Learning Discretized Neural Networks under Ricci Flow

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

[1] *Institute of Cyber-Systems and Control, Zhejiang University, Hangzhou, China.*
[2] *State Grid Corporation of China.*
[3] *Centre for Frontier Artificial Intelligence Research, A*STAR, Singapore.*
[4] *Australian Artificial Intelligence Institute, University of Technology, Sydney Australia.*

Abstract

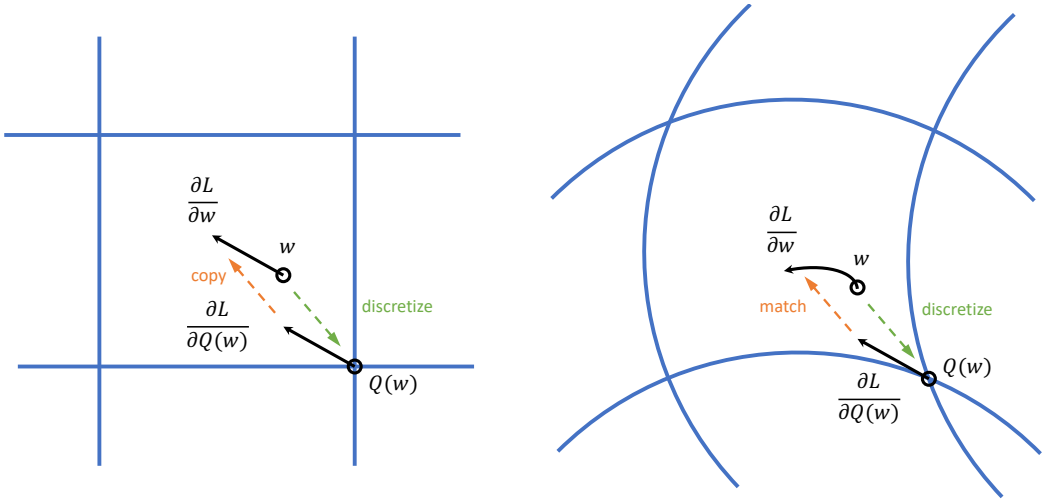
In this paper, we consider Discretized Neural Networks (DNNs) consisting of low-precision weights and activations, which suffer from either infinite or zero gradients due to the non-differentiable discrete function in the training process. In this case, most training-based DNNs employ the standard Straight-Through Estimator (STE) to approximate the gradient w.r.t. discrete values. However, the STE gives rise to the problem of gradient mismatch, due to the perturbations of the approximated gradient. To address this problem, this paper reveals that this mismatch can be viewed as a metric perturbation in a Riemannian manifold through the lens of duality theory. Further, on the basis of the information geometry, we construct the Linearly Nearly Euclidean (LNE) manifold for DNNs as a background to deal with perturbations. By introducing a partial differential equation on metrics, i.e., the Ricci flow, we prove the dynamical stability and convergence of the LNE metric with the L^2-norm perturbation. Unlike the previous perturbation theory whose convergence rate is the fractional powers, the metric perturbation under the Ricci flow can be exponentially decayed in the LNE manifold. The experimental results on various datasets demonstrate that our method achieves better and more stable performance for DNNs than other representative training-based methods.

Keywords: Discretized Neural Networks, Gradient Perturbation, Information Geometry, Ricci Flow, 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, which can significantly reduce computational complexity, storage space, power consumption, resources, etc (Chen et al., 2020). Considering a Discretized neural network (DNN) which is able to be well-trained, based on the standard chain rule, 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 deriva-

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).



(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 of DNNs, the continuous weight $\boldsymbol{w}$ is mapped to a discrete weight $Q(\boldsymbol{w})$ via a discrete function. In the backward, 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, on the other hand, matches the gradient by introducing the proper metric $g_{\boldsymbol{w}}$, i.e., $\partial L/\partial \boldsymbol{w} = g_{\boldsymbol{w}}^{-1}\partial L/\partial Q(\boldsymbol{w})$, while taking into account the gradient mismatch caused by STE in a Riemannian manifold.

tives. Its root is that the derivative $\partial Q(\boldsymbol{w})/\partial \boldsymbol{w}$ can not be calculated. In the backward, one can obtain the gradient $\partial L/\partial Q(\boldsymbol{w})$, but need to update the continuous weight $\boldsymbol{w}$ via the gradient $\partial L/\partial \boldsymbol{w}$. Since the gradient $\partial L/\partial \boldsymbol{w}$ can not be obtained explicitly, one needs the derivative $\partial Q(\boldsymbol{w})/\partial \boldsymbol{w}$ as a bridge to calculate $\partial L/\partial \boldsymbol{w}$ via the 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 Straight-Through Estimator (STE) that yields an immediate connection between $\partial L/\partial \boldsymbol{w}$ and $\partial L/\partial Q(\boldsymbol{w})$ in back-propagation such that bypassing the derivative $\partial Q(\boldsymbol{w})/\partial \boldsymbol{w}$. The definition of STE was then given by Bengio et al. (2013), which 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:

$$\frac{\partial L}{\partial \boldsymbol{w}} = \frac{\partial L}{\partial \operatorname{sign}(\boldsymbol{w})} \cdot \mathbb{I}, \quad \text{where } \mathbb{I} := \begin{cases} 1 & \text{if} \quad |\boldsymbol{w}| \leq 1 \\ 0 & \text{otherwise} \end{cases}, \tag{1}$$

where $\mathbb{I}$ is the indicator function. Note that $Q(\cdot)$ will degenerate to $\operatorname{sign}(\cdot)$ which is equal to $+1$ for $\boldsymbol{w} \geq 0$ and -1 otherwise 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).

On the other hand, Non-STE methods consist in all techniques that do not rely on STE, e.g., (Hou et al., 2016; Bai et al., 2018; Leng et al., 2018). However, the learning process of Non-STE methods depends heavily 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). Hence, STE methods are widely used in DNNs rather than Non-STE methods owing to their simplicity and versatility.

However, STE introduced into DNNs, inevitably gives rise 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 when $|\boldsymbol{w}| \leq 1$ (Chen et al., 2019), which compromises the training stability of DNNs (Cai et al., 2017; Liu et al., 2018; Qin et al., 2020). Furthermore, the formula of STE tells us that this problem is able to be alleviated by modifying the gradient $\partial L/\partial \boldsymbol{w}$.

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 purpose of the authors is to manually redefine the indicator function $\mathbb{I}$ as $\frac{1-\tanh^2(\boldsymbol{w})}{\max(|\tanh(\boldsymbol{w})|)}$ such that the function $\frac{1-\tanh^2(\boldsymbol{w})}{\max(|\tanh(\boldsymbol{w})|)}$ provides a smooth transition to avoid abrupt clipping of the indicator function near ± 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_ψ parameterized by ψ 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 will output 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 (Beck and Teboulle, 2003) to perform gradient descent in the dual space (unconstrained space) with gradients computed in the primal space (discrete space). In particular, by projecting the primal space $\boldsymbol{w}$ into the dual space $\tilde{\boldsymbol{w}} = \tanh(\beta_k \boldsymbol{w})$, the gradient can be yielded 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 $\beta_k \to \infty$, $\tilde{\boldsymbol{w}}$ gradually approaches $\text{sign}(\boldsymbol{w})$ until the corresponding neural network is fully binarized with an adaptive mirror map.

However, Zhou et al. (2016) obtained the new weight by manually setting the function tanh, which can only scale the gradient as a whole and do not fundamentally alleviate the gradient mismatch. On the other hand, although Chen et al. (2019) automatically matched the gradient by learning a new neural network (a meta quantizer), the additional errors are also engendered in the gradient propagation due to the meta quantizer, leading to further intensifying the problem of gradient mismatch. Subsequently, Ajanthan et al. (2021) bypassed the problem of gradient mismatch because the derivative $\partial\tilde{\boldsymbol{w}}/\partial\boldsymbol{w} = \left(1 - \tanh^2(\beta_k \boldsymbol{w})\right)$ can be calculated directly, which implies that this method does not maintain discrete weights during the training. Therefore, the problem of gradient mismatch still remains to be solved.

1.1 Contributions

In this work, we regard the gradient mismatch between $\partial L/\partial \boldsymbol{w}$ and $\partial L/\partial Q(\boldsymbol{w})$ as a perturbation phenomenon between these two gradients. By introducing the framework of Riemannian geometry in Figure 1(b), the gradient mismatch is further viewed as a metric perturbation in a Riemannian manifold (Section 2.2) through the lens of duality theory (Amari and Nagaoka, 2000; Amari, 2016). 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 such that the problem of gradient mismatch is theoretically solved. 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 as a background to deal 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 relationship 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. Furthermore, unlike the previous perturbation theory whose convergence rate is the fractional powers ($t^{-3/2}$), the metric perturbation under the Ricci flow can give rise to exponentially decaying in the LNE manifold (e^{-t}), further bringing about theoretical assurance 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 by calculating 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 in traditional DNNs.
- The experiments are implemented on several classification benchmark datasets and network structures. Experimental results demonstrate the effectiveness of our geometric method RF-DNN compared with other representative training-based methods.

1.2 Overall Organization

This paper is organized as follows. In Section 2, we introduce the motivation and Ricci flow. According to the geometric structure measured by the LNE divergence, we deduce the corresponding LNE manifold for neural networks in Section 3. 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. In Section 6, we present how to introduce discrete Ricci flow into DNNs and yield 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^∞-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). Stability of LNE manifolds under the Ricci-DeTurck flow is then proved (Corollary 15 and Theorem 16). A discretization of the Ricci flow is therefore proposed, that leads to a practical algorithm (RF-DNNs, Algorithm 2).

2. Motivation and Formulation

2.1 Background

We start with the basic background for feed-forward DNNs that will be used throughout the paper. This background is on the basis of the work (Martens and Grosse, 2015). And we list the important notations in Appendix B.

A neural network can be regarded as a function to transform 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 via a nonlinear function.

For a DNN, we need to add a discrete function $Q(\cdot)$ to discretize the weight matrix $\boldsymbol{W}_i$ and the activation vector $\boldsymbol{a}_i$. Furthermore, we mark 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:

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

where f is a nonlinear (activation) function, $\odot$ represents the element-wise multiplication. We vectorize $\hat{\boldsymbol{W}}_i$ as $\mathrm{vec}(\hat{\boldsymbol{W}}_i)$ and stack their columns together, where $\mathrm{vec}(*)$ is the operator that vectorizes a matrix as a column vector. Note that the vectorized weights in each layer before and after discretization are expressed as $\boldsymbol{w}$ and $\hat{\boldsymbol{w}}$, respectively. We define the new discretized parameter vector as $\hat{\boldsymbol{\xi}} = \left[\mathrm{vec}\left(\hat{\boldsymbol{W}}_1\right)^\top, \mathrm{vec}\left(\hat{\boldsymbol{W}}_2\right)^\top, \ldots, \mathrm{vec}\left(\hat{\boldsymbol{W}}_l\right)^\top\right]^\top$, where we ignore the bias vector for brevity. Similarly, we can denote the parameter vector as $\boldsymbol{\xi} = \left[\mathrm{vec}\left(\boldsymbol{W}_1\right)^\top, \mathrm{vec}\left(\boldsymbol{W}_2\right)^\top, \ldots, \mathrm{vec}\left(\boldsymbol{W}_l\right)^\top\right]^\top$ where $\hat{\boldsymbol{\xi}} = Q(\boldsymbol{\xi})$.

In frequentist statistics, we represent the loss function as the negative log likelihood w.r.t. discretized parameter $\hat{\boldsymbol{\xi}}$, where the input $\boldsymbol{a}_0$ can be observed. The way minimizing the loss function is to maximize the likelihood:

$$L(\boldsymbol{y}, \boldsymbol{z}) = -\log p\left(\boldsymbol{y}|\boldsymbol{z}, \hat{\boldsymbol{\xi}}\right) \tag{3}$$

where $\boldsymbol{y}$ is the target and $\boldsymbol{z}$ is the prediction calculated by DNNs. Note that $p\left(\boldsymbol{y}|\boldsymbol{z}, \hat{\boldsymbol{\xi}}\right)$ is the probability density function defined by the conditional probability distribution $P_{\boldsymbol{y}|\boldsymbol{z}}\left(\hat{\boldsymbol{\xi}}\right)$ of DNNs. As for the back-propagation of DNNs, the details are given in later sections.

2.2 Motivation

We consider that the root of the gradient mismatch is that there is a perturbation phenomenon between $\partial L/\partial \boldsymbol{w}$ and $\partial L/\partial Q(\boldsymbol{w})$, i.e.,

$$\frac{\partial L}{\partial \boldsymbol{w}} = \mathcal{P}\left(\frac{\partial L}{\partial Q(\boldsymbol{w})}\right)\Big|_{O\left(\frac{\partial L}{\partial Q(\boldsymbol{w})}\right)} \tag{4}$$

where the input of the perturbation function $\mathcal{P}$ is the gradient $\partial L/\partial Q(\boldsymbol{w})$, and the corresponding perturbation range is $O\left(\partial L/\partial Q(\boldsymbol{w})\right)$. If the perturbation range $O\left(\partial L/\partial Q(\boldsymbol{w})\right)$ can be significantly eliminated or decayed, then there is an elegant solution to the problem of gradient mismatch. In the framework of perturbation theory of the linear space (Kato, 2013), the rate of convergence for perturbations is the fractional powers, e.g., $t^{-3/2}$.

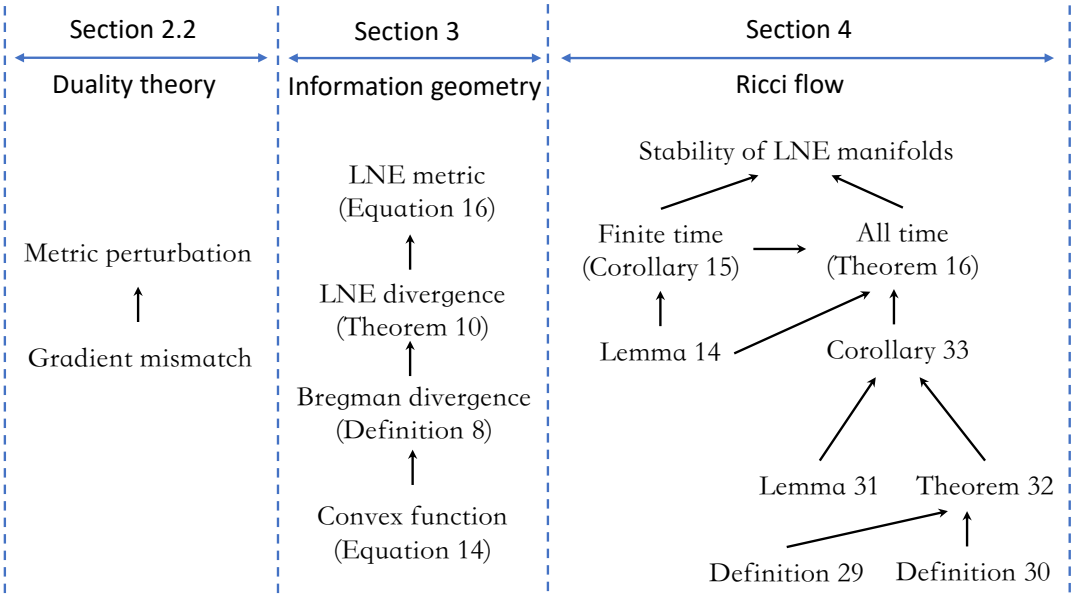


Figure 2: The overview of the theoretical ideas in this paper.

Inspired by the mirror descent [2] framework (Beck and Teboulle, 2003), one can map the parameter from the primal space to the dual space, and then calculate the gradient in the dual space. When the Riemannian metric structures are introduced by means of information geometry (Amari, 2016), naturally [3], the gradient mismatch is conclusively viewed as a metric perturbation in a Riemannian manifold. Specifically, with the help of Definition 1, it is obvious that the gradient $\partial L/\partial \boldsymbol{w}$ will be rewritten as $\tilde{\partial} L/\tilde{\partial} \boldsymbol{w}$ in a Riemannian manifold. Therefore, based on Equation (4), the problem of gradient mismatch can be defined as

$$\frac{\tilde{\partial} L}{\tilde{\partial} \boldsymbol{w}} = g_{\boldsymbol{w}}^{-1} \frac{\partial L}{\partial Q(\boldsymbol{w})} \tag{5}$$

where the perturbation item is implied in the metric $g_{\boldsymbol{w}}$ and the term $\frac{\partial L}{\partial Q(\boldsymbol{w})}$ can be considered as the conventional gradient $\partial L(\boldsymbol{w})$ in Definition 1. Then the metric perturbation phenomenon 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 δ.

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 conventional 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 is what kind of manifolds we need to construct to deal with metric perturbations naturally and easily. Or what manifold is a "good" manifold in the presence of perturbations. In practice, it seems that general relativity gives an excellent example in nature to deal with small gravitational perturbations under the framework of a Riemannian manifold (Wald, 2010). To treat the approximation in which gravity is "weak", the spacetime metric is nearly flat in the context of general relativity, which is enough for most cases, except for phenomena dealing with gravitational collapse and the large scale structure of the universe. By assuming that the deviation γ_{ij} of the actual spacetime metric $g_{ij} = \eta_{ij} + \gamma_{ij}$ from a flat metric η_{ij} is "small", the linearized gravity is presented to approximately calculate the gravity in general relativity. An adequate definition of "smallness" in this context is that the components of γ_{ij} are much smaller than 1 in the global inertial coordinate system of η_{ij}. Such a linearly nearly flat metric greatly simplifies

2. Mirror descent induces non-Euclidean structure by solving iterative optimization problems using different proximity functions (Bubeck et al., 2015).
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 to each other (Raskutti and Mukherjee, 2015), which implies that mirror descent is the steepest descent direction along the Riemannian manifold corresponding to the choice of Bregman divergence.

the calculation of "weak" gravity, and the manifolds constructed with such metrics are considered to be sufficient for approximating the manifold with perturbations.

Similarly, in this paper, by regarding the Euclidean metric δ_{ij} as the flat metric η_{ij}, we can naturally define the Linearly Nearly Euclidean (LNE) metric. If we can construct such a metric for DNNs, then a metric perturbation can be handled in the background of LNE manifolds. Motivated by the natural gradient descent (Amari, 2016) that connects neural networks with the Riemannian metric, LNE metrics can be mathematically constructed in neural networks. Specifically, our aim is to first introduce a convex function to derive the LNE divergence with the help of Bregman divergence (Bregman, 1967), and then construct the LNE metric by introducing the LNE divergence into a neural network [4]. Consequently, the LNE metric appears in the gradient of the LNE manifold, similar to Definition 1. Finally, based on the constructed manifold for DNNs, the only remaining problem is how to rapidly decay the metric perturbation by employing a geometric tool, i.e., Ricci flow.

In addition, a series of proofs about the 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 is decayed, the metric perturbation can be alleviated such that a theoretical solution for the problem of gradient mismatch in the training of DNNs is further given. Unlike the previous perturbation theory whose the convergence rate is the fractional powers, the metric perturbation under the Ricci flow can be exponentially decayed in the LNE manifold. In Figure 2, we give the 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 $\mathcal{M}$, based on Definition 2, of a time-dependent metric $g(t)$. Given the initial metric g_0, the Ricci flow is a partial differential equation that evolves the metric tensor:

$$\begin{aligned} \frac{\partial}{\partial t} g(t) &= -2\,\mathrm{Ric}(g(t)) \\ g(0) &= g_0 \end{aligned} \tag{6}$$

where Ric denotes the Ricci curvature tensor and the detailed definition can be found in Appendix A. The purpose of Ricci flow is to prove Thurston's Geometrization Conjecture and Poincaré Conjecture, which is to evolve the metric towards certain geometric structures and topological properties (Sheridan and Rubinstein, 2006).

4. The part from a convex function to the LNE divergence is in the mirror descent framework. Then, the part from the LNE divergence to the LNE metric introduces the idea of information geometry.

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 [5]) $h_{ij} = \frac{\partial}{\partial t} g_{ij}(t) \neq 0$, 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 knows whether there exists a unique solution of Ricci flow for a short time by checking 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. Since the inequality can not always be satisfied, Ricci flow is not always strongly parabolic, which makes us unable to assure the existence of the solution according to Theorem 4. In what follows, the non-poarabolic root is analyzed and the solution can be found on the basis of the relationship between the Ricc flow and the Ricci-DeTurck flow. Now, it is possible to understand which parts have an impact on its non-parabolic nature by the linearization of the Ricci curvature tensor. We define the linearization of the Ricci curvature as $\mathcal{D}[\mathrm{Ric}]$ so 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* [6]

$$\begin{aligned} &\mathcal{D}[-2\,\mathrm{Ric}](h)_{ij} = g^{pq}\nabla_p\nabla_q h_{ij} + \nabla_i V_j + \nabla_j V_i + O(h_{ij}) \\ &\text{where} \quad V_i = g^{pq}\left(\frac{1}{2}\nabla_i h_{pq} - \nabla_q h_{pi}\right) \text{and} \quad h_{ij} = \frac{\partial}{\partial t} g_{ij}(t). \end{aligned} \tag{7}$$

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

By carefully observing Lemma 5, the impact on the non-parabolic of Ricci flow comes from the terms V_i and V_j (Sheridan and Rubinstein, 2006), instead of 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 try to eliminate the impact of the non-parabolic nature on the Ricci flow.

5. 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.
6. 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.

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

$$g(t) = \varphi^*(t)\bar{g}(t), \tag{8}$$

which satisfies 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. $\bar{g}(t)$) that is strongly parabolic. Furthermore, the solution is followed by the DeTurck Trick (DeTurck, 1983) that has a time-dependent reparameterization of the manifold:

$$\begin{aligned} \frac{\partial}{\partial t}\bar{g}(t) &= -2\,\mathrm{Ric}(\bar{g}(t)) - \mathcal{L}_{\frac{\partial \varphi(t)}{\partial t}}\bar{g}(t) \\ \bar{g}(0) &= \bar{g}_0, \end{aligned} \tag{9}$$

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. (Schnürer et al., 2007) have showed the stability of Euclidean space under the Ricci flow for a small C^0 perturbation. Koch et al. (Koch and Lamm, 2012) have given the stability of the Euclidean space along with the Ricci flow in the L^∞-norm. Moreover, for the decay of the L^∞-norm on Euclidean space, Appleton (Appleton, 2018) has provided a proof using a different method.

On the other hand, for a Ricci-flat metric with small perturbations, Guenther et al. (Guenther et al., 2002) proved that such metrics would converge under Ricci flow. Considering the stability of integrable and closed Ricci-flat metrics, Sesum (Sesum, 2006) has proved that the convergence rate is exponential because the spectrum of the Lichnerowicz operator is discrete. Furthermore, Deruelle et al. (Deruelle and Kröncke, 2021) have proved 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. Our work is also discussed on Ricci-flat manifolds.

3. Neural Networks in LNE Manifolds

The aim of this section is to bulid a LNE manifold via the information geometry (Amari, 2016) such that paving the way for the introduction of Ricci flow. Specifically, we first introduce a convex function (Equation (14)) to derive the LNE divergence (Theorem 10) with the help of Bregman divergence (Definition 8), and then construct the LNE metric (Equation (16)) by introducing the LNE divergence into a neural networks. Consequently, the LNE metric then appears in the steepest descent gradient (Lemma 11) of the LNE manifold. Of course, the mirror descent algorithm (Beck and Teboulle, 2003) can also equivalently construct the relationship between divergences and gradients, but it lacks the geometric meaning (manifold and metric) that is exactly what we need.

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.

For the $n \times n$ matrix, all such matrices form an n^2-dimensional manifold. Specifically, the matrix forms the $n(n+1)/2$-dimensional manifold, i.e., a submanifold embedded in the manifold of all the matrices when this matrix is symmetric and positive-definite (Sheridan and Rubinstein, 2006). Note that such a matrix is described as the metric in the geometry theory. Since symmetric and positive-definite matrices have many advantages, which lead to wide implementations in real-world applications, our method will also construct such symmetric and positive-definite matrices (metrics) with some appealing geometric characteristics in neural networks.

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 the curved coordinate system is selected in a Riemannian manifold, the symmetric and positive-definite metric is also given based on Definition 2. That geometry-based metric can describe the properties of manifolds, such as curvature (Helgason, 2001).*

3.2 Euclidean Space and Divergence

From the viewpoint of the information geometry, the metric can be deduced by the divergence satisfying the certain criteria (Basseville, 2013), which is summarized as 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

$$D[\boldsymbol{\xi}_P : \boldsymbol{\xi}_Q] = \frac{1}{2} \sum_{i,j} g_{ij}(\boldsymbol{\xi}_P) d\xi_i d\xi_j + O(|d\boldsymbol{\xi}|^3), \tag{10}$$

and the Riemannian metric g_{ij} is positive-definite, depending on $\boldsymbol{\xi}_P$.

When P and Q are sufficiently close where these two points are expressed in coordinates as $\boldsymbol{\xi}_P$ and $\boldsymbol{\xi}_Q$ (column vectors), the square of an infinitesimal distance ds^2 between them by using Definition 7 can be defined as:

$$ds^2 = 2D[\boldsymbol{\xi}_P : \boldsymbol{\xi}_Q] = \sum_{i,j} g_{ij}(\boldsymbol{\xi}_P) d\xi_i d\xi_j \tag{11}$$

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 higher order term $O(|d\boldsymbol{\xi}|^3)$ for the sake of simplicity and convenience. 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 two nearby points $\boldsymbol{\xi}_P$ and $\boldsymbol{\xi}_Q$ is given by Equation (11). A divergence D provides $\mathcal{M}$ with a Riemannian structure.

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

$$D_E[\boldsymbol{\xi} : \boldsymbol{\xi}'] = \frac{1}{2}\sum_i (\xi_i - \xi_i')^2. \tag{12}$$

The Riemannian metric g_{ij} degenerates into the Euclidean metric δ_{ij} in this case, so that

$$ds^2 = 2D_E[\boldsymbol{\xi} : \boldsymbol{\xi} + d\boldsymbol{\xi}] = \sum_i (d\xi)^2 = \sum_{i,j} \delta_{ij} d\xi_i d\xi_j. \tag{13}$$

Obviously, the Euclidean divergence satisfies the criteria of divergence. Note that the Euclidean metric δ_{ij} is the identity matrix $\boldsymbol{I}$ where we still retain the notation of metrics for the sake of the representation habit in the geometry theory.

3.3 LNE Manifold and Divergence

Recall that [7], in general relativity (Wald, 2010), the complete Riemannian manifold $(\mathcal{M}, g)$ that is endowed with a linearly nearly flat spacetime metric, is considered to solve the Newtonian limit by the linearized gravity. The form of this metric is $g_{ij} = \eta_{ij} + \gamma_{ij}$ where η_{ij} represents a flat Minkowski metric whose background is special relativity and γ_{ij} is smallness. In practice, this is an excellent theory of small gravitational perturbations when gravity is "weak".

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

3.3.1 Convex Function and Bregman Divergence

In order to construct the LNE manifold endowed with the LNE metric in the neural network, according to Definition 7, we can introduce the divergence to obtain the expression of the LNE metric, similar to the relationship between Euclidean metric and divergence.

The premise of constructing a divergence is to find a suitable convex function. There are many applications for different convex functions in physics and optimization (Bubeck et al., 2015). Thus, we can introduce a nonlinear function $\Phi(\boldsymbol{\xi})$ of coordinates $\boldsymbol{\xi}$ as the convex function with a certain geometric structure to satisfy the needs of constructing the LNE divergence. When considering a differentiable function, it is said to be convex if and only if its Hessian is positive-definite

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

7. The link between the LNE manifold and general relativity can be found in Section 2.2.

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* $z = \Phi(\boldsymbol{\xi}') + (\boldsymbol{\xi} - \boldsymbol{\xi}')\nabla\Phi(\boldsymbol{\xi}')$*, depending on the Taylor expansion at the point* $\boldsymbol{\xi}'$*:*

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

By drawing a tangent hyperplane touching it at the point $\boldsymbol{\xi}'$

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

we can write the distance between the convex function $\Phi(\boldsymbol{\xi})$ and the tangent hyperplane z as the Bregman divergence. Considering it as a function of two points $\boldsymbol{\xi}$ and $\boldsymbol{\xi}'$, we can easily know that it satisfies the criteria of 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 graph describing their relationship is shown in Figure 3, where z is the vertical axis of the graph.

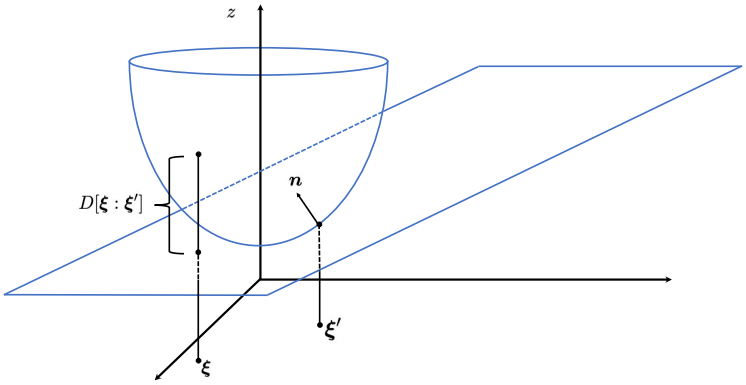


Figure 3: The divergence $D[\boldsymbol{\xi} : \boldsymbol{\xi}']$ is viewed as the distance between the convex function $\Phi(\boldsymbol{\xi})$ and the tangent hyperplane z. And z depends on the point $\boldsymbol{\xi}'$ at which the supporting hyperplane with normal vector $\boldsymbol{n} = \nabla\Phi(\boldsymbol{\xi}')$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

Remark 9 *In fact, a divergence is derived from a convex function in the form of the Bregman divergence. By choosing different convex functions, for example, we can easily obtain the Euclidean divergence from the Bregman divergence, by defining the convex function:* $\Phi(\boldsymbol{\xi}) = 1/2\sum_i \xi_i^2$ *in a Euclidean space. Besides, given the convex function:* $\Phi(\boldsymbol{\xi}) = \sum_i p(\boldsymbol{x}, \xi_i)\log p(\boldsymbol{x}, \xi_i)$ *where* $\sum_i p(\boldsymbol{x}, \xi_i) = \sum_i p(\boldsymbol{x}, \xi_i') = \int p(\boldsymbol{x}, \boldsymbol{\xi})d\boldsymbol{\xi} = \int p(\boldsymbol{x}, \boldsymbol{\xi}')d\boldsymbol{\xi}' = 1$ *is satisfied, the Bregman divergence is the same as the KL divergence.*

3.3.2 LNE Divergence and Gradient

Similar to the Bregman divergence with a convex function, we try to construct a new convex function to obtain the LNE divergence on which the LNE metric emerges naturally on the basis of Definition 7. Inspired by the previous work (Ajanthan et al., 2021), we propose a

novel convex function, which satisfies symmetry and is able to geometrically construct an easy-to-compute metric with the linearly nearly Euclidean nature:

$$\Phi(\boldsymbol{\xi}) = \sum_i \frac{1}{\tau^2} \log\left(\cosh(\tau\xi_i)\right) \tag{14}$$

where τ is a constant parameter that is used to control how linearly close to Euclidean structure.

Theorem 10 *By introducing a convex function Φ defined by Equation (14) into Definition 8, the **LNE divergence** between two points $\boldsymbol{\xi}$ and $\boldsymbol{\xi}'$ yields*

$$\begin{aligned} D_{LNE}[\boldsymbol{\xi}' : \boldsymbol{\xi}] &= \sum_i \left[\frac{1}{\tau^2} \log \frac{\cosh(\tau\xi_i')}{\cosh(\tau\xi_i)} - \frac{1}{\tau}(\xi_i' - \xi_i)\tanh(\tau\xi_i) \right] \\ &\approx \frac{1}{2} \sum_{i,j} \left\{ \delta_{ij} - \left[\tanh(\tau\boldsymbol{\xi}) \tanh(\tau\boldsymbol{\xi})^\top \right]_{ij} d\xi_i d\xi_j \right\}. \end{aligned} \tag{15}$$

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

Comparing with Definition 7, we know that the **LNE metric** corresponding to the LNE divergence is

$$\begin{aligned} g(\boldsymbol{\xi}) &= \delta_{ij} - \left[\tanh(\tau\boldsymbol{\xi}) \tanh(\tau\boldsymbol{\xi})^\top \right]_{ij} \\ &= \begin{bmatrix} 1 - \tanh(\tau\xi_1)\tanh(\tau\xi_1) & \cdots & -\tanh(\tau\xi_1)\tanh(\tau\xi_n) \\ \vdots & \ddots & \vdots \\ -\tanh(\tau\xi_n)\tanh(\tau\xi_1) & \cdots & 1 - \tanh(\tau\xi_n)\tanh(\tau\xi_n) \end{bmatrix}. \end{aligned} \tag{16}$$

Based on Section 3.1, we are able to employ the parameters of a neural network to construct the LNE metric (for a neural network, the coordinate $\boldsymbol{\xi}$ is the parameter vector). As a result, we can describe the neural network in the LNE manifold, as measured by the LNE divergence based on Theorem 10. Naturally, the steepest descent gradient in the LNE manifold is given by Lemma 11, which is similar to the natural gradient in Definition 1.

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

$$\tilde{\partial}_{\boldsymbol{\xi}} = g(\boldsymbol{\xi})^{-1} \partial_{\boldsymbol{\xi}} = \left[\delta - \tanh(\tau\boldsymbol{\xi}) \tanh(\tau\boldsymbol{\xi})^\top \right]^{-1} \partial_{\boldsymbol{\xi}}. \tag{17}$$

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

On the constructed LNE manifold, we can introduce the Ricci flow to decay the metric perturbation w.r.t. the LNE metric.

3.4 Convergence Analysis

We can now describe the mirror descent via the Bregman divergence associated to Φ. Let $\mathcal{C} \subset \mathbb{R}^n$ be a convex open set such that $\mathcal{X} \subset \bar{\mathcal{C}}$ ($\bar{\mathcal{C}}$ is the closure of $\mathcal{C}$), and $\mathcal{X} \cap \mathcal{C} \neq \emptyset$. When $\boldsymbol{x}^0 \in \arg\min_{\boldsymbol{x} \in \mathcal{X} \cap \mathcal{C}} \Phi(\boldsymbol{x})$ is the initial point, for iteration $k \geq 0$ and step size η, the update of the mirror descent can be written as:

$$\begin{aligned} \nabla \Phi(\boldsymbol{y}^{k+1}) &= \nabla \Phi(\boldsymbol{x}^k) - \eta \partial^k \\ \boldsymbol{x}^{k+1} &= \arg \min_{\boldsymbol{x} \in \mathcal{X} \cap \mathcal{C}} D_B[\boldsymbol{x}, \boldsymbol{y}^{k+1}] \end{aligned} \tag{18}$$

where $\partial^k \in \partial F(\boldsymbol{x}^k)$ and $\boldsymbol{y}^{k+1} \in \mathcal{C}$.

Theorem 12 *(Bubeck et al., 2015) Let Φ be a mirror map ρ-strongly convex on $\mathcal{X} \cap \mathcal{C}$ w.r.t. $\|\cdot\|$. Let $R^2 = \sup_{\boldsymbol{x} \in \mathcal{X} \cap \mathcal{C}} \Phi(\boldsymbol{x}) - \Phi(\boldsymbol{x}^0)$, and F be convex and L-Lipschitz w.r.t. $\|\cdot\|$. Then mirror descent with $\eta = \frac{R}{L}\sqrt{\frac{2\rho}{t}}$ satisfies*

$$F\left(\frac{1}{t}\sum_{k=0}^{t-1} \boldsymbol{x}^k\right) - F(\boldsymbol{x}^*) \leq RL\sqrt{\frac{2}{\rho t}}.$$

In this paper, the mirror descent is obtained by taking Equation (14) on $\mathcal{C} = \mathbb{R}^n$. The function $\Phi(\boldsymbol{\xi})$ is a mirror map 1-strongly convex w.r.t. $\|\cdot\|_2$, and furthermore the associated Bregman divergence is given by Theorem 10. In that case, the mirror descent is equivalent to projected subgradient descent, and the rate of convergence is $O(1/\sqrt{t})$, which is consistent with the standard mirror descent (Theorem 12). Of course, this gives the same convergence as (Ajanthan et al., 2021).

It is easy to verify that the projection w.r.t. this Bregman divergence on the simplex $\Delta_n = \{\xi \in \mathbb{R}^n : \sum_{i=1}^n \xi_i = 1\}$ amounts to a simple renormalization (Bubeck et al., 2015). For $\mathcal{X} = \Delta_n$, one has $\boldsymbol{x}^0 = (1/n, \cdots, 1/n)$ and $R^2 = \log(\cosh n)$. In that case, the mirror descent with the Equation (14) achieves a rate of convergence of order $\sqrt{\frac{\log(\cosh n)}{t}}$.

4. Evolution of LNE Manifolds under Ricci Flow

In this section, we will focus on LNE metrics under Ricci flow, and prove that the evolution of LNE manifolds can achieve good performances in terms of stability (all time), i.e., the Ricci flow can exponentially decay the L^2-norm perturbation to the LNE metric.

4.1 Relationship between LNE Metrics and Ricci Flow

To facilitate the handling of the metric perturbation, we have given the LNE metric in Equation (16), a special form of Ricci-flat metrics (Guenther et al., 2002; Deruelle and Kröncke, 2021), which means that the Ricci-flat metrics on noncompact manifolds are LNE. In particular, the constructed metric $g(\boldsymbol{\xi})$ in Equation (16) is LNE, which is equivalent to either g_0 under the Ricci flow or $\bar{g}_0$ under the Ricci-DeTurck flow since g_0 and $\bar{g}_0$ are diffeomorphic [8] to each other on the basis of Equation (8).

8. When there exists a Ricci flow, the corresponding Ricci-DeTurck flow exists, and vice versa.

The full statement of the LNE metric is the linearly nearly Euclidean Ricci-flat metric in Definition 13 given on the basis of the previous work (Deruelle and Kröncke, 2021).

Definition 13 *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*

$$\left|\partial^k(\phi_*\gamma)\right|_\delta = O(r^{-\iota-k}) \quad \forall k \geq 0 \tag{19}$$

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 δ and the deviation γ:

$$\bar{g}_0 = \delta + \gamma. \tag{20}$$

4.2 Finite Time Existence

In this subsection, we consider the linear stability and integrability of the LNE manifold $(\mathcal{M}^n, g_0)$. Fortunately, similar to the previous proofs (Koiso, 1983; Besse, 2007), we can know that $(\mathcal{M}^n, g_0)$ is integral and linearly stable based on Definition 29 and Definition 30.

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 (9) can be further reformulated as follows:

$$\begin{aligned} \frac{\partial}{\partial t}\bar{g}(t) &= -2\,\mathrm{Ric}(\bar{g}(t)) - \mathcal{L}_{\frac{\partial\varphi(t)}{\partial t}}\bar{g}(t) \\ \bar{g}(0) &= \bar{g}_0 + d \end{aligned} \tag{21}$$

where $d = \bar{g}(0) - \bar{g}_0$ is a metric perturbation deviated from the LNE metric $\bar{g}_0$. Corollary 15 guarantees the finite time existence of the Ricci-DeTurck flow w.r.t. L^2-norm perturbations, and then provides the necessary premise for proving its all time convergence.

Lemma 14 *(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*

$$\|\bar{g}(t) - \bar{g}_0\|_{L^\infty} < C\|\bar{g}(0) - \bar{g}_0\|_{L^\infty} < C \cdot \epsilon. \tag{22}$$

Corollary 15 *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*

$$\|\nabla^k d(t)\|_{L^2} \leq C_k \cdot t^{-k/2} \tag{23}$$

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

Proof When Lemma 14 satisfies in a finite time, according to (Deruelle and Kröncke, 2021), the Ricci-DeTurck flow with the LNE metric w.r.t. the L^2-norm perturbation exists. ■

4.3 All Time Convergence for L^2-norm Perturbations

In order to prove the all time stability of LNE metrics under Ricci–DeTurck flow, we need to construct $\bar{g}_0(t)$ that is 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 are able to establish a manifold via an L^2-neighbourhood $\mathcal{U}$ of integral $\bar{g}_0$ in the space of metrics

$$\tilde{\mathcal{F}} = \mathcal{F} \cap \mathcal{U}. \tag{24}$$

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, based on the previous work (Deruelle and Kröncke, 2021). In this way, $d(t) - d_0(t) = \bar{g}(t) - \bar{g}_0(t)$ holds because we write $d_0(t) = \bar{g}_0(t) - \bar{g}_0$.

According to Theorem 16, it is obvious that the L^2-norm metric perturbation w.r.t. the LNE metric can be dynamically decayed by the Ricci-DeTurck flow. See the details in Appendix D.

Theorem 16 *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 ϵ_1 is a small enough constant.*

Proof By Lemma 14, we have a constant $\epsilon_2 > 0$ such that $d(t) \in \mathcal{B}_{L^2}(0, \epsilon_2)$ holds. By Lemma 31 (in the second step) and Corollary 33 (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 15 and Lemma 31, 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, $\bar{g}(t)$ will converge to $\bar{g}(\infty) = \bar{g}_0 + d_0(\infty)$ as t approaches $+\infty$ based on the elliptic regularity. In other words, $d(t) - d_0(t)$ will converge to 0 as t goes to $+\infty$ 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 starting close to the LNE metric exists for all time, and the Ricci-DeTurck flow will converge to the LNE metric following with (Deruelle and Kröncke, 2021). ■

4.4 Perturbation Analysis

By proving the finite time existence of the Ricci-DeTurck flow with L^2-norm perturbations (Corollary 15), we then prove that L^2-norm perturbations w.r.t. the LNE metric converges for all time under the Ricci-DeTurck flow (Theorem 16). Based on (Sesum, 2006), we further yield $|\bar{g}(t) - \bar{g}_0(\infty)| < Ce^{-\epsilon_2 t}$ such that the metric perturbation exponentially converges.

Recall that we can reparameterize $\bar{g}(t)$ to $g(t) = \varphi^*(t)\bar{g}(t)$ via the pullback. Since the perturbation entirely comes from $\bar{g}(t)$ and has nothing to do with the time-dependent diffeomorphism $\varphi^*(t)$ in essence, it also exponentially converges for $g(t)$ under the Ricci flow when the existence of the solution of Ricci flow satisfies. And in Section 3, the metric $g(\boldsymbol{\xi}) = \delta_{ij} - \left[\tanh(\tau\boldsymbol{\xi})\tanh(\tau\boldsymbol{\xi})^\top\right]_{ij}$ we construct for the neural network is a kind of LNE metrics (based on Definition 13), so the perturbation for this metric satisfies the exponential decay under the Ricci flow. In what follows, the gradient mismatch in DNNs can be effectively overcome by employing the LNE manifolds under Ricci flow.

5. Discretized Neural Networks in LNE Manifolds

Up to now, we have theoretically solved the problem of gradient mismatch. Specifically, we have completed the construction of LNE manifolds for neural networks in Section 3 and exponentially decayed the metric perturbation in LNE manifolds in Section 4.

However, based on Lemma 11, there is the inverse of the LNE metric for the steepest descent gradient in the LNE manifold, which makes it difficult to be calculated in practice. Therefore, in this section, our aim is to approximate the inverse of the LNE metric, and further obtain the approximated gradient in the LNE manifold, that leads to the practical algorithm for training 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, whose form is given in Equation (1). Then, Zhou et al. (2016) applied STE to arbitrary bit-width discretized neural networks. Similarly, the generalization of STE in discretized neural networks yields

$$\frac{\partial L}{\partial \boldsymbol{w}} = \frac{\partial L}{\partial Q(\boldsymbol{w})} \cdot \mathbb{I}. \tag{25}$$

There is still a contradiction to be solved before the LNE manifold is introduced into the DNN. Based on Lemma 11, the LNE manifold is defined on the parameter $\boldsymbol{\xi}$ of all layers in a neural network. However, the gradient in back-propagation is defined layer-by-layer, i.e., on the weight $\boldsymbol{w}$ of each layer, which causes that the gradient update can not be associated with the LNE manifold. Fortunately, the LNE manifold can be layer-by-layer redefined by replacing $\boldsymbol{\xi}$ with $\boldsymbol{w}$, which is equivalent to defining the LNE manifold for each layer. And

on the basis of Lemma 11, the steepest descent gradient is rewritten as

$$\tilde{\partial}_{\boldsymbol{w}} = g^{-1}(\boldsymbol{w})\partial_{\boldsymbol{w}} = \left[\delta - \tanh(\tau \boldsymbol{w}) \tanh(\tau \boldsymbol{w})^{\top}\right]^{-1} \partial_{\boldsymbol{w}}, \tag{26}$$

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

$$\frac{\tilde{\partial} L}{\tilde{\partial} \boldsymbol{w}} = \left[\delta - \tanh(\tau \boldsymbol{w}) \tanh(\tau \boldsymbol{w})^{\top}\right]^{-1} \frac{\partial L}{\partial Q(\boldsymbol{w})}. \tag{27}$$

Further, the proposed gradient in above means that we are in the framework of solving the problem of gradient mismatch, similar to Equation (5). Note that the metric at this time is layer-by-layer LNE. On the other hand, gradient computation involves the inverse of the LNE metric, which greatly consumes computing resources. Hence, we propose 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 gives the largest variation in the objective per unit of variation along the layer-by-layer LNE manifold.

5.2 Strong Approximation

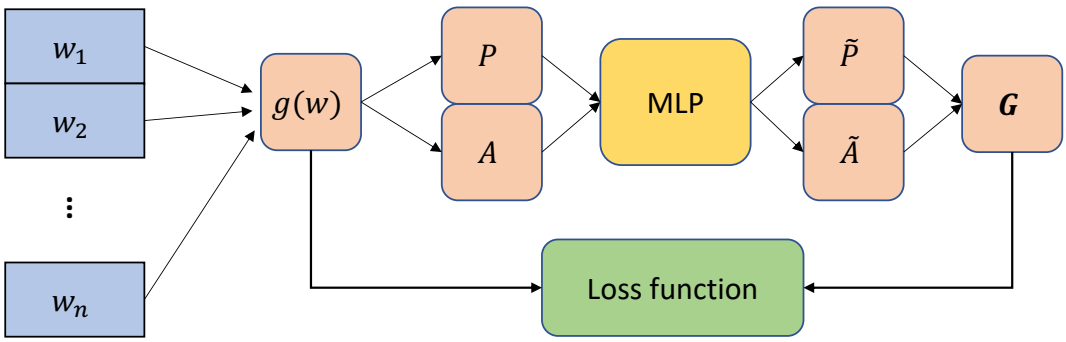


Figure 4: Flow chart of strong approximation of $g^{-1}(\boldsymbol{w})$. The new entries $\tilde{P}$ and $\tilde{A}$ produced by neural network form a matrix $\boldsymbol{G}$, which will multiply by the metric $g(\boldsymbol{w})$. As the loss function defined by Equation (28) decreases, the matrix $\boldsymbol{G}$ can be used to approximate the inverse of the metric $g(\boldsymbol{w})$.

For the $n \times n$ symmetric metric $g(\boldsymbol{w})$, it can be decomposed in terms of the combination of entries P and A, where P is the entries made up of the elements of the lower triangular matrix that contains $n(n-1)/2$ real parameters and A is the entries made up of the elements of the diagonal matrix that contains n real parameters.

Our aim is to approximate the inverse of the LNE metric, and further approximate the gradient in Equation (27). Based on the universal approximation theorem (Cybenko, 1989;

Hornik, 1991), 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). Inspired by this work, we also introduce a multi-layer perceptron (MLP) neural network, as shown in Figure 4, to minimize the loss function

$$\tilde{L} = \|\boldsymbol{I} - g(\boldsymbol{w})\boldsymbol{G}\|^2. \tag{28}$$

Hence, the matrix $\boldsymbol{G}$ can be used to strongly approximate the inverse of the metric $g(\boldsymbol{w})$.

5.3 Weak Approximation

In this subsection, we can also present a weak approximation for the inverse of the LNE metric with the efficient calculation.

Definition 17 *(Bhatia, 2013) For $\boldsymbol{A} \in \mathcal{R}^{n\times n}$, $\boldsymbol{A}$ is called **diagonally dominant** when it satisfies*

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

Definition 18 *(Bhatia, 2013) 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.*

Considering the properties of the LNE metric, we can easily make $g^{-1}(\boldsymbol{w})$ satisfy Definition 17 by adjusting the parameter τ. Furthermore, the existence of the inverse of the LNE metric can be guaranteed on the basis of Definition 18. By placing a higher requirement, the LNE metric is diagonally dominant. According to Corollary 19, the weak approximation of the gradient in the LNE manifold can be calculated, giving rise to a nice feature to facilitate the fast computation of the inverse.

Corollary 19 *Based on Definition 17 and Definition 18, the weak approximation of the gradient in the LNE manifold is defined as*

$$\tilde{\partial}_{\boldsymbol{w}} = \Big[\delta - \tanh(\tau\boldsymbol{w})\tanh(\tau\boldsymbol{w})^\top\Big]^{-1}\partial_{\boldsymbol{w}} \approx \Big[\delta + \tanh(\tau\boldsymbol{w})\tanh(\tau\boldsymbol{w})^\top\Big]\partial_{\boldsymbol{w}} \tag{29}$$

if the LNE metric is diagonally dominant.

Proof Considering the inverse of the LNE metric, due to the diagonally dominant property in Definition 17 and Definition 18, we can approximate $\big[\delta - \tanh(\tau\boldsymbol{w})\tanh(\tau\boldsymbol{w})^\top\big]^{-1}$ by ignoring the fourth-order small quantity $\sum O(\rho_a\rho_b\rho_c\rho_d)$, i.e.,

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

■

5.4 Training

Algorithm 1 An algorithm for the training of DNNs in the LNE manifold. We represent the gradient in the LNE manifold as $\tilde{\partial}$. For brevity, we ignore 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 τ and a learning rate η.

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

1: {Forward propagation}
2: **for** $i = 1; i \leq l; i++$ **do**
3: Discretize $\hat{\boldsymbol{W}}_i = Q(\boldsymbol{W}_i)$;
4: Compute $\boldsymbol{s}_i = \hat{\boldsymbol{W}}_i \hat{\boldsymbol{a}}_{i-1}$;
5: Discretize $\hat{\boldsymbol{a}}_i = Q\left(f \odot \boldsymbol{s}_i\right)$;
6: **end for**
7: {Loss derivative}
8: Compute $L = L(\boldsymbol{y}, \boldsymbol{z})$;
9: Compute $\partial_{\boldsymbol{a}_l} L = \frac{\partial L(\boldsymbol{y}, \boldsymbol{z})}{\partial \boldsymbol{z}}\big|_{\boldsymbol{z}=\hat{\boldsymbol{a}}_l}$;
10: {Backward propagation}
11: **for** $i = l; i \geq 1; i--$ **do**
12: Compute $\partial_{\boldsymbol{s}_i} L = \partial_{\boldsymbol{a}_i} L \odot f'(\boldsymbol{s}_i)$;
13: Compute $\partial_{\hat{\boldsymbol{W}}_i} L = \left(\nabla_{\boldsymbol{s}_i} L\right) \hat{\boldsymbol{a}}_{i-1}^\top$;
14: Compute $\tilde{\partial}_{\boldsymbol{W}_i} L = g^{-1}(\boldsymbol{W}_i) \partial_{\hat{\boldsymbol{W}}_i} L$ based on Equation (27);
15: Compute $\partial_{\hat{\boldsymbol{a}}_{i-1}} L = \hat{\boldsymbol{W}}_i^\top \left(\partial_{\boldsymbol{s}_i} L\right)$;
16: **end for**
17: {The parameters update}
18: **for** $i = l; i \geq 1; i--$ **do**
19: Update $\boldsymbol{W}_i \leftarrow \boldsymbol{W}_i - \eta \cdot \tilde{\partial}_{\boldsymbol{W}_i} L$;
20: **end for**
21: Update $\hat{\boldsymbol{\xi}} = \left[\mathrm{vec}\left(\hat{\boldsymbol{W}}_1\right)^\top, \mathrm{vec}\left(\hat{\boldsymbol{W}}_2\right)^\top, \ldots, \mathrm{vec}\left(\hat{\boldsymbol{W}}_l\right)^\top\right]^\top$;

Consequently, based on the previous work (Courbariaux et al., 2016), we give the practical algorithm of DNNs in the LNE manifold. As shown in Algorithm 1, this algorithm is similar to the general algorithm of DNNs, except for Line 14. Recall that, in Figure 1, the general DNNs use STE to simply copy the gradient, i.e., $\tilde{\partial}_{\boldsymbol{W}_i} L = \partial_{\hat{\boldsymbol{W}}_i} L$, but we match the gradient by introducing the LNE metric. Furthermore, we can practically calculate this gradient in Line 14 via the strong approximation or the weak approximation in above.

6. Ricci Flow Discretized Neural Networks

In this section, we propose Ricci flow discretized neural networks (RF-DNNs). When the Ricci flow is introduced, it implies that the background of DNNs discussed here is the LNE manifold. Our aim is to provide a practical solution for the metric perturbation and further for the problem of gradient mismatch. Therefore, we will focus on how to calculate the discrete Ricci flow in practice rather than just staying on theoretical analysis.

To relate the Ricci flow to neural networks, we need to discretize the Ricci flow and choose a suitable coordinate system. For the left hand side of the Ricci flow, we have established the connection between the LNE metric and neural networks in Section 3. Note that we use the form of the LNE metric in the calculation, and such metrics at this time are with perturbations. For the right hand side of the Ricci flow, we need to calculate the Ricci curvature tensor with the coordinate system. And this coordinate system is the key to linking the Ricci curvature and neural networks. Specifically, we define a method for calculating the Ricci curvature in such a way that the selection of coordinate systems is related to the input transformations, which implies that the Ricci curvature in neural networks reflects the effect of different input transformations on the parameter.

6.1 Ricci Curvature in Neural Networks

Now we consider the Ricci curvature tensor on the Riemannian metric g. According to Appendix A, its coordinate form can be expressed as

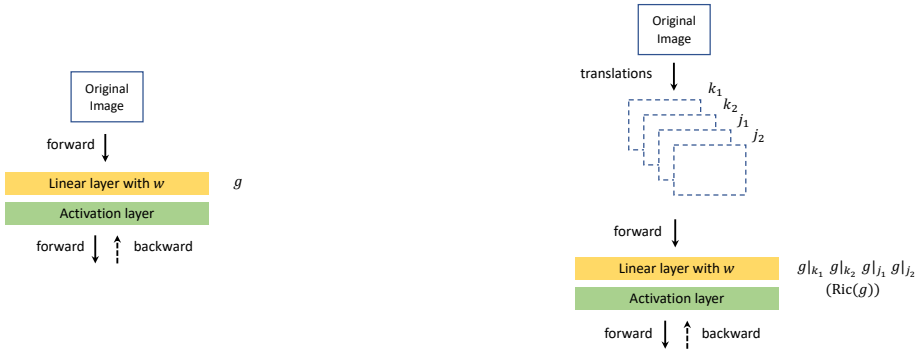
$$\begin{aligned} &-2\,\mathrm{Ric}(g) = -2R^i_{ikj} = 2R^i_{kij} \\ &= g^{ip}\left(\partial_i\partial_k g_{pj} - \partial_i\partial_j g_{pk} + \partial_p\partial_j g_{ik} + \partial_p\partial_k g_{ij}\right). \end{aligned} \tag{30}$$

In order to relate the Ricci curvature to neural networks, we define a method for calculating the Ricci curvature in such a way that the selection of coordinate systems is related to the input transformations. When the Ricci curvature is equal to zero, it means that different input transformations will not cause variations of the parameter.

Inspired by the previous work (Kaul and Lall, 2019), we can treat the terms ∂_i and ∂_p as variations for the translation and rotation of each input, respectively. In general, since the data augmentation does not involve the rotation in real-world applications such as image classification tasks (He et al., 2016; Shorten and Khoshgoftaar, 2019), we consider the translation instead of the rotation by discarding the index p, i.e., $\partial_p(\partial_j g_{ik} + \partial_k g_{ij}) = 0$ for the fairness of ablation studies. When considering one of the translation and rotation, g^{ip} will degenerate into δ^{ip} (identity matrix). Subsequently, $\partial_i\partial_k g$ and $\partial_i\partial_j g$ can be treated as variations for the row and column transformation of the input data w.r.t. the metric g, respectively. Further the Ricci curvature can be rewritten as

$$-2\,\mathrm{Ric}(g) = \partial_i\partial_k g_{pj} - \partial_i\partial_j g_{pk}. \tag{31}$$

Remark 20 *The choice of i and p is arbitrary (including k and j), and can even be other coordinate representations. Here, we just give it 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: By feeding the original image into the neural network and performing a forward and backward on the linear layer to update the weights $\boldsymbol{w}$, we can construct the metric structure $g(\boldsymbol{w})$ based on Section 5.1. In addition, we first process 4 different small translation transformations (k_1, k_2, j_1, and j_2) on the original image, and then input them into the neural network. By sequentially performing a forward and backward, we can obtain four metric structures ($g|_{k_1}$, $g|_{k_2}$, $g|_{j_1}$, and $g|_{j_2}$) under the corresponding translations. Combined with these metric structures, we can present the Ricci curvature structure $\mathrm{Ric}(g)$.

As shown in Figure 5, with the help of Equation (31), we yield the Ricci curvature with coordinate systems in terms of the difference equation:

$$-2\,\mathrm{Ric}(g) = \frac{g|_{k_1} - g|_{k_2}}{k_1 - k_2} - \frac{g|_{j_1} - g|_{j_2}}{j_1 - j_2} \tag{32}$$

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. Note that $g|_{k_1}$, $g|_{k_2}$, $g|_{j_1}$, and $g|_{j_2}$ are 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 $(j_1 - j_2)$ are translations less than 4 pixels, which is consistent with data augmentation (He et al., 2016).

6.2 Existence of Discrete Ricci Flow in Neural Networks

Recall that we previously considered the Ricci-DeTurck flow instead of the Ricci flow as the solution of the Ricci flow does not always exist based on Section 2.3. If we can guarantee that the solution of Ricci flow exists in neural networks, then we can use the Ricci flow to exponentially decay the metric perturbation based on Section 4.4.

We have considered the right hand side of the Ricci flow in Equation (6), i.e., the Ricci curvature tensor. Now we define the equivalent form of the left hand side of the Ricci flow

in terms of the difference equation:

$$\frac{\partial}{\partial t}g(t) := g(t+1) - g(t). \tag{33}$$

where $t \in \{0, 1, \cdots, T-1\}$ is a uniform partition of the interval $[0, T]$. In the training of neural networks, T is the total number of iterations. As the limit $T \to \infty$, the above formula holds on.

Combining Equation (32) and Equation (33) in neural networks, we present the expression of the discrete Ricci flow in terms of the difference equation:

$$\begin{aligned} g(t+1)|_{k_1} - g(t)|_{k_1} &= \frac{g(t)|_{k_1} - g(t)|_{k_2}}{k_1 - k_2} - \frac{g(t)|_{j_1} - g(t)|_{j_2}}{j_1 - j_2} \\ g(0)|_{k_1} &= \delta - \tanh(\tau \boldsymbol{w}) \tanh(\tau \boldsymbol{w})^\top \end{aligned} \tag{34}$$

To ensure the existence of the solution of the discrete Ricci flow, we are able to achieve this goal by adding a regularization into the loss function to constrain the discrete Ricci flow in DNNs. Following Equation (34), we further present the regularization:

$$N = \left\| g(t+1)|_{k_1} - g(t)|_{k_1} - \frac{g(t)|_{k_1} - g(t)|_{k_2}}{k_1 - k_2} + \frac{g(t)|_{j_1} - g(t)|_{j_2}}{j_1 - j_2} \right\|_{L^2}^2, \tag{35}$$

where $g(t)$ is ϵ-close to the LNE metric g_0 based on Definition 21. In other words, $g(t)$ is LNE metric with perturbations.

Definition 21 *(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 ϵ-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 ϵ-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 N in DNNs, the solution of the discrete Ricci flow exists when $N \to 0$. Simultaneously, the metric perturbation will exponentially converges $(g(t) \to g_0)$ as the evolution of the discrete Ricci flow.

6.3 Algorithm Design

By applying the constraints of the discrete Ricci Flow in layer-by-layer LNE manifold, we can finally alleviate the problem of gradient mismatch. Since the background is the LNE manifold, we can construct the satisfied gradient on the basis of Equation (27). Note that, at this time, the metric is time-dependent under the Ricci flow, i.e., $g_{\boldsymbol{w}}(t)$.

With the indicator function $\mathbb{I}$ to represent the constraints of the discrete Ricci flow, we yield the gradient under the discrete Ricci flow as follows:

$$\tilde{\partial}_{\boldsymbol{w}} L = g_{\boldsymbol{w}}^{-1}(t) \partial_{Q(\boldsymbol{w})} \cdot \mathbb{I}, \quad \text{where} \quad \mathbb{I} := \begin{cases} 1 & \text{if } |N| \leq \varepsilon \\ 0 & \text{otherwise} \end{cases} \tag{36}$$

Algorithm 2 An algorithm for the training our RF-DNNs in the LNE manifold. We define a parameter α for balancing the regularization to ensure the existence of the solution of the discrete Ricci flow. For brevity, we ignore 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 τ and a learning rate η.

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 $\hat{\boldsymbol{a}}_i = Q\left(f \odot \boldsymbol{s}_i\right)$;
6: **end for**
7: Compute the regularization N based on Equation (35);
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 $\tilde{\partial}_{\boldsymbol{W}_i} L = g_{\boldsymbol{W}_i}^{-1}(t) \partial_{\hat{\boldsymbol{W}}_i} L \cdot \mathbb{I}$ based on Equation (36);
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 $\hat{\boldsymbol{\xi}} = \left[\mathrm{vec}\left(\hat{\boldsymbol{W}}_1\right)^{\top}, \mathrm{vec}\left(\hat{\boldsymbol{W}}_2\right)^{\top}, \ldots, \mathrm{vec}\left(\hat{\boldsymbol{W}}_l\right)^{\top}\right]^{\top}$;

where we define a small constant ε. Furthermore, due to the error in estimating the the regularization with finite training time, we allow an ε-bounded error in the above formulation, which also implies that the cases are less than or equal to ε (rather than strictly equal to 0), all satisfy the discrete Ricci flow.

The overall process is shown in Algorithm 2. Compared with Algorithm 1, we add Line 7 and Line 15. In Line 7, we need to calculate the regularization to ensure that the solution of discrete Ricci flow exists. On the other hand, in Line 15, we calculate the gradient in the LNE manifold under the discrete Ricci flow, unlike Algorithm 1 which just calculates the gradient in the LNE manifold with perturbations. When we apply the Ricci flow, it means that the LNE manifold at this time is dynamic and anti-perturbation.

Remark 22 *In addition to using the discretized weight and activation, DNNs need to save the non-discretized weight and activation for gradient update. Note that the gradient of a DNN is non-discretized.*

6.4 Complexity Analysis

Based on Algorithm 2, one knows that the forward time complexity is about $\mathcal{O}(n^2)$ where the time complexity 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 about $\mathcal{O}(n)$. Then the time complexity of the gradients w.r.t. the weight (Line 15) and activation (Line 17) both are about $\mathcal{O}(n^2)$. Therefore, the backward time complexity is about $\mathcal{O}(2n^2)$. For a 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 way, the updated weights are only used to calculate the constraints of discrete Ricci flow, and the final weights can be obtained by a backward pass again. 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 our RF-DNN is still $\mathcal{O}(n^2)$. Hence, the complexity of the RF-DNN keep consistency with the general neural network.

7. Experiments

In this section, we design ablation studies to compare our RF-DNN [9] trained from scratch with other STE methods. Furthermore, given a pre-trained model, we evaluate the performance of the RF-DNN in comparison with several representative training-based methods on classification benchmark datasets. All the experiments implemented in Python are conducted with PyTorch (Paszke et al., 2019). The hardware environment is conducted on a Workstation with 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) consisting of natural color images with 32×32 pixels, respectively, 50k training and 10k test images, and we pick out 5k training images as a validation set from the training set. CIFAR-10 consists of images organized into 10 classes and CIFAR-100 into 100 classes. We adopt a standard data augmentation scheme (random corner cropping and random flipping) that is widely used for these two datasets. We normalize the images with the means of the channel and standard deviations in preprocessing.

ImageNet dataset: The ImageNet benchmark (Russakovsky et al., 2015) consists of 1.2 million high-resolution natural images, where the validation set contains 50k images. These images are organized into 1000 categories of the objects for training, which are resized

9. In order to calculate the gradient conveniently, we use the weak approximation of the inverse of the LNE metric in all experiments.

to 224 × 224 pixels before fed into the network. In the next experiments, we report our single-crop evaluation results using top-1 and top-5 accuracies.

We specify the discrete function, the composition of which has a significant influence on the performance and computation of DNNs. Specifically, the discrete function is able to simplify the calculations which are also varied depending on the different discrete values, e.g., fixed-point multiplication, SHIFT operation (Elhoushi et al., 2019) and XNOR operation (Rastegari et al., 2016) etc.

We mark Q^1 as the 1-bit discrete function:

$$Q^1(\cdot) = \text{sign}(\cdot) = \{-1, +1\}. \tag{37}$$

The k-bit, over 1-bit, discrete function can be marked as Q^k,

$$Q^{k>1}(\cdot) = \frac{2}{2^k - 1} \text{round}\left[(2^k - 1)\left(\frac{\cdot}{2\max|\cdot|} + \frac{1}{2}\right)\right] - 1 \tag{38}$$

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

7.2 Ablation Studies with STE Methods

In order to illustrate the superiority of RF-DNN against the problem of gradient mismatch, we compare our RF-DNN with three other methods by training from scratch. In Table 1, Table 2 and Table 3, we mark $\{-1, +1\}$ in '**Forward**', which indicates that the weights are binarized using Equation (37), i.e., -1 or $+1$, in the forward of DNNs. In the backward, the methods (Dorefa (Zhou et al., 2016), MultiFCG (Chen et al., 2019) and FCGrad (Chen et al., 2019)) use the different approximated gradients to update the weights. Here, we apply different ResNet models (He et al., 2016) to compare ablation studies.

The batch normalization with a batch size of 128 is used in the learning strategy, and Nesterov momentum of 0.9 (Dozat, 2016) is used in SGD optimization. For CIFAR, we set total training epochs as 200 and set a weight decay of 0.0005 where the learning rate is lowered by 10 times at epoch 80, 150, and 190, with the initial value of 0.1. For ImageNet, we set total training epochs as 100 and set the learning rate of each parameter group using a cosine annealing schedule with a weight decay of 0.0001. All experiments are conducted for 5 times, and the statistics of the last 10/5 epochs' test accuracies are reported for a fair comparison. Hence, we evaluate the accuracy performance in terms of (mean ± 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 setting and optimizer for fairness in the forward. The only difference is the gradient in the backward propagation. The performance in various models and datasets shows that our RF-DNN has significant improvement over other STE methods. The average results of multiple experiments are better than that of other methods, which may benefit from the alleviation of the gradient mismatch to make the loss function of DNNs more fully descended. In addition, the minor variances show that our training method is relatively stable, which also confirms our point of view.

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**	

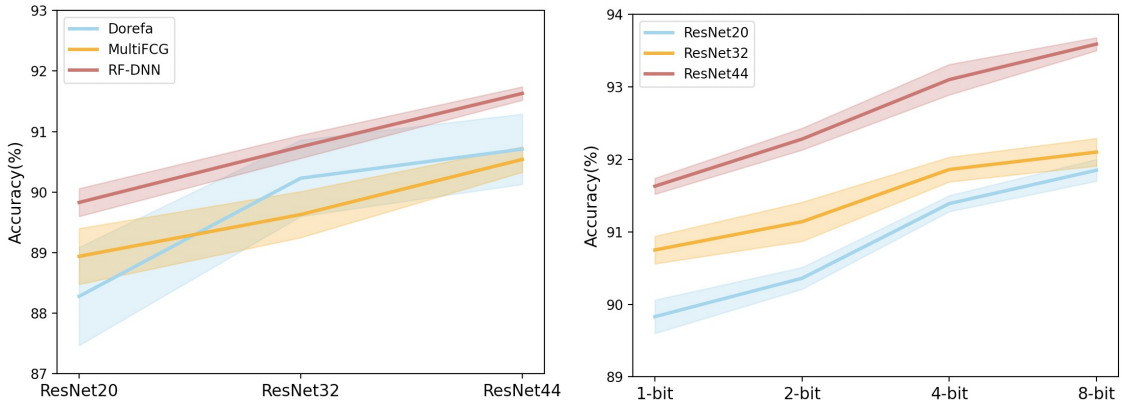
Table 2: The experimental results on CIFAR100 with ResNet56/110. The accuracy of full-precision (FP) baseline is reported by (Chen et al., 2019).

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**	

7.3 Convergence and Stability Analysis

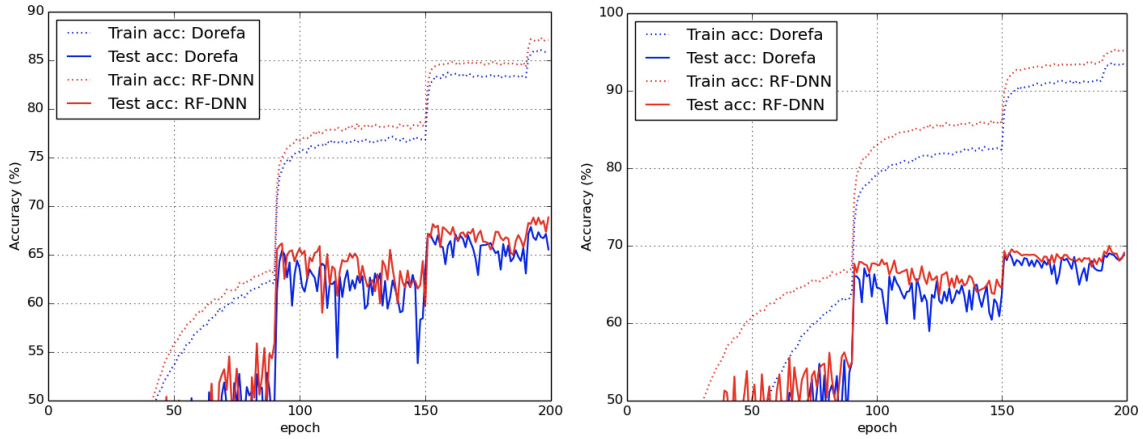
Since the standard deviations can reflect the convergence and stability of training to a certain extent, we visualize the Table 1 as shown in Figure 6(a). Intuitively, compared to Dorefa and MultiFCG, our proposed RF-DNN can better alleviate the perturbations caused by the gradient mismatch to achieve more stable performance. Furthermore, we also present the accuracy performance of RF-DNN with different bit width weight representations in Figure 6(b). And we see fairly consistent stability across different bit width and backbone models.

As shown in Figure 7, the RF-DNN can achieve 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



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

Figure 6: Accuracy performance (mean ± std) for ResNet20/32/44 on CIFAR10. The line and bar represent the mean and standard deviation of the different random seed results, respectively. (a) We compare our RF-DNN with Dorefa and MultiFCG via the 1bit weight representation, which is also the visualization of Table 1. (b) We present our RF-DNN via different bit width weight representations. Note that higher mean and lower deviation always implies 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 has a 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**	

and 1.05% higher on the test dataset with ResNet110. In addition, it can be seen from the fluctuation of the test curves in Figure 7 that the RF-DNN has a tremendous improvement compared with the Dorefa on the stability of training. From the training curve, our method significantly outperforms Dorefa, of course, this needs to be combined with the test curve to explain the superiority of RF-DNN. From the test curve, the accuracy of our method is always higher than that of Dorefa, and the corresponding volatility has improved. The experimental results verify that our theoretical framework is an effective solution against the gradient mismatch, further improving the training performance of DNNs.

7.4 Comparisons with Training-based Methods

In this experiment, based on a full-precision pre-trained model, we compare the RF-DNN with several state-of-the-art DNNs, e.g., 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), under the same bit width using Equation (37) or Equation (38). Note that **W** and **A** represent the bit width of weights and activations respectively in Table 4. Consequently, the experimental results show that our RF-DNN is able to achieve better performance than other recent state-of-the-art training-based methods, which seems to benefit from our effective solution of the gradient mismatch.

8. Conclusion and Future Work

Traditional discretized neural networks (DNNs) tell us that the weights can only take low-precision discrete values, as well as the activations. This makes DNNs' memory footprint very light compared to full-precision floating point networks. However, it makes their training difficult: to maintain discrete weights, in general, the gradient w.r.t. discrete weights is approximated with the Straight-Through Estimator (STE), which causes the update weight to differ from the gradient w.r.t. continuous weights (*gradient mismatch*).

This paper proposes a novel analysis of the gradient mismatch phenomenon through the lens of duality theory. This mismatch is then viewed as a metric perturbation in a Riemannian manifold. In theory, on the basis of the information geometry, we construct the LNE manifold for neural networks, such that forming the background to effecively

Table 4: The classification accuracy results on ImageNet and comparison with other training-based methods, with 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%

deal with a metric perturbation. By revealing the stability of LNE metrics with the L^2-norm perturbation under the Ricci-DeTurck flow, we subsequently introduce the Ricci flow Discretized Neural Network (RF-DNN) in practice by using the constraints of the discrete Ricci flow in the LNE manifold to alleviate the metric perturbation with the exponential convergence rate, giving rise to an appealing solution for STE in DNNs. Experimentally, our RF-DNN achieves improvements in both the stability and performance of DNNs.

In this paper, information geometry is a very important part of combining the geometric tool (Ricci flow) and neural networks. Our future research will continue to explore the

connection between neural networks and manifolds and aim to introduce geometric ideas to solve practical problems in machine 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 ω:

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

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 for the Ricci Flow

C.1 Proof for Lemma 5

Lemma 23 *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)
δ:	Euclidean metric (identity matrix)	Φ:	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^∞:	norm
∂:	partial derivative	∇:	covariant derivative
$\mathcal{L}$:	Lie derivative	Δ_{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
φ^*:	pullback	ϕ_*:	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 ϵ-ball with respect to the L^2-norm induced by $\bar{g}_0$ and centred at $\bar{g}_0$

Combining with Lemma 23, we can obtain the deformation equation because of $\nabla g = 0$,

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

■

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 (6), we have the following expression for the pullback metric $g(t)$

$$\frac{\partial}{\partial t}g(t) = \varphi^*(t)\frac{\partial}{\partial t}\bar{g}(t) + \varphi^*(t)\mathcal{L}_{\frac{\partial \varphi(t)}{\partial t}}\bar{g}(t) = -2\,\mathrm{Ric}(\varphi^*(t)\bar{g}(t)) = -2\varphi^*(t)\,\mathrm{Ric}(\bar{g}(t)). \tag{39}$$

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 24, 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 24 *(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 ∇ 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 28.

Theorem 25 *(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 26 *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*

$$\int_0^T \left|\frac{\partial}{\partial t}g_x(t)\right|_{g(t)} dt \leq C, \tag{40}$$

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

$$e^{-C}g_x(0) \leq g_x(T) \leq e^{C}g_x(0). \tag{41}$$

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

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

■

Corollary 27 *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 (6), 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 26. ■

Theorem 28 *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*

$$\lim_{t \to T} \left(\sup_{x \in \mathcal{M}} |\operatorname{Rm}_x(t)| \right) = \infty. \tag{42}$$

Proof For a contradiction, we assume that $|\operatorname{Rm}_x(t)|$ is bounded by a constant. It follows from Corollary 27 that the metrics $g(t)$ converges smoothly to a smooth metric $g(T)$. Based on Theorem 25, 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. ■

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

Appendix D. Proof for All Time Convergence in LNE Manifolds

Definition 29 *(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 $L_{g_0} := \Delta_{g_0} + 2\operatorname{Rm}(g_0)*$ is in $(-\infty, 0]$ where Δ_{g_0} is the Laplacian, when L_{g_0} acting on d_{ij} satisfies*

$$\begin{aligned} L_{g_0}(d) &= \Delta_{g_0} d + 2\operatorname{Rm}(g_0) * d \\ &= \Delta_{g_0} d + 2\operatorname{Rm}(g_0)_{iklj} d_{mn} g_0^{km} g_0^{ln}. \end{aligned} \tag{43}$$

Definition 30 *(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.*

We rewrite the Ricci-DeTurck flow (21) as an evolution of the difference $d(t) := \bar{g}(t) - \bar{g}_0$, such that

$$\begin{aligned} \frac{\partial}{\partial t} d(t) = \frac{\partial}{\partial t} \bar{g}(t) &= -2\operatorname{Ric}(\bar{g}(t)) + 2\operatorname{Ric}(\bar{g}_0) + \mathcal{L}_{\frac{\partial \varphi'(t)}{\partial t}} \bar{g}_0 - \mathcal{L}_{\frac{\partial \varphi(t)}{\partial t}} \bar{g}(t) \\ &= \Delta d(t) + \operatorname{Rm} * d(t) + F_{\bar{g}^{-1}} * \nabla^{\bar{g}_0} d(t) * \nabla^{\bar{g}_0} d(t) + \nabla^{\bar{g}_0} \left(G_{\Gamma(\bar{g}_0)} * d(t) * \nabla^{\bar{g}_0} d(t) \right), \end{aligned} \tag{44}$$

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^∞-norm w.r.t. the LNE metric $\bar{g}_0$, and mark generic constants as C or C_1.

Lemma 31 *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:*

$$\left\| \frac{\partial}{\partial t} d_0(t) \right\|_{L^2} \leq C \left\| \nabla^{\bar{g}_0(t)} \left(d(t) - d_0(t) \right) \right\|_{L^2}^2 . \tag{45}$$

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

Theorem 32 *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*

$$\left(\Delta d(t) + \mathrm{Rm}(\bar{g}_0) * d(t), d(t) \right)_{L^2} \leq -\alpha_{\bar{g}_0} \left\| \nabla^{\bar{g}_0} d(t) \right\|_{L^2}^2 \tag{46}$$

for all $\bar{g}(t) \in \tilde{\mathcal{F}}$ whose definition is given in Equation (24).

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 29 and Definition 30, $-L_{\bar{g}_0}$ 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. ■

Corollary 33 *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*

$$\|d(t) - d_0(t)\|_{L^2}^2 \geq C \int_0^T \left\| \nabla^{\bar{g}_0(t)} \left(d(t) - d_0(t) \right) \right\|_{L^2}^2 \mathrm{d}t. \tag{47}$$

Proof Based on Equation (44), 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 31 and Theorem 32, we further obtain

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

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

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

holds. ■

Appendix E. Proof for the Information Geometry

E.1 Proof for 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:

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

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

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

E.2 Proof for 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

T. Ajanthan, K. Gupta, P. Torr, R. Hartley, and P. Dokania. Mirror descent view for neural network quantization. In *International Conference on Artificial Intelligence and Statistics*, pages 2809–2817. PMLR, 2021.

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

S.-i. Amari. *Information geometry and its applications*, volume 194. Springer, 2016.

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.

A. 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.

J. L. Ba, J. R. Kiros, and G. E. Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

Y. Bai, Y.-X. Wang, and E. Liberty. Proxquant: Quantized neural networks via proximal operators. *arXiv preprint arXiv:1810.00861*, 2018.

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

R. H. Bamler. *Stability of Einstein metrics of negative curvature*. Princeton University, 2011.

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

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

Y. Bengio, N. Léonard, and A. Courville. Estimating or propagating gradients through stochastic neurons for conditional computation. *arXiv preprint arXiv:1308.3432*, 2013.

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

R. Bhatia. *Matrix analysis*, volume 169. Springer Science & Business Media, 2013.

L. 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.

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

Z. Cai, X. He, J. Sun, and N. 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.

J. Chen, L. Liu, Y. Liu, and X. Zeng. A learning framework for n-bit quantized neural networks toward fpgas. *IEEE Transactions on Neural Networks and Learning Systems*, pages 1–15, 2020. doi: 10.1109/TNNLS.2020.2980041.

S. Chen, W. Wang, and S. J. 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.

M. Courbariaux, I. Hubara, D. Soudry, R. El-Yaniv, and Y. Bengio. Binarized neural networks: Training deep neural networks with weights and activations constrained to+ 1 or-1. *arXiv preprint arXiv:1602.02830*, 2016.

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

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

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

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

T. Dozat. Incorporating nesterov momentum into adam. 2016.

M. Elhoushi, Z. Chen, F. Shafiq, Y. H. Tian, and J. Y. Li. Deepshift: Towards multiplication-less neural networks. *arXiv preprint arXiv:1905.13298*, 2019.

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

P. Gysel, J. Pimentel, M. Motamedi, and S. 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.

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

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

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

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

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

L. Hou, Q. Yao, and J. T. Kwok. Loss-aware binarization of deep networks. *arXiv preprint arXiv:1611.01600*, 2016.

A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017.

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

V. Jejjala, D. K. M. Pena, and C. Mishra. Neural network approximations for calabi-yau metrics. *arXiv preprint arXiv:2012.15821*, 2020.

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

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

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

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

A. Krizhevsky, G. Hinton, et al. Learning multiple layers of features from tiny images. 2009.

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

O. A. Ladyzhenskaia, V. A. Solonnikov, and N. N. Ural'tseva. *Linear and quasi-linear equations of parabolic type*, volume 23. American Mathematical Soc., 1988.

C. Leng, Z. Dou, H. Li, S. Zhu, and R. 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.

F. Li, B. Zhang, and B. Liu. Ternary weight networks. *arXiv preprint arXiv:1605.04711*, 2016.

Z. Liu, B. Wu, W. Luo, X. Yang, W. Liu, and K.-T. 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.

C. Louizos, M. Reisser, T. Blankevoort, E. Gavves, and M. Welling. Relaxed quantization for discretized neural networks. In *International Conference on Learning Representations*, 2019. URL `https://openreview.net/forum?id=HkxjYoCqKX`.

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

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

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

A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *arXiv preprint arXiv:1912.01703*, 2019.

H. Qin, R. Gong, X. Liu, M. Shen, Z. Wei, F. Yu, and J. 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.

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

M. Rastegari, V. Ordonez, J. Redmon, and A. Farhadi. Xnor-net: Imagenet classification using binary convolutional neural networks. In *European conference on computer vision*, pages 525–542. Springer, 2016.

O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, et al. Imagenet large scale visual recognition challenge. *International journal of computer vision*, 115(3):211–252, 2015.

H. Sak, A. W. Senior, and F. Beaufays. Long short-term memory recurrent neural network architectures for large scale acoustic modeling. 2014.

O. C. Schnürer, F. Schulze, and M. Simon. Stability of euclidean space under ricci flow. *arXiv preprint arXiv:0706.0421*, 2007.

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

N. Sheridan and H. Rubinstein. Hamilton's ricci flow. *Honour thesis*, 2006.

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

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

J. Yang, X. Shen, J. Xing, X. Tian, H. Li, B. Deng, J. Huang, and X.-s. Hua. Quantization networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

A. Zhou, A. Yao, Y. Guo, L. Xu, and Y. Chen. Incremental network quantization: Towards lossless cnns with low-precision weights. *arXiv preprint arXiv:1702.03044*, 2017.

A. Zhou, A. Yao, K. Wang, and Y. 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.

S. Zhou, Y. Wu, Z. Ni, X. Zhou, H. Wen, and Y. Zou. Dorefa-net: Training low bitwidth convolutional neural networks with low bitwidth gradients. *arXiv preprint arXiv:1606.06160*, 2016.

C. Zhu, S. Han, H. Mao, and W. J. Dally. Trained ternary quantization. *arXiv preprint arXiv:1612.01064*, 2016.

F. Zhu, R. Gong, F. Yu, X. Liu, Y. Wang, Z. Li, X. Yang, and J. 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.