Training CNNs with Normalized Kernels

Mete Ozay [1] and Takayuki Okatani [1,2]
[1]Tohoku University, Sendai, Miyagi, Japan
[2]RIKEN Center for AIP, Tokyo, Japan
{mozay,okatani}@vision.is.tohoku.ac.jp

Abstract

Several methods of normalizing convolution kernels have been proposed in the literature to train convolutional neural networks (CNNs), and have shown some success. However, our understanding of these methods has lagged behind their success in application; there are a lot of open questions, such as why a certain type of kernel normalization is effective and what type of normalization should be employed for each (e.g., higher or lower) layer of a CNN. As the first step towards answering these questions, we propose a framework that enables us to use a variety of kernel normalization methods at any layer of a CNN. A naive integration of kernel normalization with a general optimization method, such as SGD, often entails instability while updating parameters. Thus, existing methods employ ad-hoc procedures to empirically assure convergence. In this study, we pose estimation of convolution kernels under normalization constraints as constraint-free optimization on kernel submanifolds that are identified by the employed constraints. Note that naive application of the established optimization methods for matrix manifolds to the aforementioned problems is not feasible because of the hierarchical nature of CNNs. To this end, we propose an algorithm for optimization on kernel manifolds in CNNs by appropriate scaling of the space of kernels based on structure of CNNs and statistics of data. We theoretically prove that the proposed algorithm has assurance of almost sure convergence to a solution at single minimum. Our experimental results show that the proposed method can successfully train popular CNN models using several different types of kernel normalization methods. Moreover, they show that the proposed method improves classification performance of baseline CNNs, and provides state-of-the-art performance for major image classification benchmarks.

Introduction

Over the last decade, convolutional neural networks (CNNs) have been utilized to perform various tasks such as image classification (Wei et al. 2015). While performing optimization using stochastic gradient descent (SGD) algorithms with backpropagation (BP) in CNNs, we observe that norm of gradients may exponentially increase or decrease (Glorot and Bengio 2010; Pascanu, Mikolov, and Bengio 2013). Exploding and vanishing gradients trigger several open problems such as convergence of SGD and its robustness to reparametrization of convolution kernels, and internal covariate shift. In order to cope with these problems, various methods have been proposed for normalization of kernels[1] using different orthogonality constraints (Arpit et al. 2016; Salimans and Kingma 2016).

However, different kernel normalization methods may affect geometry of search spaces, dissimilarly. More precisely, kernel normalization methods may provide kernel spaces with different geometric properties according to the constraints used in the normalization operations. *In order to assure convergence of SGD algorithms to solutions at single minimum, we need to identify search spaces and compute steps according to the geometry of kernel spaces in CNNs*. Consequently, convergence properties of SGD may vary among different kernel normalization methods. Since various orthogonality constraints are nonconvex and nonlinear, embedding constraints into cost functions may lead to many local minimizers (Wen and Yin 2013). These theoretical challenges imply more crucial problems in implementation of optimization methods for training CNNs with normalized kernels. SGD methods will not work as intended if they are implemented without proper consideration of geometry of normalized kernels. For example, if projections defined for unit norm kernels are employed to perform SGD steps using column-wise normalized or orthonormal kernels in CNNs, then gradients explode and vanish in the first few epochs of the SGD.

We address the aforementioned problems by identifying kernel spaces as Riemannian manifolds under a geometric optimization framework for training of CNNs. We pose the kernel estimation problem in CNNs (LeCun et al. 1998) as optimization on Riemannian submanifolds which are described according to different geometric properties of the kernels, such as orthonormal rectangular or orthogonal square kernels. Thereby, we can define constraints of optimization problems of CNNs in search spaces of SGD algorithms, instead of embedding the constraints into cost functions of the problems (Mishra and Sepulchre 2016).

However, when we employ well-known Riemannian optimization methods developed for shallow SGD (Absil, Mahony, and Sepulchre 2007) to train CNNs, we encounter another challenge. In our experiments, we observe that appli-

[1]We refer to *convolution kernels* used in CNNs by *kernels*.

cation of these methods directly to train CNNs result in vanishing and exploding gradient problems as well. Thus, SGD diverges in early epochs of training. One of the main reasons of occurrence of this problem is the multi-scale hierarchical architecture of CNNs. Although kernels with the same size are applied at different layers, size of receptive fields of the corresponding units (neurons) of CNNs increases as the depth of the CNNs increases. For instance, at the lower layers, *small* kernels (e.g. matrices of size 3×3 or 7×7) are applied to *small* image patches, each of which covers small parts of objects. However, at higher layers, kernels are applied to features which can represent larger patterns such as groups of objects. In addition, at the fully connected layers, kernels are used to train classifiers. Therefore, geometric and statistical properties of features input to kernels vary among different layers of CNNs. Moreover, additional feature normalization methods, such as batch normalization (BN), can be used in CNNs to remove statistical variance of features. In (Salimans and Kingma 2016), it was reported that keeping statistical variance of features using mean only BN improves the performance. Similarly, in our experimental analyses, we observe that employing mean only BN and using statistical properties of features while constructing kernel spaces (e.g. the sphere) by scaling the kernels (e.g. changing the radius of the sphere) improves the performance. Therefore, varying geometric and statistical properties of features and kernels need to be considered while employing the Riemannian optimization methods in CNNs.

For this purpose, we propose a modular SGD algorithm which can be used to perform optimization on various kernel submanifolds at different layers of CNNs. We also provide assurance of its convergence to a single minimum of classification loss. In our framework, we first construct kernel submanifolds at each layer of a CNN such that a kernel resides as a point on a kernel submanifold. Then, we employ our proposed SGD algorithm for optimization on kernel submanifolds using BP in CNNs. The proposed algorithm is used to train various different types of CNNs on different image classification datasets. Experimental results show that normalized kernels can be used boost performance of base-line CNNs.

Related Work and Our Contributions

Kernel (weight) normalization methods have been implemented using reparametrizations (Salimans and Kingma 2016), and additional constraints, such as orthogonality (Neyshabur, Salakhutdinov, and Srebro 2016), in order to preserve unit norm property of the kernels for forward propagation (Arpit et al. 2016) or at initialization (Mishkin and Matas 2016). Removal of scale and translation from kernels by normalization can be interpreted as imposition of a geometric structure such that the kernels lie on the sphere (Stoyan, Kendall, and Mecke 2013). Compact Stiefel manifolds were employed to learn features represented by symmetric positive definite (SPD) matrices and semi-orthogonal weights in (Huang and Gool 2017). Since they designed a particular network to learn SPD features, generalization of their method to train a wider class of CNNs with convergence properties is not trivial.

In our approach, CNNs can be trained using different kernel submanifolds such as the sphere, the oblique and/or the Stiefel manifold. Additional constraints can also be imposed using immersed submanifolds such as rotation groups. Thus, our approach can be considered as generalization of the aforementioned approaches such that we can employ our methods to model different submanifolds according to various constraints, such as orthonormal kernels. Thereby, we employ geometry of kernels to identify the constraints on the optimization problem of CNNs. Our contributions can be summarized as follows:

- Using the proposed approach, we provide a geometric view of normalization methods by describing the geometry of convolution kernels normalized using different normalization methods.
- In order to perform optimization in CNNs using kernels that are normalized by different methods, we propose an algorithm which enables stable training of CNNs by proper scaling of kernel spaces to avoid gradient vanishing/explosion. The algorithm has a modular design such that different scaling operations can be employed for different layers.
- In the experimental analyses, we examine the proposed methods and theoretical results for different manifolds using several benchmark image classification datasets. The results show that the proposed method can be used to boost performance of various baseline CNN models. In the supplemental material, proofs of theorems, implementation details of the experiments, additional theoretical and experimental results are given.

Geometric Properties of Normalized Convolution Kernels in CNNs

We contemplate subspaces of convolution kernels endowed with differentiable structures, i.e. Riemannian kernel submanifolds. Suppose that we are given a set of training samples $S = \{s_i = (\mathbf{I}_i, y_i)\}_{i=1}^{N}$ of a random variable s drawn from a distribution $\mathcal{P}$ on a measurable space $\mathfrak{S}$, where y_i is a class label of the i^{th} image $\mathbf{I}_i$. An L-layer CNN consists of a set of tensors $\mathcal{W} = \{\mathcal{W}_l\}_{l=1}^{L}$, where $\mathcal{W}_l = \{\mathbf{W}_{d,l} \in \mathbb{R}^{A_l \times B_l \times C_l}\}_{d=1}^{D_l}$, and $\mathbf{W}_{d,l} = [W_{c,d,l} \in \mathbb{R}^{A_l \times B_l}]_{c=1}^{C_l}$ is a tensor[2] composed of kernels (weight matrices) $W_{c,d,l}$ constructed at each layer $l = 1, 2, \ldots, L$, for each c^{th} channel $c = 1, 2, \ldots, C_l$ and each d^{th} kernel $d = 1, 2, \ldots, D_l$.

At each l^{th} convolution layer, a feature representation $f_l(\mathbf{X}_l; \mathcal{W}_l)$ is computed by compositionally employing nonlinear functions, and convolving an image $\mathbf{I}$ with kernels as

$$f_l(\mathbf{X}_l; \mathcal{W}_l) = f_l(\cdot; \mathcal{W}_l) \circ f_{l-1}(\cdot; \mathcal{W}_{l-1}) \circ \cdots \circ f_1(\mathbf{X}_1; \mathcal{W}_1), \tag{1}$$

where $\mathbf{X}_l = [X_{c,l}]_{c=1}^{C_l}$, and $\mathbf{X}_1 := \mathbf{I}$ is an image at the first layer ($l = 1$). The c^{th} channel of the data matrix $X_{c,l}$ is con-

[2] We use shorthand notation for matrix concatenation such that $[W_{c,d,l}]_{c=1}^{C_l} \triangleq [W_{1,d,l}, \cdots, W_{C_l,d,l}]$, where $\triangleq$ means *equal to by definition*.

volved with the kernel $W_{c,d,l}$ to obtain the d^{th} feature map $X_{c,l+1} := \hat{X}_{d,l}$ by $\hat{X}_{d,l} = W_{c,d,l} * X_{c,l}, \forall c, d, l$ [3].

Given a batch of samples $\mathfrak{s} \subseteq S$, we denote a value of a classification loss function for a kernel $\omega \triangleq W_{c,d,l}$ by $\mathcal{L}(\omega, \mathfrak{s})$. We denote the loss function of kernels $\mathcal{W}$ utilized in the CNN by $\mathcal{L}(\mathcal{W}, \mathfrak{s})$. If we assume that $\mathfrak{s}$ consists of a single sample s_i, then, an expected loss or cost function of the CNN is computed by

$$\mathcal{L}(\mathcal{W}) \triangleq E_{\mathcal{P}}\{\mathcal{L}(\mathcal{W}, s)\} = \int \mathcal{L}(\mathcal{W}, s) d\mathcal{P}. \tag{2}$$

The expected loss for ω is denoted by $\mathcal{L}(\omega)$. Then, feature representations are learned by solving

$$\min_{\mathcal{W}} \mathcal{L}(\mathcal{W}). \tag{3}$$

In other words, feature representations are learned in CNNs by solving (3) in a search space. Restriction on the search space can be imposed into (3) by defining constraints expressed as a function of the search variables $\mathcal{W}$, e.g. using normalization constraints.

Kernel (weight) normalization methods have been employed to learn deep feature representations (Salimans and Kingma 2016; Neyshabur, Salakhutdinov, and Srebro 2016). Analysis and understanding of effect of normalization methods on geometry of kernel spaces is crucial for training of CNNs using SGD since gradient steps should be determined according to their geometric properties. The next theorem characterizes the geometry of spaces of kernels normalized by different normalization methods.

Proposition 1 (Geometry of spaces of normalized kernels)**.** *Suppose that we are given a set of kernels $\omega \triangleq W_{d,c,l} \in \mathbb{R}^{A_l \times B_l}, \forall c, d$ computed at the l^{th} layer of a CNN. Moreover, suppose that an ambient kernel manifold $\mathcal{M}$ is identified by a Euclidean space of $A \times B$ matrices $\mathbb{R}^{A \times B}$.*

(i) If the kernels ω are normalized to the unit Frobenius norm, then each kernel resides on the $A_l B_l - 1$ dimensional sphere $\mathcal{S}(A_l, B_l) = \{\omega \in \mathbb{R}^{A_l \times B_l} : \|\omega\|_F^2 = 1\}$, where $\|\cdot\|_F$ is the squared Frobenius norm.

(ii) If the columns $\omega_b, \forall b = 1, 2, \ldots, B_l$ of the kernels ω are normalized with the unit norm, then each ω_b resides on the unit sphere $\mathcal{S}(A_l) = \{\omega_b \in \mathbb{R}^{A_l} : \|\omega_b\|_F^2 = 1\}$. In addition, the space of the kernels ω is isometric to the oblique manifold $\mathcal{OB}(A_l, B_l) = \{\omega \in \mathbb{R}^{A_l \times B_l} : \mathrm{ddiag}(\omega^{\mathrm{T}}\omega) = I_{B_l}\}$, where $\mathrm{ddiag}(\omega)$ denotes the diagonal matrix whose diagonal elements are those of ω, and I_{B_l} is a $B_l \times B_l$ identity matrix (Absil and Gallivan 2006).

(iii) If the kernels are orthonormal, then they reside on the compact Stiefel manifold

$$St(A_l, B_l) = \{\omega \in \mathbb{R}^{A_l \times B_l} : \omega^T \omega = I_{B_l}\}.$$

Furthermore, if they are square kernels such that $A_l = B_l = n$, then they reside on the orthogonal group $\mathcal{O}(n) = \{\omega \in \mathbb{R}^{n \times n} : \omega^T \omega = I_n\}$.

[3] We ignore the bias terms in the notation for the sake of simplicity. We consider each kernel as a weight matrix.

This theorem shows that different normalization methods, even if they are implemented in an intuitively similar manner, imply different geometric properties. Thus, if a normalization method is used with optimization for training CNNs without paying attention to the geometry of the kernel spaces, it usually does not work. In oder to cope with this problem, some studies use ad-hoc transformations, as utilized in weight normalization methods (Salimans and Kingma 2016; Arandjelovic et al. 2017). Instead, we propose to perform optimization on Riemannian submanifolds of normalized convolution kernels.

Training CNNs by Optimization on Normalized Kernel Submanifolds

We found through our preliminary experiments that naive application of well-established optimization methods for matrix manifolds (Absil, Mahony, and Sepulchre 2007) to our problem does not work, as mentioned in the first section. We need to develop a method which enables proper training of CNNs by considering geometric and statistical properties of feature and kernel spaces. In addition, we need to assure convergence of the developed algorithm. In order to address these problems, we propose an SGD algorithm considering optimization of kernels at each l^{th} convolution layer of an L-layer CNN. Let R_l be the size of the receptive field of the unit that employs ω_l, F_l be the dimension of output feature maps, and $\mathfrak{c}_l$ and $\hat{\mathfrak{c}}_l$ be the number of input and output channels used at the l^{th} layer, respectively. We determine geometric scaling γ_l of the kernel space of ω_l by

$$\gamma_l = \sqrt{\frac{R_l \mathfrak{c}_l}{F_l \hat{\mathfrak{c}}_l}}. \tag{4}$$

In addition, we compute standard deviation λ_l^t of features input to a kernel ω_l at each t^{th} iteration of the algorithm. Then, the kernel space is scaled by the following function[4]

$$\Gamma_l^t = \frac{\gamma_l}{\lambda_l^t}. \tag{5}$$

Note that, scaling of the kernel space affects geometry of the corresponding tangent space at ω_l as well. Therefore, a step of an SGD, which includes back-propagation and projection of gradients, movement of kernels on the tangent space, and their back-projection to the kernel space, is also affected.

In order to perform SGD according to the dynamic change of structural properties of kernel spaces during training of CNNs, we employ the following steps:

- In the gradient transformation step, we apply a scaled projection operation to map the Euclidean gradients of kernels obtained by BP to tangent spaces at kernels according to the scaled submanifolds.
- In the kernel movement step, we move the kernels on the scaled tangent spaces.
- In the retraction step, we apply a scaled projection operator (a retraction or an exponential map) to map kernels moved on tangent spaces to the scaled submanifolds.

[4] Details are given in the supplemental material.

Algorithm 1: Training CNNs using SGD on Riemannian submanifolds of normalized kernels.

Input: T (number of iterations), S (training set), Θ (set of hyperparameters used for computation of momentum, Euclidean gradient decay and learning rate) and $\mathcal{L}$ (a loss function).

1 **Initialization:** Construct Riemannian kernel submanifolds $\{\hat{\mathcal{M}}_l\}_{l=1}^{L}$. Compute the initial scale function Γ_l^0, and initialize $\omega_l^t \in \hat{\mathcal{M}}_l$, where $\omega_l^t \triangleq W_{d,c,l}^t$, $\forall c = 1, 2, \ldots, C_l$, $\forall d = 1, 2, \ldots, D_l$, $\forall l = 1, 2, \ldots, L$. Initialize momentum variables μ_1.

2 **for** *each iteration* $t = 1, 2, \ldots, T$ **do**

3 **for** *each layer* $l = 1, 2, \ldots, L$ **do**

4 Compute the Euclidean gradient $\text{grad}_E \mathcal{L}(\omega_l^t)$, and the scale function Γ_l^t.

5 $\mu_t := q\Big(\text{grad}_E\ \mathcal{L}(\omega_l^t), \mu_t, \Theta\Big)$.

6 $\text{grad}\mathcal{L}(\omega_l^t) := \Pi_{\omega_l^t}(\mu_t, \Gamma_l^t)$.

7 $\alpha_t := g(t, \Theta)$.

8 $v_t := h(\text{grad}\mathcal{L}(\omega_l^t), \alpha_t)$.

9 $\omega_l^{t+1} := \phi_{\omega_l^t}(v_t, \Gamma_l^t), \forall \omega_l^t \in \hat{\mathcal{M}}_l$.

10 $t := t + 1$.

11 **end**

12 **end**

Output: The set of estimated kernels $\{\omega_l^T\}_{l=1}^{L}$.

An algorithmic description of the proposed SGD is given in Algorithm 1:

• **Initialization:** We first compute the initial scaling function Γ_l^0 and construct a Riemannian kernel submanifold $\hat{\mathcal{M}}_l$, for each convolution layer $l = 1, 2, \ldots, \mathcal{L}$ whose members are $\omega_l^t \in \hat{\mathcal{M}}_l$, where $\omega_l^t \triangleq W_{d,c,l}^t$, $\forall c = 1, 2, \ldots, C_l$, $\forall d = 1, 2, \ldots, D_l$, $\forall l$.

• For each epoch $t = 1, 2, \ldots, T$, and for each $l = 1, 2, \ldots, L$, following steps are performed (see Fig. 1);

- Line 4: The Euclidean gradient $\text{grad}_E \mathcal{L}(\omega_l^t)$ is computed and obtained using BP (see Fig. 1). In addition, the scaling function Γ_l^t is updated using (5).

- Line 5: Momentum and Euclidean gradient decay methods are employed on the Euclidean gradient $\text{grad}_E \mathcal{L}(\omega_l^t)$ as follows:

$$\mu_t := q\Big(\text{grad}_E\ \mathcal{L}(\omega_l^t), \mu_t, \Theta\Big). \tag{6}$$

We can employ state-of-the-art acceleration methods (Sutskever et al. 2013) in this step. For example, momentum can be employed with the Euclidean gradient decay using

$$q\Big(\text{grad}_E\ \mathcal{L}(\omega_l^t), \mu_t, \Theta\Big) = \theta_\mu \mu_t - \theta_E \text{grad}_E\ \mathcal{L}(\omega_l^t), \tag{7}$$

where $\theta_\mu \in \Theta$ is the parameter employed on the momentum variable μ_t. We consider $\theta_E \in \Theta$ as the decay parameter for the Euclidean gradient. The reason is that θ_μ and θ_E affect the step performed in the ambient Euclidean space while the learning rate (LR) is employed on the submanifold gradient. A detailed discussion of methods that are used to compute $q(\cdot)$ is given in the supplemental material (see Figure 1).

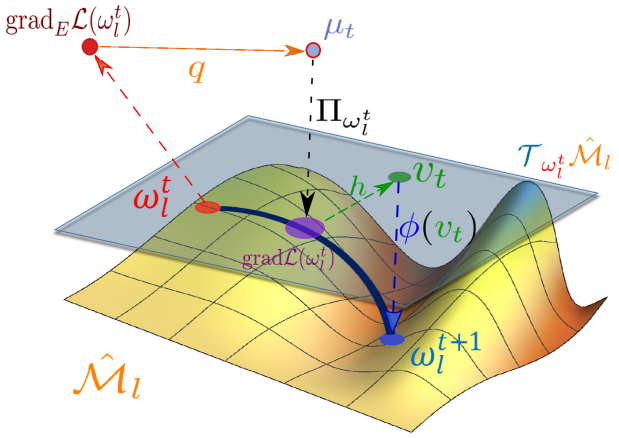

Figure 1: Updating kernels on a kernel submanifold $\hat{\mathcal{M}}_l$, $\forall l$ (Lines 4-9 in the Algorithm 1) at the t^{th} epoch.

- Line 6: The moved vector μ_t is projected to the tangent space $\mathcal{T}_{\omega_l^t}\hat{\mathcal{M}}_l$, to compute the submanifold gradient $\text{grad}\mathcal{L}(\omega_l^t)$ by $\text{grad}\mathcal{L}(\omega_l^t) := \Pi_{\omega_l^t}(\mu_t, \Gamma_l^t)$, where $\Pi_{\omega_l^t}$ is a projection operator defined according to the geometry of $\hat{\mathcal{M}}_l$ as explained in Proposition 1 (see Figure 1).

- Line 7: The learning rate α_t is updated by $\alpha_t := g(t, \Theta)$, where $g(t, \Theta)$ is a function that controls the convergence rate (Bonnabel 2013). We choose $g(t, \Theta)$ which satisfies the following as suggested in (Bonnabel 2013; LeCun 1998; Kiefer and Wolfowitz 1952);

$$\sum_{t=0}^{\infty} \alpha_t = +\infty \text{ and } \sum_{t=0}^{\infty} \alpha_t^2 < \infty. \tag{8}$$

- Line 8: A tangent vector $v_t \in \mathcal{T}_{\omega_l^t}\hat{\mathcal{M}}_l$ is computed using $v_t = h(\text{grad}\mathcal{L}(\omega_l^t), \alpha_t)$, where $h(\cdot)$ is a function that defines the next step on $\mathcal{T}_{\omega_l^t}\hat{\mathcal{M}}_l$ (see Figure 1). In this work, we employed $h(\text{grad}\mathcal{L}(\omega_l^t), \alpha_t) = -\alpha_t \text{grad}\mathcal{L}(\omega_l^t)$ to move the solution in a gradient descent direction with step size α_t.

- Line 9: Compute the next iterate using

$$\omega_l^{t+1} = \phi_{\omega_l^t}(v_t, \Gamma_l^t), \forall \omega_l^t \in \hat{\mathcal{M}}_l, \tag{9}$$

where ϕ is a mapping from $\mathcal{T}_{\omega_l^t}\hat{\mathcal{M}}_l$ onto $\hat{\mathcal{M}}_l$ (see Fig. 1). We employ retractions for implementation of ϕ. Therefore, this step enables us also to keep ω_l^{t+1} on $\hat{\mathcal{M}}_l$, $\forall l$.

Convergence Properties of Algorithm 1

In (Bonnabel 2013), convergence properties of SGD methods were analyzed for a particular class of manifolds following the proof methods suggested in (Saad 1998). In this work, we extend and employ their results to train CNNs using different kernel submanifolds. We first consider a collection of kernels $\{\omega_l^t\}_{t\geq 1}$ computed at the l^{th} layer of a CNN as a stochastic process. Then, the expected value of a submanifold gradient of a classification loss can be computed

by $\nabla\mathcal{L}(\omega_l^t) = E_{\mathcal{P}}\{\text{grad}\mathcal{L}(\omega_l^t, s)\}$[5]. In the following theorems, we provide convergence properties of Algorithm 1 for two cases where we use i) exponential maps, and ii) retractions for ϕ at the 9^{th} step of the algorithm.

i) Exponential maps for ϕ : An exponential map is used to map a vector $v_t \in \mathcal{T}_{\omega_l^t}\hat{\mathcal{M}}_l$ to a kernel along a geodesic curve on $\mathcal{M}_l$ which goes through ω_l^t in the direction of v_t.

Theorem 2 (Convergence of Algorithm 1 using exponential maps for ϕ)**.** *Suppose that the following conditions are satisfied;*
(1) Condition for maps onto kernel submanifolds: *$\hat{\mathcal{M}}_l$ is a connected compact Riemannian kernel submanifold.*
(2) Condition for kernels: *There exists a compact set $\mathcal{K}$ such that $\omega_l^t \in \mathcal{K}$, $\forall t \geq 1$. The minimal distance between conjugate kernels ω_l^t and ω_l^{t+1} denoted by $\rho(\omega_l^t, \omega_l^{t+1})$ is a geodesic that satisfies $\rho(\omega_l^t, \omega_l^{t+1}) > 0, \forall t, l$.*
(3) Condition for gradients: *The gradient is bounded on $\mathcal{K}$, such that $\exists \mathfrak{K} > 0$, $\|\text{grad}\mathcal{L}(\omega_l^t, s)\| \leq \mathfrak{K}$, $\forall s$ and $\forall \omega_l^t \in \mathcal{K}$.*
(4) Condition for the classification loss function*: We use a three times continuously differentiable function $\mathcal{L}(\omega_l) \geq 0$.*

Then, the loss function and the gradient converges almost surely (a.s.) by $\mathcal{L}(\omega_l^t) \xrightarrow[t\to\infty]{\text{a.s.}} \mathcal{L}(\hat{\omega}_l)$, where $\hat{\omega}_l$ is a minimum, and $\nabla\mathcal{L}(\omega_l^t) \xrightarrow[t\to\infty]{\text{a.s.}} 0$.

ii) Retractions for ϕ : In the experimental analysis, we used retractions to compute numerical approximations to exponential maps onto kernel submanifolds. Thus, we next provide the convergence properties for the case where ϕ is implemented using retractions.

Theorem 3 (Convergence of Algorithm 1 using retraction for ϕ)**.** *Suppose that the conditions **(1)-(4)** of Theorem 2 are satisfied, and that ϕ is a twice continuously differentiable retraction. Then, we have $\mathcal{L}(\omega_l^t) \xrightarrow[t\to\infty]{\text{a.s.}} \mathcal{L}(\hat{\omega}_l)$ and $\nabla\mathcal{L}(\omega_l^t) \xrightarrow[t\to\infty]{\text{a.s.}} 0$.*

An analysis of computational complexity of Algorithm 1, proofs of the theorems and implementation details of momentum, decay and learning rate are given in the supp. mat.

Experimental Analyses and Results

The proposed framework and the algorithm can be employed to train CNNs using different Riemannian manifolds. We train state-of-the-art CNNs using our algorithm on benchmark datasets and three manifolds, namely the sphere, the oblique and the Stiefel manifold. For a fair comparison with state-of-the-art methods, we used the same code and hyperparameters provided by the authors. Implementation details and additional results are given in the supp. mat.

We can explore the relationship between learned features by expounding geometry of submanifolds of kernels $\{\mathbf{W}_{d,l} \in \mathbb{R}^{A_l \times B_l \times C_l}\}_{d=1}^{D_l}, \forall l = 1, 2, \ldots, L$, computed at convolution layers and fully connected (FC) layers of a CNN consisting of L layers. We note that we identify kernel submanifolds by kernels $\mathbf{W}_l^{fc} \in \mathbb{R}^{C_l \times D_l}$ at FC layers, since $A_l = B_l = 1$ for FC layers. Then, we delineate the structure of patterns learned at different layers according to the constraints imposed on kernels by the submanifolds (see Proposition 1):

• **At the classification layer** ($l = L$), constraints imposed on FC kernels by manifold structures enable us to perform regularization (Lahlou and Oppenheim 2016). Thereby, our proposed framework and methods enable us to explore and utilize the relationship between two properties of regularization methods, namely i) regularization of models using data augmentation (Allen 1974), and ii) learning of models endowed with geometric invariants (Ng 2004). For instance, the Stiefel manifold implies an ℓ_2 norm regularization for classification (Bakır et al. 2004). Moreover, compact class conditional probability density functions can be learned over the Stiefel manifold (Turaga et al. 2011). If the sphere is used, then we perform trace norm regularization (Grave, Obozinski, and Bach 2011) since kernels residing on the sphere are normalized using the Frobenius norm (Golub and Van Loan 2013). Since generalized trace norms can be considered as the analog for matrices of what the weighted ℓ_1 norm is for vectors (Angst, Zach, and Pollefeys 2011), we perform regularization using kernels of the sphere as performed by Lasso type algorithms (Hastie, Tibshirani, and Wainwright 2015). Moreover, off-diagonal elements of kernels of the oblique manifold are regularized by assuming independence between covariates (Bickel and Levina 2008).
• **At the lower layers** ($l < L$), if we use kernels with $A_l > 1$ and $B_l > 1$, then we perform additional regularization on spatially distributed patterns (Richter and Roth 2015) within a neighborhood determined by A_l and B_l. For instance, square shape kernels of the Stiefel manifold construct the orthogonal group by Proposition 1. Then, the orthogonality constraints imposed by these kernels were used to learn representations of shape variation caused by both shape deformation and viewpoint changes (Bhattacharya and Bhattacharya 2012). Moreover, translation and scaling variability is removed from kernels of the sphere. This property has been used for statistical shape analysis to learn representations of unit length curves, shape primitives, and deformable shapes using the sphere (Srivastava et al. 2011). The constraints determined by the oblique manifold induce oblique rotation. Thus, oblique manifold was used to extract features which are mutually independent (Absil and Gallivan 2006). Since a detailed analysis of these properties is beyond the scope of this work, we explore them experimentally by training CNNs with the aforementioned manifolds for image classification, and analyzing their performance.

Comparison with Normalization Methods

In a recent work (Salimans and Kingma 2016), kernels are reparameterized by a fixed norm r that is initialized by the inverse of standard deviation of pre-activations. Their proposed method constructs a space of kernels identified by the sphere with a fixed radius. In this aspect, their proposed method can be considered as an instance of our proposed algorithm for the sphere. In other words, we can perform

[5] In practice, we receive a batch of samples $\mathfrak{s}^t \subseteq S$ at each t^{th} epoch. Assuming that each batch contains a single sample, $\nabla\mathcal{L}(\omega_l^t)$ denotes an average gradient computed by $\frac{1}{|S|}\sum_{i=1}^{|S|}\text{grad}\mathcal{L}(\omega_l^t, s_i)$.

Table 1: Results for Cifar-10 without DA. The results marked by † indicate the results reproduced by our implementation of the associated algorithm using the code provided by the authors of the related work. Classification error obtained using the baseline CNN is marked by red, and our best error is marked by blue.

Model	Class. Error (%)
NiN/NiN †	10.41/10.68
NiN + MOBN(Sp/Ob/St)†	9.03/8.95/**8.57**
NormProp (Arpit et al. 2016) / All-CNN-C(Springenberg et al. 2015)	9.11/9.08
SK/SK†/SK + MOBN	8.43/ 8.45/8.52
SK(WN)/SK(WN)†	8.46/8.51
SK(Sp/Ob/St) †	8.24/8.11/**7.94**
SK(BN)	8.05
SK+MOBN(WN) /†	7.31/7.33
SK+MOBN(Sp/Ob/St) †	6.88/6.75/**6.02**

optimization on other manifolds such as the oblique and the Stiefel manifold in addition to the sphere. In our method, each kernel can also reside in a different manifold endowed with a different geometry. For instance, we can perform optimization on different kernels that reside on the spheres with different radii. Therefore, our proposed methods enable us to have a better control on the geometry of kernel spaces to train CNNs compared to their method.

We examine this property by training their proposed CNN (Salimans and Kingma 2016) (denoted by SK) using our proposed methods for the sphere (Sp), the oblique manifold (Ob) and the Stiefel manifold (St). The results given in Table 1 are obtained by training CNNs on the Cifar-10 dataset without using data augmentation (DA). In Table 1, we observe that our methods that use the sphere (SK†(Sphere)) outperform the methods proposed in (Salimans and Kingma 2016) (SK†(WN)). This observation supports our claim for the benefit of employment of kernels belonging to spaces with varying manifold structures, e.g. the sphere with varying radii (which are determined by statistical properties of the data and the gradients of the loss). We also observe that we further boost the performance using the oblique and the Stiefel manifolds. This result also propounds conjectures and results provided in the previous works (Minh and Murino 2016) regarding regularization and invariance properties of models learned using different manifolds.

We aim to learn representations robust to statistical variance and mean of pre-activations by normalizing them using batch normalization (BN). If features input to a layer are i.i.d. with zero mean and unit variance, then we can equivalently obtain these robust representations using normalized kernels at that layer (Salimans and Kingma 2016). This property is explored in (Arpit et al. 2016) by proving that approximation error to covariance of pre-activations is upper bounded by a function of kernel norms. Therefore, normalized kernels that reside in the sphere are used to train CNNs in (Arpit et al. 2016). Following this property, BN is employed by removing just mean for the features obtained using normalized kernels, and this method is called mean-only BN (MOBN) (Salimans and Kingma 2016). The results given in Table 1 show that we can further boost the performance using MOBN. We also notice that, MOBN boosts the performance only if normalized kernels are used for training, such that the error of SK (MOBN) is 8.52% while that of SK (BN) is 8.05%.

In addition, we compare our methods with the normalized propagation (NormProp) method suggested in (Arpit et al. 2016). Briefly, NormProp implements a kernel normalization method using the ℓ_2 norm of kernels at FC layers, and the Frobenius norm at the other convolution layers. In this aspect, they identify kernels as elements of the sphere. However, they do not perform gradient projections and retractions used in SGD steps during BP, but they perform spherical projections during forward propagation. For comparison, we train the same Network in Network (NiN) (Lin, Chen, and Yan 2014) architecture utilized in (Arpit et al. 2016) using our methods. The results show that we obtain similar error for the sphere (9.03%) compared to their reported error (9.11%), and we can further boost the performance using the oblique (8.95%) and the Stiefel manifolds (8.57%). In addition, proposed methods boost the performance of NiN and SK by 2.29% and 2.43%, respectively. Note that, nine convolution layers are used in both NiN and SK, using kernels with different sizes (see (Salimans and Kingma 2016) and supplemental material.). In order to analyze the effect of number of layers to the performance boost, we perform experiments using larger networks in the next section.

Results for Training Large-scale CNNs

We first employ our methods for training of residual networks (Res) with constant depth (RCD) and stochastic depth (RSD) consisting of 110 layers (Huang, Liu, and Weinberger 2016; Huang et al. 2016). In order to explore how the proposed methods enable us to learn invariance properties as discussed above, we analyze the results for Cifar and Imagenet datasets that are augmented using standard DA methods (details are given in the supp. mat.). The results given in Table 2, show that the performance boost is larger for datasets prepared w/o DA compared to the augmented datasets. In addition, we can further boost the performance even for augmented datasets, since data augmentation is conducted using transformations on images (Mahendran and Vedaldi 2015), while the kernels computed at different layers can learn the invariants at different resolutions.

Since Res with less number of layers do not perform as well as SK and NiN on the Cifar dataset, we also provide the results for the Cifar-10 with DA in Table 3. The results show that our methods can boost the performance of the baseline Res. However, for a smaller Res (Res-20), the kernels of the sphere may outperform the kernels of the oblique as also observed in Table 2. Moreover, we observe that the amount of performance boost (for St) decreases from 0.78% to 0.35% as the number of layers increases to 44 in Table 3. On the other hand, for St, we obtain 0.65% and 2.11% boost for Cifar 10 and 100 with DA, and 0.72% and 4.98% boost for

Table 2: Classification error (%) for larger networks trained on Cifar-10 and Cifar-100 datasets with and without using DA.

Model	Cifar-10 w.DA	Cifar-100 w.DA	Cifar-10w/o DA	Cifar-100w/o DA
NormProp (Arpit et al. 2016)	7.47	29.24	9.11	32.19
PRONG (Desjardins et al. 2015)	7.32	-	-	-
RCD(Huang, Liu, and Weinberger 2016)/RCD(Huang et al. 2016)/RCD†	6.41/~/6.58	27.22/27.76/27.52	13.63/~/13.60	44.74/~/45.09
RCD+MOBN(Sp/Ob/St)†	6.22/6.07/**5.93**	26.44/25.99/**25.41**	13.11/12.94/**12.88**	42.51/42.30/**40.11**
RSD(Huang, Liu, and Weinberger 2016)/RSD(Huang et al. 2016)/RSD†	5.23/5.25/5.63	24.58/24.98/25.03	11.66/~/11.68	37.80/~/38.15
RSD+MOBN(Sp/Ob/St)†	5.20/5.14/**4.79**	23.77/23.81/**23.16**	10.91/10.93/**10.46**	36.90/36.47/**35.92**

Table 3: Results for residual networks (Res) on Cifar-10 with DA.

Model	Class. Error (%)
Res-20 (He et al. 2016)/†	8.75/8.81
Res-20+MOBN(Sp/Ob/St)†	8.25/8.43/**8.03**
Res-44 (He et al. 2016)/†	7.17/7.16
Res-44+MOBN(Sp/Ob/St)†	6.99/6.89/ **6.81**

Table 4: Results for Imagenet for single crop.

Model	Top-1 Error (%)
Res-18† / Res-34† / Res-50†	30.59/26.88/24.52
PRONG (Inception) (Desjardins et al. 2015)	28.90
Res-18+MOBN (Sp/Ob/St)†	29.13/28.97/**28.14**
Res-34+MOBN (Sp/Ob/St)†	26.04/25.73/**25.16**
Res-50+MOBN (Sp/Ob/St)†	23.79/23.70/**23.02**

Cifar 10 and 100 without DA, using Res consisting of 110 layers equipped with pre-activations (RCD) (see Table 2). Therefore, the amount of boost also depends on the number of classes and use of the augmentation methods.

We observe that performance boosts more for Cifar-100 compared to the results obtained for Cifar-10. This result suggests that we can learn feature representations of diverse patterns observed in large number of classes using kernels belonging to the manifolds. In order to scrutinize this observation, we provide the results for training of residual networks (Res) (He et al. 2016) using the Imagenet dataset in Table 4. The results[6] given in Table 4 complement the previous observations such that we have 2.45%, 1.72% and 1.50% performance boost (for St) using Res-18, Res-34 and Res-50, respectively. We also provide the performance of a recent method proposed for optimization on a probabilistic manifold of network parameters, called PRONG (Desjardins et al. 2015). In other words, PRONG implements an approximation for natural gradient descent. An interesting result is that Res-18 with 18 convolution layers, which was trained using our methods with manifolds, outperforms Inception (22 conv. layers) which was trained using PRONG (see Table 4).

[6]We compute Top-1 classification error for single crop using ILSVRC 2012 validation data. Implementation details are given in the supplemental material

Conclusion

We proposed a mathematical framework to explore and use geometric properties of spaces of convolution kernels in CNNs. In our theoretical results, we showed that spaces of normalized kernels that are computed by various normalization methods can be characterized by Riemannian submanifolds of convolution kernels. Following our theoretical results, we proposed a SGD algorithm for optimization on Riemannian kernel submanifolds to train CNNs with assurance of convergence to a solution at single minimum of loss.

We employed our algorithm to train state-of-the-art CNNs using benchmark datasets. We observed that our methods boost the performance of CNNs for various datasets prepared with and without using data augmentation methods. We believe that our results will guide researchers to develop geometry-aware training algorithms that employ powerful regularization methods and take advantage of invariance properties of kernels. In the feature work, we plan to apply our framework for other tasks such as segmentation, detection, action recognition and video analysis. Moreover, we will use our algorithm to train other deep networks such as auto-encoders and recurrent neural networks.

Acknowledgments

This work was partly supported by CREST, JST Grant Number JPMJCR14D1, JSPS KAKENHI Grant Number JP15H05919, and the ImPACT Program "Tough Robotics Challenge" of the Council for Science, Technology, and Innovation (Cabinet Office, Government of Japan).

References

Absil, P. A., and Gallivan, K. A. 2006. Joint diagonalization on the oblique manifold for independent component analysis. In *ICASSP*, volume 5, 945–948.

Absil, P.-A.; Mahony, R.; and Sepulchre, R. 2007. *Optimization Algorithms on Matrix Manifolds*. PUP.

Allen, D. M. 1974. The relationship between variable selection and data agumentation and a method for prediction. *Technometrics* 16(1):125–127.

Angst, R.; Zach, C.; and Pollefeys, M. 2011. The generalized trace-norm and its application to structure-from-motion problems. In *ICCV*, 2502–2509.

Arandjelovic, R.; Gronat, P.; Torii, A.; Pajdla, T.; and Sivic, J. 2017. Netvlad: Cnn architecture for weakly supervised place recognition. *IEEE Trans. Pattern Anal. Mach. Intell.* PP(99):1–1.

Arpit, D.; Zhou, Y.; Kota, B. U.; and Govindaraju, V. 2016. Normalization propagation: A parametric technique for removing internal covariate shift in deep networks. In *ICML*, 1168–1176.

Bakır, G. H.; Gretton, A.; Franz, M.; and Schölkopf, B. 2004. Multivariate regression via stiefel manifold constraints. In *Proc. of the 26th DAGM Symp. on Patt. Recog.*, 262–269.

Bhattacharya, A., and Bhattacharya, R. 2012. *Nonparametric Inference on Manifolds: With Applications to Shape Spaces*. Cambridge University Press.

Bickel, P. J., and Levina, E. 2008. Regularized estimation of large covariance matrices. *Ann. Statist.* 36(1):199–227.

Bonnabel, S. 2013. Stochastic gradient descent on riemannian manifolds. *IEEE Trans. Autom. Control* 58(9):2217–2229.

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

Glorot, X., and Bengio, Y. 2010. Understanding the difficulty of training deep feedforward neural networks. In *AISTATS*, volume 9, 249–256.

Golub, G., and Van Loan, C. 2013. *Matrix computations*. Johns Hopkins University Press, Baltimore, MD.

Grave, E.; Obozinski, G. R.; and Bach, F. R. 2011. Trace lasso: a trace norm regularization for correlated designs. In Shawe-taylor, J.; Zemel, R.; Bartlett, P.; Pereira, F.; and Weinberger, K., eds., *NIPS*. 2187–2195.

Hastie, T.; Tibshirani, R.; and Wainwright, M. 2015. *Statistical Learning with Sparsity: The Lasso and Generalizations*. Chapman and Hall/CRC.

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

Huang, Z., and Gool, L. V. 2017. A riemannian network for spd matrix learning. In *AAAI*.

Huang, G.; Sun, Y.; Liu, Z.; Sedra, D.; and Weinberger, K. Q. 2016. Deep networks with stochastic depth. In *ECCV*, 646–661.

Huang, G.; Liu, Z.; and Weinberger, K. Q. 2016. Densely connected convolutional networks. *CoRR* abs/1608.06993.

Kiefer, J., and Wolfowitz, J. 1952. Stochastic estimation of the maximum of a regression function. *Ann. Math. Statist.* 23(3):462–466.

Lahlou, T. A., and Oppenheim, A. V. 2016. Trading accuracy for numerical stability: Orthogonalization, biorthogonalization and regularization. In *ICASSP*, 4747–4751.

LeCun, Y.; Bottou, L.; Bengio, Y.; and Haffner, P. 1998. Gradient-based learning applied to document recognition. *Proceedings of the IEEE* 86(11):2278–2324.

LeCun, Y. 1998. Efficient backprop. In *Neural Networks: Tricks of the Trade*. Springer.

Lin, M.; Chen, Q.; and Yan, S. 2014. Network in network. In *ICLR*.

Mahendran, A., and Vedaldi, A. 2015. Understanding deep image representations by inverting them. In *CVPR*, 5188–5196.

Minh, H., and Murino, V. 2016. *Algorithmic Advances in Riemannian Geometry and Applications: For Machine Learning, Computer Vision, Statistics, and Optimization*. Springer International Publishing.

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

Mishra, B., and Sepulchre, R. 2016. Riemannian preconditioning. *SIAM Journal on Optimization* 26(1):635–660.

Neyshabur, B.; Salakhutdinov, R.; and Srebro, N. 2016. Data-dependent path normalization in neural networks. In *ICLR*.

Ng, A. Y. 2004. Feature selection, l1 vs. l2 regularization, and rotational invariance. In *ICML*.

Pascanu, R.; Mikolov, T.; and Bengio, Y. 2013. On the difficulty of training recurrent neural networks. In *ICML*, 1310–1318.

Richter, S. R., and Roth, S. 2015. Discriminative shape from shading in uncalibrated illumination. In *CVPR*, 1128–1136.

Saad, D. 1998. *Online Learning in Neural Networks*. Cambridge University Press.

Salimans, T., and Kingma, D. P. 2016. Weight normalization: A simple reparameterization to accelerate training of deep neural networks. In *NIPS*.

Springenberg, J.; Dosovitskiy, A.; Brox, T.; and Riedmiller, M. 2015. Striving for simplicity: The all convolutional net. In *ICLR*.

Srivastava, A.; Klassen, E.; Joshi, S. H.; and Jermyn, I. H. 2011. Shape analysis of elastic curves in euclidean spaces. *IEEE Trans. Pattern Anal. Mach. Intell.* 33(7):1415–1428.

Stoyan, D.; Kendall, W. S.; and Mecke, J. 2013. *Stochastic geometry and its applications*. Wiley, 3rd edition.

Sutskever, I.; Martens, J.; Dahl, G.; and Hinton, G. 2013. On the importance of initialization and momentum in deep learning. In *ICML*, volume 28, 1139–1147.

Turaga, P. K.; Veeraraghavan, A.; Srivastava, A.; and Chellappa, R. 2011. Statistical computations on grassmann and stiefel manifolds for image and video-based recognition. *IEEE Trans. Pattern Anal. Mach. Intell.* 33(11):2273–2286.

Wei, Y.; Xia, W.; Lin, M.; Huang, J.; Ni, B.; Dong, J.; Zhao, Y.; and Yan, S. 2015. Hcp: A flexible cnn framework for multi-label image classification. *IEEE Trans. Pattern Anal. Mach. Intell.* PP(99):1–1.

Wen, Z., and Yin, W. 2013. A feasible method for optimization with orthogonality constraints. *Mathematical Programming* 142(1):397–434.