Ricci Flow

**Definition 2** *(Sheridan and Rubinstein, 2006) A* ***Riemannian metric*** *on a smooth manifold $\mathcal{M}$ is a smoothly-varying inner product on the tangent space $T_p\mathcal{M}$ at each point $p \in \mathcal{M}$, i.e., a (0,2)-tensor which is symmetric and positive-definite at each point of $\mathcal{M}$. One will usually write $g$ for a Riemannian metric, and $g_{ij}$ for it coordinate representation. A manifold together with a Riemannian metric, $(\mathcal{M}, g)$, is called a* ***Riemannian manifold****.*

The concept of Ricci flow was first proposed by Hamilton (Hamilton et al., 1982) on the Riemannian manifold $\mathcal{M}$, building upon Definition 2 for a time-dependent metric $g(t)$. Given the initial metric $g_0$, the Ricci flow is described by a partial differential equation that evolves the metric tensor:

$$\begin{aligned} \frac{\partial}{\partial t} g(t) &= -2\,\mathrm{Ric}(g(t)) \\ g(0) &= g_0 \end{aligned} \tag{5}$$

where Ric denotes the Ricci curvature tensor, with a detailed definition available in Appendix A. The purpose of the Ricci flow is to prove Thurston's Geometrization Conjecture and Poincaré Conjecture, guiding the evolution of the metric towards specific geometric structures and topological properties (Sheridan and Rubinstein, 2006).

**Corollary 3** *(Sheridan and Rubinstein, 2006) The Ricci flow is* ***strongly parabolic*** *if there exists $\delta > 0$ such that for all covectors $\varphi \neq 0$ and all (symmetric[6]) $h_{ij} = \frac{\partial}{\partial t} g_{ij}(t) \neq 0$, the principal symbol of the differential operator $-2\,\mathrm{Ric}$ satisfies*

$$[-2\,\mathrm{Ric}](\varphi)(h)_{ij} h^{ij} = g^{pq} \left(\varphi_p \varphi_q h_{ij} + \varphi_i \varphi_j h_{pq} - \varphi_q \varphi_i h_{jp} - \varphi_q \varphi_j h_{ip}\right) h^{ij} > \delta \varphi_k \varphi^k h_{rs} h^{rs}$$

*where $h^{ij}$ is the inverse of $h_{ij}$.*

**Theorem 4** *(Ladyzhenskaia et al., 1988) Suppose that $u(t) : \mathcal{M} \times [0, T) \to \mathcal{E}$ is a time-dependent section of the vector bundle $\mathcal{E}$ where $\mathcal{M}$ is a Riemannian manifold. If the system of the Ricci flow is strongly parabolic at $u_0$ where $u_0 = u(0) : \mathcal{M} \to \mathcal{E}$, then there exists a unique solution on the time interval $[0, T)$.*

Combined with Corollary 3 and Theorem 4, one can determine the existence of a unique solution of the Ricci flow over a short time by verifying whether it is strongly parabolic. However, if we choose $h_{ij} = \varphi_i \varphi_j$, it is clear that the left hand side of the inequality in Corollary 3 is 0, thus the inequality can not hold. As a consequence, Ricci flow is not always strongly parabolic, and this lack of guarantee for the existence of a solution is highlighted by Theorem 4. In the following analysis, we delve into the non-parabolic nature and find a solution based on the relationship between the Ricci flow and the Ricci-DeTurck flow. The impact of its non-parabolic nature on different parts can be understood through the

6. The Riemannian metric $g_{ij}$ is always symmetric based on Definition 2. Hence, $h_{ij} = \frac{\partial}{\partial t} g_{ij}(t)$ is required to be symmetric.

linearization of the Ricci curvature tensor. We define the linearization of the Ricci curvature as $\mathcal{D}[\mathrm{Ric}]$ such that

$$\mathcal{D}[\mathrm{Ric}]\left(\frac{\partial}{\partial t}g_{ij}(t)\right) = \frac{\partial}{\partial t}\mathrm{Ric}(g_{ij}(t)).$$

**Lemma 5** *The linearization of* $-2\,\mathrm{Ric}$ *can be rewritten as*[7]

$$\begin{aligned} &\mathcal{D}[-2\,\mathrm{Ric}](h)_{ij} = g^{pq}\nabla_p\nabla_q h_{ij} + \nabla_i V_j + \nabla_j V_i + O(h_{ij}) \\ &\text{where} \quad V_i = g^{pq}\left(\frac{1}{2}\nabla_i h_{pq} - \nabla_q h_{pi}\right) \text{and} \quad h_{ij} = \frac{\partial}{\partial t}g_{ij}(t). \end{aligned} \tag{6}$$

**Proof** The proofs can be found in Appendix C.1. ■

By carefully observing Lemma 5, the impact on the non-parabolic nature of the Ricci flow comes from the terms $V_i$ and $V_j$ (Sheridan and Rubinstein, 2006), rather than the term $g^{pq}\nabla_p\nabla_q h_{ij}$. On the other hand, the term $O(h_{ij})$ will have no contributions to the principal symbol of $-2\,\mathrm{Ric}$, so we can ignore it in this problem. Next, we attempt to eliminate the impact of the non-parabolic nature on the Ricci flow.

Using a time-dependent diffeomorphism $\varphi(t) : \mathcal{M} \to \mathcal{M}$ (with $\varphi(0) = \mathrm{id}$), the pullback metrics $g(t)$ can be expressed as

$$g(t) = \varphi^*(t)\bar{g}(t), \tag{7}$$

satisfying the Ricci flow equation, where $\varphi^*(t)$ is the pullback through $\varphi(t)$. The above formula yields the new metric $\bar{g}(t)$ via the pullback, and the terms $V_i$ and $V_j$ can be reparameterized by choosing $\varphi(t)$ to form the Ricci-DeTurck flow (w.r.t. $\bar{g}(t)$), which is strongly parabolic. Furthermore, the solution is followed by the DeTurck Trick (DeTurck, 1983), involving a time-dependent reparameterization of the manifold:

$$\begin{aligned} \frac{\partial}{\partial t}\bar{g}(t) &= -2\,\mathrm{Ric}(\bar{g}(t)) - \mathcal{L}_{\frac{\partial\varphi(t)}{\partial t}}\bar{g}(t) \\ \bar{g}(0) &= \bar{g}_0, \end{aligned} \tag{8}$$

See Appendix C.2 for details. Thus, the Ricci-DeTurck flow has a unique solution for a short time. For the long time behavior, please refer to Appendix C.3.

### 2.4 Literature

For the Riemannian $n$-dimensional manifold $(\mathcal{M}^n, g)$ that is isometric to the Euclidean $n$-dimensional space $(\mathbb{R}^n, \delta)$, Schnürer et al. (2007) showed the stability of the Euclidean space under the Ricci flow for a small $C^0$ perturbation. Koch and Lamm (2012) demonstrated the stability of the Euclidean space along with the Ricci flow in the $L^\infty$-norm. Moreover, for the decay of the $L^\infty$-norm on Euclidean space, Appleton (2018) provided a proof from another idea.

7. In this paper, we use the Einstein summation convention (for example, $(AB)^j_i = A^k_i B^j_k$). When the same index appears twice in one term, once as an upper index and the other time as a lower index, summation is automatically taken over this index even without the summation symbol.

On the other hand, for a Ricci-flat metric with small perturbations, Guenther et al. (2002) proved that such metrics converge under Ricci flow. Considering the stability of integrable and closed Ricci-flat metrics, Sesum (2006) proved that the convergence rate is exponential because the spectrum of the Lichnerowicz operator is discrete. Furthermore, Deruelle and Kröncke (2021) demonstrated that an asymptotically locally Euclidean Ricci-flat metric is dynamically stable under the Ricci flow, with the $L^2 \cap L^\infty$ perturbation on non-flat and non-compact Ricci-flat manifolds. In our work, we discuss aspects related to Ricci-flat manifolds.

## 3. Neural Networks in LNE Manifolds

The aim of this section is to bulid an LNE manifold via information geometry, laying the foundation for the introduction of the Ricci flow. Specifically, we first introduce a convex function (Equation (13)) to derive the LNE divergence (Theorem 10) with the assistance of Bregman divergence (Definition 8). We then construct the LNE metric (Equation (15)) by incorporating the LNE divergence into neural networks. Consequently, the LNE metric emerges in the steepest descent gradient (Lemma 11) of the LNE manifold. Certainly, the mirror descent algorithm can equivalently establish the link between divergences and gradients, but it lacks the geometric meaning (manifold and metric) crucial for our purposes.

### 3.1 Neural Network Manifold

A neural network is composed of a large number of neurons connected with each other. The set of all such networks forms a manifold, where the weights represented by the neuron connections can be regarded as the coordinate system.

**Remark 6** *Comparing straight lines in Euclidean space, geodesics are the straightest possible lines that we can draw in a Riemannian manifold. Given a geodesic, there exists a unique non-Euclidean coordinate system. Once a curved coordinate system is selected in a Riemannian manifold, the symmetric and positive-definite metric is also defined based on Definition 2. This geometry-based metric can describe the properties of manifolds, such as curvature (Helgason, 2001).*

### 3.2 Euclidean Space and Divergence

From the viewpoint of information geometry, the metric can be deduced by the divergence satisfying the certain criteria (Basseville, 2013), summarized in Definition 7.

**Definition 7** *(Amari, 2016) $D[P:Q]$ is called a* ***divergence*** *when it satisfies the following criteria:*

*(1) $D[P:Q] \geq 0$,*

*(2) $D[P:Q] = 0$ when and only when $P = Q$,*

*(3) When $P$ and $Q$ are sufficiently close to each other, and their coordinates are denoted by $\boldsymbol{\xi}_P$ and $\boldsymbol{\xi}_Q = \boldsymbol{\xi}_P + d\boldsymbol{\xi}$ respectively, the Taylor expansion of the divergence can be written as*

$$D[\boldsymbol{\xi}_P : \boldsymbol{\xi}_Q] = \frac{1}{2} \sum_{i,j} g_{ij}(\boldsymbol{\xi}_P) d\xi_i d\xi_j + O(|d\boldsymbol{\xi}|^3), \tag{9}$$

*and the Riemannian metric $g_{ij}$ is symmetric and positive-definite*[8]*, acting on $\boldsymbol{\xi}_P$.*

When $P$ and $Q$ are sufficiently close, expressed in coordinates as column vectors $\boldsymbol{\xi}_P$ and $\boldsymbol{\xi}_Q$ based on Definition 7, the square of an infinitesimal distance $ds^2$ between them can be defined as:

$$ds^2 = 2D[\boldsymbol{\xi}_P : \boldsymbol{\xi}_Q] = \sum_{i,j} g_{ij}(\boldsymbol{\xi}_P) d\xi_i d\xi_j \tag{10}$$

where $d\boldsymbol{\xi}$ denotes a sufficiently small coordinate variation between the coordinates $\boldsymbol{\xi}_P$ and $\boldsymbol{\xi}_Q$. Here, we can ignore the third-order term $O(|d\boldsymbol{\xi}|^3)$ followed by Amari (2016) because the second-order approximation can give sufficiently accurate results. A manifold $\mathcal{M}$ is said to be Riemannian when a postive-definite metric $g_{ij}$ is defined on $\mathcal{M}$, and the square of the local distance between $\boldsymbol{\xi}_P$ and $\boldsymbol{\xi}_Q$ is given by Equation (10). Geometrically, the divergence $D[\boldsymbol{\xi}_P : \boldsymbol{\xi}_Q]$ provides the manifold with a Riemannian structure.

Using an orthonormal Cartesian coordinate system in Euclidean space, the Euclidean divergence is defined as half of the square of the Euclidean distance between $\boldsymbol{\xi}$ and $\boldsymbol{\xi}'$

$$D_E[\boldsymbol{\xi} : \boldsymbol{\xi}'] = \frac{1}{2} \sum_i (\xi_i - \xi_i')^2. \tag{11}$$

In this context, the Riemannian metric $g_{ij}$ degenerates into the Euclidean metric $\delta_{ij}$, resulting in the squared infinitesimal distance $ds^2$ expressed as:

$$ds^2 = 2D_E[\boldsymbol{\xi} : \boldsymbol{\xi} + d\boldsymbol{\xi}] = \sum_i (d\xi)^2 = \sum_{i,j} \delta_{ij} d\xi_i d\xi_j. \tag{12}$$

It is worth noting that the Euclidean metric $\delta_{ij}$ is equivalent to the identity matrix $\boldsymbol{I}$, and we use the notation of metrics here for consistency with geometry theory conventions.

### 3.3 LNE Manifold and Divergence

Recall that[9], in general relativity (Wald, 2010), the complete Riemannian manifold $(\mathcal{M}, g)$ endowed with a linearly nearly flat spacetime metric is considered to address the Newtonian limit through the linearized gravity. The form of this metric is $g_{ij} = \eta_{ij} + \gamma_{ij}$, where $\eta_{ij}$ represents the Minkowski metric (background to special relativity in flat spacetime), and $\gamma_{ij}$ denotes small perturbations. In practice, this theory is excellent for describing small gravitational perturbations when gravity is "weak".

Similarly, we define a metric $g_{ij} = \delta_{ij} + \gamma_{ij}$ in a Riemannian manifold, where $\delta_{ij}$ represents a flat Euclidean metric. An adequate definition of "smallness" in this context is that the components of $\gamma_{ij}$ are much smaller than 1 in the global inertial coordinate system of $\delta_{ij}$. Therefore, we can systematically develop the LNE metric to address small perturbations.

#### 3.3.1 Convex Function and Bregman Divergence

To construct the LNE manifold endowed with the LNE metric in the neural network, in accordance with Definition 7, we introduce a divergence to express the LNE metric, drawing an analogy to the relationship between the Euclidean metric and its divergence.

8. The components of a Riemannian metric in a coordinate basis take on the form of a symmetric and positive-definite matrix in differential geometry (Helgason, 2001).
9. The link between the LNE manifold and general relativity can be found in Section 2.2.

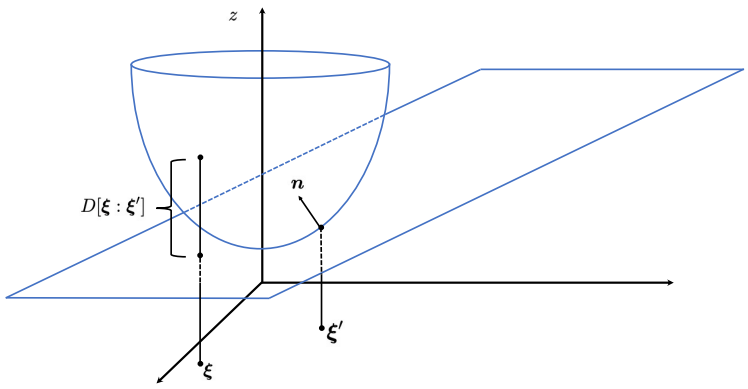



Figure 3: The divergence $D[\boldsymbol{\xi} : \boldsymbol{\xi}']$ is viewed as the distance between the convex function $\Phi(\boldsymbol{\xi})$ and its tangent hyperplane $z$, where the supporting hyperplane with normal vector $\boldsymbol{n} = \nabla\Phi(\boldsymbol{\xi}')$ at the point $\boldsymbol{\xi}'$ is defined.

The construction of a divergence relies on finding a suitable convex function (Bubeck et al., 2015). Here, we introduce a nonlinear function $\Phi(\boldsymbol{\xi})$ of coordinates $\boldsymbol{\xi}$ as the convex function, possessing a specific geometric structure to fulfill the requirements for constructing the LNE divergence. For a twice differentiable function, it is considered convex if and only if its Hessian is positive-definite

$$H(\boldsymbol{\xi}) = \left(\frac{\partial^2}{\partial\xi_i \partial\xi_j}\Phi(\boldsymbol{\xi})\right).$$

**Definition 8** *(Bregman, 1967) The* ***Bregman divergence*** $D_B[\boldsymbol{\xi} : \boldsymbol{\xi}']$ *is defined as the difference between a convex function* $\Phi(\boldsymbol{\xi})$ *and its tangent hyperplane* $z = \Phi(\boldsymbol{\xi}') + (\boldsymbol{\xi} - \boldsymbol{\xi}') \cdot \nabla\Phi(\boldsymbol{\xi}')$*, depending on the Taylor expansion at the point* $\boldsymbol{\xi}'$*:*

$$D_B[\boldsymbol{\xi} : \boldsymbol{\xi}'] = \Phi(\boldsymbol{\xi}) - \Phi(\boldsymbol{\xi}') - (\boldsymbol{\xi} - \boldsymbol{\xi}') \cdot \nabla\Phi(\boldsymbol{\xi}').$$

By drawing a tangent hyperplane that touches the convex function at the point $\boldsymbol{\xi}'$

$$z = \Phi(\boldsymbol{\xi}') + (\boldsymbol{\xi} - \boldsymbol{\xi}') \cdot \nabla\Phi(\boldsymbol{\xi}'),$$

we can express the distance between the convex function $\Phi(\boldsymbol{\xi})$ and the tangent hyperplane $z$ as the Bregman divergence. Since $\Phi(\boldsymbol{\xi})$ is convex, the graph of $\Phi(\boldsymbol{\xi})$ is always above the tangent hyperplane, touching it at $\boldsymbol{\xi}'$. The relationship between $\Phi(\boldsymbol{\xi})$ and $z$ is illustrated in Figure 3, with $z$ representing the vertical axis of the graph.

**Remark 9** *We show examples of Bregman divergence (Amari, 2016). For a convex function* $\Phi(\boldsymbol{\xi}) = 1/2 \sum_i \xi_i^2$ *in a Euclidean space, the Bregman divergence coincides with the Euclidean divergence, equivalently, the square of the Euclidean distance. When considering a convex function* $\Phi(\boldsymbol{\xi}) = -\sum_i \log \xi_i$*, the Bregman divergence is equivalent to the Logarithmic divergence. For another convex function* $\Phi(\boldsymbol{\xi}) = \sum_i \xi_i \log \xi_i$ *satisfying* $\sum_i \xi_i = 1$*, the Bregman divergence is the same as the KL divergence.*

### 3.3.2 LNE Divergence and Gradient

Similar to the Bregman divergence associated with a convex function, we aim to construct a new convex function to derive the LNE divergence, from which the LNE metric naturally emerges based on Definition 7. Inspired by the work of Ajanthan et al. (2021), we propose a novel convex function that satisfies symmetry and allows the geometric construction of an easy-to-compute metric with the linearly nearly Euclidean nature:

$$\Phi(\boldsymbol{\xi}) = \sum_i \frac{1}{\tau^2} \log\left(\cosh(\tau \xi_i)\right) \tag{13}$$

where $\tau$ is a constant parameter controlling the linearity closeness to Euclidean structure.

**Theorem 10** *By introducing a convex function $\Phi$ defined by Equation (13) into Definition 8, the **LNE divergence** between two points $\boldsymbol{\xi}$ and $\boldsymbol{\xi}'$ can be expressed as:*

$$\begin{aligned} D_{LNE}[\boldsymbol{\xi}' : \boldsymbol{\xi}] &= \sum_i \left[ \frac{1}{\tau^2} \log \frac{\cosh(\tau \xi_i')}{\cosh(\tau \xi_i)} - \frac{1}{\tau}(\xi_i' - \xi_i) \tanh(\tau \xi_i) \right] \\ &\approx \frac{1}{2} \sum_{i,j} \left[ \delta_{ij} - \left( \tanh(\tau \boldsymbol{\xi}) \tanh(\tau \boldsymbol{\xi})^\top \right)_{ij} d\xi_i d\xi_j \right]. \end{aligned} \tag{14}$$

**Proof** The detailed proofs can be found in Appendix E.1. ■

Combined with Definition 7, it is evident that the **LNE metric** corresponding to the LNE divergence is given by

$$\begin{aligned} g(\boldsymbol{\xi}) &= \delta_{ij} - \left[ \tanh(\tau \boldsymbol{\xi}) \tanh(\tau \boldsymbol{\xi})^\top \right]_{ij} \\ &= \begin{bmatrix} 1 - \tanh(\tau \xi_1) \tanh(\tau \xi_1) & \cdots & -\tanh(\tau \xi_1) \tanh(\tau \xi_n) \\ \vdots & \ddots & \vdots \\ -\tanh(\tau \xi_n) \tanh(\tau \xi_1) & \cdots & 1 - \tanh(\tau \xi_n) \tanh(\tau \xi_n) \end{bmatrix}. \end{aligned} \tag{15}$$

Building upon the concepts introduced in Section 3.1, we can leverage the parameters of a neural network to construct the LNE metric (with the neural network's parameter vector $\boldsymbol{\xi}$). Consequently, the neural network can be characterized within the LNE manifold, as measured by the LNE divergence based on Theorem 10. The steepest descent gradient in the LNE manifold is given by Lemma 11, resembling the natural gradient defined in Definition 1.

**Lemma 11** *The steepest descent gradient measured by the LNE divergence is defined as*

$$\tilde{\partial}_{\boldsymbol{\xi}} = g(\boldsymbol{\xi})^{-1} \partial_{\boldsymbol{\xi}} = \left[ \delta - \tanh(\tau \boldsymbol{\xi}) \tanh(\tau \boldsymbol{\xi})^\top \right]^{-1} \partial_{\boldsymbol{\xi}}. \tag{16}$$

**Proof** The proofs can be found in Appendix E.2. ■

Within the constructed LNE manifold, the introduction of the Ricci flow facilitates the decay of metric perturbations w.r.t. the LNE metric, which will be elaborated on in the following section.

## 4. Evolution of LNE Manifolds under Ricci Flow

This section focuses on LNE metrics under Ricci flow, aiming to demonstrate that the evolution of LNE manifolds exhibits strong stability properties over time. Specifically, we prove that the Ricci flow exponentially decays the $L^2$-norm perturbation to the LNE metric.

### 4.1 LNE Metrics and Ricci Flow

To facilitate the handling of metric perturbations, we have presented the LNE metric $g(\boldsymbol{\xi})$ in Equation (15), which takes the form of Ricci-flat metrics (Guenther et al., 2002; Deruelle and Kröncke, 2021). Furthermore, the definition of the LNE metric corresponds to the linearly nearly Euclidean Ricci-flat metric as per Definition 12, building upon prior work of Deruelle and Kröncke (2021). Notably, the equivalence of the LNE metric $g(\boldsymbol{\xi})$ extends to either $g_0$ under the Ricci flow or $\bar{g}_0$ under the Ricci-DeTurck flow, as they are diffeomorphic[10] to each other based on Equation (7).

**Definition 12** *A complete Riemannian n-manifold $(\mathcal{M}^n, \bar{g}_0)$ is said to be LNE with one end of order $\iota > 0$ if there exists a compact set $K \subset \mathcal{M}$, a radius $r$, a point $x$ in $\mathcal{M}$ and a diffeomorphism satisfying $\phi : \mathcal{M}\backslash K \to (\mathbb{R}^n\backslash B(x,r))/SO(n)$. Note that $B(x,r)$ is the ball and $SO(n)$ is a finite group acting freely on $\mathbb{R}^n\backslash\{0\}$. Then*

$$\left|\partial^k(\phi_*\gamma)\right|_\delta = O(r^{-\iota-k}) \quad \forall k \geq 0 \tag{17}$$

*holds on $(\mathbb{R}^n\backslash B(x,r))/SO(n)$. The LNE metric $\bar{g}_0$ can be linearly decomposed into a form containing the Euclidean metric $\delta$ and the deviation $\gamma$:*

$$\bar{g}_0 = \delta + \gamma. \tag{18}$$

### 4.2 All Time Convergence for $L^2$-norm Perturbations

Firstly, buliding upon previous proofs (Koiso, 1983; Besse, 2007), we can establish that the LNE manifold $(\mathcal{M}^n, g_0)$ is integral and linearly stable, as defined in Definition 13 and Definition 14.

**Definition 13** *(Deruelle and Kröncke, 2021) A complete LNE n-manifold $(\mathcal{M}^n, g_0)$ is said to be linearly stable if the $L^2$ spectrum of the Lichnerowicz operator $L_{g_0} := \Delta_{g_0} + 2\,\mathrm{Rm}(g_0)*$ is in $(-\infty, 0]$ where $\Delta_{g_0}$ is the Laplacian, when $L_{g_0}$ acting on $d_{ij}$ satisfies*

$$\begin{aligned} L_{g_0}(d) &= \Delta_{g_0} d + 2\,\mathrm{Rm}(g_0) * d \\ &= \Delta_{g_0} d + 2\,\mathrm{Rm}(g_0)_{iklj} d_{mn} g_0^{km} g_0^{ln}. \end{aligned} \tag{19}$$

**Definition 14** *(Deruelle and Kröncke, 2021) A n-manifold $(\mathcal{M}^n, g_0)$ is said to be integrable if a neighbourhood of $g_0$ has a smooth structure.*

10. When a Ricci flow exists, a corresponding Ricci-DeTurck flow exists, and vice versa.

Due to the diffeomorphic relationship between the Ricci flow and the Ricci–DeTurck flow, we introduce a metric perturbation for the Ricci-DeTurck flow, and Equation (8) can be further reformulated as follows:

$$\begin{aligned} \frac{\partial}{\partial t}\bar{g}(t) &= -2\,\mathrm{Ric}(\bar{g}(t)) - \mathcal{L}_{\frac{\partial \varphi(t)}{\partial t}}\bar{g}(t) \\ \bar{g}(0) &= \bar{g}_0 + d \end{aligned} \tag{20}$$

where $d = \bar{g}(0) - \bar{g}_0$ is a metric perturbation deviated from the LNE metric $\bar{g}_0$. In this way, $d(t) - d_0(t) = \bar{g}(t) - \bar{g}_0(t)$ holds because we define $d_0(t) = \bar{g}_0(t) - \bar{g}_0$.

**Theorem 15** *Let $(\mathcal{M}^n, \bar{g}_0)$ be the LNE $n$-manifold which is linearly stable and integrable. For any metric $\bar{g}(t) \in \mathcal{B}_{L^2}(\bar{g}_0, \epsilon_2)$ where a constant $\epsilon_2 > 0$, there is a complete Ricci–DeTurck flow $(\mathcal{M}^n, \bar{g}(t))$ starting from $\bar{g}(t)$ converging to the LNE metric $\bar{g}(\infty) \in \mathcal{B}_{L^2}(\bar{g}_0, \epsilon_1)$ where $\epsilon_1$ is a small enough constant.*

**Proof** The proofs can be found in Appendix D.3. ■

According to Theorem 15, the $L^2$-norm metric perturbation w.r.t. the LNE metric can be dynamically decayed by the Ricci-DeTurck flow in all time. For more details, refer to Appendix D (finite-time stability in Appendix D.1 and all-time stability in Appendix D.2). By proving the finite time existence of the Ricci-DeTurck flow with $L^2$-norm perturbations (Corollary 29), we then establish the convergence of $L^2$-norm perturbations w.r.t. the LNE metric for all time under the Ricci-DeTurck flow (Theorem 15).

### 4.3 Perturbation Analysis

Following the analysis in (Sesum, 2006), we further obtain $|\bar{g}(t) - \bar{g}_0(\infty)| < Ce^{-\epsilon_2 t}$, indicating exponential convergence of the metric perturbation. Consequently, it also exhibits exponential convergence for $g(t)$ under the Ricci flow, assuming the existence of a solution of the Ricci flow. Recall that by reparameterizing $\bar{g}(t)$ to $g(t) = \varphi^*(t)\bar{g}(t)$ via the pullback, the perturbation entirely originates from $\bar{g}(t)$ and is independent of the time-dependent diffeomorphism $\varphi^*(t)$.

In Section 3, the metric $g(\boldsymbol{\xi}) = \delta_{ij} - \left[\tanh(\tau\boldsymbol{\xi})\tanh(\tau\boldsymbol{\xi})^\top\right]_{ij}$ constructed for the neural network is a kind of LNE metrics (as per Definition 12), thereby ensuring the perturbation for this metric undergoes exponential decay under the Ricci flow.

## 5. Discretized Neural Networks in LNE Manifolds

Up to this point, we have tackled the problem of gradient mismatch by constructing LNE manifolds for neural networks (Section 3) and implementing an exponential decay mechanism for metric perturbations (Section 4). However, the practical computation of the steepest descent gradient in the LNE manifold, as indicated by Lemma 11, poses challenges due to the involovement of the inverse of the LNE metric. In this section, our objective is to approximate the inverse of the LNE metric and sebsequently derive the approximated gradient in the LNE manifold. This step is crucial for developing a practical algorithm to train DNNs in the LNE manifold.

### 5.1 Gradient Computation in Discretized Neural Networks

Recall that Courbariaux et al. (2016) applied STE to binarized neural networks, formulated as in Equation (1). Subsequently, Zhou et al. (2016) extended STE to arbitrary bit-width discretized neural networks. The generalized form of STE in discretized neural networks is expressed as:

$$\frac{\partial L}{\partial \boldsymbol{w}} = \frac{\partial L}{\partial Q(\boldsymbol{w})}. \tag{21}$$

Before introducing the LNE manifold to DNNs, a contradiction needs resolution. According to Lemma 11, the LNE manifold is defined based on the parameter $\boldsymbol{\xi}$ across all layers in a neural network. However, back-propagation computes the gradient layer-by-layer, specifically on the weight $\boldsymbol{w}$ of each layer. This misalignment prevents the direct association of gradient updates with the LNE manifold. Fortunately, we can redefine the LNE manifold layer-by-layer by substituting $\boldsymbol{\xi}$ with $\boldsymbol{w}$, effectively defining the LNE manifold for each layer. Building upon Lemma 11, the steepest descent gradient is then reformulated as:

$$\tilde{\partial}_{\boldsymbol{w}} = g^{-1}(\boldsymbol{w})\partial_{\boldsymbol{w}} = \left[\delta - \tanh(\tau \boldsymbol{w}) \tanh(\tau \boldsymbol{w})^{\top}\right]^{-1} \partial_{\boldsymbol{w}}, \tag{22}$$

which can be used for the gradient computation in DNNs, i.e.,

$$\frac{\tilde{\partial} L}{\tilde{\partial} \boldsymbol{w}} = \left[\delta - \tanh(\tau \boldsymbol{w}) \tanh(\tau \boldsymbol{w})^{\top}\right]^{-1} \frac{\partial L}{\partial Q(\boldsymbol{w})}. \tag{23}$$

Furthermore, the proposed gradient, as described above, is part of our framework to address the problem of gradient mismatch, based on Equation (4). In this context, the metric is layer-by-layer LNE. However, the computation of the gradient involves the inverse of the LNE metric, a process that demands significant computational resources. Hence, we introduce two methods for approximating the gradient of DNNs in LNE manifolds: weak approximation and strong approximation, respectively. The approximated gradient is defined as the direction in parameter space that maximizes the objective's variation per unit change along the layer-by-layer LNE manifold.

### 5.2 Strong Approximation

Our objective is to approximate the inverse of the LNE metric and subsequently approximate the gradient in Equation (23). Based on the universal approximation theorem (Cybenko, 1989; Hornik, 1991), which asserts that a continuous function on compact subsets can be approximated by a neural network with a single hidden layer and a finite number of neurons (Jejjala et al., 2020), we introduce a Multi-Layer Perceptron (MLP) neural network, depicted in Figure 4, to minimize the loss function:

$$\tilde{L} = \|\boldsymbol{I} - g(\boldsymbol{w})\boldsymbol{G}\|^2. \tag{24}$$

For the $n \times n$ symmetric metric $g(\boldsymbol{w})$, it can be decomposed into the combination of entries $P$ and $A$, where $P$ consists of the elements of the lower triangular matrix, containing $n(n-1)/2$ real parameters, and $A$ consists of the elements of the diagonal matrix, containing $n$ real parameters. Therefore, the matrix $\boldsymbol{G}$ can be effectively utilized to strongly approximate the inverse of the metric $g(\boldsymbol{w})$.

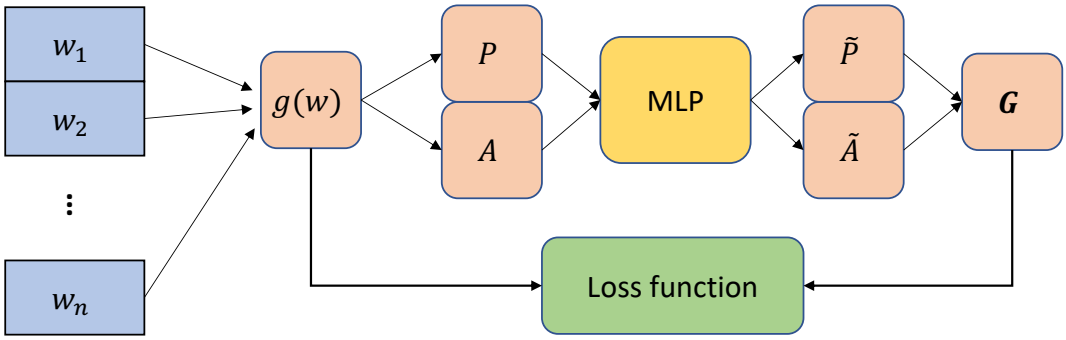



Figure 4: The flow chart of strong approximation of $g^{-1}(\boldsymbol{w})$. The new entries $\tilde{P}$ and $\tilde{A}$ generated by the neural network constitute a matrix $\boldsymbol{G}$, which is multiplied by the metric $g(\boldsymbol{w})$. As the loss function, defined by Equation (24), decreases, the matrix $\boldsymbol{G}$ serves to approximate the inverse of the metric $g(\boldsymbol{w})$.

### 5.3 Weak Approximation

In this subsection, we present a method for the weak approximation of the inverse of the LNE metric with efficient calculations.

**Definition 16** *For $\boldsymbol{A} \in \mathcal{R}^{n\times n}$, $\boldsymbol{A}$ is called **diagonally dominant** when it satisfies*

$$\left|a_{ii}\right| > \sum_{j=1, j\neq i}^{n} \left|a_{ij}\right|, \quad i = 1, 2, \ldots, n.$$

**Definition 17** *If $\boldsymbol{A} \in \mathcal{R}^{n\times n}$ is a diagonally dominant matrix, then $\boldsymbol{A}$ is a **nonsingular matrix** together, i.e., $\boldsymbol{A}^{-1}$ exists.*

By considering the properties of the LNE metric, adjusting the parameter $\tau$ allows us to easily ensure that the LNE metric $g(\boldsymbol{w})$ is diagonally dominant based on Definition 16. Moreover, the existence of $g^{-1}(\boldsymbol{w})$ can be guaranteed based on Definition 17. According to Corollary 18, the weak approximation of the gradient in the LNE manifold can be calculated, offering a convenient feature for accelerating the computation of the inverse.

**Corollary 18** *Based on Definition 16 and Definition 17, the weak approximation of the gradient in the LNE manifold is defined as*

$$\tilde{\partial}_{\boldsymbol{w}} = \left[\delta - \tanh(\tau\boldsymbol{w})\tanh(\tau\boldsymbol{w})^{\top}\right]^{-1} \partial_{\boldsymbol{w}} \approx \left[\delta + \tanh(\tau\boldsymbol{w})\tanh(\tau\boldsymbol{w})^{\top}\right] \partial_{\boldsymbol{w}} \tag{25}$$

*if the LNE metric is diagonally dominant.*

**Proof** Considering the inverse of the LNE metric, due to the diagonally dominant property in Definition 16 and Definition 17, we can approximate $\left[\delta - \tanh(\tau\boldsymbol{w})\tanh(\tau\boldsymbol{w})^{\top}\right]^{-1}$ by

ignoring the fourth-order small quantity $\sum O(\rho_a\rho_b\rho_c\rho_d)$, i.e.,

$$\begin{aligned}
&\left[\delta - \tanh(\tau\boldsymbol{w})\tanh(\tau\boldsymbol{w})^\top\right]\left[\delta + \tanh(\tau\boldsymbol{w})\tanh(\tau\boldsymbol{w})^\top\right] \\
&= \begin{bmatrix} 1-\rho_1\rho_1 & -\rho_1\rho_2 & \cdots \\ -\rho_2\rho_1 & 1-\rho_2\rho_2 & \cdots \\ \vdots & \vdots & \ddots \end{bmatrix}\begin{bmatrix} 1+\rho_1\rho_1 & \rho_1\rho_2 & \cdots \\ \rho_2\rho_1 & 1+\rho_2\rho_2 & \cdots \\ \vdots & \vdots & \ddots \end{bmatrix} \\
&= \begin{bmatrix} 1-\sum O(\rho_a\rho_b\rho_c\rho_d) & \rho_1\rho_2-\rho_1\rho_2-\sum O(\rho_a\rho_b\rho_c\rho_d) & \cdots \\ -\rho_2\rho_1+\rho_2\rho_1-\sum O(\rho_a\rho_b\rho_c\rho_d) & 1-\sum O(\rho_a\rho_b\rho_c\rho_d) & \cdots \\ \vdots & \vdots & \ddots \end{bmatrix} \approx \boldsymbol{I}.
\end{aligned}$$

The proof is completed. ■

### 5.4 Training

Building upon previous work (Courbariaux et al., 2016), we present a practical algorithm for training DNNs in the LNE manifold. As outlined in Algorithm 1, this algorithm closely resembles the general DNN training algorithm, with the key difference lying in Line 14. Recall that, in Figure 1, conventional DNNs utilize STE to directly copy the gradient, i.e., $\tilde{\partial}_{\boldsymbol{W}_i} L = \partial_{\hat{\boldsymbol{W}}_i} L$. In contrast, our method involves matching the gradient by introducing the LNE metric. Moreover, we can practically compute this gradient in Line 14 using either the strong approximation or weak approximation mentioned above.

## 6. Ricci Flow Discretized Neural Networks

In this section, we introduce Ricci flow discretized neural networks (RF-DNNs). The introduction of the Ricci flow implies that the background of the discussed DNNs is the LNE manifold. Our primary goal is to offer a practical solution for metric perturbations, thereby addressing the problem of gradient mismatch. Thus, we will focus on the practical calculation of discrete Ricci flow, rather than solely engaging in theoretical analysis.

To establish the connection between the Ricci flow and neural networks, we discretize the Ricci flow and select a suitable coordinate system. In Section 3, we have established the relationship between the LNE metric and neural networks for the left-hand side of the Ricci flow. Notably, we utilize the form of the LNE metric in these calculations, and such metrics at this stage incorporate perturbations. Moving to the right-hand side of the Ricci flow, we need to compute the Ricci curvature tensor with the chosen coordinate system. This coordinate system is important for linking Ricci curvature to neural networks. Specifically, we define a method for calculating the Ricci curvature, where the selection of coordinate systems is associated with input transformations. This implies that the Ricci curvature in neural networks reflects the impact of different input transformations on the parameters.

**Algorithm 1** An algorithm for training DNNs in the LNE manifold. We denote the gradient in the LNE manifold as $\tilde{\partial}$. For brevity, we omit the normalization operation (Ioffe and Szegedy, 2015; Ba et al., 2016).

**Input:** A minibatch of inputs and targets $(\boldsymbol{x} = \boldsymbol{a_0}, \boldsymbol{y})$, $\boldsymbol{\xi}$ mapped to $(\boldsymbol{W}_1, \boldsymbol{W}_2, \ldots, \boldsymbol{W}_l)$, $\hat{\boldsymbol{\xi}}$ mapped to $\left(\hat{\boldsymbol{W}}_1, \hat{\boldsymbol{W}}_2, \ldots, \hat{\boldsymbol{W}}_l\right)$, a nonlinear function $f$, a constant factor $\tau$ and a learning rate $\eta$.

**Output:** The updated discretized parameters $\hat{\boldsymbol{\xi}}$.

{Forward propagation}
**for** $i = 1; i \leq l; i++$ **do**
  Discretize $\hat{\boldsymbol{W}}_i = Q(\boldsymbol{W}_i)$;
  Compute $\boldsymbol{s}_i = \hat{\boldsymbol{W}}_i \hat{\boldsymbol{a}}_{i-1}$;
  Discretize $\hat{\boldsymbol{a}}_i = Q\left(f(\boldsymbol{s}_i)\right)$;
**end for**
{Loss derivative}
Compute $L = L(\boldsymbol{y}, \boldsymbol{z})$;
Compute $\partial_{\boldsymbol{a}_l} L = \frac{\partial L(\boldsymbol{y}, \boldsymbol{z})}{\partial \boldsymbol{z}}\big|_{\boldsymbol{z}=\hat{\boldsymbol{a}}_l}$;
{Backward propagation}
**for** $i = l; i \geq 1; i--$ **do**
  Compute $\partial_{\boldsymbol{s}_i} L = \partial_{\boldsymbol{a}_i} L \odot f'(\boldsymbol{s}_i)$;
  Compute $\partial_{\hat{\boldsymbol{W}}_i} L = (\nabla_{\boldsymbol{s}_i} L)\, \hat{\boldsymbol{a}}_{i-1}^\top$;
  Compute $\tilde{\partial}_{\boldsymbol{W}_i} L = g^{-1}(\boldsymbol{W}_i) \partial_{\hat{\boldsymbol{W}}_i} L$ based on Equation (23);
  Compute $\partial_{\hat{\boldsymbol{a}}_{i-1}} L = \hat{\boldsymbol{W}}_i^\top \left(\partial_{\boldsymbol{s}_i} L\right)$;
**end for**
{The parameters update}
**for** $i = l; i \geq 1; i--$ **do**
  Update $\boldsymbol{W}_i \leftarrow \boldsymbol{W}_i - \eta \cdot \tilde{\partial}_{\boldsymbol{W}_i} L$;
**end for**
Update $\hat{\boldsymbol{\xi}} = \left[\mathrm{vec}\left(\hat{\boldsymbol{W}}_1\right)^\top, \mathrm{vec}\left(\hat{\boldsymbol{W}}_2\right)^\top, \ldots, \mathrm{vec}\left(\hat{\boldsymbol{W}}_l\right)^\top\right]^\top$;

### 6.1 Ricci Curvature in Neural Networks

Now, let's consider the Ricci curvature tensor on the Riemannian metric $g$. According to Appendix A, its coordinate form can be expressed as follows:

$$\begin{aligned} &-2\,\mathrm{Ric}(g) = -2R^i_{ikj} = 2R^i_{kij} \\ &= g^{ip}\left(\partial_i \partial_k g_{pj} - \partial_i \partial_j g_{pk} + \partial_p \partial_j g_{ik} + \partial_p \partial_k g_{ij}\right). \end{aligned} \tag{26}$$

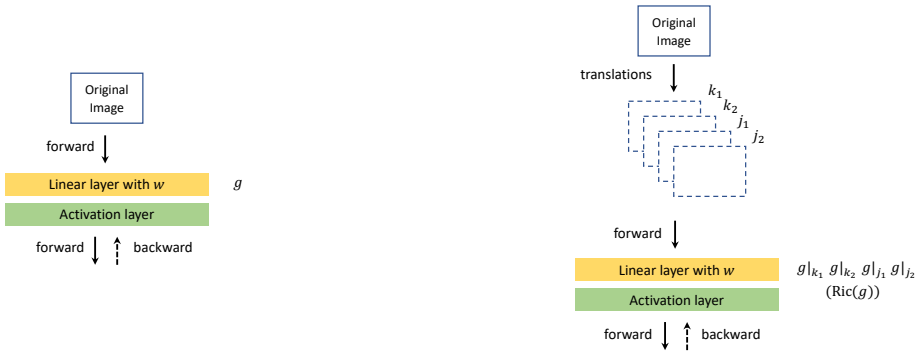
To establish a connection between the Ricci curvature and neural networks, we introduce a method for calculating the Ricci curvature such that the selection of coordinate systems is linked to input transformations. When the Ricci curvature is equal to zero, it implies that different input transformations will not induce variations in the parameters.

Inspired by prior work (Kaul and Lall, 2019), we interpret the terms $\partial_i$ and $\partial_p$ as changes representing translation and rotation of each input, respectively. Typically, data

augmentation in real-world applications like image classification tasks (He et al., 2016; Shorten and Khoshgoftaar, 2019) does not involve rotation. For the sake of fairness in ablation studies, we focus on translation by discarding the index $p$, i.e., $\partial_p(\partial_j g_{ik} + \partial_k g_{ij}) = 0$. When considering either translation or rotation, $g^{ip}$ degenerates into $\delta^{ip}$ (the identity matrix). Consequently, $\partial_i \partial_k g$ and $\partial_i \partial_j g$ can be treated as changes representing row and column transformations of the input data w.r.t. the metric $g$, respectively. The Ricci curvature can be rewritten as:

$$-2\,\mathrm{Ric}(g) = \partial_i \partial_k g_{pj} - \partial_i \partial_j g_{pk}. \tag{27}$$

**Remark 19** *The selection of $i$ and $p$ (as well as $k$ and $j$) is arbitrary and can even be represented in other coordinate systems. Here, we provide a specific geometric meaning by considering the characteristic of the image classification task.*




(a) Metric structure ($g$) (b) Ricci curvature structure ($\mathrm{Ric}(g)$)

Figure 5: Upon feeding the original image into the neural network and performing a forward and backward pass on the linear layer to update the weights $\boldsymbol{w}$, we construct the metric structure $g(\boldsymbol{w})$ based on Section 5.1. Furthermore, we subject the original image to four distinct small translation transformations ($k_1$, $k_2$, $j_1$, and $j_2$) before inputting them into the neural network. By sequentially performing a forward and backward passes, we obtain four metric structures ($g|_{k_1}$, $g|_{k_2}$, $g|_{j_1}$, and $g|_{j_2}$) corresponding to these translations. The combination of these metrics allows us to characterize the Ricci curvature $\mathrm{Ric}(g)$.

As shown in Figure 5 and leveraging Equation (27), we express the Ricci curvature with coordinate systems using a difference equation:

$$-2\,\mathrm{Ric}(g) = \frac{g|_{k_1} - g|_{k_2}}{k_1 - k_2} - \frac{g|_{j_1} - g|_{j_2}}{j_1 - j_2} \tag{28}$$

where we approximate partial derivatives with difference equations (Kaul and Lall, 2019), i.e., $\partial_i \partial_k g = (g|_{k_1} - g|_{k_2})/(k_1 - k_2)$ and $\partial_i \partial_j g = (g|_{j_1} - g|_{j_2})/(j_1 - j_2)$ corresponding to the

input translation dimensions $k$ and $j$, respectively. Here, $g|_{k_1}$, $g|_{k_2}$, $g|_{j_1}$, and $g|_{j_2}$ represent four metric structures under different small translation transformations $k_1$, $k_2$, $j_1$, and $j_2$, respectively. In general, $(k_1 - k_2)$ and $(j_1 - j_2)$ denote translations of fewer than 4 pixels, aligning with common data augmentation practices (He et al., 2016).

### 6.2 Existence of Discrete Ricci Flow in Neural Networks

Recall that we considered the Ricci-DeTurck flow instead of the Ricci flow, as the solution of the Ricci flow does not always exist, as discussed in Section 2.3. Assuming that the solution of the Ricci flow exists in neural networks, we can utilize the Ricci flow to exponentially decay the metric perturbation, as explained in Section 4.3.

In terms of the Ricci flow equation, we have previously examined the right-hand side, namely, the Ricci curvature tensor. Now, we define the equivalent form of the left-hand side of the Ricci flow using a difference equation:

$$\frac{\partial}{\partial t} g(t) := g(t+1) - g(t), \tag{29}$$

which represents the consecutive iterations in the training process, where $t \in \{0, 1, \cdots, T-1\}$ is a uniform partition of the interval $[0, T]$, with $T$ being the total number of iterations. As the number of iterations $T$ approaches infinity, the formula above holds.

Combining Equation (28) and Equation (29) in neural networks, we present the discrete Ricci flow as a difference equation:

$$\begin{aligned} g(t+1)|_{k_1} - g(t)|_{k_1} &= \frac{g(t)|_{k_1} - g(t)|_{k_2}}{k_1 - k_2} - \frac{g(t)|_{j_1} - g(t)|_{j_2}}{j_1 - j_2} \\ g(0)|_{k_1} &= \delta - \tanh(\tau \boldsymbol{w}) \tanh(\tau \boldsymbol{w})^\top \end{aligned} \tag{30}$$

To ensure the existence of the solution of the discrete Ricci flow, we achieve this goal by adding a regularization term to the loss function, constraining the discrete Ricci flow in DNNs. Following Equation (30), we present the regularization term:

$$N = \left\| g(t+1)|_{k_1} - g(t)|_{k_1} - \frac{g(t)|_{k_1} - g(t)|_{k_2}}{k_1 - k_2} + \frac{g(t)|_{j_1} - g(t)|_{j_2}}{j_1 - j_2} \right\|_{L^2}^2, \tag{31}$$

where $g(t)$ is $\epsilon$-close to the LNE metric $g_0$ based on Definition 20. In other words, $g(t)$ is the LNE metric with perturbations.

**Definition 20** *(Sheridan and Rubinstein, 2006) Let $g(t)$ be the metrics on the LNE manifold. For $\epsilon > 0$, $\mathcal{B}_{L^2}(g_0, \epsilon)$ is the $\epsilon$-ball with respect to the $L^2$-norm induced by $g_0$ and centred at $g_0$, where any metric $g(t) \in \mathcal{B}_{L^2}(g_0, \epsilon)$ is $\epsilon$-close to $g_0$ if*

$$(1+\epsilon)^{-1} g_0 \leq g(t) \leq (1+\epsilon) g_0$$

*in the sense of matrices.*

By constraining the regularization term $N$ in DNNs, the solution of the discrete Ricci flow exists when $N \to 0$. Simultaneously, the metric perturbation exponentially converges ($g(t) \to g_0$) as the discrete Ricci flow evolves.

**Algorithm 2** An algorithm for training our RF-DNNs in the LNE manifold. We introduce a parameter $\alpha$ to balance the regularization and ensure the existence of the solution for the discrete Ricci flow. For brevity, we omit the normalization operation (Ioffe and Szegedy, 2015; Ba et al., 2016).

**Input:** A minibatch of inputs and targets ($\boldsymbol{x} = \boldsymbol{a_0}, \boldsymbol{y}$), $\boldsymbol{\xi}$ mapped to $(\boldsymbol{W}_1, \boldsymbol{W}_2, \ldots, \boldsymbol{W}_l)$, $\hat{\boldsymbol{\xi}}$ mapped to $\left(\hat{\boldsymbol{W}}_1, \hat{\boldsymbol{W}}_2, \ldots, \hat{\boldsymbol{W}}_l\right)$, a nonlinear function $f$, a constant factor $\tau$ and a learning rate $\eta$.

**Output:** The updated discretized parameters $\hat{\boldsymbol{\xi}}$.

1: {Forward propagation}
2: **for** $i = 1; i \leq l; i++$ **do**
3: Compute $\hat{\boldsymbol{W}}_i = Q(\boldsymbol{W}_i)$;
4: Compute $\boldsymbol{s}_i = \hat{\boldsymbol{W}}_i \hat{\boldsymbol{a}}_{i-1}$;
5: Compute $\hat{\boldsymbol{a}}_i = Q\left(f(\boldsymbol{s}_i)\right)$;
6: **end for**
7: Compute the regularization term $N$ based on Equation (31);
8: {Loss derivative}
9: Compute $L = L(\boldsymbol{y}, \boldsymbol{z}) + \alpha \cdot N$;
10: Compute $\partial_{\boldsymbol{a}_l} L = \frac{\partial L(\boldsymbol{y}, \boldsymbol{z})}{\partial \boldsymbol{z}}\big|_{\boldsymbol{z}=\hat{\boldsymbol{a}}_l}$;
11: {Backward propagation}
12: **for** $i = l; i \geq 1; i--$ **do**
13: Compute $\partial_{\boldsymbol{s}_i} L = \partial_{\boldsymbol{a}_i} L \odot f'(\boldsymbol{s}_i)$;
14: Compute $\partial_{\hat{\boldsymbol{W}}_i} L = \left(\nabla_{\boldsymbol{s}_i} L\right) \hat{\boldsymbol{a}}_{i-1}^{\top}$;
15: Compute $\tilde{\partial}_{\boldsymbol{W}_i} L = g_{\boldsymbol{W}_i}^{-1}(t) \partial_{\hat{\boldsymbol{W}}_i} L$ based on Equation (32);
16: Compute $\partial_{\hat{\boldsymbol{a}}_{i-1}} L = \hat{\boldsymbol{W}}_i^{\top} \left(\partial_{\boldsymbol{s}_i} L\right)$;
17: **end for**
18: {The parameters update}
19: **for** $i = l; i \geq 1; i--$ **do**
20: Update $\boldsymbol{W}_i \leftarrow \boldsymbol{W}_i - \eta \cdot \tilde{\partial}_{\boldsymbol{W}_i} L$;
21: **end for**
22: Update $\hat{\boldsymbol{\xi}} = \left[\mathrm{vec}\left(\hat{\boldsymbol{W}}_1\right)^{\top}, \mathrm{vec}\left(\hat{\boldsymbol{W}}_2\right)^{\top}, \ldots, \mathrm{vec}\left(\hat{\boldsymbol{W}}_l\right)^{\top}\right]^{\top}$;

### 6.3 Algorithm Design

By imposing constraints on the discrete Ricci Flow in layer-by-layer LNE manifold, we can effectively address the problem of gradient mismatch. Given that the background is the LNE manifold, we can construct the satisfied gradient based on Equation (23). Note that, at this point, the metric becomes time-dependent under the Ricci flow, i.e., $g_{\boldsymbol{w}}(t)$. And we obtain the gradient under the discrete Ricci flow as follows:

$$\tilde{\partial}_{\boldsymbol{w}} L = g_{\boldsymbol{w}}^{-1}(t) \partial_{Q(\boldsymbol{w})}. \tag{32}$$

The overall process is shown in Algorithm 2. Compared with Algorithm 1, we have introduced Line 7 and Line 15. In Line 7, the regularization term is calculated to ensure the

existence of the solution for the discrete Ricci flow. On the other hand, in Line 15, the gradient is computed in the LNE manifold under the discrete Ricci flow. This is in contrast to Algorithm 1, which only calculates the gradient in the LNE manifold with perturbations. Applying the Ricci flow indicates that the LNE manifold at this point is dynamic and anti-perturbative.

**Remark 21** *In addition to using discretized weights and activations, DNNs need to store non-discretized weights and activations for gradient updates. It is important to note that the gradients of a DNN are non-discretized.*

### 6.4 Complexity Analysis

Based on Algorithm 2, it is evident that the forward time complexity is approximately $\mathcal{O}(n^2)$, where the time complexities of Line 4 and Line 5 are about $\mathcal{O}(n^2)$ and $\mathcal{O}(n)$, respectively. In the backward pass, the time complexity of Line 13 is around $\mathcal{O}(n)$. Consequently, the time complexities of computing the gradients w.r.t. the weights (Line 15) and activations (Line 17) are both approximately $\mathcal{O}(n^2)$. Therefore, the backward time complexity is roughly $\mathcal{O}(2n^2)$. For the training process of a neural network, its total complexity is $\mathcal{O}(n^2)$.

Since the computation of the Ricci curvature involves four different translations of input data w.r.t. the metric, its time complexity is about $\mathcal{O}(n^2)$. In this manner, the updated weights are only used to calculate the constraints of the discrete Ricci flow, and the final weights can be obtained by a subsequent backward pass. The time complexity of Line 16 is $\mathcal{O}(n^2)$ when we use the weak approximation to calculate the gradient. Thus, the total complexity of RF-DNN remains $\mathcal{O}(n^2)$, which is consistent with that of a neural network.

## 7. Experiments

In this section, we conduct ablation studies to compare our RF-DNN[11] trained from scratch with other STE methods. Additionally, when evaluating the performance of the RF-DNN with a pre-trained model, we compare it with several representative training-based methods on classification benchmark datasets. All experiments are implemented in Python using PyTorch (Paszke et al., 2019). The hardware environment includes an Intel(R) Xeon(R) Silver 4214 CPU(2.20 GHz), GeForce GTX 2080Ti GPU, and 128GB RAM.

### 7.1 Experimental Settings

The two datasets used in our experiments are introduced as follows.

**CIFAR datasets:** There are two CIFAR benchmarks (Krizhevsky et al., 2009), each consisting of natural color images with $32 \times 32$ pixels. Both datasets comprise 50k training images, 10k test images, and a validation set of 5k images selected from the training set. CIFAR-10 is organized into 10 classes, while CIFAR-100 has 100 classes. We apply a standard data augmentation scheme (random corner cropping and random flipping), widely used for these two datasets. Images are normalized during preprocessing using the means and standard deviations of the channels.

11. For convenient gradient calculation, we utilize the weak approximation of the inverse of the LNE metric in all experiments.

**ImageNet dataset:** The ImageNet benchmark (Russakovsky et al., 2015) consists of 1.2 million high-resolution natural images, with a validation set containing 50k images. These images are organized into 1000 object categories for training and re resized to 224 $\times$ 224 pixels before fed into the network. In the subsequent experiments, we report our single-crop evaluation results using top-1 and top-5 accuracies.

We specify the discrete function, the composition of which significantly influences the performance and computation of DNNs. Specifically, the discrete function can simplify calculations, which vary depending on different discrete values, such as fixed-point multiplication, SHIFT operation (Elhoushi et al., 2019), and XNOR operation (Rastegari et al., 2016), etc.

We denote $Q^1$ as the 1-bit discrete function:

$$Q^1(\cdot) = \text{sign}(\cdot) = \{-1, +1\}. \tag{33}$$

The $k$-bit, for $k > 1$, discrete function is denoted as $Q^k$:

$$Q^{k>1}(\cdot) = \frac{2}{2^k - 1} \text{round}\left[(2^k - 1)\left(\frac{\cdot}{2\max|\cdot|} + \frac{1}{2}\right)\right] - 1 \tag{34}$$

where round$[\cdot]$ is the rounding function and max$|\cdot|$ refers to calculating the absolute value of the input first, and then finding its maximum value. In this way, a DNN using the discrete function $Q^1(\cdot)$ can be computed with the XNOR operation, while a DNN using the discrete function $Q^{k>1}(\cdot)$ can be computed with fixed-point multiplication.

### 7.2 Ablation Studies with STE Methods

To showcase the superiority of RF-DNN in addressing the problem of gradient mismatch, we compare it with three other methods by training from scratch. In Table 1, Table 2, and Table 3, we mark $\{-1, +1\}$ in '**Forward**' to indicate that the weights are binarized using Equation (33), i.e., $-1$ or $+1$, in the forward pass of DNNs. In the backward pass, the methods (Dorefa (Zhou et al., 2016), MultiFCG (Chen et al., 2019), and FCGrad (Chen et al., 2019)) use different approximated gradients to update the weights. Here, we apply different ResNet models (He et al., 2016) for ablation studies.

Batch normalization with a batch size of 128 is employed in the learning strategy, and Nesterov momentum of 0.9 (Dozat, 2016) is used in SGD optimization. For CIFAR, we set the total training epochs to 200 and a weight decay of 0.0005. The learning rate is reduced by a factor of 10 at epoch 80, 150, and 190, starting with an initial value of 0.1. For ImageNet, we set the total training epochs to 100 and use a cosine annealing schedule for the learning rate of each parameter group with a weight decay of 0.0001. All experiments are conducted 5 times, and the statistics of the test accuracies from the last 10/5 epochs are reported for a fair comparison. Hence, we evaluate the accuracy performance in terms of (mean $\pm$ std). Note that we perform standard data augmentation and pre-processing on CIFAR and ImageNet datasets.

In Table 1, Table 2, and Table 3, we use the same the discrete function $Q^1(\cdot)$, parameter settings, and optimizer for fairness in the forward pass. The only difference is the gradient in the backward propagation. The performance across various models and datasets demonstrates that RF-DNN exhibits significant improvement over other STE methods. The

Table 1: The experimental results on CIFAR10 with ResNet20/32/44. The accuracy of full-precision (FP) baseline is reported by (Chen et al., 2019).

| Network | Forward | Backward | Test Acc (%) | FP Acc (%) |
|---|---|---|---|---|
| ResNet20 | {−1,+1} | Dorefa | 88.28±0.81 | 91.50 |
| | | MultiFCG | 88.94±0.46 | |
| | | RF-DNN | **89.83±0.23** | |
| ResNet32 | {−1,+1} | Dorefa | 90.23±0.63 | 92.13 |
| | | MultiFCG | 89.63±0.38 | |
| | | RF-DNN | **90.75±0.19** | |
| ResNet44 | {−1,+1} | Dorefa | 90.71±0.58 | 93.56 |
| | | MultiFCG | 90.54±0.21 | |
| | | RF-DNN | **91.63±0.11** | |

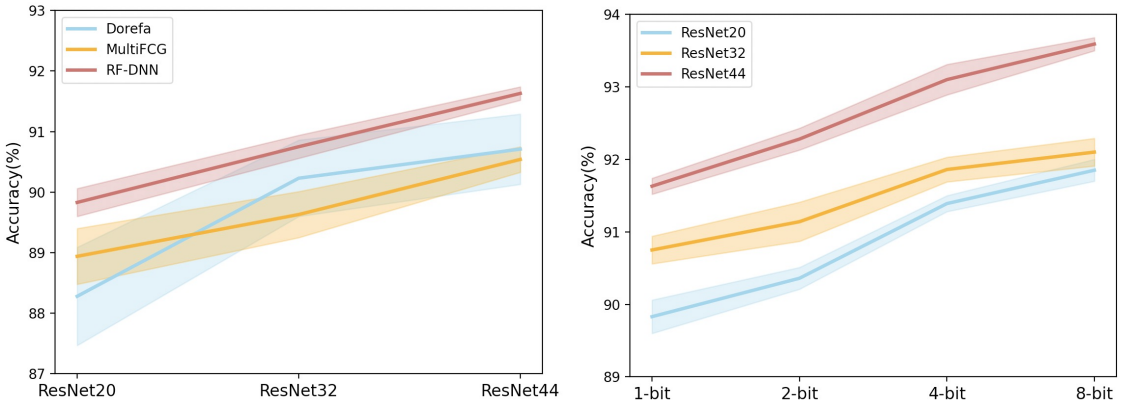
Table 2: The experimental results on CIFAR100 with ResNet56/110. The accuracy of full-precision (FP) baseline is reported by (Chen et al., 2019).

| Network | Forward | Backward | Test Acc (%) | FP Acc (%) |
|---|---|---|---|---|
| ResNet56 | {−1,+1} | Dorefa | 66.71±2.32 | 71.22 |
| | | MultiFCG | 66.58±0.37 | |
| | | FCGrad | 66.56±0.35 | |
| | | RF-DNN | **68.56±0.32** | |
| ResNet110 | {−1,+1} | Dorefa | 68.15±0.50 | 72.54 |
| | | MultiFCG | 68.27±0.14 | |
| | | FCGrad | 68.74±0.36 | |
| | | RF-DNN | **69.20±0.28** | |

average results of multiple experiments surpass those of other methods, which likely benefit from the alleviation of the gradient mismatch, making the loss function of DNNs more fully descended. Additionally, the minor variances indicate that our training method is relatively stable such that confirming our point of view.

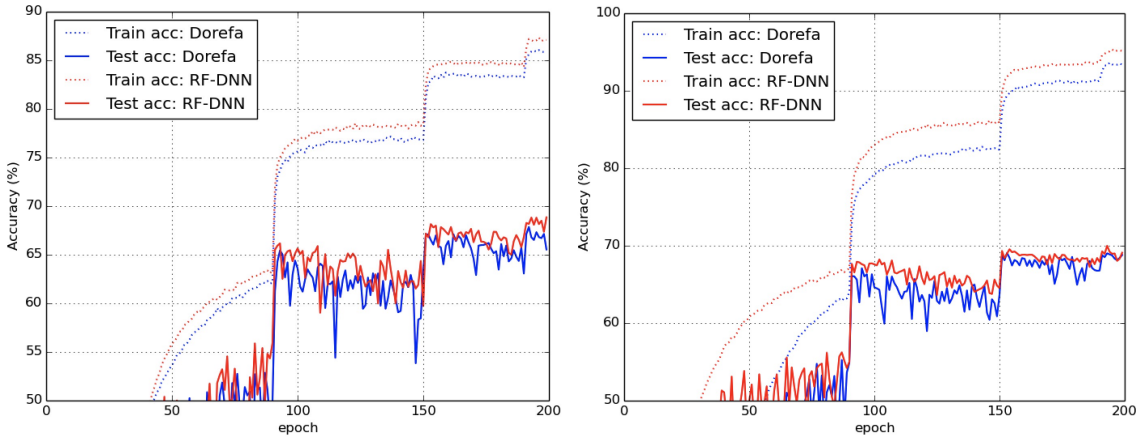
### 7.3 Convergence and Stability Analysis

Since standard deviations can reflect the convergence and stability of training to a certain extent, we visualize the data from Table 1 in Figure 6(a). Intuitively, when compared to Dorefa and MultiFCG, our proposed RF-DNN better alleviates perturbations caused by gradient mismatch, leading to more stable performance. Furthermore, we present the accu-




(a) Comparison of different methods (b) Comparison of different bit widths

Figure 6: Accuracy performance (mean ± std) for ResNet20/32/44 on CIFAR10. The lines and bars represent the mean and standard deviation of the results from different random seeds, respectively. (a) We compare RF-DNN with Dorefa and MultiFCG using 1-bit weight representation, also visualized in Table 1. (b) RF-DNN is presented with different bit-width weight representations. Note that a higher mean and lower deviation typically imply better convergence and stability.




(a) ResNet56 (b) ResNet110

Figure 7: Training and test curves of ResNet56/110 on CIFAR100 compared between Dorefa and RF-DNN. Intuitively, RF-DNN exhibits more stable training performance than Dorefa.

Table 3: The experimental results on ImageNet with ResNet18. The accuracy of full-precision (FP) baseline is reported by (Chen et al., 2019).

| Network | Forward | Backward | Test Top1/Top5 (%) | FP Top1/Top5 (%) |
|---|---|---|---|---|
| ResNet18 | {−1,+1} | Dorefa | 58.34±2.07/81.47±1.56 | 69.76/89.08 |
| | | MultiFCG | 59.47±0.02/82.41±0.01 | |
| | | FCGrad | 59.83±0.36/82.67±0.23 | |
| | | RF-DNN | **60.83±0.41/83.54±0.18** | |

racy performance of RF-DNN with different bit width weight representations in Figure 6(b). We observe fairly consistent stability across different bit widths and backbone models.

As depicted in Figure 7, RF-DNN achieves higher accuracies than Dorefa on CIFAR100 dataset, i.e., 1.25% higher on the training dataset with ResNet56, 1.85% higher on the test dataset with ResNet56, 1.97% higher on the training dataset with ResNet110, and 1.05% higher on the test dataset with ResNet110. Additionally, the fluctuation of the test curves in Figure 7 indicates that RF-DNN shows tremendous improvement compared to Dorefa in terms of training stability. From the training curve, our method significantly outperforms Dorefa. However, this superiority needs to be considered in conjunction with the test curve. The accuracy of our method is consistently higher than that of Dorefa in the test curve, thereby indicating an improvement in stability. The experimental results verify that our theoretical framework is an effective solution against gradient mismatch, further enhancing the training performance of DNNs.

### 7.4 Comparisons with Training-based Methods

Here, we compare RF-DNN with several state-of-the-art DNNs, such as DeepShift (Elhoushi et al., 2019), QN (Yang et al., 2019), ADMM (Leng et al., 2018), MetaQuant (Chen et al., 2019), INT8 (Zhu et al., 2020), SR+DR (Gysel et al., 2018), ELQ (Zhou et al., 2018), MD (Ajanthan et al., 2021), and RQ (Louizos et al., 2019), all under the same bit width using Equation (33) or Equation (34). Note that **W** and **A** represent the bit width of weights and activations, respectively, in Table 4. The experimental results demonstrate that RF-DNN outperforms other recent state-of-the-art training-based methods, which appears to be attributed to our effective solution for addressing gradient mismatch.

## 8. Conclusion and Future Work

Traditional discretized neural networks (DNNs) suggest that both weights and activations can only take low-precision discrete values, reducing the memory footprint compared to full-precision floating-point networks. However, training such networks becomes challenging due to the need to maintain discrete weights. Generally, the gradient w.r.t. discrete weights is approximated using the Straight-Through Estimator (STE), resulting in a *gradient mismatch* compared to the gradient w.r.t. continuous weights.

Table 4: The classification accuracy results on ImageNet are compared with other training-based methods, including AlexNet (Krizhevsky et al., 2012), ResNet18, ResNet50 and MobileNet (Howard et al., 2017). Note that the accuracy of full-precision baseline is reported by Elhoushi et al. (2019).

| **Method** | **W** | **A** | **Top-1** | | **Top-5** | |
|---|---|---|---|---|---|---|
| | | | Accuracy | Gap | Accuracy | Gap |
| **AlexNet** (Original) | 32 | 32 | 56.52% | - | 79.07% | - |
| RF-DNN (ours) | 6 | 32 | **56.39%** | **−0.13%** | **78.78%** | **−0.29%** |
| DeepShift (Elhoushi et al., 2019) | 6 | 32 | 54.97% | −1.55% | 78.26% | −0.81% |
| **ResNet18** (Original) | 32 | 32 | 69.76% | - | 89.08% | - |
| RF-DNN (ours) | 1 | 32 | **67.05%** | **−2.71%** | **88.09%** | **−0.99%** |
| MD (Ajanthan et al., 2021) | 1 | 32 | 66.78% | −2.98% | 87.01% | −2.07% |
| ELQ (Zhou et al., 2018) | 1 | 32 | 66.21% | −3.55% | 86.43% | −2.65% |
| ADMM (Leng et al., 2018) | 1 | 32 | 64.80% | −4.96% | 86.20% | −2.88% |
| QN (Yang et al., 2019) | 1 | 32 | 66.50% | −3.26% | 87.30% | −1.78% |
| MetaQuant (Chen et al., 2019) | 1 | 32 | 63.44% | −6.32% | 84.77% | −4.31% |
| RF-DNN (ours) | 4 | 4 | **66.75%** | **−3.01%** | **87.02%** | **−2.06%** |
| RQ ST (Louizos et al., 2019) | 4 | 4 | 62.46% | −7.30% | 84.78% | −4.30% |
| **ResNet50** (Original) | 32 | 32 | 76.13% | - | 92.86% | - |
| RF-DNN (ours) | 8 | 8 | **76.07%** | **−0.06%** | **92.87%** | **+0.01%** |
| INT8 (Zhu et al., 2020) | 8 | 8 | 75.87% | −0.26% | - | - |
| **MobileNet** (Original) | 32 | 32 | 70.61% | - | 89.47% | - |
| RF-DNN (ours) | 5 | 5 | **61.32%** | **−9.29%** | **84.08%** | **−5.39%** |
| SR+DR (Gysel et al., 2018) | 5 | 5 | 59.39% | −11.22% | 82.35% | −7.12% |
| RQ ST (Louizos et al., 2019) | 5 | 5 | 56.85% | −13.76% | 80.35% | −9.12% |
| RF-DNN (ours) | 8 | 8 | **70.76%** | **+0.15%** | **89.54%** | **+0.07%** |
| RQ (Louizos et al., 2019) | 8 | 8 | 70.43% | −0.18% | 89.42% | −0.05% |

This paper introduces a novel analysis of the gradient mismatch phenomenon through the lens of duality theory. The mismatch is interpreted as metric perturbations in a Riemannian manifold. Theoretical insights, rooted in information geometry, lead to the construction of the LNE manifold for neural networks. This manifold forms the background to effectively address metric perturbations. The stability of LNE metrics with the $L^2$-norm perturbation under the Ricci-DeTurck flow is revealed, paving the way for practical introduction of the Ricci flow Discretized Neural Network (RF-DNN). The constraints of the discrete Ricci flow in the LNE manifold are used to alleviate metric perturbations, achieving an

exponential convergence rate and providing a compelling solution for DNNs. Experimental results demonstrate improvements in both the stability and performance of DNNs.

In this paper, information geometry plays a crucial role in combining geometric tool (Ricci flow) with neural networks. For future research, we aim to further explore the connection between neural networks and manifolds, leveraging geometric ideas to address practical challenges in deep learning.

## Acknowledgments

We thank all reviewers and the editor for excellent contributions.

## Appendix A. Differential Geometry

1. Riemann curvature tensor (Rm) is a (1,3)-tensor defined for a 1-form $\omega$:

$$R^l_{ijk}\omega_l = \nabla_i\nabla_j\omega_k - \nabla_j\nabla_i\omega_k$$

where the covariant derivative of $F$ satisfies

$$\nabla_p F^{j_1\dots j_l}_{i_1\dots i_k} = \partial_p F^{j_1\dots j_l}_{i_1\dots i_k} + \sum_{s=1}^{l} F^{j_1\dots q\dots j_l}_{i_1\dots i_k}\Gamma^{j_s}_{pq} - \sum_{s=1}^{k} F^{j_1\dots j_l}_{i_1\dots q\dots i_k}\Gamma^{q}_{pi_s}.$$

In particular, coordinate form of the Riemann curvature tensor is:

$$R^l_{ijk} = \partial_i\Gamma^l_{jk} - \partial_j\Gamma^l_{ik} + \Gamma^p_{jk}\Gamma^l_{ip} - \Gamma^p_{ik}\Gamma^l_{jp}.$$

2. Christoffel symbol in terms of an ordinary derivative operator is:

$$\Gamma^k_{ij} = \frac{1}{2}g^{kl}(\partial_i g_{jl} + \partial_j g_{il} - \partial_l g_{ij}).$$

3. Ricci curvature tensor (Ric) is a (0,2)-tensor:

$$R_{ij} = R^p_{pij}.$$

4. Scalar curvature is the trace of the Ricci curvature tensor:

$$R = g^{ij}R_{ij}.$$

5. Lie derivative of $F$ in the direction $\frac{d\varphi(t)}{dt}$:

$$\mathcal{L}_{\frac{d\varphi(t)}{dt}}F = \left(\frac{d}{dt}\varphi^*(t)F\right)_{t=0}$$

where $\varphi(t) : \mathcal{M} \to \mathcal{M}$ for $t \in (-\epsilon, \epsilon)$ is a time-dependent diffeomorphism of $\mathcal{M}$ to $\mathcal{M}$.

## Appendix B. Notation

For clarity of definitions in this paper, we list the important notations as shown in Table 5.

## Appendix C. Proof of the Ricci Flow

### C.1 Proof of Lemma 5

**Lemma 22** *The linearization of the Ricci curvature tensor is given by*

$$\mathcal{D}[\mathrm{Ric}](h)_{ij} = -\frac{1}{2}g^{pq}(\nabla_p\nabla_q h_{ij} + \nabla_i\nabla_j h_{pq} - \nabla_q\nabla_i h_{jp} - \nabla_q\nabla_j h_{ip}).$$

**Proof** Based on Appendix A, we have

$$\nabla_q\nabla_i h_{jp} = \nabla_i\nabla_q h_{jp} - R^r_{qij}h_{rp} - R^r_{qip}h_{jm}.$$

Table 5: Definitions of notations

| | | | |
|---|---|---|---|
| $\boldsymbol{W}_i$: | weight matrix for the $i$-th layer | $\hat{\boldsymbol{W}}_i$: | discretized weight matrix for the $i$-th layer |
| $\boldsymbol{w}$: | vectorized weights in each layer | $\hat{\boldsymbol{w}}$: | discretized vectorized weights in each layer |
| $\boldsymbol{a}_i$: | activation vector for the $i$-th layer | $\hat{\boldsymbol{a}}_i$: | discretized activation vector for the $i$-th layer |
| $\boldsymbol{\xi}$: | parameter vector | $\hat{\boldsymbol{\xi}}$: | discretized parameter vector |
| $Q^1$: | 1-bit discrete function | $Q^{k>1}$: | k-bit discrete function (over 1-bit) |
| $\delta$: | Euclidean metric (identity matrix) | $\Phi$: | convex function |
| $g_0$: | LNE metric under Ricci flow | $\bar{g}_0$: | LNE metric under Ricci-DeTurck flow |
| $g$ or $g(t)$: | the metrics under Ricci flow | $\bar{g}$ or $\bar{g}(t)$: | the metrics under Ricci-DeTurck flow |
| $g(0)$: | initial metric under Ricci flow | $\bar{g}(0)$: | initial metric under Ricci-DeTurck flow |
| $d(0)$: | the initial perturbation | $d(t)$: | the time-evolving perturbation |
| $D$: | divergence | $L$ or $\tilde{L}$: | loss function |
| $L_{g_0}$: | Lichnerowicz operator | $L^2$ or $L^\infty$: | norm |
| $\partial$: | partial derivative | $\nabla$: | covariant derivative |
| $\mathcal{L}$: | Lie derivative | $\Delta_{g_0}$: | the Laplacian |
| Rm: | Riemann curvature tensor | $f$: | nonlinear function |
| Ric: | Ricci curvature tensor | $\mathcal{D}$[Ric]: | the linearization of the Ricci curvature tensor |
| $\varphi^*$: | pullback | $\phi_*$: | pushforward |
| $B(x,r)$: | the ball with a radius $r$ and a point $x \in \mathcal{M}$ | $\mathcal{B}_{L^2}(\bar{g}_0, \epsilon)$: | the $\epsilon$-ball with respect to the $L^2$-norm induced by $\bar{g}_0$ and centred at $\bar{g}_0$ |

Combining with Lemma 22, we can obtain the deformation equation because of $\nabla g = 0$,

$$\begin{aligned}\mathcal{D}[-2\mathrm{Ric}](h)_{ij} =& g^{pq}\nabla_p\nabla_q h_{ij} + \nabla_i\left(\frac{1}{2}\nabla_j h_{pq} - \nabla_q h_{jp}\right) + \nabla_j\left(\frac{1}{2}\nabla_i h_{pq} - \nabla_q h_{ip}\right) + O(h_{ij})\\ =& g^{pq}\nabla_p\nabla_q h_{ij} + \nabla_i V_j + \nabla_j V_i + O(h_{ij}).\end{aligned}$$

The proof is completed. ■

### C.2 Description of the DeTurck Trick

Based on the chain rule for the Lie derivative in Appendix A, we can calculate

$$\begin{aligned}\frac{\partial}{\partial t}g(t) &= \frac{\partial\left(\varphi^*(t)\bar{g}(t)\right)}{\partial t} \\ &= \left(\frac{\partial\left(\varphi^*(t+\tau)\bar{g}(t+\tau)\right)}{\partial \tau}\right)_{\tau=0} \\ &= \left(\varphi^*(t)\frac{\partial \bar{g}(t+\tau)}{\partial \tau}\right)_{\tau=0} + \left(\frac{\partial\left(\varphi^*(t+\tau)\bar{g}(t)\right)}{\partial \tau}\right)_{\tau=0} \\ &= \varphi^*(t)\frac{\partial}{\partial t}\bar{g}(t) + \varphi^*(t)\mathcal{L}_{\frac{\partial \varphi(t)}{\partial t}}\bar{g}(t)\end{aligned}$$

where $\frac{\partial \varphi(t)}{\partial t}$ is equal to $V(t)$ (Sheridan and Rubinstein, 2006). With the help of Equation (5), we have the following expression for the pullback metric $g(t)$

$$\frac{\partial}{\partial t}g(t) = \varphi^*(t)\frac{\partial}{\partial t}\bar{g}(t) + \varphi^*(t)\mathcal{L}_{\frac{\partial \varphi(t)}{\partial t}}\bar{g}(t) = -2\,\mathrm{Ric}(\varphi^*(t)\bar{g}(t)) = -2\varphi^*(t)\,\mathrm{Ric}(\bar{g}(t)). \tag{35}$$

The diffeomorphism invariance of the Ricci curvature tensor is used in the last step. The above equation is equivalent to

$$\frac{\partial}{\partial t}\bar{g}(t) = -2\,\mathrm{Ric}(\bar{g}(t)) - \mathcal{L}_{\frac{\partial \varphi(t)}{\partial t}}\bar{g}(t).$$

Based on Definition 23, we further yield

$$\frac{\partial}{\partial t}\bar{g}(t) = -2\,\mathrm{Ric}(\bar{g}(t)) - \nabla_i V_j - \nabla_j V_i.$$

**Definition 23** *(Sheridan and Rubinstein, 2006) On a Riemannian manifold $(\mathcal{M}, g)$, we have*

$$(\mathcal{L}_X g)_{ij} = \nabla_i X_j + \nabla_j X_i,$$

*where $\nabla$ denotes the Levi-Civita connection of the metric $g$, for any vector field $X$.*

### C.3 Curvature Explosion at Singularity

In general, we present the behavior of Ricci flow in finite time and show that the evolution of the curvature is close to divergence. The core demonstration is followed with Theorem 27.

**Theorem 24** *(Sheridan and Rubinstein, 2006) Given a smooth Riemannian metric $g_0$ on a closed manifold $\mathcal{M}$, there exists a maximal time interval $[0, T)$ such that a solution $g(t)$ of the Ricci flow, with $g(0) = g_0$, exists and is smooth on $[0, T)$, and this solution is unique.*

**Theorem 25** *Let $\mathcal{M}$ be a closed manifold and $g(t)$ a smooth time-dependent metric on $\mathcal{M}$, defined for $t \in [0, T)$. If there exists a constant $C < \infty$ for all $x \in \mathcal{M}$ such that*

$$\int_0^T \left|\frac{\partial}{\partial t}g_x(t)\right|_{g(t)} dt \leq C, \tag{36}$$

*then the metrics $g(t)$ converge uniformly as $t$ approaches $T$ to a continuous metric $g(T)$ that is uniformly equivalent to $g(0)$ and satisfies*

$$e^{-C}g_x(0) \leq g_x(T) \leq e^{C}g_x(0). \tag{37}$$

**Proof** Considering any $x \in \mathcal{M}$, $t_0 \in [0, T)$, $V \in T_x\mathcal{M}$, we have

$$\begin{aligned}
\left|\log\left(\frac{g_x(t_0)(V,V)}{g_x(0)(V,V)}\right)\right| &= \left|\int_0^{t_0} \frac{\partial}{\partial t}\left[\log g_x(t)(V,V)\right] dt\right| \\
&= \left|\int_0^{t_0} \frac{\frac{\partial}{\partial t}g_x(t)(V,V)}{g_x(t)(V,V)} dt\right| \\
&\leq \int_0^{t_0} \left|\frac{\partial}{\partial t}g_x(t)\left(\frac{V}{|V|_{g(t)}}, \frac{V}{|V|_{g(t)}}\right)\right| dt \\
&\leq \int_0^{t_0} \left|\frac{\partial}{\partial t}g_x(t)\right|_{g(t)} dt \\
&\leq C.
\end{aligned}$$

By exponentiating both sides of the above inequality, we have

$$e^{-C}g_x(0)(V,V) \leq g_x(t_0)(V,V) \leq e^{C}g_x(0)(V,V).$$

This inequality can be rewritten as

$$e^{-C}g_x(0) \leq g_x(t_0)(V,V) \leq e^{C}g_x(0)(V,V)$$

because it holds for any $V$. Thus, the metrics $g(t)$ are uniformly equivalent to $g(0)$.

Consequently, we have the well-defined integral:

$$g_x(T) - g_x(0) = \int_0^T \frac{\partial}{\partial t}g_x(t)dt.$$

We can show that this integral is well-defined from two perspectives. Firstly, as long as the metrics are smooth, the integral exists. Secondly, the integral is absolutely integrable. Based on the norm inequality induced by $g(0)$, we can obtain

$$|g_x(T) - g_x(t)|_{g(0)} \leq \int_t^T \left|\frac{\partial}{\partial t}g_x(t)\right|_{g(0)} dt.$$

For each $x \in \mathcal{M}$, the above integral will approach zero as $t$ approaches $T$. Since $\mathcal{M}$ is compact, the metrics $g(t)$ converge uniformly to a continuous metric $g(T)$ which is uniformly equivalent to $g(0)$ on $\mathcal{M}$. Moreover, we can show that

$$e^{-C}g_x(0) \leq g_x(T) \leq e^{C}g_x(0).$$

The proof is completed. ■

**Corollary 26** *Let $(\mathcal{M}, g(t))$ be a solution of the Ricci flow on a closed manifold. If $|\operatorname{Rm}|_{g(t)}$ is bounded on a finite time $[0, T)$, then $g(t)$ converges uniformly as $t$ approaches $T$ to a continuous metric $g(T)$ which is uniformly equivalent to $g(0)$.*

**Proof** The bound on $|\operatorname{Rm}|_{g(t)}$ implies one on $|\operatorname{Ric}|_{g(t)}$. Based on Equation (5), we can extend the bound on $|\frac{\partial}{\partial t} g(t)|_{g(t)}$. Therefore, we obtain an integral of a bounded quantity over a finite interval is also bounded, by Theorem 25. The proof is completed. ■

**Theorem 27** *If $g_0$ is a smooth metric on a compact manifold $\mathcal{M}$, the Ricci flow with $g(0) = g_0$ has a unique solution $g(t)$ on a maximal time interval $t \in [0, T)$. If $T < \infty$, then*

$$\lim_{t \to T} \left( \sup_{x \in \mathcal{M}} |\operatorname{Rm}_x(t)| \right) = \infty. \tag{38}$$

**Proof** For a contradiction, we assume that $|\operatorname{Rm}_x(t)|$ is bounded by a constant. It follows from Corollary 26 that the metrics $g(t)$ converges smoothly to a smooth metric $g(T)$. Based on Theorem 24, it is possible to find a solution to the Ricci flow on $t \in [0, T)$, as the smooth metric $g(T)$ is uniformly equivalent to the initial metric $g(0)$.

Hence, we can extend the solution of the Ricci flow after the time point $t = T$, which contradicts the choice of $T$ as the maximal time for the existence of the Ricci flow on $[0, T)$. In other words, $|\operatorname{Rm}_x(t)|$ is unbounded. The proof is completed. ■

As approaching the singular time $T$, the Riemann curvature $|\operatorname{Rm}|_{g(t)}$ becomes no longer convergent and tends to explode.

## Appendix D. Proof of All Time Convergence in LNE Manifolds

### D.1 Finite Time Stability

We first prove the finite-time stability of LNE manifolds.

**Lemma 28** *(Bamler, 2010, 2011) Let $(\mathcal{M}^n, \bar{g}_0)$ be a complete Ricci-flat $n$-manifold. If $\bar{g}(0)$ is a metric satisfying $\|\bar{g}(0) - \bar{g}_0\|_{L^\infty} < \epsilon$ where $\epsilon > 0$, then there exists a constant $C < \infty$ and a unique Ricci–DeTurck flow $\bar{g}(t)$ that satisfies*

$$\|\bar{g}(t) - \bar{g}_0\|_{L^\infty} < C \|\bar{g}(0) - \bar{g}_0\|_{L^\infty} < C \cdot \epsilon. \tag{39}$$

**Corollary 29** *Let $(\mathcal{M}^n, \bar{g}_0)$ be the LNE $n$-manifold. For a Ricci–DeTurck flow $\bar{g}(t)$ on a maximal time interval $t \in [0, T)$ and $k \in \mathbb{N}$, there exists constants $C_k = C_k(\bar{g}_0, T)$ such that*

$$\|\nabla^k d(t)\|_{L^2} \leq C_k \cdot t^{-k/2} \tag{40}$$

*where $d(t) = \bar{g}(t) - \bar{g}_0$ is the time-evolving perturbation.*

**Proof** When Lemma 28 is satisfied in a finite time, based on (Deruelle and Kröncke, 2021), the Ricci-DeTurck flow with the LNE metric w.r.t. the $L^2$-norm perturbation exists. The proof is completed. ■

Corollary 29 guarantees the finite time existence of the Ricci-DeTurck flow w.r.t. $L^2$-norm perturbations and provides the necessary premise for proving its all time convergence.

### D.2 All time Stability

Then, we prove the all-time stability of LNE manifolds. By rewriting the Ricci-DeTurck flow (20) as an evolution of the difference $d(t) := \bar{g}(t) - \bar{g}_0$, we have

$$\begin{aligned}\frac{\partial}{\partial t}d(t) &= \frac{\partial}{\partial t}\bar{g}(t) = -2\,\mathrm{Ric}(\bar{g}(t)) + 2\,\mathrm{Ric}(\bar{g}_0) + \mathcal{L}_{\frac{\partial\varphi'(t)}{\partial t}}\bar{g}_0 - \mathcal{L}_{\frac{\partial\varphi(t)}{\partial t}}\bar{g}(t)\\ &= \Delta d(t) + \mathrm{Rm} * d(t) + F_{\bar{g}^{-1}} * \nabla^{\bar{g}_0} d(t) * \nabla^{\bar{g}_0} d(t) + \nabla^{\bar{g}_0}\left(G_{\Gamma(\bar{g}_0)} * d(t) * \nabla^{\bar{g}_0} d(t)\right),\end{aligned} \tag{41}$$

where the tensors $F$ and $G$ depend on $\bar{g}^{-1}$ and $\Gamma(\bar{g}_0)$. Note that $\bar{g}_0$ is the LNE metric which satisfies the above formula.

In the follwing, we denote $\|\cdot\|_{L^2}$ or $\|\cdot\|_{L^\infty}$ as the $L^2$-norm or $L^\infty$-norm w.r.t. the LNE metric $\bar{g}_0$, and mark generic constants as $C$ or $C_1$.

**Lemma 30** *Let $\bar{g}(t)$ be a Ricci–DeTurck flow on a maximal time interval $t \in (0, T)$ in an $L^2$-neighbourhood of $\bar{g}_0$. We have the following estimate:*

$$\left\|\frac{\partial}{\partial t}d_0(t)\right\|_{L^2} \leq C\left\|\nabla^{\bar{g}_0(t)}\left(d(t) - d_0(t)\right)\right\|_{L^2}^2. \tag{42}$$

**Proof** According to the Hardy inequality (Minerbe, 2009), we have the same proofs by referring the details (Deruelle and Kröncke, 2021). ■

To establish the all time stability of LNE metrics under Ricci–DeTurck flow, we need to construct $\bar{g}_0(t)$ as a family of Ricci-flat reference metrics with $\frac{\partial}{\partial t}\bar{g}_0(t) = O((\bar{g}(t) - \bar{g}_0(t))^2)$. Let

$$\mathcal{F} = \left\{\bar{g}(t) \in \mathcal{M}^n \;\middle|\; 2\,\mathrm{Ric}(\bar{g}(t)) + \mathcal{L}_{\frac{\partial\varphi(t)}{\partial t}}\bar{g}(t) = 0\right\}$$

be the set of stationary points under the Ricci-DeTurck flow. Then, we establish a manifold via an $L^2$-neighbourhood $\mathcal{U}$ of integral $\bar{g}_0$ in the space of metrics:

$$\tilde{\mathcal{F}} = \mathcal{F} \cap \mathcal{U}. \tag{43}$$

For all $\bar{g} \in \tilde{\mathcal{F}}$, the terms $\mathrm{Ric}(\bar{g}(t)) = 0$ and $\mathcal{L}_{\frac{\partial\varphi(t)}{\partial t}}\bar{g}(t) = 0$ hold individually, as established in the previous work (Deruelle and Kröncke, 2021).

**Theorem 31** *Let $(\mathcal{M}^n, \bar{g}_0)$ be the LNE $n$-manifold which is linearly stable and integrable. Then, there exists a constant $\alpha_{\bar{g}_0}$ satisfying*

$$(\Delta d(t) + \mathrm{Rm}(\bar{g}_0) * d(t), d(t))_{L^2} \leq -\alpha_{\bar{g}_0}\left\|\nabla^{\bar{g}_0} d(t)\right\|_{L^2}^2 \tag{44}$$

*for all $\bar{g}(t) \in \tilde{\mathcal{F}}$ whose definition is given in Equation (43).*

**Proof** The similar proofs can be found in (Devyver, 2014) with some minor modifications. Due to the linear stability requirement of LNE manifolds in Definition 13 and Definition 14, $-L_{\bar{g}_0}$ is non-negative. Then there exists a positive constant $\alpha_{\bar{g}_0}$ satisfying

$$\alpha_{\bar{g}_0}\left(-\Delta d(t), d(t)\right)_{L^2} \leq \left(-\Delta d(t) - \mathrm{Rm}(\bar{g}_0) * d(t), d(t)\right)_{L^2}.$$

By Taylor expansion, we repeatedly use elliptic regularity and Sobolev embedding (Pacini, 2010) to obtain the estimate. The proof is completed. ■

**Corollary 32** *Let $(\mathcal{M}^n, \bar{g}_0)$ be the LNE $n$-manifold which is integrable. For a Ricci–DeTurck flow $\bar{g}(t)$ on a maximal time interval $t \in [0, T]$, if it satisfies $\|\bar{g}(t) - \bar{g}_0\|_{L^\infty} < \epsilon$ where $\epsilon > 0$, then there exists a constant $C < \infty$ for $t \in [0, T]$ such that the evolution inequality satisfies*

$$\|d(t) - d_0(t)\|_{L^2}^2 \geq C \int_0^T \left\|\nabla^{\bar{g}_0(t)}\left(d(t) - d_0(t)\right)\right\|_{L^2}^2 \mathrm{d}t. \tag{45}$$

**Proof** Based on Equation (41), we know

$$\begin{aligned}
\frac{\partial}{\partial t}(d(t) - d_0) =& \Delta(d(t) - d_0) + \mathrm{Rm} * (d(t) - d_0) \\
&+ F_{\bar{g}^{-1}} * \nabla^{\bar{g}_0}(d(t) - d_0) * \nabla^{\bar{g}_0}(d(t) - d_0) \\
&+ \nabla^{\bar{g}_0}\left(G_{\Gamma(\bar{g}_0)} * (d(t) - d_0) * \nabla^{\bar{g}_0}(d(t) - d_0)\right).
\end{aligned}$$

Followed by Lemma 30 and Theorem 31, we further obtain

$$\begin{aligned}
\frac{\partial}{\partial t}\|d(t) - d_0\|_{L^2}^2 =& 2\left(\Delta(d(t) - d_0) + \mathrm{Rm} * (d(t) - d_0), d(t) - d_0\right)_{L^2} \\
&+ \left(F_{\bar{g}^{-1}} * \nabla^{\bar{g}_0}(d(t) - d_0) * \nabla^{\bar{g}_0}(d(t) - d_0), d(t) - d_0\right)_{L^2} \\
&+ \left(\nabla^{\bar{g}_0}\left(G_{\Gamma(\bar{g}_0)} * (d(t) - d_0) * \nabla^{\bar{g}_0}(d(t) - d_0)\right), d(t) - d_0\right)_{L^2} \\
&+ \left(d(t) - d_0, \frac{\partial}{\partial t} d_0(t)\right)_{L^2} + \int_{\mathcal{M}} (d(t) - d_0) * (d(t) - d_0) * \frac{\partial}{\partial t} d_0(t) \mathrm{d}\mu \\
\leq& -2\alpha_{\bar{g}_0}\left\|\nabla^{\bar{g}_0}\left(d(t) - d_0\right)\right\|_{L^2}^2 \\
&+ C\left\|(d(t) - d_0)\right\|_{L^\infty}\left\|\nabla^{\bar{g}_0}\left(d(t) - d_0\right)\right\|_{L^2}^2 \\
&+ \left\|\frac{\partial}{\partial t} d_0(t)\right\|_{L^2}\|d(t) - d_0\|_{L^2} \\
\leq& \left(-2\alpha_{\bar{g}_0} + C \cdot \epsilon\right)\left\|\nabla^{\bar{g}_0}\left(d(t) - d_0\right)\right\|_{L^2}^2.
\end{aligned}$$

Let $\epsilon$ be a small enough constant that $-2\alpha_{\bar{g}_0} + C \cdot \epsilon < 0$ holds, we can find

$$\frac{\partial}{\partial t}\|d(t) - d_0\|_{L^2}^2 \leq -C\left\|\nabla^{\bar{g}_0}\left(d(t) - d_0\right)\right\|_{L^2}^2$$

holds. The proof is completed. ■

### D.3 Proof of Theorem 15

By Lemma 28, we have a constant $\epsilon_2 > 0$ such that $d(t) \in \mathcal{B}_{L^2}(0, \epsilon_2)$ holds. By Lemma 30 (in the second step) and Corollary 32 (in the third step), we can obtain

$$\begin{aligned}\|d_0(T)\|_{L^2} &\leq C \int_1^T \left\| \frac{\partial}{\partial t} d_0(t) \right\|_{L^2} \mathrm{d}t \\ &\leq C \int_1^T \left\| \nabla^{\bar{g}_0} \left( d(t) - d_0(t) \right) \right\|_{L^2}^2 \mathrm{d}t \\ &\leq C \left\| d(1) - d_0(1) \right\|_{L^2}^2 \leq C \|d(1)\|_{L^2}^2 \leq C \cdot (\epsilon_2)^2 .\end{aligned}$$

Furthermore, we can obtain from the above formulas

$$\|d(T) - d_0(T)\|_{L^2} \leq \|d(1) - d_0(1)\|_{L^2} \leq C \cdot \epsilon_2.$$

By the triangle inequality, we get

$$\|d(T)\|_{L^2} \leq C \cdot (\epsilon_2)^2 + C \cdot \epsilon_2.$$

Followed by Corollary 29 and Lemma 30, $T$ should be pushed further outward, i.e.,

$$\lim_{t \to +\infty} \sup \left\| \frac{\partial}{\partial t} d_0(t) \right\|_{L^2} \leq \lim_{t \to +\infty} \sup \left\| \nabla^{\bar{g}_0} \left( d(t) - d_0(t) \right) \right\|_{L^2}^2 = 0.$$

Thus, as $t$ approaches $+\infty$ based on the elliptic regularity, $\bar{g}(t)$ will converge to $\bar{g}(\infty) = \bar{g}_0 + d_0(\infty)$. In other words, $d(t) - d_0(t)$ will converge to 0 as $t$ approaches $+\infty$ w.r.t. all Sobolev norms (Minerbe, 2009),

$$\lim_{t \to +\infty} \|d(t) - d_0(t)\|_{L^2} \leq \lim_{t \to +\infty} C \left\| \nabla^{\bar{g}_0} \left( d(t) - d_0(t) \right) \right\|_{L^2} = 0.$$

Any Ricci-DeTurck flow that starts close to the LNE metric exists for all time, and it will converge to the LNE metric, as discussed in (Deruelle and Kröncke, 2021).

## Appendix E. Proof of the Information Geometry

### E.1 Proof of Theorem 10

The LNE divergence can be defined between two nearby points $\boldsymbol{\xi}$ and $\boldsymbol{\xi}'$, where the first derivative of the LNE divergence w.r.t. $\boldsymbol{\xi}'$ is:

$$\begin{aligned}&\partial_{\boldsymbol{\xi}'} D_{LNE}[\boldsymbol{\xi}' : \boldsymbol{\xi}] \\ &= \sum_i \left[ \partial_{\boldsymbol{\xi}'} \frac{1}{\tau^2} \log\cosh(\tau \xi_i') - \partial_{\boldsymbol{\xi}'} \frac{1}{\tau^2} \log\cosh(\tau \xi_i) - \frac{1}{\tau} \partial_{\boldsymbol{\xi}'} (\xi_i' - \xi_i) \tanh(\tau \xi_i) \right] \\ &= \sum_i \partial_{\boldsymbol{\xi}'} \frac{1}{\tau^2} \log\cosh(\tau \xi_i') - \frac{1}{\tau} \tanh(\tau \boldsymbol{\xi}).\end{aligned}$$

The second derivative of the LNE divergence w.r.t. $\boldsymbol{\xi}'$ is:

$$\partial_{\boldsymbol{\xi}'}^2 D_{LNE}[\boldsymbol{\xi}' : \boldsymbol{\xi}] = \sum_i \partial_{\boldsymbol{\xi}'}^2 \frac{1}{\tau^2} \log\cosh(\tau \xi_i').$$

We deduce the Taylor expansion of the LNE divergence at $\boldsymbol{\xi}' = \boldsymbol{\xi}$:

$$\begin{aligned}
D_{LNE}[\boldsymbol{\xi}' : \boldsymbol{\xi}] &\approx D_{LNE}[\boldsymbol{\xi} : \boldsymbol{\xi}] + \left( \sum_i \partial_{\boldsymbol{\xi}'} \frac{1}{\tau^2} \log\cosh(\tau \xi_i') - \frac{1}{\tau} \tanh(\tau \boldsymbol{\xi}) \right)^\top \Big|_{\boldsymbol{\xi}'=\boldsymbol{\xi}} d\boldsymbol{\xi} \\
&\quad + \frac{1}{2} d\boldsymbol{\xi}^\top \left( \sum_i \partial^2_{\boldsymbol{\xi}'} \frac{1}{\tau^2} \log\cosh(\tau \xi_i') \right) \Big|_{\boldsymbol{\xi}'=\boldsymbol{\xi}} d\boldsymbol{\xi} \\
&= 0 + 0 + \frac{1}{2\tau^2} d\boldsymbol{\xi}^\top \partial \left[ \frac{\partial \cosh(\tau \boldsymbol{\xi})}{\cosh(\tau \boldsymbol{\xi})} \right] d\boldsymbol{\xi} \\
&= \frac{1}{2\tau^2} d\boldsymbol{\xi}^\top \frac{\partial^2 \cosh(\tau \boldsymbol{\xi}) \cosh(\tau \boldsymbol{\xi}) - \partial \cosh(\tau \boldsymbol{\xi}) \partial \cosh(\tau \boldsymbol{\xi})^\top}{\cosh^2(\tau \boldsymbol{\xi})} d\boldsymbol{\xi} \\
&= \frac{1}{2\tau^2} d\boldsymbol{\xi}^\top \left( \frac{\partial^2 \cosh(\tau \boldsymbol{\xi})}{\cosh(\tau \boldsymbol{\xi})} - \tau^2 \left[ \frac{\sinh(\tau \boldsymbol{\xi})}{\cosh(\tau \boldsymbol{\xi})} \right] \left[ \frac{\sinh(\tau \boldsymbol{\xi})}{\cosh(\tau \boldsymbol{\xi})} \right]^\top \right) d\boldsymbol{\xi} \\
&= \frac{1}{2} \sum_{i,j} \left[ \delta_{ij} - \left( \tanh(\tau \boldsymbol{\xi}) \tanh(\tau \boldsymbol{\xi})^\top \right)_{ij} d\xi_i d\xi_j \right].
\end{aligned}$$

### E.2 Proof of Lemma 11

We would like to know in which direction minimizes the loss function with the constraints of the LNE divergence, so that we do the minimization:

$$d\boldsymbol{\xi}^* = \underset{d\boldsymbol{\xi} \text{ s.t. } D_{LNE}[\boldsymbol{\xi} : \boldsymbol{\xi} + d\boldsymbol{\xi}] = c}{\arg\min} L(\boldsymbol{\xi} + d\boldsymbol{\xi})$$

where $c$ is the constant. The loss function descends along the manifold with constant speed, regardless the curvature.

Furthermore, we can write the minimization in Lagrangian form. Combined with Theorem 10, the LNE divergence can be approximated by its second order Taylor expansion. Approximating $L(\boldsymbol{\xi} + d\boldsymbol{\xi})$ with it first order Taylor expansion, we get:

$$\begin{aligned}
d\boldsymbol{\xi}^* &= \arg\min_{d\boldsymbol{\xi}} \; L(\boldsymbol{\xi} + d\boldsymbol{\xi}) + \lambda \left( D_{LNE}[\boldsymbol{\xi} : \boldsymbol{\xi} + d\boldsymbol{\xi}] - c \right) \\
&\approx \arg\min_{d\boldsymbol{\xi}} \; L(\boldsymbol{\xi}) + \partial_{\boldsymbol{\xi}} L(\boldsymbol{\xi})^\top d\boldsymbol{\xi} + \frac{\lambda}{2} d\boldsymbol{\xi}^\top g(\boldsymbol{\xi}) d\boldsymbol{\xi} - c\lambda.
\end{aligned}$$

To solve this minimization, we set its derivative w.r.t. $d\boldsymbol{\xi}$ to zero:

$$\begin{aligned}
0 &= \frac{\partial}{\partial d\boldsymbol{\xi}} L(\boldsymbol{\xi}) + \partial_{\boldsymbol{\xi}} L(\boldsymbol{\xi})^\top d\boldsymbol{\xi} + \frac{\lambda}{2} d\boldsymbol{\xi}^\top \left[ \delta - \tanh(\tau \boldsymbol{\xi}) \tanh(\tau \boldsymbol{\xi})^\top \right] d\boldsymbol{\xi} - c\lambda \\
&= \partial_{\boldsymbol{\xi}} L(\boldsymbol{\xi}) + \lambda \left[ \delta - \tanh(\tau \boldsymbol{\xi}) \tanh(\tau \boldsymbol{\xi})^\top \right] d\boldsymbol{\xi} \\
d\boldsymbol{\xi} &= -\frac{1}{\lambda} \left[ \delta - \tanh(\tau \boldsymbol{\xi}) \tanh(\tau \boldsymbol{\xi})^\top \right]^{-1} \partial_{\boldsymbol{\xi}} L(\boldsymbol{\xi})
\end{aligned}$$

where a constant factor $1/\lambda$ can be absorbed into learning rate. Therefore, we get the optimal descent direction, i.e., the opposite direction of gradient, which takes into account the local curvature defined by $\left[ \delta - \tanh(\tau \boldsymbol{\xi}) \tanh(\tau \boldsymbol{\xi})^\top \right]^{-1}$.

## References

Thalaiyasingam Ajanthan, Kartik Gupta, Philip Torr, Richad Hartley, and Puneet Dokania. Mirror descent view for neural network quantization. In *International Conference on Artificial Intelligence and Statistics*, pages 2809–2817. PMLR, 2021.

S-i Amari and H Nagaoka. Methods of information geometry, volume 191 of translations of mathematical monographs, s. kobayashi and m. takesaki, editors. *American Mathematical Society, Providence, RI, USA*, pages 2–19, 2000.

Shun-Ichi Amari. Natural gradient works efficiently in learning. *Neural computation*, 10(2): 251–276, 1998.

Shun-ichi Amari. *Information geometry and its applications*, volume 194. Springer, 2016.

Alexander Appleton. Scalar curvature rigidity and ricci deturck flow on perturbations of euclidean space. *Calculus of Variations and Partial Differential Equations*, 57(5):1–23, 2018.

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

Yu Bai, Yu-Xiang Wang, and Edo Liberty. Proxquant: Quantized neural networks via proximal operators. *arXiv preprint arXiv:1810.00861*, 2018.

Richard H Bamler. Stability of hyperbolic manifolds with cusps under ricci flow. *arXiv preprint arXiv:1004.2058*, 2010.

Richard Heiner Bamler. *Stability of Einstein metrics of negative curvature*. Princeton University, 2011.

Michéle Basseville. Divergence measures for statistical data processing—an annotated bibliography. *Signal Processing*, 93(4):621–633, 2013.

Amir Beck and Marc Teboulle. Mirror descent and nonlinear projected subgradient methods for convex optimization. *Operations Research Letters*, 31(3):167–175, 2003.

Yoshua Bengio, Nicholas Léonard, and Aaron Courville. Estimating or propagating gradients through stochastic neurons for conditional computation. *arXiv preprint arXiv:1308.3432*, 2013.

Arthur L Besse. *Einstein manifolds*. Springer Science & Business Media, 2007.

Lev M Bregman. The relaxation method of finding the common point of convex sets and its application to the solution of problems in convex programming. *USSR computational mathematics and mathematical physics*, 7(3):200–217, 1967.

Sébastien Bubeck et al. Convex optimization: Algorithms and complexity. *Foundations and Trends® in Machine Learning*, 8(3-4):231–357, 2015.

Zhaowei Cai, Xiaodong He, Jian Sun, and Nuno Vasconcelos. Deep learning with low precision by half-wave gaussian quantization. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5918–5926, 2017.

Jun Chen, Liang Liu, Yong Liu, and Xianfang Zeng. A learning framework for n-bit quantized neural networks toward fpgas. *IEEE transactions on neural networks and learning systems*, 32(3):1067–1081, 2021.

Shangyu Chen, Wenya Wang, and Sinno Jialin Pan. Metaquant: Learning to quantize by learning to penetrate non-differentiable quantization. In *Advances in Neural Information Processing Systems*, volume 32, pages 3916–3926. Curran Associates, Inc., 2019.

Matthieu Courbariaux, Itay Hubara, Daniel Soudry, Ran El-Yaniv, and Yoshua Bengio. Binarized neural networks: Training deep neural networks with weights and activations constrained to+ 1 or-1. *arXiv preprint arXiv:1602.02830*, 2016.

George Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of control, signals and systems*, 2(4):303–314, 1989.

Alix Deruelle and Klaus Kröncke. Stability of ale ricci-flat manifolds under ricci flow. *The Journal of Geometric Analysis*, 31(3):2829–2870, 2021.

Dennis M DeTurck. Deforming metrics in the direction of their ricci tensors. *Journal of Differential Geometry*, 18(1):157–162, 1983.

Baptiste Devyver. A gaussian estimate for the heat kernel on differential forms and application to the riesz transform. *Mathematische Annalen*, 358(1):25–68, 2014.

Timothy Dozat. Incorporating nesterov momentum into adam. 2016.

Mostafa Elhoushi, Zihao Chen, Farhan Shafiq, Ye Henry Tian, and Joey Yiwei Li. Deepshift: Towards multiplication-less neural networks. *arXiv preprint arXiv:1905.13298*, 2019.

Christine Guenther, James Isenberg, and Dan Knopf. Stability of the ricci flow at ricci-flat metrics. *Communications in Analysis and Geometry*, 10(4):741–777, 2002.

Philipp Gysel, Jon Pimentel, Mohammad Motamedi, and Soheil Ghiasi. Ristretto: A framework for empirical study of resource-efficient inference in convolutional neural networks. *IEEE transactions on neural networks and learning systems*, 29(11):5784–5789, 2018.

Richard S Hamilton et al. Three-manifolds with positive ricci curvature. *J. Differential geom*, 17(2):255–306, 1982.

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*, pages 770–778, 2016.

Sigurdur Helgason. *Differential geometry and symmetric spaces*, volume 341. American Mathematical Soc., 2001.

G Hinton. Neural networks for machine learning. coursera,[video lectures], 2012.

Kurt Hornik. Approximation capabilities of multilayer feedforward networks. *Neural networks*, 4(2):251–257, 1991.

Lu Hou, Quanming Yao, and James T Kwok. Loss-aware binarization of deep networks. *arXiv preprint arXiv:1611.01600*, 2016.

Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017.

Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

Vishnu Jejjala, Damian Kaloni Mayorga Pena, and Challenger Mishra. Neural network approximations for calabi-yau metrics. *arXiv preprint arXiv:2012.15821*, 2020.

Tosio Kato. *Perturbation theory for linear operators*, volume 132. Springer Science & Business Media, 2013.

Piyush Kaul and Brejesh Lall. Riemannian curvature of deep neural networks. *IEEE transactions on neural networks and learning systems*, 31(4):1410–1416, 2019.

Herbert Koch and Tobias Lamm. Geometric flows with rough initial data. *Asian Journal of Mathematics*, 16(2):209–235, 2012.

Norihito Koiso. Einstein metrics and complex structures. *Inventiones mathematicae*, 73(1): 71–106, 1983.

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

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

Olga Aleksandrovna Ladyzhenskaia, Vsevolod Alekseevich Solonnikov, and Nina N Ural'tseva. *Linear and quasi-linear equations of parabolic type*, volume 23. American Mathematical Soc., 1988.

Cong Leng, Zesheng Dou, Hao Li, Shenghuo Zhu, and Rong Jin. Extremely low bit neural network: Squeeze the last bit out with admm. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

Fengfu Li, Bo Zhang, and Bin Liu. Ternary weight networks. *arXiv preprint arXiv:1605.04711*, 2016.

Zechun Liu, Baoyuan Wu, Wenhan Luo, Xin Yang, Wei Liu, and Kwang-Ting Cheng. Bi-real net: Enhancing the performance of 1-bit cnns with improved representational capability and advanced training algorithm. In *Proceedings of the European conference on computer vision (ECCV)*, pages 722–737, 2018.

Christos Louizos, Matthias Reisser, Tijmen Blankevoort, Efstratios Gavves, and Max Welling. Relaxed quantization for discretized neural networks. In *International Conference on Learning Representations*, 2019. URL `https://openreview.net/forum?id=HkxjYoCqKX`.

James Martens and Roger Grosse. Optimizing neural networks with kronecker-factored approximate curvature. In *International conference on machine learning*, pages 2408–2417, 2015.

Vincent Minerbe. Weighted sobolev inequalities and ricci flat manifolds. *Geometric and Functional Analysis*, 18(5):1696–1749, 2009.

Arkadi Nemirovsky and David Yudin. Informational complexity and efficient methods for solution of convex extremal problems. *Ékonomika i Mathematicheskie Metody*, 12, 1983.

Tommaso Pacini. Desingularizing isolated conical singularities: uniform estimates via weighted sobolev spaces. *arXiv preprint arXiv:1005.3511*, 2010.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *arXiv preprint arXiv:1912.01703*, 2019.

Haotong Qin, Ruihao Gong, Xianglong Liu, Mingzhu Shen, Ziran Wei, Fengwei Yu, and Jingkuan Song. Forward and backward information retention for accurate binary neural networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 2250–2259, 2020.

Garvesh Raskutti and Sayan Mukherjee. The information geometry of mirror descent. *IEEE Transactions on Information Theory*, 61(3):1451–1457, 2015.

Mohammad Rastegari, Vicente Ordonez, Joseph Redmon, and Ali Farhadi. Xnor-net: Imagenet classification using binary convolutional neural networks. In *European conference on computer vision*, pages 525–542. Springer, 2016.

Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International journal of computer vision*, 115(3):211–252, 2015.

Hasim Sak, Andrew W Senior, and Françoise Beaufays. Long short-term memory recurrent neural network architectures for large scale acoustic modeling. 2014.

Oliver C Schnürer, Felix Schulze, and Miles Simon. Stability of euclidean space under ricci flow. *arXiv preprint arXiv:0706.0421*, 2007.

Natasa Sesum. Linear and dynamical stability of ricci-flat metrics. *Duke Mathematical Journal*, 133(1):1–26, 2006.

Nick Sheridan and Hyam Rubinstein. Hamilton's ricci flow. *Honour thesis*, 2006.

Connor Shorten and Taghi M Khoshgoftaar. A survey on image data augmentation for deep learning. *Journal of big data*, 6(1):1–48, 2019.

Robert M Wald. *General relativity*. University of Chicago press, 2010.

Jiwei Yang, Xu Shen, Jun Xing, Xinmei Tian, Houqiang Li, Bing Deng, Jianqiang Huang, and Xian-sheng Hua. Quantization networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

Aojun Zhou, Anbang Yao, Yiwen Guo, Lin Xu, and Yurong Chen. Incremental network quantization: Towards lossless cnns with low-precision weights. *arXiv preprint arXiv:1702.03044*, 2017.

Aojun Zhou, Anbang Yao, Kuan Wang, and Yurong Chen. Explicit loss-error-aware quantization for low-bit deep neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 9426–9435, 2018.

Shuchang Zhou, Yuxin Wu, Zekun Ni, Xinyu Zhou, He Wen, and Yuheng Zou. Dorefa-net: Training low bitwidth convolutional neural networks with low bitwidth gradients. *arXiv preprint arXiv:1606.06160*, 2016.

Chenzhuo Zhu, Song Han, Huizi Mao, and William J Dally. Trained ternary quantization. *arXiv preprint arXiv:1612.01064*, 2016.

Feng Zhu, Ruihao Gong, Fengwei Yu, Xianglong Liu, Yanfei Wang, Zhelong Li, Xiuqi Yang, and Junjie Yan. Towards unified int8 training for convolutional neural network. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1969–1979, 2020.