A Kernel-based Weight Decorrelation for Regularizing CNNs

Yanhong Zhang[a,b], Fei Zhu[b,*]

[a] *College of Healthy Science and Engineering, Tianjin University of Traditional Chinese Medicine, China.*
[b] *Center for Applied Mathematics, Tianjin University, China.*

Abstract

Recent years has witnessed the success of convolutional neural networks (CNNs) in many machine learning and pattern recognition applications, especially in image recognition. However, due to the increasing model complexity, the parameter redundancy problem arises, and greatly degrades the performance of CNNs. To alleviate this problem, various regularization techniques, such as Dropout, have been proposed and proved their effectiveness. In this paper, we propose a novel adaptive kernel-based weight decorrelation (AKWD) framework, in order to regularize CNNs for better generalization. Different from existing works, the correlation between paring weights is measured by the cosine distance defined in RKHS associated with a specific kernel. The case with the well-known Gaussian kernel is investigated in detail, where the bandwidth parameter is adaptively estimated. By regularizing CNN models of different capacities using AKWD, better performance is achieved on several benchmark databases for both object classification and face verification tasks. In particular, when Dropout or BatchNorm is present, even higher improvements are obtained using the proposed AKWD, that demonstrates a good compatibility of the proposed regularizer with other regularization techniques.

Keywords:
Convolutional neural networks, regularization, weight decorrelation, object recognition, face verification

*Corresponding author
Email address: fei.zhu@tju.edu.cn (Fei Zhu)

1. Introduction

Convolutional neural networks (CNNs) [1, 2] have achieved remarkable success in a plenty of large-scale visual recognition tasks, *e.g.,* object recognition [3, 4], scene recognition [5, 6] and face verification [7, 8, 9], to name a few. Of particular note is the power of CNNs on image classification task, as it even outperforms human-level performance on well-known datasets such as ImageNet [10]. The strong representation ability of CNNs mainly benefits from the layered learning structure, in which the features are built hierarchically with the high-level representations generated from a cascade of lower-level ones [11, 12].

With the increasing data volume and the improving computational capacity, CNNs are getting wider and deeper to achieve better performance [13, 14]. However, large network structures contain a great number of learnable parameters, thus making CNNs prone to the overfitting problem especially when training data is relatively limited. Worse still, even trained with an enormous amount of data, some deep networks can be still inclined to overfitting [15]. To this end, many training techniques have been proposed to prevent overfitting and to boost the performance of CNNs, including data augmentation [3, 16], image preprocessing [17, 18, 19], parameter initialization [20, 4], learning decay policy and most importantly, the regularization techniques. A majority of these training techniques focus on reducing the parameter redundancy directly or un-directly to enhance the representation power of neural networks.

Various regularization techniques have been proposed and proved to be effective not only in mitigating overfitting, but also in improving the generalization of the trained models [11]. The first category of regularization techniques reduces the model complexity by decreasing the parameter redundancy of CNN models. The earliest regularizer weight decay applies a ℓ_2-norm penalty on the weights, such that the model is re-parameterized with less effective number of parameters [21]. In recent works [22, 23], sparse regularization was introduced to deep neural networks (DNN) to zero out the redundant parameters during the process of training, and thus remove unnecessary connections. Proposed by Hinton *et al* in [24], Dropout randomly discards a subset of neuron activations in order to prevent them from co-adapting too much. Several improvements are developed based on Dropout,

including Maxout [25] and Shakeout [26]. Instead of randomly dropping a portion of activations as in Dropout, the so-called DropConnect generalizes Dropout by randomly discarding a portion of weights [27].

An alternative category of regularization focuses on preventing overfitting while keeping the full capacity of the model, mainly by weight (or feature) decorrelation. As pointed out in [28], a model easy to overfit usually contains high-level redundancy in their weights, tending to learn similar patterns that generally correspond to noise in the training data. By far, a limited variety of studies have been dedicated to alleviate the overfitting problem from the aspect of weight decorrelation. In [29], the authors proposed two incoherent training methods either by minimizing the coherence of weight matrice, or by minimizing the correlation coefficients of bottleneck (BN) features in the context of speech recognition. In [15], the so-called DeCov regularizer attempts to decorrelate the representations by minimizing the cross-covariance of hidden activations. However, DeCov has the shortcoming of mass computation when features are in a high dimensional space. Differently, Rodríguez *et al* [28] proposed to directly decorrelate the weights by enforcing the global/local orthogonality regularizations, where the correlation (or similarity) between any two weight vectors is measured by the cosine angle between them. However, the use of the conventional cosine angle as weight correlation measurement is obviously not always the best choice, due to its insensitivity w.r.t the magnitudes of input vectors, as to be shown later. Worse still, a direct use of cosine-based decorrelation as in [28] leads to a global weight regularization term that jointly penalizes both the positively and negatively correlated weight pairs, although the latter ones contribute to competitive learning and self-organization [30].

Kernel machines shed light on extending the linear algorithms to the nonlinear scope [31, 32, 33]. By exploiting some nonlinear function, the original data is transformed from the input space to a reproducing kernel Hilbert space (RKHS), namely feature space $\mathcal{H}$. Implicitly determined by a specific kernel, the resulting feature space is usually of high-dimension, where existing linear methods can be performed on the mapped data [34]. For example, the feature space associated with the Gaussian kernel is an infinite-dimensional space. The primary merit of kernel method is that, it allows to easily compute the inner product between any pair of transformed data in $\mathcal{H}$, by only using the kernel function, without explicit knowledge on neither the nonlinear mapping function nor the feature space.

In this paper, we propose a generalized weight decorrelation regulariza-

tion framework based on kernel machines, enhance the representation ability of CNN models. When linear kernel is adopted, the proposed method is reduced to the global weight regularization in [28]. Specifically, the weight regularization with Gaussian kernel is investigated in detail. The resulting method, referred as adaptive kernel-based weight decorrelation (AKWD), essentially corresponds to a local weight decorrelation, that exclusively regularizes the correlated weight pairs whose angles are within a certain region, its range being controlled by the bandwidth parameter in the kernel. To appropriately set this parameter, an adaptive bandwidth tuning scheme is also presented. The generalization of the proposed method is verified by extensive experiments performed using CNNs of different capacities, for both object recognition and face verification tasks. The compatibility of the proposed method with other regularization techniques, *e.g.*, Dropout and BatchNorm, is also studied.

The contributions of this paper are summarized as follows:

- A novel adaptive kernel-based weight decorrelation framework is proposed, which mitigates the overfitting issue and improves the performance of CNNs.

- The proposed weight decorrelation framework improves the existing global weight regularization [28], as the former is more flexible and generalized. When a different nonlinear kernel is employed, the corresponding kernelized weight decorrelation method accordingly characterizes the nonlinear correlation between weight vectors in the associated RKHS. The case with the linear kernel reduces to the existing cosine similarity-based weight decorrelation method.

- The weight decorrelation with the Gaussian kernel is naturally a local method that exclusively regularizes the positively correlated weight pairs whose angles are within a certain region. By exploring the nonlinear correlation between weights, the insensitivity to the magnitudes of the weight vectors, which is a main drawback of the cosine similarity-based method is overcome. In addition, the bandwidth parameter in the Gaussian kernel, that controls the local property, is adaptively learned during updates.

- Extensive experiments are conducted on several public datasets for both object classification and face verification tasks. The proposed

algorithm achieves not only higher accuracy but also comparable convergence rate with the non-regularized model. Comparisons with several common regularizations and state-of-the-art methods demonstrate the effectiveness of the proposed regularizer, as well as its good compatibility with Dropout and BatchNorm.

The remainder of the paper is organized as follows. We first succinctly review the related works in Section 2. In Section 3, the proposed kernel-based weight decorrelation regularization framework is presented, and the case with Gaussian kernel is studied in detail. Section 4 and 5 validate the performance of the proposed method on two visual recognition tasks. Finally, Section 6 provides the conclusions and future works.

2. Weight decorrelation: from global to local

Recently, Rodríguez *et al.* [28] proposed an orthogonal weight (feature) decorrelation approach based on the cosine similarity. Before revisiting the method, some notations are firstly given. Let $W \in \mathbb{R}^{N\times M}$ be a matrix recoding the weight parameters connecting the $(I-1)$-th layer to the I-th layer, where the former consists of M neurons and the latter is composed by N neurons $\{h_1, h_2, ...h_N\}$. Let $\boldsymbol{w}_i$ be the i-th row of W, representing the weight vector connecting the neurons of the $(I-1)$-th layer to the neuron h_i. If $\boldsymbol{w}_i = \boldsymbol{w}_j$ when $i \neq j$, then h_i and h_j is said to be positively correlated. They are said to be negatively correlated, if $\boldsymbol{w}_i = -\boldsymbol{w}_j$. To encode more information, the output neurons are expected to be mutually dissimilar, namely the correlation between them should be as small as possible. To this end, the authors in [28] proposed to directly perform decorrelation on weight vectors instead of seeking for activation independence.

In the relevant paper [28], both a global and a local weight regularization strategies were developed. Given a pair of weight vectors $\boldsymbol{w}_i$ and $\boldsymbol{w}_j$, the corresponding weight correlation is measured by the cosine of the angle between them, namely

$$\cos(\boldsymbol{w}_i, \boldsymbol{w}_j) = \frac{\langle \boldsymbol{w}_i, \boldsymbol{w}_j \rangle}{\|\boldsymbol{w}_i\| \cdot \|\boldsymbol{w}_j\|}, \tag{1}$$

where $\langle \cdot, \cdot \rangle$ denotes the inner product of two entries.

Considering the correlations of all the paring weights for the l-th layer, a global orthogonal weight regularization term is expressed by

$$\mathcal{L}_{global}(W) = \sum_{i=1}^{N} \sum_{j=1,\ j\neq i}^{N} \cos^2(\boldsymbol{w}_i, \boldsymbol{w}_j), \tag{2}$$

which is included to the target loss function. This loss reaches its minimum when the paring weight vectors are orthogonal to each other. It is noteworthy that the above formulation uniformly regularizes all kinds of correlations between all pairs of weight vectors $(\boldsymbol{w}_i, \boldsymbol{w}_j)$. However, there are evidences that negative correlations should be kept, since they are advantageous to inhibitory connections, competitive learning and self-organization, as indicated in [30, 28].

A local orthogonal weight decorrelation penalizes exclusively the positively correlated weight pairs whose angles are within $[-\frac{\pi}{2}, \frac{\pi}{2}]$, whiling retaining the negatively correlated ones. Towards this end, the authors improve (2) by elaborating the following regularizer that penalizes the gradients for angles smaller than $\frac{\pi}{2}$, with

$$\mathcal{L}_{local}(W) = \sum_{i=1}^{N} \sum_{j=1,\ j\neq i}^{N} \log(1 + e^{\lambda(\cos(\boldsymbol{w}_i, \boldsymbol{w}_j)-1)}), \tag{3}$$

where $\|\boldsymbol{w}_i\| = 1$, $\|\boldsymbol{w}_j\| = 1$, and λ is a predefined hyperparameter controlling the minimum angle between paring weights to be regularized. In [28], λ is empirically set to 10 such that the regularization effect approximately vanishes for the angles over $\frac{\pi}{2}$.

3. Proposed method

3.1. On the limitations of cosine-based weight (de)correlation

To represent the correlation between weight vectors by the conventional cosine similarity, as did in [28], has several limitations. First, due to the measure's insensitivity against the magnitudes of the weight vectors, it lacks discriminative ability especially when the ℓ_2-norm of the paring weight vectors are of great difference. For example, as shown in Fig. 1, $\alpha < \beta$ holds, thus $\cos(\boldsymbol{b}_2, \boldsymbol{b}_1) < \cos(\boldsymbol{b}_1, \boldsymbol{a})$ representing that vector $\boldsymbol{b}_1$ is more correlated and similar to vector $\boldsymbol{a}$ than to $\boldsymbol{b}_2$. However, it is more reasonable to say that

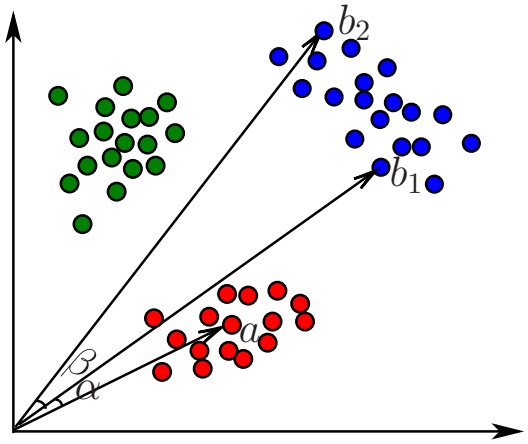


Fig. 1. Comparison between the conventional cosine similarity and the kernelized cosine similarity of 2D weight vectors. The points with different colors remark the features from different classes, concretely $\boldsymbol{a} = (2,1)^\top, \boldsymbol{b}_1 = (5,5)^\top, \boldsymbol{b}_2 = (3,7)^\top$. When using the conventional cosine similarity, $\alpha < \beta$ holds, which is less reasonable. However, the cosine similarity with Gaussian kernel can better characterize their correlation with $\kappa(\boldsymbol{b}_2, \boldsymbol{b}_1) > \kappa(\boldsymbol{b}_1, \boldsymbol{a})$.

vector $\boldsymbol{b}_1$ is more similar to vector $\boldsymbol{b}_2$ than to $\boldsymbol{a}$, as observed from the figure. Although such problem caused by various vector scales can be relieved by the mean-centered normalization [35], it heavily depends on the available vectors and does not get the root of the problem. Moreover, the cosine similarity can merely reflect the linear correlation between weight vectors in the Euclidean space. However, it is possible that to measure the correlation in another nonlinear space can better reveal the intrinsic nonlinear correlation between weight vectors. Lastly, the decorrelation based on cosine similarity in (2) essentially results a global weight regularizer that penalizes both the positively and negatively correlated weight pairs, regardless of the fact that the latter ones are proved useful in competitive learning and self-organization [30].

3.2. General problem formulation

To overcome the aforementioned limitations of cosine-based weight decorrelation, we propose a novel and generalized weight decorrelation framework mainly by taking advantage of the kernel machines, where the cosine similarity is defined in the reproducing kernel Hilbert space (RKHS), *i.e.*, feature space associated to a specific kernel.

Let $\varphi(\cdot)$ be a nonlinear function that transforms the weight vector $\boldsymbol{w}_i$

into $\varphi(\boldsymbol{w}_i)$, for $i = 1, \ldots, N$. Let $\kappa(\cdot, \cdot)$ represent the reproducing kernel associated with this nonlinear map, and $\mathcal{H}$ be the resulting feature space. Analogous to the cosine similarity between two weight vectors in the input space as in (1), the kernelized cosine similarity is defined in the RKHS $\mathcal{H}$, with

$$\begin{aligned} \text{kernel\_cosine}(\boldsymbol{w}_i, \boldsymbol{w}_j) &= \frac{\langle \varphi(\boldsymbol{w}_i), \varphi(\boldsymbol{w}_j) \rangle_{\mathcal{H}}}{\|\varphi(\boldsymbol{w}_i)\|_{\mathcal{H}} \cdot \|\varphi(\boldsymbol{w}_j)\|_{\mathcal{H}}} \\ &= \frac{\kappa(\boldsymbol{w}_i, \boldsymbol{w}_j)}{\sqrt{\kappa(\boldsymbol{w}_i, \boldsymbol{w}_i)}\sqrt{\kappa(\boldsymbol{w}_j, \boldsymbol{w}_j)}}, \end{aligned} \tag{4}$$

where $\langle \cdot, \cdot \rangle$ represents the inner product in $\mathcal{H}$, and $\| \cdot \|_{\mathcal{H}}$ is the associated norm. The well-known kernel trick, namely

$$\kappa(\boldsymbol{x}, \boldsymbol{y}) = \langle \varphi(\boldsymbol{x}), \varphi(\boldsymbol{y}) \rangle_{\mathcal{H}} \tag{5}$$

is applied. Fig. 2 provides a schematic illustration of the kernelized similarity. Compared with the conventional cosine similarity used in [28], the kernelized cosine similarity is more flexible and generalized. For the latter, the similarity between weight vectors is evaluated in the RKHS associated to a specific kernel. When a different nonlinear kernel is employed, *e.g.* the ones listed in Table 1, the corresponding kernelized cosine similarity is able to accordingly characterize the nonlinear correlation between weight vectors in the associated feature space. It is also noteworthy that the case with the linear inner product kernel results the linear technique, *i.e.*, the conventional cosine angle that measures the linear correlation between weight vectors in the Euclidean space, as in [28].

By adopting the kernelized cosine similarity in (4), the resulting correlation between $\boldsymbol{w}_i$ and all the other weights of the l-th layer is evaluated by

$$\mathcal{L}(\boldsymbol{w}_i) = \sum_{j=1,\ j \neq i}^{N} \text{kernel\_cosine}(\boldsymbol{w}_i, \boldsymbol{w}_j), \tag{6}$$

and the correlations of all the paring weights for the l-th layer are expressed

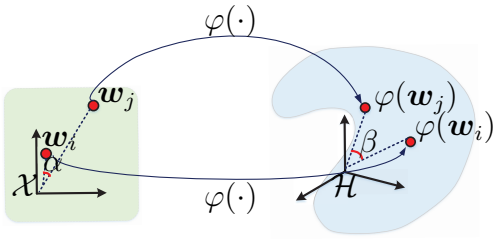


Fig. 2. Illustration of the kernelized cosine similarity. Kernel machines transform the input space $\mathcal{X}$ (represented by the green region) to a higher-dimensional RKHS space $\mathcal{H}$ (represented by the blue region). The conventional cosine similarity, *i.e.* $\cos\alpha$, measures the cosine of angle between weight vectors $\boldsymbol{w}_i$ and $\boldsymbol{w}_j$ in $\mathcal{X}$. The kernelized cosine similarity, *i.e.* $\cos\beta$, evaluates the cosine of angle between the mapped data $\varphi(\boldsymbol{w}_i)$ and $\varphi(\boldsymbol{w}_j)$ in $\mathcal{H}$.

Table 1. Several common kernels and their gradients with respect to $\boldsymbol{w}_i$

	Kernel	$k(\boldsymbol{w}_i, \boldsymbol{w}_j)$	$\nabla_{\boldsymbol{w}_i} k(\boldsymbol{w}_i, \boldsymbol{w}_j)$
	Linear	$\boldsymbol{w}_i^T \boldsymbol{w}_j$	$\boldsymbol{w}_j$
	Polynomial	$(\boldsymbol{w}_i^T \boldsymbol{w}_j + c)^d$	$d(\boldsymbol{w}_i^T \boldsymbol{w}_j + c)^{(d-1)} \boldsymbol{w}_j$
	Sigmoid	$\tanh(\gamma \boldsymbol{w}_i^T \boldsymbol{w}_j + c)$	$\frac{\gamma}{\cosh^2(\gamma \boldsymbol{w}_i^T \boldsymbol{w}_j + c)} \boldsymbol{w}_j$
RBF kernel	Gaussian	$\exp\left(-\frac{\|\boldsymbol{w}_i - \boldsymbol{w}_j\|^2}{2\sigma^2}\right)$	$-\frac{1}{\sigma^2} k(\boldsymbol{w}_i, \boldsymbol{w}_j)(\boldsymbol{w}_i - \boldsymbol{w}_j)$
RBF kernel	Exponential	$\exp\left(-\frac{\|\boldsymbol{w}_i - \boldsymbol{w}_j\|}{2\sigma^2}\right)$	$-\frac{1}{2\sigma^2} k(\boldsymbol{w}_i, \boldsymbol{w}_j) sgn(\boldsymbol{w}_i - \boldsymbol{w}_j)$
RBF kernel	Laplacian	$\exp\left(-\frac{\|\boldsymbol{w}_i - \boldsymbol{w}_j\|}{\sigma}\right)$	$-\frac{1}{\sigma} k(\boldsymbol{w}_i, \boldsymbol{w}_j) sgn(\boldsymbol{w}_i - \boldsymbol{w}_j)$

by

$$\begin{aligned} \mathcal{L}(W) &= \sum_{i=1}^{N} \sum_{j=1,\ j\neq i}^{N} \text{kernel\_cosine}(\boldsymbol{w}_i, \boldsymbol{w}_j) \\ &= \sum_{i=1}^{N} \sum_{j=1,\ j\neq i}^{N} \frac{\kappa(\boldsymbol{w}_i, \boldsymbol{w}_j)}{\sqrt{\kappa(\boldsymbol{w}_i, \boldsymbol{w}_i)}\sqrt{\kappa(\boldsymbol{w}_j, \boldsymbol{w}_j)}}. \end{aligned} \tag{7}$$

Next, we utilize $\mathcal{L}(W)$ as a regularization term and derive the corresponding gradient for back-propagation, by which the weight correlation of W defined in RHKS $\mathcal{H}$ is expected to be suppressed. Let X be the inputs and y be the labels. In the training process, the regularization term in (7) will be added to the original target cost $\mathcal{J}(W; X, y)$ for joint supervision, yielding

$$\widehat{\mathcal{J}}(W; X, y) = \mathcal{J}(W; X, y) + \lambda \mathcal{L}(W), \tag{8}$$

where λ is the hyper-parameter which controls the degree of the regularization. Regarding back-propagation, the gradient of $\mathcal{L}(W)$ with respect to the

weight vector $\boldsymbol{w}_i$ computes

$$\frac{\partial \mathcal{L}(W)}{\partial \boldsymbol{w}_i} = \sum_{j=1, j\neq i}^{N} \frac{1}{\sqrt{K_{ii}}\sqrt{K_{jj}}} \left(\frac{\partial K_{ij}}{\partial \boldsymbol{w}_i} - \frac{\partial K_{ii}}{\partial \boldsymbol{w}_i} \frac{K_{ij}}{2K_{ii}} \right), \tag{9}$$

where we denote $K_{ij} \triangleq \kappa(\boldsymbol{w}_i, \boldsymbol{w}_j)$ for notation simplicity. By adopting (9), we obtain the back-propagation gradient of the new target cost $\widehat{\mathcal{J}}$ in terms of weight vector $\boldsymbol{w}_i$ with

$$\triangle \boldsymbol{w}_i = -\alpha \Big(\frac{\partial \mathcal{J}}{\partial \boldsymbol{w}_i} + \lambda \frac{\partial \mathcal{L}}{\partial \boldsymbol{w}_i} \Big), \tag{10}$$

where α represents the global learning rate.

It is noteworthy that the proposed kernel-based weight decorrelation is a flexible and generalized framework suitable for different kernels. By adopting a specific kernel function and its gradient with respect to the first entry, explicit formulas can be easily obtained from (9) and (10). We list in Table 1 the expressions of several valid kernels and their gradients with respect to $\boldsymbol{w}_i$, that can be brought into formulas (9) and (10) directly. For all the kernels in Table 1, the complexity of the back-propagation update computes $\mathcal{O}(N^2)$. It is noteworthy that the global orthogonal weight regularization method in [28] can be taken as a special case of the proposed framework, where the linear kernel is applied with additional constraints $\|\boldsymbol{w}_i\| = 1$ and $\|\boldsymbol{w}_j\| = 1$.

3.3. Local weight decorrelation with Gaussian kernel

We study in detail a special case of the proposed weight deccorelation framework with the well-known Gaussian kernel, given by

$$\kappa(\boldsymbol{w}_i, \boldsymbol{w}_j) = \exp\left(-\frac{\|\boldsymbol{w}_i - \boldsymbol{w}_j\|_2^2}{2\sigma^2} \right), \tag{11}$$

where σ is the bandwidth parameter of the kernel. When Gaussian kernel is applied, the corresponding kernelized cosine similarity (4) is simplified to the the kernel function itself, since $\kappa(\boldsymbol{w}_i, \boldsymbol{w}_i) = \kappa(\boldsymbol{w}_j, \boldsymbol{w}_j) = 1$ in this case.

By integrating expression in (11) to (7), the regularization term, which is defined as the correlations of all the paring weights for the l-th layer, is

given by

$$\begin{aligned}\mathcal{L}(W) &= \sum_{i=1}^{N}\sum_{j=1,\ j\neq i}^{N}\kappa(\boldsymbol{w}_i,\boldsymbol{w}_j)\\ &= \sum_{i=1}^{N}\sum_{j=1,\ j\neq i}^{N}\exp\left(-\frac{\|\boldsymbol{w}_i-\boldsymbol{w}_j\|_2^2}{2\sigma^2}\right).\end{aligned} \tag{12}$$

This regularizer will be added to the target cost function for joint supervision during the training process. In this case, the expression of the gradient with respect to the weight vector $\boldsymbol{w}_i$ in (9) becomes

$$\begin{aligned}\frac{\partial\mathcal{L}(W)}{\partial\boldsymbol{w}_i} &= \sum_{j=1,j\neq i}^{N}\frac{\partial\kappa(\boldsymbol{w}_i,\boldsymbol{w}_j)}{\partial\boldsymbol{w}_i}\\ &= -\frac{1}{\sigma^2}\sum_{j=1,j\neq i}^{N}\kappa(\boldsymbol{w}_i,\boldsymbol{w}_j)(\boldsymbol{w}_i-\boldsymbol{w}_j),\end{aligned} \tag{13}$$

and the back-propagation gradient in (10) becomes

$$\triangle\boldsymbol{w}_i = -\alpha\Big(\frac{\partial\mathcal{J}}{\partial\boldsymbol{w}_i}-\frac{\lambda}{\sigma^2}\sum_{j\neq i}k(\boldsymbol{w}_i,\boldsymbol{w}_j)(\boldsymbol{w}_i-\boldsymbol{w}_j)\Big), \tag{14}$$

where, $\mathcal{J}$ is the original target loss function, λ represents the degree of the regularization, and α is the global learning rate. In the experiments, we empirically normalize the weight vectors at each iteration by $\|\boldsymbol{w}_i\| = 1$, for $i = 1, 2, ..., N$.

The resulting method improves the existing weight decorrelation regularization from two aspects. Firstly, compared to the conventional cosine similarity used in [28], the cosine similarity with Gaussian kernel is more sensitive to the magnitudes of the input weight vectors. Therefore, it can better characterize the underlying nonlinear correlation between weights. Secondly, the weight decorrelation regularizer with Gaussian kernel is naturally a local one. Here, "local" means the regularization can exclusively punish the positively correlated weight pairs while retaining the negatively correlated ones. In [28], the authors achieved this purpose by using an elaborately designed correlation function, as in (3). Differently, the proposed weight decorrelation with Gaussian kernel yields a local regularization in a straighter and simpler

way, thanks to the good property induced by the applied kernel. Fig. 3 illustrates the kernelized cosine similarity in terms of $\epsilon \triangleq \|\boldsymbol{w}_i - \boldsymbol{w}_j\|_2$ with Gaussian kernel, under different bandwidth σ. For a fixed σ, $\epsilon = 0$ means the strongest correlation between weight vectors, with $\kappa(\boldsymbol{w}_i, \boldsymbol{w}_j) = 1$; and an arbitrarily large ϵ signifies that two weight vectors are non-correlated at all, with $\kappa(\boldsymbol{w}_i, \boldsymbol{w}_j) = 0$. The correlation function is of relatively important value only when the variable ϵ is within a small region around zero, with the range of region being controlled by the bandwidth parameter σ. When ϵ is far from zero, the function will dramatically drop to zero. Reflecting in the back-propagation gradient, this signifies that only the highly-correlated (or positively correlated) weight pairs will be regularized, while the over-dissimilar (or negatively correlated) ones will be kept in the training process.

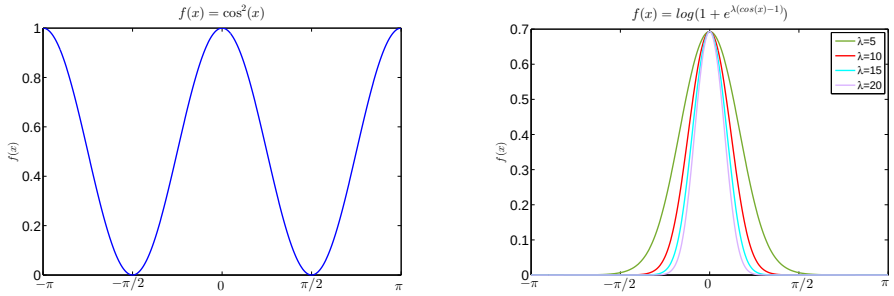
Concerning the estimation of σ, an adaptive method is applied, where the parameter is updated according to the mean distance among weight vectors, as recommend in [36]. At each iteration, the bandwidth σ is adptively estimated by

$$\sigma^2 = \frac{1}{N} \sum_{i=1}^{N} (\boldsymbol{w}_i - \bar{\boldsymbol{w}}_i)^2, \tag{15}$$

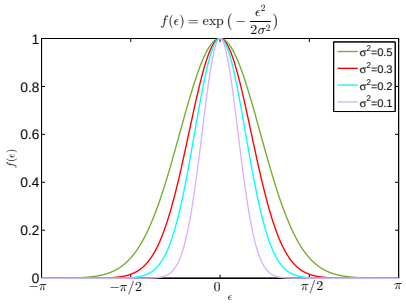
where $\bar{\boldsymbol{w}}_i = \frac{1}{M} \sum_{j=1}^{M} \boldsymbol{w}_{ij}$, N and M denote the number and the dimensionality of the weight vectors, respectively. Here, the parameter α controls the minimum angle-of-influence of the regularizer and is learnable in an adaptive way. This is an advantage over the existing local weight decorrelation method [28], where the hyperparameter λ in (3), which controls the minimum angle between paring weights, should be set manually. The proposed method is referred as the adaptive kernel-based weight decorrelation (AKWD). The optimization steps of the proposed AKWD method with Gaussian kernel are summarized in Algorithm 1.

4. Experiments for object classification

We investigate the effectiveness of the proposed weight decorrelation regularizer with Gaussian kernel, on two typical visual recognition tasks, namely the object classification and the face verification. In the following, all the experiments are implemented on TiTan-X GPUs using stochastic gradient descent (SGD) method, and the open Caffe library is applied [37]. For each task, the same training setting is applied to every comparing method with



(a) Global loss (eq. (2)) (b) Local loss (eq. (3))



(c) Kernelized loss (eq. (12))

Fig. 3. Comparison among the three loss functions. **(a)** global loss with conventional cosine similarity, where $x \triangleq \langle \boldsymbol{w}_i, \boldsymbol{w}_j \rangle$ is the cosine angle [28]; **(b)** local loss that penalizes exclusively positive correlations given for different λ value, where $x \triangleq \langle \boldsymbol{w}_i, \boldsymbol{w}_j \rangle$ is the cosine angle [28]; **(c)** kernelized loss with Gaussian kernel in terms of $\epsilon \triangleq \|\boldsymbol{w}_i - \boldsymbol{w}_j\|_2$ under various bandwidth σ, which is naturally a local one.

or without regularization, in order to keep a fair comparison. If not specially mentioned, the activation function is chosen as the commonly-used ReLU [38], and the weight decay and the momentum are set to be 0.0005 and 0.9, respectively. More training details for each experiment will be specified in corresponding sections. For testing, the softmax classifier is utilized for the object classification task, and the similarity score is measured by cosine distance for the face verification task.

This section focuses on studying the effectiveness of the proposed method on the object classification task, on three popular benchmark datasets, *i.e.*, MNIST, CIFAR-10 and CIFAR-100.

Algorithm 1 The algorithm of the proposed AKWD with Gaussian kernel.

Require: Layer parameters W to be regularized, hyperparameters λ and learning rate α.

for each iteration $t = 1, 2, \ldots, T$ **do**
  normalize each row of W, s.t. $\|\boldsymbol{w}_i\| = 1$
  compute σ_t^2 and $\mathcal{L}^t(W)$, then compute loss function $\widehat{\mathcal{J}}^t = \mathcal{J}^t + \lambda\mathcal{L}^t$
  compute the gradients $\nabla\mathcal{L}^t_{\boldsymbol{w}_i}$ using (13)
  update parameters by
  $\boldsymbol{w}_i^{t+1} = \boldsymbol{w}_i^t - \alpha(\nabla\mathcal{J}^t_{\boldsymbol{w}_i} + \lambda\nabla\mathcal{L}^t_{\boldsymbol{w}_i})$
end for

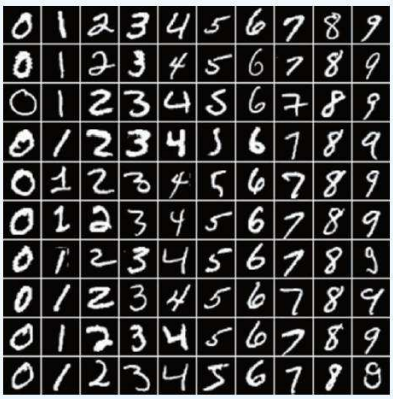


Fig. 4. Samples in MNIST.

4.1. A first study of AKWD using simple network on MNIST

In this subsection, we design a series of experiments by adopting a simple network on MNIST dataset, in order to study the behaviors of the proposed weight decorrelation regularizer. The MNIST dataset [1] is popular for handwritten digit recognition, which consists of 60,000 grayscale images drawn from 10 classes (0-9), including 50,000 training samples and 10,000 testing samples. The digits are size-normalized and centered to an uniform size with 28×28 pixels, as shown in Fig. 4. Divided by 256, the original data are again scaled to $[0, 1]$ as inputs. A simple network, *i.e.*, LeNet [1] is adopted with slight modifications: the filter number of the first two convolutional layers are increased from 20 and 50 to 32 and 64, respectively; the number of units of the first fully-connected layer is expanded to 512. The resulting network structure is detailed in Table 2. The model is trained according to the default

Table 2. The simple network architectures for MNIST and CIFAR-10/100

MNIST	conv1	pool1	conv2	pool2	conv3	pool3	fc1
Filt Size	5×5	2×2	5×5	2×2	-	-	1×1
Num Filt	32	-	64	-	-	-	512
Stride	1	2	1	2	-	-	1
Padding	0	-	0	-	-	-	-
CIFAR	conv1	pool1	conv2	pool2	conv3	pool3	fc1
Filt Size	5×5	3×3	5×5	3×3	5×5	3×3	1×1
Num Filt	32	-	32	-	64	-	64
Stride	1	2	1	2	1	2	1
Padding	2	-	2	-	2	-	-

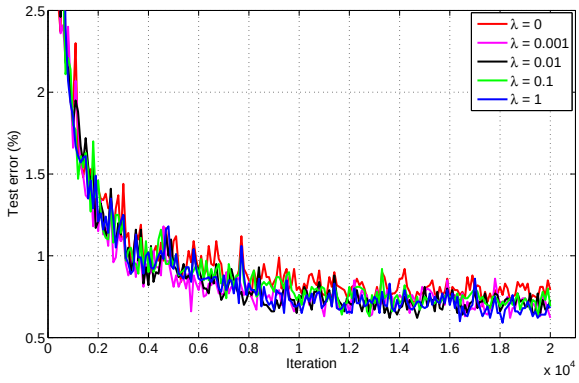


Fig. 5. Comparison of test error rate on MNIST validation set in terms of different values of regularization parameter λ, along with iterations.

learning rate policy and parameter initialization, and the maximum iteration number is set as 20k.

We firstly examine the classification performance of the proposed weight decorrelation regularizer AKWD under different levels of regularization, with the regularization parameter λ ranging within the set $[0, 0.001, 0.01, 0.1, 1]$. The proposed regularizer is imposed to one convolutional layer, namely *conv2*. Fig. 5 reports the classification results in terms of error rate on MNIST validation set, using different values of λ. It is observed that the regularized models with positive values of λ all yield smaller testing errors, when compared to the original, un-regularized model with $\lambda = 0$. In particular, on

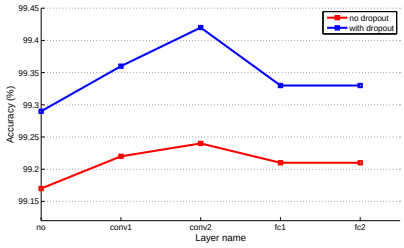


Fig. 6. The effectiveness of AKWD to different layers on MNIST. **Blue**: joint use of dropout and AKWD at different layers. **Red**: single use of AKWD without dropout at different layers.

MNIST dataset, the best classification result is achieved at the highest regularization level with $\lambda = 1$.

Next, we study the performance of the proposed regularizer on different layers. To this end, AKWD is applied to one of the layers of the network, namely to the layer *conv1*, *conv2*, *fc1* and the final fully-connected layer *fc2*. As illustrated by the red line in Fig. 6, regardless of the layer where the regularizer is placed to, AKWD always leads to an improved testing accuracy when compared to the original un-regularized network. In particular, better results are achieved when regularizing the convolutional layers than regularizing the fully-connected layers by AKWD.

Furthermore, the compatibility of the proposed AKWD with other regularization techniques is explored, and the well-known dropout is considered as representative. To this end, we apply dropout after the first fully-connected layer with the drop radio set to 0.5 by default. Again, AKWD is applied to one of the layers, namely to *conv1*, *conv2*, *fc1* and *fc2*. The effectiveness of combining dropout and AKWD at different layers is presented in Fig. 6 by the blue line. We observe that, the accuracy improvements obtained by a joint use of dropout and AKWD at different layers (blue line) are consistent with that obtained by using AKWD solely (red line). More importantly, the combined use of dropout and AKWD always leads to larger margins of improvement over its baseline model (the case where **Layer name** being **no**), when compared to the case with single use of AKWD. It demonstrates that the proposed AKWD has a good compatibility with dropout, as the combined regularization scheme can further boost the classification performance than the single use of AKWD. Also, Table 3 reports the averaged test

Table 3. The averaged recognition error rate (%) over five runs on MNIST dataset

Method	Error Rate
Baseline	0.83
OrthoReg	0.79
AKWD	0.76
Baseline + Dropout	0.71
OrthoReg + Dropout	0.63
AKWD + Dropout	**0.58**

error rate obtained by using the baseline model, the regularized model by OrthoReg [28], and the regularized model by the proposed AKWD. For each model, two cases are considered, namely adding dropout or not. It is observed that whether dropout is present or not, AKWD outperforms both the baseline and the OrthoReg. And it is the joint use of dropout and AKWD that achieves the best classification performance with the lowest recognition error rate.

Lastly, we investigate the relationship between weight decorrelation and overfitting. To this end, Fig. 7 traces the performance gap between training and testing over iteration, using the the baseline model, the regularized model by OrthoReg, and the regularized model by AKWD. It is noticed that the model regularized by AKWD has the lowest level of overfitting among the three comparing methods, as it generally provides the lowest train-test accuracy gap. In addition, both OrthoReg and AKWD lead to lower levels of overfitting than the un-regularized baseline model, indicating that the regularization effect brought by weight decorrelation helps to reduce the overfitting.

4.2. Evaluation on CIFAR-10

On CIFAR-10 dataset, the generalization of the proposed method is verified by using two networks of different capacities. The CIFAR-10 dataset [39] contains 60,000 32×32 RGB images from 10 object classes, with 6,000 images per class. It consists of a training set of 50,000 images and a testing set of 10,000 images. Fig. 8 shows the examples from every object class.

Experiments are carried out using two types of CNNs: a shallow network and a deeper one. The shallow CNN network is chosen as CIFAR-10 network

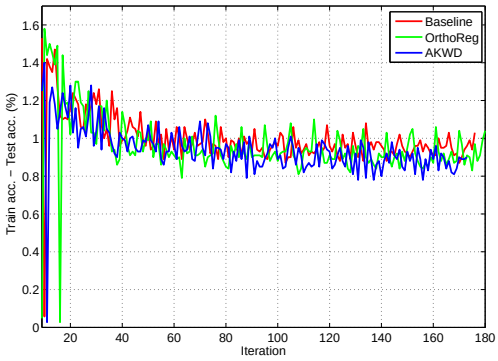


Fig. 7. The overfitting of different models.

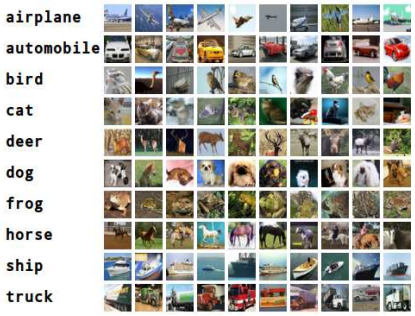


Fig. 8. Samples in CIFAR-10/100.

built in Caffe library, with the network structure given in Table 2. For this model, we start the training with a learning rate of 0.001, which is divided by 10 at 8k and 14k iterations. The maximum iteration number is set to 20k. The input data are preprocessed by a per-pixel mean subtraction, and no data augmentation is applied. The deeper CNN network is chosen as WRN-28-10 in [40] with slight differences. The input data is preprocessed by simple mean/std normalization. Also, we follow the standard data augmentation for training: four pixels are padded on each side and a 32×32 crop is randomly sampled from the padded image or its horizontal flipping, as in [19]. This model is trained with the mini-batch size of 128 using two GPUs, and 64 in parenthesis. The learning rate is initially set to 0.1, and reduces by multiplying 0.2 at 24k, 48k and 64k iterations. The maximum iteration number is set to 80k. Different to training, only the single view

of the original 32×32 image is used for testing. In the following, the shallow network and the deeper one are referred as [S] and [D], respectively, for simplicity. According to the experimental results analysis in MNIST, the proposed AKWD is applied to regularize only the last convolutional layer on both networks, with the regularization parameter λ set to 1.

For each type of CNN, comparative study is carried out among the baseline model, the regularized model by OrthoReg and the regularized model by AKWD. As data augmentation, image preprocessing and parameter initialization techniques also reduce the overfitting effect, the comparison with these three strategies is performed using the shallow CNN. Specifically, the data augmentation as in [19] is applied, and image preprocessing is carried on by randomly adjusting brightness and contrast with whitening subsequently. Regarding parameter initialization, the "Xavier" initialization is considered for comparison. The combined group and exclusive sparsity regularization (CGES) in [22] is also compared using the shallow network. Moreover, other six state-of-the-art methods, namely Maxout [25], DropConnect [27], Highway Network [41], ELU-Network [42], LSUV [43] and All-CNN [44], are compared. The recognition results of all the comparing methods are listed in Table 4, where DA and PP are abbreviations for data augmentation and image preprocessing, respectively.

It is observed that on the shallow networks, the proposed AKWD achieves best accuracy among all the comparing methods. Of particular note is the image preprocessing with whitening operation, which has both regularization and decorrelation effects but is surpassed by AKWD. On both shallow and deeper networks, the regularized model by AKWD provides smaller test error over the original model and the regularized model by OrthoReg. Among all the comparing methods, it is the regularized deeper CNN model with AKWD (AKWD[D]) that yields the best classification result, demonstrating the effectiveness of the proposed approach in improving the generalization of deep CNNs. As BatchNorm is used in WRN-28-10, the results also imply that our regularizer has good compatibility with BatchNorm.

4.3. Evaluation on CIFAR-100

The CIFAR-100 dataset [39] has the same size and format as CIFAR-10, except that it has 100 classes with 600 images per class. For each class, there are 500 training images and 100 testing images. Similar as the experiments on CIFAR-10, the aforementioned shallow and deeper CNNs are adopted. Meanwhile, all the training and testing setups also follow the settings on

Table 4. Recognition error rate (%) on CIFAR-10

Method	Error Rate
Maxout [25]	11.68
DropConnect [27]	9.41
Highway Network [41]	7.60
ELU-Network [42]	6.55
LSUV [43]	5.84
All-CNN [44]	4.41
Baseline [S]	23.37
DA [S]	22.90
PP [S]	22.57
Xavier [S]	22.49
CGES [S]	22.40
OrthoReg [S]	22.31
AKWD [S]	22.07
Baseline [D]	4.30
OrthoReg [D]	4.15
AKWD [D]	3.95

CIFAR-10. Again, the proposed AKWD is applied to regularize only the last convolutional layer, on both networks. The testing results are presented in Table 5. As observed, similar results are obtained as on CIFAR-10. For both CNNs, the regularized model by AKWD outperforms the baseline model and all the regularized models, namely OrthoReg, CGES, and the models using different training techniques. Among all the comparing methods, the regularized deeper CNN model with AKWD (AKWD[D]) leads to the best classification performance.

Based on the above results on MNIST and CIFAR-10/100, we summarize that the proposed regularizer performs better than other regularizations, including Xavier initialization, data augmentation, image preprocessing, CGES, and OrthoReg for the object recognition task. To illustrate the regularization effect of our method, the training and testing curves are shown in Fig. 9 for CIFAR-10, and in Fig. 10 for CIFAR-100. From the training curve, it can be observed that the convergence rate of our method is comparable with the original model and other regularizations. This confirms that our method can be optimized easily, coinciding with the theoretical analysis.

Table 5. Recognition accuracy (%) on CIFAR-100

Method	Accuracy
Maxout [25]	61.43
All-CNN [44]	66.29
Highway Network [41]	67.76
LSUV [43]	70.04
ELU-Network [42]	75.72
Baseline [S]	43.66
DA [S]	45.18
PP [S]	45.06
Xavier [S]	44.83
CGES[S]	44.02
OrthoReg [S]	46.59
AKWD [S]	47.68
Baseline [D]	80.41
OrthoReg [D]	80.71
AKWD [D]	80.96

Unfortunately, the method takes more training time than the non-regularized model, especially for the deep models. We also observe that the proposed AKWD regularizer is more stable with less fluctuations than OrthoReg at the training stage, especially on CIFAR-100. Considering the accuracy gap between training and testing in Fig. 9 and Fig. 10, it can be seen that AKWD can mitigate the overfitting problem of the baseline model and the instable training of OrthoReg.

5. Experiments for Face verification

Face verification task determines whether two face images are from the same person, by comparing the feature similarity between them and the threshold, as illustrated in Fig. 11. In this section, the evaluation of our approach on face verification task is conducted on two typical benchmarks: the LFW dataset and the YTF dataset.

5.1. Experimental settings

In this experiment, the models are trained on the widely used public CASIA-WebFace dataset [45], and evaluated on LFW [46] and YTF [47].

Table 6. The network architectures for face verification. $Block(3,3)$ consists of two stacked convolutional layers with kernel size of 3×3

Layer	Block Type	Output Size	Param
conv1	3×3, stride 1	$110 \times 94 \times 32$	0.86K
conv2	3×3, stride 1	$108 \times 92 \times 64$	18.4K
max-pool1	2×2, stride 2	$54 \times 46 \times 64$	–
resblock1	$Block(3,3)$	$54 \times 46 \times 64$	73.7K
conv3	3×3, stride 1	$52 \times 44 \times 128$	73.7K
max-pool2	2×2, stride 2	$26 \times 22 \times 128$	–
resblock2	$Block(3,3)$	$26 \times 22 \times 128$	294.9K
resblock3	$Block(3,3)$	$26 \times 22 \times 128$	294.9K
conv4	3×3, stride 1	$24 \times 20 \times 256$	294.9K
max-pool3	2×2, stride 2	$12 \times 10 \times 256$	–
resblock4	$Block(3,3)$	$12 \times 10 \times 256$	1179.6K
resblock5	$Block(3,3)$	$12 \times 10 \times 256$	1179.6K
resblock6	$Block(3,3)$	$12 \times 10 \times 256$	1179.6K
resblock7	$Block(3,3)$	$12 \times 10 \times 256$	1179.6K
resblock8	$Block(3,3)$	$12 \times 10 \times 256$	1179.6K
conv5	3×3, stride 1	$10 \times 8 \times 512$	1179.6K
max-pool4	2×2, stride 2	$5 \times 4 \times 512$	–
resblock9	$Block(3,3)$	$5 \times 4 \times 512$	4718.5K
resblock10	$Block(3,3)$	$5 \times 4 \times 512$	4718.5K
resblock11	$Block(3,3)$	$5 \times 4 \times 512$	4718.5K
fc1	–	$1 \times 1 \times 512$	5242.8K

Some face images from the three datasets are shown in Fig. 12. The CASIA-WebFace dataset is a typical training set for the optimization of neural network parameters in the field of face recognition. It contains 10,575 subjects with 494,414 face images of celebrities from the web. After removing the mislabeled and false detected face images, the remaining 437,633 images of 10,575 subjects are used for training. Before training, the faces of all the images are firstly detected as in [48], and then aligned by similarity transformation according to the five detected key landmarks. [1] After that, the face

[1]The detection and alignment tools are available at: https://github.com/seetaface/SeetaFaceEngine.

Table 7. Recognition accuracy (%) on LFW and YTF

Method	Train. Images	Networks	LFW	YTF
DeepFace [18]	4.4M	7	97.35	91.40
WebFace [45]	0.4M	1	97.73	92.24
Web-Scale [49]	4.5M	4	98.37	-
DeepID2 [50]	-	25	98.97	-
DeepID2+ [51]	0.3M	25	99.47	93.20
FaceNet [52]	200M	1	99.63	95.10
Baseline	0.4M	1	98.88	93.82
OrthoReg	0.4M	1	98.88	93.72
AKWD	0.4M	1	99.08	94.04

images are cropped to 112×96 RGB images, and normalized by subtracting 127.5 and then dividing by 128. Again, we flip the input images horizontally to augment data.

A reduced version of ResNet [19] is used as the baseline for face verification, which contains 27 convolutional layers. The detailed network architectures are given in Table 6. The activation function PReLU [4] is applied after every convolutional layer. Specifically, the network is trained by joint supervision of the softmax loss and the center loss [7], and the weight of center loss is set to 0.008. The training is started with a initial learning rate of 0.1 and a batch-size of 128. The learning rate is decreased by 10% at 16k and 24k iterations, and the maximum iteration number is set to 28k. Analogous to the experimental settings in object recognition task, the regularized model by AKWD consists of regularizing only the last convolutional layer of the network, with the regularization hyperparameter $\lambda = 1$. At the testing stage, the features of both the original image and its horizontally flipped version are firstly extracted using the learned model, and then concatenated together for testing. Specifically, the feature similarity of face pairs is measured by the cosine distance matrix after transforming the representation by PCA [7]. In the following, the effectiveness of the proposed method will be verified on LFW and YTF from several aspects.

5.2. Evaluation on LFW

LFW (the Labeled Faces in the Wild) is an acknowledged challenging dataset for studying the problem of unconstrained face recognition. It con-

tains 13,233 face images of 5,749 people collected from websites. Every face is taken under an unconstrained environment and is labeled with the name of the person pictured. In this dataset, 1,680 people have two or more photos and 4,096 people have only one photo. Following the standard protocol of unrestricted with labeled outside data, we test on the provided 6,000 face pairs consisting of 3,000 genuine matches and 3,000 impostor matches (see Fig. 12(b)). To be specific, the testing pairs are split into 10-fold, where nine splits are randomly chosen to train a classifier, and the tenth is used to perform the classification decision. Repeating the experiment ten times, the mean classification accuracy is estimated as the final decision. The effectiveness of the proposed AKWD regularizer on face verification is compared with the baseline model, the regularized model by OrthoReg, and six state-of-the-art methods, *i.e.*, DeepFace [18], WebFace [45] , Web-Scale[49], DeepID2 [50], DeepID2+ [51] and FaceNet [52].

The classification accuracies of all the comparing approaches are reported in Table 7. It is observed that our approach achieves a verification accuracy of 99.08% on the LFW dataset, surpassing the result of the baseline model by 0.2%. Of particular note is that, as the baseline model has very low overfitting issue with relatively good performance, even a small margin of improvement in accuracy demonstrates the effectiveness of our method. This is further confirmed by the observation that OrthoReg seems to have no effect on improving the baseline model.

Furthermore, we interpret the proposed regularizer from the perspective of the neural responses. Fig. 13 compares the neural responses of the regularized model by AKWD and the baseline model on both the positive testing pair and negative testing pair. For illustration, 32 neurons are subsampled from the total 512 neurons from the output of the top hidden layer (fc1). Regarding the case for positive pair (top), we observe that AKWD is able to capture more similar activation patterns for the same person than the baseline model. As for the negative pair (bottom), AKWD yields more different activated neurons for a pair of different persons, when compared to the baseline model. The above observations demonstrate that AKWD can help to improve the generalization of CNN models.

5.3. Evaluation on YTF

We also evaluate our method on the YTF (YouTube Faces) dataset. The YTF dataset contains 3,425 videos of 1,595 different subjects downloaded from YouTube. Each subject has several videos with various sizes of frames

ranging from 48 to 6,070. Besides, the frames of YTF have various face variations and poor resolutions. Consequently, it is regarded as a more challenging testing benchmark for face verification. As for LFW, we also follow the standard protocol of unrestricted with labeled outside data, and take a 10-fold cross validation on the provided 5,000 testing video pairs ten times. The only difference is that we use the average accuracy of the randomly selected 100 pairs of frames per video to estimate the similarity score of a test video pair.

As in LFW, the same comparing methods are involved for comparison, and the testing results are given in Table 7. It is observed that OrthoReg is not applicable for face verification on the YTF dataset, as it leads to a slight decrease in accuracy compared to the un-regularized baseline model. On the contrary, the proposed AKWD regularizer obtains the verification accuracy of 94.04%, improving the result of the baseline model by 0.22% and comparable with the most recent state-of-the-art.

Fig. 14 visualizes the first convolutional parameters for the cases without and with the AKWD regularization, where the regularization effect of AKWD on the parameters can be observed. Clearly, the original model has many very similar even "dead" convolution filters, whereas the parameters of the regularized model by AKWD contains more highlight parts, that capture more different patterns and encode more information. This coincides with the basic idea of decorrelation and reflects the regularization effect of AKWD.

6. Conclusion

In this paper, we proposed a kernel-based weight decorrelation framework to regularize the CNN models for better model generalization. Different from the existing works, the correlation between paring weights is measured by the cosine distance defined in RKHS. In particular, we investigated the case with Gaussian kernel in detail. This yields a local weight decorrelation strategy that can effectively avoid the insensitivity of the cosine similarity to magnitude of data. The bandwidth parameter of the kernel is set by an adaptive estimation. On both object classification and large-scale face verification tasks, the proposed regularization method showed its effectiveness on several public benchmarks by comparing with the state-of-the-art methods. Extensive experiments also showed the ability of the proposed AKWD in improving the generalization of CNNs of different capacities, as well as the

good compatibility with other regularization techniques, such as Dropout and BatchNorm. In the light of the performance of the combination of kernel method and deep learning, future works will concentrate on enhancing the discriminate ability of the deeply learned features in this direction.

Acknowledgment

This work was supported by the National Natural Science Foundation of China under Grant 61701337, the Natural Science Foundation of Tianjin City under Grand 18JCQNJC01600, the Major Science and Technology Project of Tianjin under Grand 18ZXRHSY00160 and the AI Key Project of Tianjin under Grand 19ZXZNGX0050.

References

[1] Y. Lecun, L. Bottou, Y. Bengio, P. Haffner, Gradient-based learning applied to document recognition, Proc. IEEE 86 (11) (1998) 2278–2324.

[2] M. D. Zeiler, R. Fergus, Visualizing and understanding convolutional networks, Eur. Conf. Comput. Vis. (ECCV) (8689) (2014) 818–833.

[3] A. Krizhevsky, I. Sutskever, G. E. Hinton, Imagenet classification with deep convolutional neural networks, in: Proc. Conf. Adv. Neural Inf. Process. Syst. (NIPS), Lake Tahoe, Nevada, United States, 2012, pp. 1097–1105.

[4] K. He, X. Zhang, S. Ren, J. Sun, Delving deep into rectifiers: Surpassing human-level performance on imagenet classification, in: IEEE Int. Conf. Comput. Vis. (ICCV), 2015, pp. 1026–1034. doi:10.1109/ICCV.2015.123.

[5] B. Zhou, A. Lapedriza, J. Xiao, A. Torralba, A. Oliva, Learning deep features for scene recognition using places database, in: Proc. Conf. Adv. Neural Inf. Process. Syst. (NIPS), Montreal, Quebec, Canada, 2014, pp. 487–495.

[6] L. Herranz, S. Jiang, X. Li, Scene recognition with cnns: Objects, scales and dataset bias, in: Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR), 2016, pp. 571–579.

[7] Y. Wen, K. Zhang, Z. Li, Y. Qiao, A discriminative feature learning approach for deep face recognition, Eur. Conf. Comput. Vis. (ECCV) (2016) 499–515.

[8] G. Hu, X. Peng, Y. Yang, T. M. Hospedales, J. Verbeek, Frankenstein: Learning deep face representations using small data, IEEE Trans. Image Process. 27 (1) (2018) 293–303. doi:10.1109/TIP.2017.2756450.

[9] H. Yao, S. Zhang, R. Hong, Y. Zhang, C. Xu, Q. Tian, Deep representation learning with part loss for person re-identification, IEEE Trans. Image Process. (2019) 1–1doi:10.1109/TIP.2019.2891888.

[10] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. S. Bernstein, et al., Imagenet large scale visual recognition challenge, Int. J. Comput. Vis. 115 (3) (2015) 211–252.

[11] Y. Lecun, Y. Bengio, G. E. Hinton, Deep learning, Nature 521 (7553) (2015) 436–444.

[12] G. Hu, Y. Yang, D. Yi, J. Kittler, W. J. Christmas, S. Z. Li, T. M. Hospedales, When face recognition meets with deep learning: An evaluation of convolutional neural networks for face recognition, in: IEEE Int. Conf. Comput. Vis. Workshop (ICCVW), 2015, pp. 384–392. doi:10.1109/ICCVW.2015.58.

[13] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. E. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, A. Rabinovich, Going deeper with convolutions, in: Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR), 2015, pp. 1–9.

[14] H. Lee, H. Kwon, Going deeper with contextual cnn for hyperspectral image classification, IEEE Trans. Image Process. 26 (10) (2017) 4843–4855. doi:10.1109/TIP.2017.2725580.

[15] M. Cogswell, F. Ahmed, R. B. Girshick, L. Zitnick, D. Batra, Reducing overfitting in deep networks by decorrelating representations, Int. Conf. Learn. Represent. (ICLR) (2016).

[16] A. G. Howard, Some improvements on deep convolutional neural network based image classification, Int. Conf. Learn. Represent. (ICLR) (2014).

[17] T. Hastie, R. Tibshirani, J. H. Friedman, The elements of statistical learning: data mining, inference, and prediction, Math. Intell. 27 (2) (2005) 83–85.

[18] Y. Taigman, M. Yang, M. Ranzato, L. Wolf, Deepface: Closing the gap to human-level performance in face verification, in: Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR), 2014, pp. 1701–1708. doi:10.1109/CVPR.2014.220.

[19] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, in: Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR), 2016, pp. 770–778.

[20] X. Glorot, Y. Bengio, Understanding the difficulty of training deep feedforward neural networks, in: Proc. Int. Conf. Artif. Intell. Stats. (AISTATS), Sardinia, Italy, 2010, pp. 249–256.

[21] A. Krogh, J. A. Hertz, A simple weight decay can improve generalization, in: Proc. Conf. Adv. Neural Inf. Process. Syst. (NIPS), Denver, Colorado, USA, 1991, pp. 950–957.

[22] J. Yoon, S. J. Hwang, Combined group and exclusive sparsity for deep neural networks, Int. Conf. Mach. Learn. (ICML) (2017) 3958–3966.

[23] R. Ma, J. Miao, L. Niu, P. Zhang, Transformed l1 regularization for learning sparse deep neural networks, Neural Networks 119 (2019) 286 – 298. doi:https://doi.org/10.1016/j.neunet.2019.08.015.

[24] N. Srivastava, G. E. Hinton, A. Krizhevsky, I. Sutskever, R. Salakhutdinov, Dropout: a simple way to prevent neural networks from overfitting, J. Mach. Learn. Research 15 (1) (2014) 1929–1958.

[25] I. J. Goodfellow, D. Warde-Farley, M. Mirza, A. Courville, Y. Bengio, Maxout networks, Int. Conf. Mach. Learn. (ICML) (2013) 1319–1327.

[26] G. Kang, J. Li, D. Tao, Shakeout: A new approach to regularized deep neural network training, IEEE Trans. Pattern Anal. Mach. Intell. 40 (5) (2018) 1245–1258.

[27] L. Wan, M. D. Zeiler, S. Zhang, Y. L. Cun, R. Fergus, Regularization of neural networks using dropconnect, Int. Conf. Mach. Learn. (ICML) (2013) 1058–1066.

[28] P. Rodriguez, J. Gonzalez, G. Cucurull, J. M. Gonfaus, F. X. Roca, Regularizing cnns with locally constrained decorrelations, Int. Conf. Learn. Represent. (ICLR) (2017).

[29] Y. Bao, H. Jiang, L. Dai, C. Liu, Incoherent training of deep neural networks to de-correlate bottleneck features for speech recognition, in: IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP), 2013, pp. 6980–6984.

[30] M. I. Chelaru, V. Dragoi, Negative correlations in visual cortical networks, Cereb. Cortex 26 (1) (2016) 246–256.

[31] V. Vapnik, The Nature of Statistical Learning Theory, Springer-Verlag, New York, NY, USA, 1995.

[32] B. Schölkopf, A. Smola, K.-R. Müller, Nonlinear component analysis as a kernel eigenvalue problem, Neural computation 10 (5) (1998) 1299–1319.

[33] R. Jenssen, Kernel entropy component analysis, IEEE Transactions on Pattern Analysis and Machine Intelligence 32 (5) (2010) 847–860. doi:10.1109/TPAMI.2009.100.

[34] J. Shawe-Taylor, N. Cristianini, Kernel Methods for Pattern Analysis, Cambridge University Press, Cambridge, UK, 2004.

[35] S. Badrul, K. George, K. Joseph, Item-based collaborative filtering recommendation algorithms, Int. World Wide Web Conf. (WWW) (2001).

[36] N. Heidenreich, A. Schindler, S. Sperlich, Bandwidth selection for kernel density estimation: a review of fully automatic selectors, ASTA-Adv. Stat. Anal. 97 (4) (2013) 403–433.

[37] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. B. Girshick, S. Guadarrama, T. Darrell, Caffe: Convolutional architecture for fast feature embedding, ACM Multimedia (2014) 675–678.

[38] V. Nair, G. E. Hinton, Rectified linear units improve restricted boltzmann machines, Int. Conf. Mach. Learn. (ICML) (2010) 807–814.

[39] A. Krizhevsky, Learning multiple layers of features from tiny images, Technical Report, University of Toronto (2009).

[40] S. Zagoruyko, N. Komodakis, Wide residual networks, in: Proc. Br. Mach. Vis. Conf. (BMVC), York, UK, 2016.

[41] R. K. Srivastava, K. Greff, J. Schmidhuber, Training very deep networks, in: Proc. Conf. Adv. Neural Inf. Process. Syst. (NIPS), Montreal, Quebec, Canada, 2015, pp. 2377–2385.

[42] D. Clevert, T. Unterthiner, S. Hochreiter, Fast and accurate deep network learning by exponential linear units (elus), Int. Conf. Learn. Represent. (ICLR) (2016).

[43] D. Mishkin, J. Matas, All you need is a good init, Int. Conf. Learn. Represent. (ICLR) (2016).

[44] J. T. Springenberg, A. Dosovitskiy, T. Brox, M. A. Riedmiller, Striving for simplicity: The all convolutional net, Int. Conf. Learn. Represent. (ICLR) (2014).

[45] D. Yi, Z. Lei, S. Liao, S. Z. Li, Learning face representation from scratch., arXiv preprint arXiv: 1411.7923 (2014).

[46] G. B. H. E. Learned-Miller, Labeled faces in the wild: Updates and new reporting procedures, Tech. Rep. UM-CS-2014-003, University of Massachusetts, Amherst (May 2014).

[47] L. Wolf, T. Hassner, I. Maoz, Face recognition in unconstrained videos with matched background similarity, in: Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR), 2011, pp. 529–534.

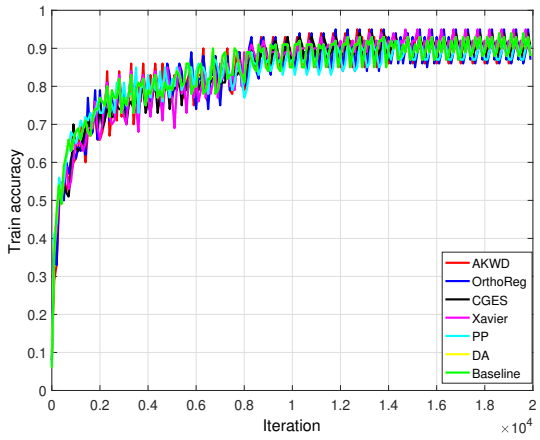
[48] J. Zhang, S. Shan, M. Kan, X. Chen, Coarse-to-fine auto-encoder networks (cfan) for real-time face alignment, Eur. Conf. Comput. Vis. (ECCV) (2014) 1–16.

[49] Y. Taigman, M. Yang, M. Ranzato, L. Wolf, Web-scale training for face identification, in: Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR), 2015, pp. 2746–2754.

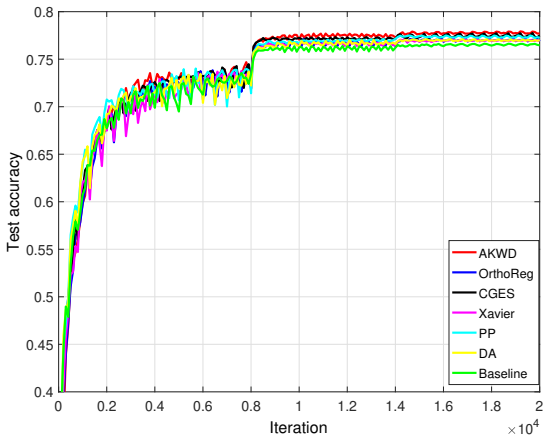
[50] Y. Sun, X. Wang, X. Tang, Deep learning face representation by joint identification-verification, in: Proc. Conf. Adv. Neural Inf. Process. Syst. (NIPS), Vol. 27, Montreal, Quebec, Canada, 2014, pp. 1988–1996.

[51] Y. Sun, X. Wang, X. Tang, Deeply learned face representations are sparse, selective, and robust, in: Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR), 2015, pp. 2892–2900.

[52] F. Schroff, D. Kalenichenko, J. Philbin, Facenet: A unified embedding for face recognition and clustering, in: Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR), 2015, pp. 815–823.

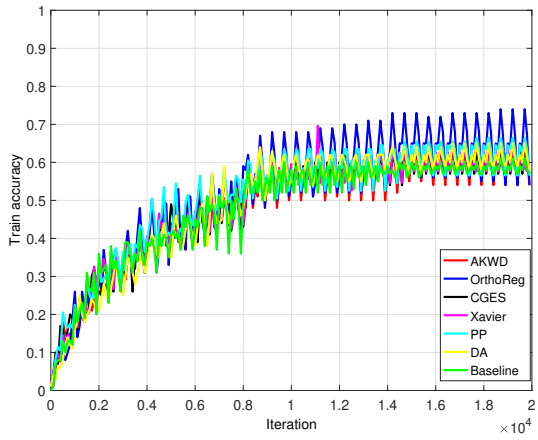


(a)

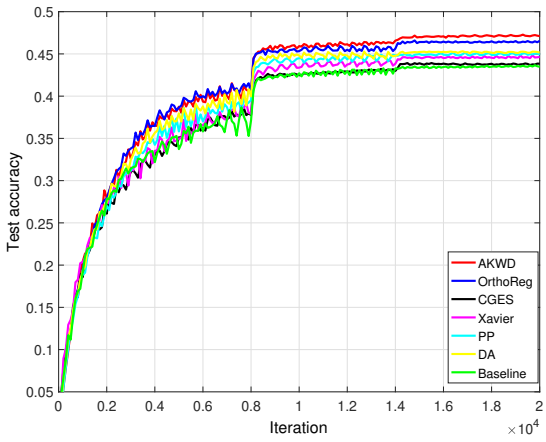


(b)

Fig. 9. Accuracy vs. iteration curves with different methods for the shallow networks on CIFAR-10.



(a)



(b)

Fig. 10. Accuracy vs. iteration curves with different methods for the shallow networks on CIFAR-100.

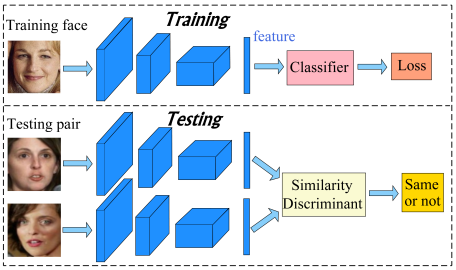


Fig. 11. The flow of face verification by CNN.

(a) CASIA-WebFace

(b) LFW

(c) YTF

Fig. 12. Face samples of the three face benchmarks.

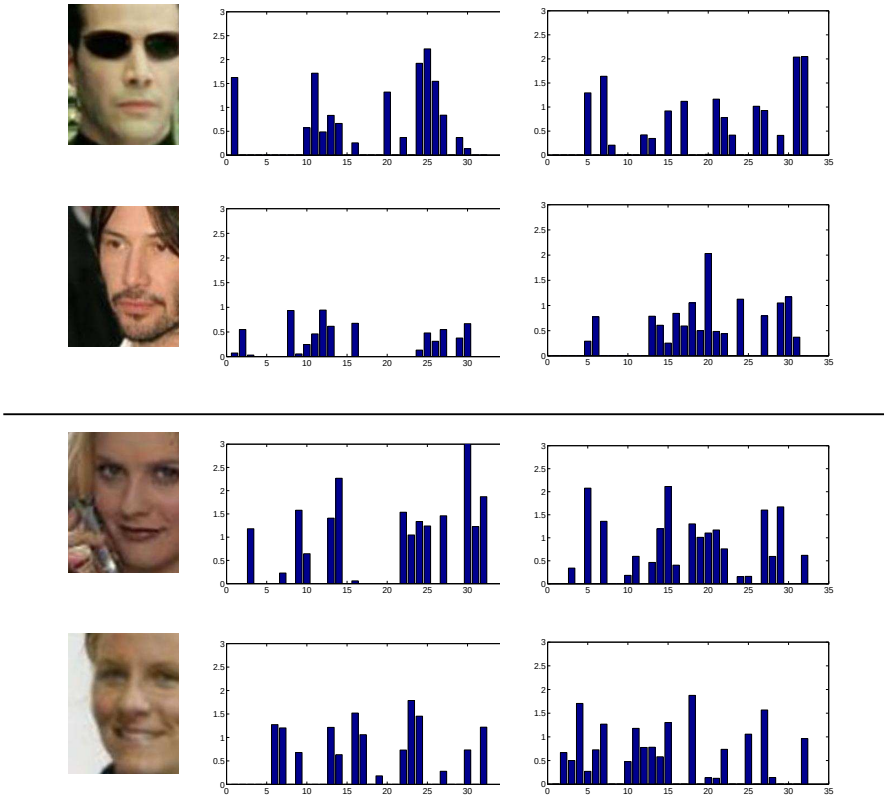
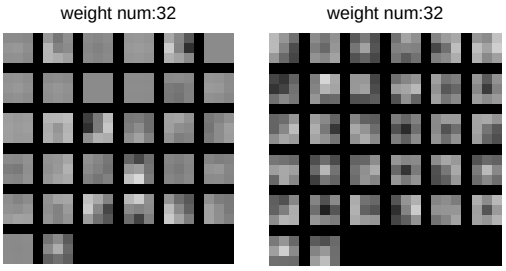


Fig. 13. Left: the testing image pairs on LFW, where the top two are for the positive pair and the bottom two are for the negative pair. **Middle:** neural responses of AKWD regularized model. **Right:** neural responses of the unregularized model. All the 32 neurons are subsampled from the 512 neurons in the top hidden layer for illustration.



(a) Without AKWD (b) With AKWD

Fig. 14. Visualization of the first convolutional parameters for the cases without and with the AKWD regularization.