Feedback Gradient Descent: Efficient and Stable Optimization with Orthogonality for DNNs

Fanchen Bu, Dong Eui Chang
School of Electrical Engineering, KAIST
{boqvezen97, dechang}@kaist.ac.kr

Abstract

The optimization with orthogonality has been shown useful in training deep neural networks (DNNs). To impose orthogonality on DNNs, both computational efficiency and stability are important. However, existing methods utilizing Riemannian optimization or hard constraints can only ensure stability while those using soft constraints can only improve efficiency. In this paper, we propose a novel method, named Feedback Gradient Descent (FGD), to our knowledge, the *first* work showing *high efficiency and stability simultaneously*. FGD induces orthogonality based on the simple yet indispensable Euler discretization of a continuous-time dynamical system on the tangent bundle of the Stiefel manifold. In particular, inspired by a numerical integration method on manifolds called Feedback Integrators, we propose to instantiate it on the tangent bundle of the Stiefel manifold for the *first* time. In the extensive image classification experiments, FGD comprehensively outperforms the existing state-of-the-art methods in terms of accuracy, efficiency, and stability.

1 Introduction

During the prosperous and ongoing development of deep neural networks (DNNs), it has been shown that imposing orthogonality on the parameters can help improve the network performance, which has attracted substantial attention with theoretical analyses (Saxe, McClelland, and Ganguli 2014; Desjardins et al. 2015; Harandi and Fernando 2016).

Many researchers have been trying to practically impose orthogonality during the training of DNNs. As orthogonality can be regarded as a property of the Stiefel manifold, Riemannian optimization is a typical and direct technique to maintain orthogonality, which has been widely studied in the optimization field (Smith 1994; Edelman, Arias, and Smith 1998; Rapcsák 2002; Absil, Mahony, and Sepulchre 2009; Absil and Malick 2012; Wen and Yin 2013; Bonnabel 2013; Jiang and Dai 2015). However, most Riemannian optimization algorithms are computationally expensive due to the complex retraction or projection. They are especially computationally prohibitive when applied to the state-of-the-art DNNs that have numerous parameters (Ozay and Okatani 2018), which has been demonstrated in the comparisons done by a previous work (Li, Fuxin, and Todorovic 2019). Without great loss of orthogonality, an alternative is to find a transformation or mapping, keeping the hard constraints on orthogonality and allowing normal Euclidean optimization on the transformed or mapped parameters (Huang et al. 2018; Li, Fuxin, and Todorovic 2019). Although these algorithms are much faster than Riemannian optimization algorithms, they are still considerably time-consuming when applied to DNNs. To further improve the efficiency, another intuitive way is to apply soft constraints (Rodríguez et al. 2017; Xie, Xiong, and Pu 2017; Bansal, Chen, and Wang 2018; Jia et al. 2017; Huang et al. 2020; Wang et al. 2020) by adding a penalty term into the loss function. Nevertheless, such a way fails to maintain orthogonality during the training process. Therefore, it is imperative to develop a new algorithm with both high efficiency and high numerical stability.

In this paper, we propose a novel method, named Feedback Gradient Descent (FGD), to our knowledge, the *first* work showing *high efficiency and stability simultaneously*. FGD induces orthogonality based on the simple Euler discretization of a continuous-time dynamical system (CTDS). In particular, we start from building a CTDS on the tangent bundle of the Stiefel manifold to represent the gradient descent (GD) process *with momentum* and orthogonality by following the idea of the Euclidean counterparts, i.e., the corresponding CTDSs of GD (Baldi 1995), especially GD *with momentum* (Qian 1999).

To construct the corresponding optimization algorithm, it is indispensable yet nontrivial to discretize the CTDS on the tangent bundle. The direct application of usual discretization techniques such as Euler's method for ordinary differential equations is inadequate because the trajectories of the discrete-time system can go off the manifold. Meanwhile, usual structure-preserving discretization methods such as variational or symplectic integrators (Haier, Lubich, and Wanner 2006) are complex and computationally expensive. Inspired by Feedback Integrators (FI) (Chang, Jiménez, and Perlmutter 2016), which is a numerical integration method for CTDSs on manifolds, we convert the discretization problem on the tangent bundle to one on a Euclidean space. Specifically, following the framework of FI, we extend the CTDS representing GD on the tangent bundle to an ambient Euclidean space and modify the extended CTDS by adding a feedback term pulling the variables back to the tangent bundle so that the tangent bundle becomes a local attractor. Moreover, we prove a new theorem, Theorem 4.3, to directly show exponential stability while FI only guarantees asymptotic stability. Although several applications of FI have been proposed (Chang 2018; Chang and Perlmutter 2019; Chang, Phogat, and Choi 2019; Ko et al. 2021; Park et al. 2021), we are the first to instantiate the FI framework for a CTDS on the tangent bundle of the Stiefel manifold, as well as the first to apply the framework to an optimization problem, especially the GD problem with orthogonality. Since the constructed CTDS is in a Euclidean space, we can efficiently discretize it without using any time-consuming operations with the stability carried over to the discretized system.

In summary, the major theoretical contributions are fourfold.

- To the best of our knowledge, FGD is the *first* optimization algorithm that imposes *orthogonality* with *momentum* for DNNs based on a *CTDS*;
- We are the *first* to instantiate the FI framework for a CTDS on the tangent bundle of the *Stiefel manifold*, and also the first to apply the framework to the *GD* problem;
- We generalize and improve the idea of FI, providing a new theorem (Theorem 4.3) showing *exponential stability* rather than the original asymptotic stability;
- FGD provides a way to apply optimization *directly* on the parameters and to maintain their *orthogonality* without using any time-consuming operations.

We conduct extensive experiments on the image classification task. We use a range of widely-used DNNs such as WideResNet (Zagoruyko and Komodakis 2016), ResNet (He et al. 2016a,b), VGG (Simonyan and Zisserman 2015), and PreActResNet (He et al. 2016b), and the popular benchmark datasets such as CIFAR-10/100 (Krizhevsky, Hinton et al. 2009), SVHN (Netzer et al. 2011), and ImageNet (Deng et al. 2009). We compare FGD with stochastic gradient descent (SGD) (Robbins and Monro 1951; Kiefer, Wolfowitz et al. 1952) *with momentum* and several state-of-the-art methods imposing soft or hard orthogonality (Rodríguez et al. 2017; Jia et al. 2017; Huang et al. 2018; Bansal, Chen, and Wang 2018; Li, Fuxin, and Todorovic 2019; Wang et al. 2020; Huang et al. 2020). FGD consistently outperforms the other baseline methods in terms of accuracy with favorable efficiency and stability. Especially, FGD achieves the highest accuracy in most cases and consumes less or similar training time than those of the methods using soft constraints. It also shows strong numerical stability comparable to previous methods imposing hard constraints.

2 Related Work

During the development of neural networks, orthogonality was first shown to be useful in mitigating the vanishing or exploding gradients problem (Bengio, Simard, and Frasconi 1994), especially on recurrent neural networks (RNNs) (Pascanu, Mikolov, and Bengio 2013; Le, Jaitly, and Hinton 2015; Wisdom et al. 2016; Arjovsky, Shah, and Bengio 2016; Jing et al. 2017; Hyland and Rätsch 2017; Vorontsov et al. 2017; Helfrich and Ye 2020). To improve the efficiency of the optimization algorithms with orthogonality, many techniques have been utilized, e.g., householder reflections (Mhammedi et al. 2017), Cayley transform (Helfrich, Willmott, and Ye 2018; Maduranga, Helfrich, and Ye 2019), and exponential-map-based parameterization (Lezcano Casado 2019; Lezcano-Casado and Martınez-Rubio 2019).

Orthogonality has also shown with theoretical foundation its ability to help the training of general deep neural networks (DNNs) (Saxe, McClelland, and Ganguli 2014; Harandi and Fernando 2016; Liu et al. 2021b), especially convolutional neural networks (CNNs) (Wang et al. 2020; Trockman and Kolter 2021; Liu et al. 2021a), by reducing the feature redundancy (Chen et al. 2017), stabilizing the distribution of activations over layers (Desjardins et al. 2015), and so on. Moreover, orthogonality has also been substantiated to be helpful in the training of generative adversarial networks (GANs) (Brock et al. 2017; Miyato et al. 2018; Brock, Donahue, and Simonyan 2018; Huang et al. 2020). It is also notable that even merely the initialization with orthogonality is helpful (Mishkin and Matas 2016). One may expect to directly apply the existing methods on RNNs to general DNNs and CNNs. However, the methods on RNNs are usually limited to square matrices of small size, which makes it difficult to apply them directly to general DNNs and CNNs with numerous sizable parameters, especially when the computational resources are limited. Many researchers have been trying to propose practically applicable optimization algorithms on DNNs, especially CNNs, with orthogonality, which we focus on in this paper.

Naturally, as orthogonality can be seen as a property of the Stiefel manifold, a straightforward way is to directly utilize Riemannian optimization algorithms on the Stiefel manifold. However, although this field has continuous theoretical development, the direct application of Riemannian optimization requires computationally expensive retraction or projection to keep the parameters on the manifold (Ozay and Okatani 2018), which significantly increases the training time as demonstrated in a previous work (Li, Fuxin, and Todorovic 2019) and can impair the stability of training (Harandi and Fernando 2016; Huang et al. 2018, 2020).

Hard Constraints. To address this problem without great loss of orthogonality, some previous methods, as some researchers have done on RNNs, use a parameter transformation or mapping from the Stiefel manifold to the Euclidean space, which still keeps hard constraints on orthogonality. For example, a previous work (Huang et al. 2018) finds a closed-form solution on the Stiefel manifold that minimizes the distance to the parameter on Euclidean space to be updated, where computationally expensive and numerically unstable eigendecomposition is required, which is also analyzed and mentioned in their later work (Huang et al. 2020); another work (Li, Fuxin, and Todorovic 2019) proposes an iterative estimation of the retraction mapping based on the Cayley transform, which still needs considerable additional time as shown in the theoretical analyses and the practical experiments.

Soft Constraints. Alternatively and intuitively, one can use soft constraints by adding a penalty term in the loss function during the training to impose orthogonality with a low computational cost, which, nevertheless, means that the algorithms can hardly guarantee the maintenance of orthogonality, just like other regularization-based methods. Many different kinds of penalty terms have been proposed, such as regularization based on the Frobenius norm and its variants (Rodríguez et al. 2017; Xie, Xiong, and Pu 2017; Bansal, Chen, and Wang 2018), regularization based the singular value (Jia et al. 2017; Huang et al. 2020), and regularization based on doubly block Toeplitz matrices (Wang et al. 2020).

Unlike all the above existing methods, our method does not use any retraction, projection, transformation, or mapping but performs the optimization directly on the orthogonal parameters.

3 Preliminaries

We use $\langle\cdot,\cdot\rangle$ to denote the Euclidean inner product of matrices, i.e., $\langle A, B\rangle = \operatorname{tr}(A^T B)$, for any two matrices A and B of the same size, and we use $\|\cdot\|$ to denote the norm induced from the above inner product. We use Sym to denote the symmetrization operator defined by $\operatorname{Sym}(A) = \frac{1}{2}(A + A^T)$, for any square matrix A.

The Stiefel manifold with parameters n and p consists of orthonormal p-frames in $\mathbb{R}^n$. Formally, in this paper, we use the compact Stiefel manifold that is defined as $\operatorname{St}(n,p) = \{X \in \mathbb{R}^{n\times p} : X^T X = I_p\}$, where $n \geq p$ and I_p is the $p \times p$ identity matrix. Taking $\operatorname{St}(n,p)$ as an embedded submanifold of $\mathbb{R}^{n\times p}$ and using the Euclidean inner product, the tangent space of $\operatorname{St}(n,p)$ at $X \in \operatorname{St}(n,p)$ is expressed as $T_X\operatorname{St}(n,p) = \{Z \in \mathbb{R}^{n\times p} : \operatorname{Sym}(X^T Z) = 0\}$, whose union over all points of the Stiefel manifold is the tangent bundle of the Stiefel manifold given by $T\operatorname{St}(n,p) = \{(X,Y) : X \in \operatorname{St}(n,p), Y \in T_X\operatorname{St}(n,p)\}$. For any $M \in \mathbb{R}^{n\times p}$, we can project M onto the tangent space $T_X\operatorname{St}(n,p)$ at X by $f_X(M) = M - X\operatorname{Sym}(X^T M)$, thus the Riemannian gradient of a differentiable function F is $\operatorname{grad} F(X) = \nabla F(X) - X\operatorname{Sym}(X^T\nabla F(X))$, where $\nabla F(\cdot)$ denotes the Euclidean gradient of F as a function on $\mathbb{R}^{n\times p}$.

4 Feedback Gradient Descent

Consider a generic optimization problem. Let $\mathcal{D}$ denote the training dataset, and let $\mathcal{L}(\cdot)$ denote the loss function that is nonnegative on Θ, where Θ is the domain in which the parameters θ are optimized, and θ can be either a vector or a matrix. The objective is to find

$$\theta^* = \arg\min_{\theta\in\Theta}\mathcal{L}(\theta). \tag{1}$$

As a well-known optimization method applicable to solve the problem (1) with Θ being Euclidean space without further restrictions, the discrete-time update of gradient descent *with momentum*

$$\begin{aligned}\theta &\leftarrow \theta + \eta\phi \\ \phi &\leftarrow (1-\gamma)\phi - \nabla\mathcal{L}(\theta)\end{aligned} \tag{2}$$

corresponds to the following CTDS through the semi-implicit Euler method with step-size η:

$$\begin{bmatrix}\dot{\theta}\\ \dot{\phi}\end{bmatrix} = \begin{bmatrix}\phi \\ (-\gamma\phi - \nabla\mathcal{L}(\theta))/\eta\end{bmatrix}, \tag{3}$$

where ϕ represents the changing rate of θ and $\gamma > 0$ is the momentum coefficient.

However, the above systems (2) and (3) are only for Euclidean space and thus cannot be directly applied to the case when Θ is the Stiefel manifold. To design a discrete-time counterpart of (2) with orthogonality, we start from the construction of a CTDS defined on the tangent bundle of the Stiefel manifold that is analogous to (3) and represents the gradient descent process with a momentum-like term.

Specifically, we have the following CTDS:

$$\begin{bmatrix}\dot{\theta}\\ \dot{\phi}\end{bmatrix} = \begin{bmatrix}\phi \\ -\theta\phi^T\phi + (D - \theta\operatorname{Sym}(\theta^T D))/\eta\end{bmatrix} \tag{4}$$

with $\theta, \phi \in \mathbb{R}^{n\times p}$, where $\eta > 0$ will be later used in the discrete-time algorithm as the step-size and

$$D = D(\theta,\phi) := -\gamma\phi - \nabla\mathcal{L}(\theta). \tag{5}$$

The following lemma shows that the dynamical system (4) is indeed well defined on $T\operatorname{St}(n,p)$.

Lemma 4.1 *For the dynamical system* (4) *defined on* $\mathbb{R}^{n\times p}\times\mathbb{R}^{n\times p}$, *if* $\theta(0)^T\theta(0) = I$ *and* $\operatorname{Sym}(\theta(0)^T\phi(0)) = 0$, *then* $\theta(t)^T\theta(t) = I$ *and* $\operatorname{Sym}(\theta(t)^T\phi(t)) = 0$ *hold for all* $t \geq 0$.

Proof. Refer to the supplementary material for the proofs that are not given in the main text. ■

The following lemma shows the asymptotic stability on $T\operatorname{St}(n,p)$ of the dynamical system (4).

Lemma 4.2 *Assume that* $\theta^* = \arg\min_{\theta\in\operatorname{St}(n,p)}\mathcal{L}(\theta)$ *uniquely exists, and let* $c_0 \geq 0$ *such that* $(\theta^*, 0)$ *is the only point in* $\{(\theta,0) \in \Omega : \operatorname{grad}_\theta\mathcal{L}(\theta) = 0\}$, *where* $\Omega = \{(\theta,\phi) \in T\operatorname{St}(n,p) : \frac{\eta}{2}\|\phi\|^2 + \mathcal{L}(\theta) \leq c_0\}$. *Assume that* $\mathcal{L}$ *is* C^2 *on* $\{\theta \in \operatorname{St}(n,p) : \mathcal{L}(\theta) \leq c_0\}$. *Then each trajectory of* (4) *starting in* Ω *stays in* Ω *for all forward time and asymptotically converges to* $(\theta^*, 0)$ *as time tends to infinity.*

Refer to the supplementary material for a local version of Lemma 4.2.

It now remains to discretize the system (4). However, we cannot just use usual discretization techniques on Euclidean space since the trajectories of the discretized system may go off the manifold, and the usual structure-preserving discretization methods can be complicated and computationally expensive. To address these problems, we propose to extend system (4) to an ambient Euclidean space and modify it by adding a feedback term so that the original domain becomes locally attractive. After doing so, we can apply any off-the-shelf Euclidean discretization method to get our final optimization algorithm with the stability carried over to the discretized system.

Let $W = S \times \mathbb{R}^{n\times p}$, where

$$S = \{\theta \in \mathbb{R}^{n\times p} : \|\theta^T\theta - I\| < 1\} \tag{6}$$

is an open neighborhood of $\operatorname{St}(n,p)$ such that $\theta^T\theta$ is invertible for all $\theta \in S$. Consider the following dynamical system defined on W:

$$\begin{bmatrix}\dot{\theta}\\ \dot{\phi}\end{bmatrix} = X(\theta,\phi) - \frac{\alpha}{4}\begin{bmatrix}\theta(I - (\theta^T\theta)^{-1}) \\ \theta(\theta^T\theta)^{-1}(\phi^T\theta(\theta^T\theta)^{-1} + \theta^T\phi)\end{bmatrix}, \tag{7}$$

where

$$X(\theta,\phi)=\begin{bmatrix}X_\theta(\theta,\phi)\\X_\phi(\theta,\phi)\end{bmatrix}$$
$$=\begin{bmatrix}\phi-\theta(\theta^T\theta)^{-1}\operatorname{Sym}(\theta^T\phi)\\ \theta(\theta^T\theta)^{-1}((\theta^T\theta)^{-1}\theta^T\phi\operatorname{Sym}(\theta^T\phi)-\phi^T\phi)+\\ (D-\theta(\theta^T\theta)^{-1}\operatorname{Sym}(\theta^T D))/\eta\end{bmatrix}$$

with D defined in (5), and $\alpha>0$ is the feedback coefficient. Note that (7) is identical to (4) on $T\mathrm{St}(n,p)$. Let $V:\mathbb{R}^{n\times p}\times\mathbb{R}^{n\times p}\to\mathbb{R}_{\geq 0}$ be a function defined as

$$V(\theta,\phi)=\frac{k_1}{4}\left\|\theta^T\theta-I\right\|^2+\frac{k_2}{2}\left\|\operatorname{Sym}(\theta^T\phi)\right\|^2, \quad (8)$$

where $k_1,k_2>0$. We have $V^{-1}(0)=T\mathrm{St}(n,p)$ and

$$\nabla V(\theta,\phi)=\begin{bmatrix}\nabla_\theta V(\theta,\phi)\\\nabla_\phi V(\theta,\phi)\end{bmatrix}$$
$$=\begin{bmatrix}k_1\theta(\theta^T\theta-I)+k_2\phi\operatorname{Sym}(\theta^T\phi)\\k_2\theta\operatorname{Sym}(\theta^T\phi)\end{bmatrix}. \quad (9)$$

We have the following theorem showing that the tangent bundle of the Stiefel manifold is a locally attractive submanifold of the dynamical system (7) on W with exponential stability.

Theorem 4.3 *For each $0<c<k_1/4$, $V^{-1}([0,c])$ is a subset of W and each trajectory of (7) starting in $V^{-1}([0,c])\subset W$ stays in $V^{-1}([0,c])$ for all forward time and exponentially converges to $V^{-1}(0)=T\mathrm{St}(n,p)$ as time tends to infinity.*

Proof. To proof the theorem, we make use of the following lemmas.

Lemma 4.4 *For each $0<c<k_1/4$, the set of all critical points of V in $V^{-1}([0,c])$ is $V^{-1}(0)$.*

Lemma 4.5 *$\langle\nabla V(\theta,\phi),X(\theta,\phi)\rangle=0,\forall(\theta,\phi)\in W$.*

We define a function $L:W\to\mathbb{R}^{2n\times 2n}$ as

$$L(\theta,\phi)=\begin{bmatrix}\frac{1}{4k_1}\theta(\theta^T\theta)^{-2}\theta^T & L_1(\theta,\phi)\\ L_1^T(\theta,\phi) & L_2(\theta,\phi)\end{bmatrix} \quad (10)$$

where $L_1(\theta,\phi)=-\frac{1}{4k_1}\theta(\theta^T\theta)^{-2}\theta^T\phi(\theta^T\theta)^{-1}\theta^T$ and $L_2(\theta,\phi)=\frac{1}{4k_1}\theta(\theta^T\theta)^{-1}\phi^T\theta(\theta^T\theta)^{-2}\theta^T\phi(\theta^T\theta)^{-1}\theta^T+\frac{1}{2k_2}\theta(\theta^T\theta)^{-2}\theta^T$.

Lemma 4.6 *$\langle\nabla V(\theta,\phi),L(\theta,\phi)\nabla V(\theta,\phi)\rangle=V(\theta,\phi),\forall(\theta,\phi)\in W$.*

By direct computation, it is easy to check that (7) can be written as

$$\begin{bmatrix}\dot\theta\\\dot\phi\end{bmatrix}=X(\theta,\phi)-\alpha L(\theta,\phi)\nabla V(\theta,\phi). \quad (11)$$

By Lemma 4.5 and 4.6, along each trajectory $(\theta(t),\phi(t))$ of (7) starting in $V^{-1}([0,c])\subset W$,

$$\frac{d}{dt}V=\langle\nabla V,X-\alpha L\nabla V\rangle=\langle\nabla V,-\alpha L\nabla V\rangle=-\alpha V, \quad (12)$$

which implies that

$$V(\theta(t),\phi(t))=e^{-\alpha t}V(\theta(0),\phi(0)) \quad (13)$$

for all $t\geq 0$. Combined with Lemma 4.4, this completes the proof. ■

Algorithm 1: Feedback Gradient Descent

```
**Input:** learning rate $\eta$, loss function $\mathcal{L}$, momentum coefficient $\gamma$, feedback coefficient $\alpha$
Initialize $\theta$ and $\phi$ such that $\theta^T\theta=I$, and $\phi=\theta\operatorname{Sym}(\theta^T\nabla\mathcal{L}(\theta))-\nabla\mathcal{L}(\theta)$
while training do
  $\theta\leftarrow\theta+\eta[X_\theta(\theta,\phi)-\frac{\alpha}{4}\theta(I-(\theta^T\theta)^{-1})]$
  $\phi\leftarrow\phi+\eta[X_\phi(\theta,\phi)-\frac{\alpha}{4}\theta(\theta^T\theta)^{-1}(\phi^T\theta(\theta^T\theta)^{-1}+\theta^T\phi)]$
end while
```

Remark 4.7 *Informally speaking, Lemma 4.5 says that the vector field X does not change the value of V and Lemma 4.6 implies that the added term $-\alpha L\nabla V$ keeps decreasing the value of V. Therefore, the dynamical system (7) consisting of these two parts can converge to the tangent bundle of the Stiefel manifold where $V=0$ and dynamical system (7) coincides with dynamical system (4). Moreover, this allows us to perform usual Euclidean discretization on system (7) with the trajectories of the corresponding discretized dynamical system kept close to the tangent bundle of the Stiefel manifold.*

By straightforward discretization of the system (7) with the semi-implicit Euler method with step size η, we have the discretized Algorithm 1 as the update rule for the parameters with orthogonality.

In the practical application of Algorithm 1, we approximately use $2I-\theta^T\theta$ for $(\theta^T\theta)^{-1}$, which is a truncation of the Neumann series (Zhu, Li, and Liang 2015). This approximation does not cause great loss of precision because $\theta^T\theta$ indeed stays around the identity matrix and it can considerably reduce the computational complexity as the inversion operation is time-consuming, especially for large matrices.

Theorem 4.8 *With the approximation $(\theta^T\theta)^{-1}\approx 2I-\theta^T\theta$, the additional time complexity of Algorithm 1 is $O(p^2n)$.*

FGD uses only matrix multiplications while the previous methods additionally use computationally expensive SVD, QR decomposition, etc. In general, this approximation is inappropriate. For example, the Cayley transform also requires the computation of matrix inversion, but the inversion is not necessarily around the identity matrix. It is notable that even if one directly uses inversion operation, it will inevitably have numerical errors when using any existing deep learning framework.

Theorem 4.9 *Assume there exists a constant $c>0$ such that $\|\phi\|\leq c$ for all timesteps when using Algorithm 1 with the approximation $(\theta^T\theta)^{-1}\approx 2I-\theta^T\theta$. For any given $0<\epsilon<1$, there exist $\eta=\eta(\epsilon)>0$ and $\alpha=\alpha(\epsilon)>0$ so that $\left\|\theta^T\theta-I\right\|\leq\epsilon$ holds for all timesteps.*

The rationality of this approximation is also bolstered by the experimental results in Section 5 in terms of accuracy, complexity, and numerical stability.

5 Experiments

We conduct comprehensive experiments using various models such as WideResNet, ResNet, VGG, and PreActResNet, on CIFAR-10/100, SVHN, and ImageNet datasets. We follow the official settings for the training-testing split of the datasets, the pre-processing, the data augmentation, etc.

To evaluate the performance, we compare our method with stochastic gradient descent (SGD) *with momentum* and several state-of-the-art optimization algorithms imposing orthogonality such as OCNN (Wang et al. 2020), ONI (Huang et al. 2020), OMDSM (Huang et al. 2018), Cayley SGD/ADAM (Li, Fuxin, and Todorovic 2019), SRIP (Bansal, Chen, and Wang 2018), SVB (Jia et al. 2017), and LCD (Rodríguez et al. 2017).

When we list the experimental results, those without $*$ are the results claimed in the original papers and those with $*$ are obtained by running the corresponding open-source code on our machines.

All the experiments are on the image classification task and all the models used are convolutional neural networks (CNNs). In CNNs, the weight parameter of a convolutional layer has a size of $c_o \times c_i \times K_1 \times K_2$, where c_o and c_i are the channel numbers of output and input, respectively, and (K_1, K_2) is the size of the kernel. To impose orthogonality with FGD, we first need to reshape each such parameter into a two-dimensional matrix. This can be simply done by flattening the last three dimensions and transposing it, which gives a matrix with a size of $p_i \times c_o$, where $p_i = c_i K_1 K_2$. We only impose orthogonality on parameters with $p_i \geq c_o$ and $\min(K_1, K_2) > 1$, which usually constitute most part of a CNN model. For the specific settings for each sub-experiment, refer to the supplementary material for the details not mentioned in the main text. Our implementation uses PyTorch (Paszke et al. 2019).

Toy Example

We first try FGD on a toy example. Fix a non-singular $V \in \mathbb{R}^{n\times p}$ and consider the following optimization problem (Huang et al. 2018) to find the matrix on the Stiefel manifold that has minimal distance with V:

$$\min_{W\in \mathrm{St}(n,p)} f(W) = \|W - V\|. \tag{14}$$

The closed-form solution of the above problem is

$$W_0 = W_0(V) = VD\Lambda^{-1/2}D^T, \tag{15}$$

where $\Lambda = \mathrm{diag}(\{\lambda_1, ..., \lambda_p\})$ is a diagonal matrix corresponding to the eigenvalues of $S = V^T V$ and D is the matrix consisting of the corresponding eigenvectors. Let $f_0 = \|W_0 - V\|$ be the minimum distance. We apply FGD on this problem with learning rate $\eta = 0.1$, momentum coefficient $\gamma = 0.1$, and feedback coefficient $\alpha = 12$. We pick 5 different random $V \in \mathbb{R}^{5\times 3}$. In Figures[1] 1a and 1b, for the variable W at the current epoch, we show the difference $f(W) - f_0$ between the current f value and the minimum, and the distance $\left\|W^T W - I\right\|$ to the Stiefel manifold during the training of 60 epochs. In each trial, FGD decreases the loss of the optimization problem and maintains the orthogonality at the same time, practically showing the correctness of FGD.

[1]We use SciencePlots (Garrett 2020) to draw the plots.

WideResNet on CIFAR-10/100

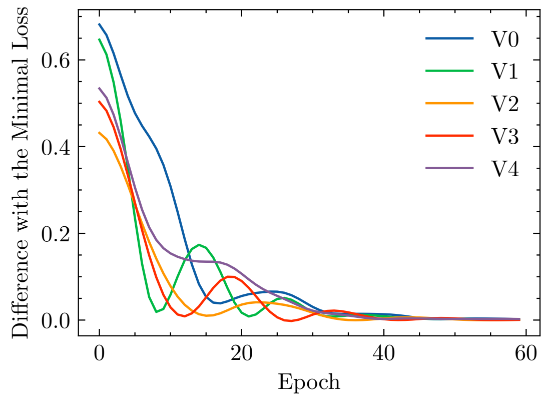
We use the WideResNet 28-10 model on CIFAR-10/100 datasets. We provide the most detailed results for this sub-experiment since the results of most baseline methods are available, which allows us to do the most comprehensive comparisons. All experiments using WideResNet 28-10 are done on one TITAN Xp GPU.

In Table 1, we list the test accuracy rates, the training time, and the distances to the Stiefel manifold of the transformed two-dimensional parameters in the final trained models of each method, where FGD shows outstanding performance in terms of accuracy, efficiency, and numerical stability.

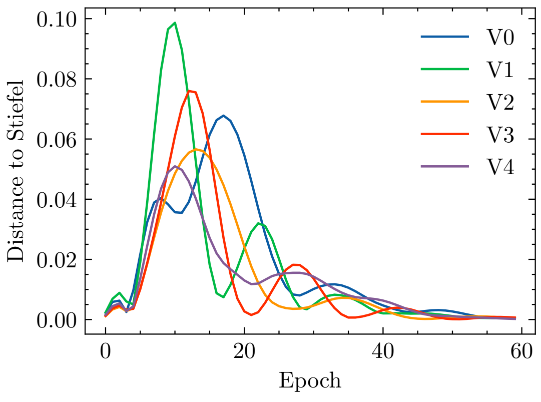
Accuracy. We report the mean value and standard deviation of the test accuracy rates on both CIFAR-10 and CIFAR-100 datasets when they are available. We list the claimed statistics in the original paper when we fail to reproduce similar ones. As some statistics are not reported in the original papers and the corresponding code is unavailable, we leave them as N/A. FGD achieves the highest accuracy rates on both CIFAR-10 and CIFAR-100 with accuracy gains of 0.24% and 0.78% over SGD, respectively. Specifically, FGD achieves an average accuracy rate of 82.25% and is the only method with an accuracy rate of more than 82% on CIFAR-100. The second best method is Cayley SGD showing an average accuracy rate of 81.90%. However, the gap with our method is favorably obvious. Figures 2a and 2b show the test accuracy rates and training losses during the training process on CIFAR-100. Specifically, in Figure 2a, we zoom in the accuracy rates in the last 20 epochs, where the superiority of FGD over others in terms of accuracy is clearly shown. Besides, the ability of FGD to mitigate the drop in the test accuracy during epochs 60 and 120 is observed. As shown in Figure 2b, the curve of FGD is much smoother than those of OCNN and SRIP, showing that FGD not only improves the accuracy rate but also stabilizes the training. Note that SRIP and OCNN include regularization terms imposing orthogonality into the loss function, but in Figure 2b we only compare the loss of the optimization problem for the sake of clarity.

Efficiency. We compare the efficiency by considering the training time for each method, as done in a previous work (Li, Fuxin, and Todorovic 2019). The efficiency of FGD is comparable to those of ONI, SRIP, and OCNN. All three methods are based on soft constraints and fail to keep orthogonality consistently. The training time of FGD is only 2.2%-2.8% longer than that of ONI or SRIP. Moreover, FGD is even 5.2% faster than OCNN. Compared with Cayley SGD/ADAM and OMDSM based on hard constraints, FGD is 37.5%-56.2% faster and achieves better numerical stability at the same time.

Numerical stability. The distance to the Stiefel manifold of each method is used to compare the numerical stability. We choose three representative layers, where the first one L_1 has original size $(160, 160, 3, 3)$ and transformed size 1440×160; the second one L_2 has original size $(320, 320, 3, 3)$ and transformed size 2880×320; and the third one L_3 has original size $(640, 640, 3, 3)$ and transformed size 5760×640. For a transformed matrix M with size $n \times p$, the distance to the Stiefel manifold is computed as $\left\|M^T M - I_p\right\|$. FGD



(a) Difference with the minimal possible distance.



(b) Distance to the Stiefel manifold.

Figure 1: The results of the toy example.

Method	CIFAR-10	CIFAR-100	Training time	L_1	L_2	L_3
SGD*	96.27 ± 0.054	81.47 ± 0.282	110.16	12.26	15.80	23.98
OCNN*	96.34 ± 0.125	81.50 ± 0.156	144.28	3.00e-4	3.70	22.14
ONI*	96.03 ± 0.153	81.18 ± 0.107	132.94	12.26	17.08	22.42
Cayley SGD	96.34	81.74	218.70(292.73*)	4.70e-6*	4.12e-6*	7.32e-6*
Cayley ADAM	96.43	81.90	224.40(292.89*)	3.50e-6*	4.48e-6*	4.52e-6*
OMDSM	96.27	81.39	312.45	N/A	N/A	N/A
SRIP	96.40	81.81	133.72*	12.57*	15.62*	24.00*
SVB	96.42	81.68	N/A	N/A	N/A	N/A
LCD	96.31	81.44	N/A	N/A	N/A	N/A
FGD*(ours)	**96.51** ± 0.062	**82.25** ± 0.220	136.71	2.12e-6	3.87e-6	6.56e-6

Table 1: Test accuracy rates (in percentages), training times per epoch (in seconds), and the distances to the Stiefel manifold of the transformed parameters using WideResNet 28-10 on CIFAR-10/100.

achieves the lowest distance to the Stiefel manifold on each representative layer, showing the highest numerical stability when imposing orthogonality.

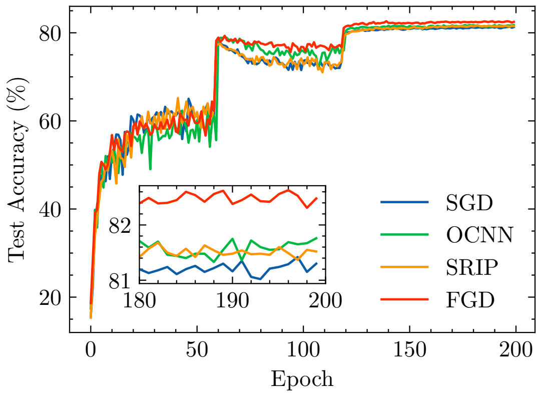
ResNet and VGG on CIFAR-10/100

We use ResNet and VGG models on CIFAR-10/100. We use ResNet110 with bottleneck layers. We use VGG models with batch normalization (Ioffe and Szegedy 2015). For each of the models, orthogonality is imposed only on the convolutional layers in the last two residual modules. The parameters of these layers constitute the majority of the whole network. This restriction on the range of parameters to impose orthogonality shows the best performance. We also apply the same restriction when using OCNN for fairness. This restriction also improves the performance of OCNN.

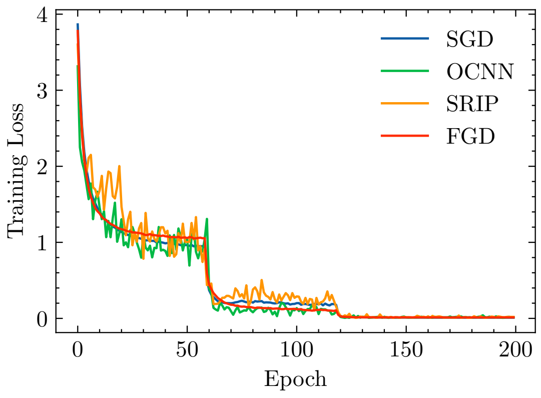
In Tables 2 and 3, we report the test accuracy rates when using ResNet and VGG, respectively. We focus on the comparisons in terms of accuracy as the statistics on the training time and the distances to the Stiefel manifold of most methods are unavailable. In each cell of both tables, the number on the left represents the accuracy on CIFAR-10 and that on the right is for CIFAR-100. In Table 2, we compare FGD with three methods: SGD, SRIP, and OCNN. The statistics and code of SRIP are only available for ResNet110. FGD consistently outperforms others in terms of accuracy. Notably, the accuracy gains of FGD over SGD are more than 1% when using ResNet18, ResNet34, and ResNet110 on CIFAR-100. In Table 3, we compare FGD with three methods: SGD, Cayley SGD, and Cayley ADAM. FGD consistently outperforms others in terms of accuracy with accuracy gains of 0.97%, 0.52%, and 1.38% over SGD when using three different VGG models on CIFAR-100.

ResNet and PreActResNet on ImageNet

To further show the performance of FGD, we conduct experiments using ResNet and PreActResNet models on the more complicated ImageNet ILSVRC-2012 dataset, where we compare FGD with four methods: SGD, OCNN, ONI, and SRIP. In Table 4, we report the test accuracy rates of different methods. In each cell, the number on the left represents the top-1 accuracy and that on the right is the top-5 accuracy. As some statistics are not reported in the original paper and the corresponding code is unavailable, we leave them as N/A. Among the five methods, FGD achieves the best performance in terms of both top-1 and top-5 accuracy rates using ResNet50 and PreActResNet34. In the experiments using ResNet34, FGD achieves the highest top-1 accuracy, and the achieved top-5 accuracy is only lower than



(a) Test accuracy rates.



(b) Training losses.

Figure 2: The results of the experiments using WideResNet 28-10 on CIFAR-100.

Method	ResNet18	ResNet34	ResNet50	ResNet101	ResNet110
SGD*	95.20/77.85	95.30/78.29	95.46/80.29	95.37/79.93	94.36/74.79
SRIP	N/A	N/A	N/A	N/A	93.45(94.44*)/74.99
OCNN*	95.20*/78.10	95.41*/78.70	95.54*/80.33*	95.45*/80.12*	94.89*/75.93*
FGD*(ours)	**95.57/78.87**	**95.52/79.52**	**95.83/80.64**	**95.68/80.17**	**95.20/76.33**

Table 2: Test accuracy rates using ResNet on CIFAR-10/100.

Method	VGG13	VGG16	VGG19
SGD*	93.92/74.27	93.81/74.03	93.73/72.99
Cayley SGD	94.10/75.14	94.23/74.52	94.15/74.32
Cayley ADAM	94.07/74.90	94.12/74.39	93.97/74.30
FGD*(ours)	**94.32/75.24**	**94.45/74.55**	**94.22/74.37**

Table 3: Test accuracy rates using VGG on CIFAR-10/100.

the claimed one of OCNN and higher than all the others including the reproduced one of OCNN. The top-1 accuracy gains of FGD over SGD are 0.47% on ResNet34, 0.67% on ResNet50, and 0.91% on PreActResNet34, while the top-5 accuracy gains are 0.46%, 0.35%, and 0.46%.

Method	ResNet34	ResNet50	PreActRes34
SGD*	73.49, 91.31	76.13, 92.98	72.62, 90.95
OCNN	73.93, **92.11** (73.81, 91.63*)	76.38, 93.18*	N/A
ONI*	73.68, 91.60	76.75, 93.28	N/A
SRIP	N/A, 91.68	N/A, 93.13	N/A, 91.21
FGD*(ours)	**73.96**, 91.77	**76.80, 93.33**	**73.53, 91.41**

Table 4: Test accuracy rates on ImageNet.

6 Conclusion and Discussion

In this paper, we have proposed Feedback Gradient Descent (FGD), an efficient and stable optimization algorithm with orthogonality for DNNs. Inspired by Feedback Integrators, we have constructed a continuous-time dynamical system in a Euclidean space containing the tangent bundle of the Stiefel manifold as a local attractor, and completed the discretization with the semi-implicit Euler method. The excellent performance of FGD in terms of accuracy, efficiency, and numerical stability has been shown through the theoretical analyses and the extensive experiments. It is hard to apply most existing momentum-based methods on manifolds to DNNs due to their problems on efficiency (Lezcano-Casado 2020). Moreover, some recent works show that the existing momentum-based methods suffer from insufficiency (Kidambi et al. 2018; Liu and Belkin 2019). FGD provides a practical and efficient solution for generalizing momentum to manifolds.

FGD has been shown to be advantageous on the image classification task. Additionally, for many other tasks where orthogonality has manifested its ability, such as graph embedding (Robles-Kelly and Hancock 2007; Shaw and Jebara 2009; Liu, Han, and Nie 2017) and matrix factorization (Ding et al. 2006; Zhang et al. 2016), FGD can hopefully also be utilized. Although we have designed FGD specifically for the Stiefel manifold, our framework can be flexibly applied to other submanifolds embedded in Euclidean space, such as the Oblique manifold (Huang et al. 2017). Some recent works (Xiao et al. 2018; Qi et al. 2020) propose techniques to leverage isometry and orthogonality for training DNNs, without using normalization or skip connections. Another recent work (Liu et al. 2021b) proposes an orthogonal over-parameterized training framework by learning an orthogonal transformation to help effectively train DNNs. We leave it as future works to explore the potential of FGD in these directions.

Acknowledgements

The authors appreciate the constructive and insightful comments from the reviewers. The authors also would like to thank Lin Wang and Yi-Ling Qiao for discussions on the PyTorch implementation.

This work was conducted by the Center for Applied Research in Artificial Intelligence (CARAI) grant funded by Defense Acquisition Program Administration (DAPA) and Agency for Defense Development (ADD) [UD190031RD].

References

Absil, P.-A.; Mahony, R.; and Sepulchre, R. 2009. *Optimization algorithms on matrix manifolds*. Princeton University Press.

Absil, P.-A.; and Malick, J. 2012. Projection-like retractions on matrix manifolds. In *SIOPT*.

Arjovsky, M.; Shah, A.; and Bengio, Y. 2016. Unitary evolution recurrent neural networks. In *ICML*.

Baldi, P. 1995. Gradient descent learning algorithm overview: A general dynamical systems perspective. In *IEEE Transactions on neural networks*.

Bansal, N.; Chen, X.; and Wang, Z. 2018. Can we gain more from orthogonality regularizations in training deep networks? In *NIPS*.

Bengio, Y.; Simard, P.; and Frasconi, P. 1994. Learning long-term dependencies with gradient descent is difficult. In *IEEE Transactions on Neural Networks*.

Bonnabel, S. 2013. Stochastic gradient descent on Riemannian manifolds. In *IEEE Transactions on Automatic Control*.

Brock, A.; Donahue, J.; and Simonyan, K. 2018. Large scale GAN training for high fidelity natural image synthesis. In *ICLR*.

Brock, A.; Lim, T.; Ritchie, J. M.; and Weston, N. 2017. Neural photo editing with introspective adversarial networks. In *ICLR*.

Chang, D. E. 2018. On controller design for systems on manifolds in Euclidean space. In *International Journal of Robust and Nonlinear Control*.

Chang, D. E.; Jiménez, F.; and Perlmutter, M. 2016. Feedback integrators. In *Journal of Nonlinear Science*.

Chang, D. E.; and Perlmutter, M. 2019. Feedback integrators for nonholonomic mechanical systems. In *Journal of Nonlinear Science*.

Chang, D. E.; Phogat, K. S.; and Choi, J. 2019. Model Predictive Tracking Control for Invariant Systems on Matrix Lie Groups via Stable Embedding into Euclidean Spaces. In *IEEE Transactions on Automatic Control*.

Chen, Y.; Jin, X.; Feng, J.; and Yan, S. 2017. Training group orthogonal neural networks with privileged information. In *IJCAI*.

Deng, J.; Dong, W.; Socher, R.; Li, L.-J.; Li, K.; and Fei-Fei, L. 2009. Imagenet: A large-scale hierarchical image database. In *CVPR*.

Desjardins, G.; Simonyan, K.; Pascanu, R.; and Kavukcuoglu, K. 2015. Natural neural networks. In *NIPS*.

Ding, C.; Li, T.; Peng, W.; and Park, H. 2006. Orthogonal nonnegative matrix t-factorizations for clustering. In *KDD*.

Edelman, A.; Arias, T. A.; and Smith, S. T. 1998. The geometry of algorithms with orthogonality constraints. In *SIMAX*.

Garrett, J. D. 2020. SciencePlots (v1.0.6). In *Zenodo*.

Haier, E.; Lubich, C.; and Wanner, G. 2006. *Geometric Numerical integration: structure-preserving algorithms for ordinary differential equations*. Springer.

Harandi, M.; and Fernando, B. 2016. Generalized backpropagation, Étude de cas: Orthogonality. In *arXiv*.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2016a. Deep residual learning for image recognition. In *CVPR*.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2016b. Identity mappings in deep residual networks. In *ECCV*.

Helfrich, K.; Willmott, D.; and Ye, Q. 2018. Orthogonal recurrent neural networks with scaled Cayley transform. In *ICML*.

Helfrich, K.; and Ye, Q. 2020. Eigenvalue Normalized Recurrent Neural Networks for Short Term Memory. In *AAAI*.

Huang, L.; Liu, L.; Zhu, F.; Wan, D.; Yuan, Z.; Li, B.; and Shao, L. 2020. Controllable orthogonalization in training dnns. In *CVPR*.

Huang, L.; Liu, X.; Lang, B.; and Li, B. 2017. Projection based weight normalization for deep neural networks. In *arXiv*.

Huang, L.; Liu, X.; Lang, B.; Yu, A. W.; Wang, Y.; and Li, B. 2018. Orthogonal weight normalization: Solution to optimization over multiple dependent stiefel manifolds in deep neural networks. In *AAAI*.

Hyland, S.; and Rätsch, G. 2017. Learning unitary operators with help from u (n). In *AAAI*.

Ioffe, S.; and Szegedy, C. 2015. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *ICML*.

Jia, K.; Tao, D.; Gao, S.; and Xu, X. 2017. Improving training of deep neural networks via singular value bounding. In *CVPR*.

Jiang, B.; and Dai, Y.-H. 2015. A framework of constraint preserving update schemes for optimization on Stiefel manifold. In *Mathematical Programming*.

Jing, L.; Shen, Y.; Dubcek, T.; Peurifoy, J.; Skirlo, S.; LeCun, Y.; Tegmark, M.; and Soljačić, M. 2017. Tunable efficient unitary neural networks (eunn) and their application to rnns. In *ICML*.

Kidambi, R.; Netrapalli, P.; Jain, P.; and Kakade, S. M. 2018. On the insufficiency of existing momentum schemes for Stochastic Optimization. In *ICLR*.

Kiefer, J.; Wolfowitz, J.; et al. 1952. Stochastic estimation of the maximum of a regression function. In *The Annals of Mathematical Statistics*.

Ko, W.; Phogat, K. S.; Petit, N.; and Chang, D. E. 2021. Tracking Controller Design for Satellite Attitude Under Unknown Constant Disturbance Using Stable Embedding. In *Journal of Electrical Engineering & Technology*.

Krizhevsky, A.; Hinton, G.; et al. 2009. *Learning multiple layers of features from tiny images*. Master's thesis, University of Toronto.

Le, Q. V.; Jaitly, N.; and Hinton, G. E. 2015. A simple way to initialize recurrent networks of rectified linear units. In *arXiv*.

Lezcano Casado, M. 2019. Trivializations for gradient-based optimization on manifolds. In *NIPS*.

Lezcano-Casado, M. 2020. Adaptive and Momentum Methods on Manifolds Through Trivializations. In *arXiv*.

Lezcano-Casado, M.; and Martınez-Rubio, D. 2019. Cheap orthogonal constraints in neural networks: A simple parametrization of the orthogonal and unitary group. In *ICML*.

Li, J.; Fuxin, L.; and Todorovic, S. 2019. Efficient Riemannian optimization on the Stiefel manifold via the Cayley transform. In *ICLR*.

Liu, C.; and Belkin, M. 2019. Accelerating SGD with momentum for over-parameterized learning. In *ICLR*.

Liu, H.; Han, J.; and Nie, F. 2017. Semi-supervised Orthogonal Graph Embedding with Recursive Projections. In *IJCAI*.

Liu, S.; Li, X.; Zhai, Y.; You, C.; Zhu, Z.; Fernandez-Granda, C.; and Qu, Q. 2021a. Convolutional normalization: Improving deep convolutional network robustness and training. In *arXiv*.

Liu, W.; Lin, R.; Liu, Z.; Rehg, J. M.; Paull, L.; Xiong, L.; Song, L.; and Weller, A. 2021b. Orthogonal over-parameterized training. In *CVPR*.

Maduranga, K. D.; Helfrich, K. E.; and Ye, Q. 2019. Complex unitary recurrent neural networks using scaled cayley transform. In *AAAI*.

Mhammedi, Z.; Hellicar, A.; Rahman, A.; and Bailey, J. 2017. Efficient orthogonal parametrisation of recurrent neural networks using householder reflections. In *ICML*.

Mishkin, D.; and Matas, J. 2016. All you need is a good init. In *ICLR*.

Miyato, T.; Kataoka, T.; Koyama, M.; and Yoshida, Y. 2018. Spectral normalization for generative adversarial networks. In *ICLR*.

Netzer, Y.; Wang, T.; Coates, A.; Bissacco, A.; Wu, B.; and Ng, A. Y. 2011. Reading digits in natural images with unsupervised feature learning. In *NIPS Workshop on Deep Learning and Unsupervised Feature Learning*.

Ozay, M.; and Okatani, T. 2018. Training cnns with normalized kernels. In *AAAI*.

Park, J.-H.; Phogat, K. S.; Kim, W.; and Chang, D. E. 2021. Transversely Stable Extended Kalman Filters for Systems on Manifolds in Euclidean Spaces. In *Journal of Dynamic Systems, Measurement, and Control*.

Pascanu, R.; Mikolov, T.; and Bengio, Y. 2013. On the difficulty of training recurrent neural networks. In *ICML*.

Paszke, A.; Gross, S.; Massa, F.; Lerer, A.; Bradbury, J.; Chanan, G.; Killeen, T.; Lin, Z.; Gimelshein, N.; Antiga, L.; Desmaison, A.; Kopf, A.; Yang, E.; DeVito, Z.; Raison, M.; Tejani, A.; Chilamkurthy, S.; Steiner, B.; Fang, L.; Bai, J.; and Chintala, S. 2019. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In *NIPS*.

Qi, H.; You, C.; Wang, X.; Ma, Y.; and Malik, J. 2020. Deep isometric learning for visual recognition. In *ICML*.

Qian, N. 1999. On the momentum term in gradient descent learning algorithms. In *Neural networks*.

Rapcsák, T. 2002. On minimization on Stiefel manifolds. In *European Journal of Operational Research*.

Robbins, H.; and Monro, S. 1951. A stochastic approximation method. In *The Annals of Mathematical Statistics*.

Robles-Kelly, A.; and Hancock, E. R. 2007. A Riemannian approach to graph embedding. In *Pattern Recognition*.

Rodríguez, P.; Gonzalez, J.; Cucurull, G.; Gonfaus, J. M.; and Roca, X. 2017. Regularizing cnns with locally constrained decorrelations. In *ICLR*.

Saxe, A. M.; McClelland, J. L.; and Ganguli, S. 2014. Exact solutions to the nonlinear dynamics of learning in deep linear neural networks. In *ICLR*.

Shaw, B.; and Jebara, T. 2009. Structure preserving embedding. In *ICML*.

Simonyan, K.; and Zisserman, A. 2015. Very deep convolutional networks for large-scale image recognition. In *ICLR*.

Smith, S. T. 1994. Optimization techniques on Riemannian manifolds. In *Fields Institute Communications*.

Trockman, A.; and Kolter, J. Z. 2021. Orthogonalizing Convolutional Layers with the Cayley Transform. In *ICLR*.

Vorontsov, E.; Trabelsi, C.; Kadoury, S.; and Pal, C. 2017. On orthogonality and learning recurrent networks with long term dependencies. In *ICML*.

Wang, J.; Chen, Y.; Chakraborty, R.; and Yu, S. X. 2020. Orthogonal Convolutional Neural Networks. In *CVPR*.

Wen, Z.; and Yin, W. 2013. A feasible method for optimization with orthogonality constraints. In *Mathematical Programming*.

Wisdom, S.; Powers, T.; Hershey, J. R.; Roux, J. L.; and Atlas, L. 2016. Full-capacity unitary recurrent neural networks. In *NIPS*.

Xiao, L.; Bahri, Y.; Sohl-Dickstein, J.; Schoenholz, S.; and Pennington, J. 2018. Dynamical isometry and a mean field theory of cnns: How to train 10,000-layer vanilla convolutional neural networks. In *ICML*.

Xie, D.; Xiong, J.; and Pu, S. 2017. All you need is beyond a good init: Exploring better solution for training extremely deep convolutional neural networks with orthonormality and modulation. In *CVPR*.

Zagoruyko, S.; and Komodakis, N. 2016. Wide residual networks. In *BMVC*.

Zhang, W. E.; Tan, M.; Sheng, Q. Z.; Yao, L.; and Shi, Q. 2016. Efficient orthogonal non-negative matrix factorization over Stiefel manifold. In *CIKM*.

Zhu, D.; Li, B.; and Liang, P. 2015. On the matrix inversion approximation based on Neumann series in massive MIMO systems. In *ICC*.