Deep Global Generalized Gaussian Networks

Qilong Wang[1], Peihua Li[2], Qinghua Hu[1,*], Pengfei Zhu[1], Wangmeng Zuo[3]
[1]Tianjin University, [2]Dalian University of Technology, [3] Harbin Institute of Technology
{qlwang,huqinghua,zhupengfei}@tju.edu.cn, peihuali@dlut.edu.cn, wmzuo@hit.edu.cn

Abstract

Recently, global covariance pooling (GCP) has shown great advance in improving classification performance of deep convolutional neural networks (CNNs). However, existing deep GCP networks compute covariance pooling of convolutional activations with assumption that activations are sampled from Gaussian distributions, which may not hold in practice and fails to fully characterize the statistics of activations. To handle this issue, this paper proposes a novel deep global generalized Gaussian network (3G-Net), whose core is to estimate a global covariance of generalized Gaussian for modeling the last convolutional activations. Compared with GCP in Gaussian setting, our 3G-Net assumes the distribution of activations follows a generalized Gaussian, which can capture more precise characteristics of activations. However, there exists no analytic solution for parameter estimation of generalized Gaussian, making our 3G-Net challenging. To this end, we first present a novel regularized maximum likelihood estimator for robust estimating covariance of generalized Gaussian, which can be optimized by a modified iterative re-weighted method. Then, to efficiently estimate the covariance of generaized Gaussian under deep CNN architectures, we approximate this re-weighted method by developing an unrolling re-weighted module and a square root covariance layer. In this way, 3G-Net can be flexibly trained in an end-to-end manner. The experiments are conducted on large-scale ImageNet-1K and Places365 datasets, and the results demonstrate our 3G-Net outperforms its counterparts while achieving very competitive performance to state-of-the-arts.

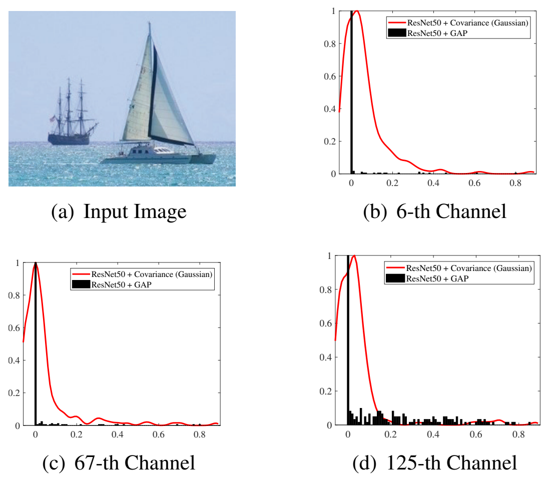

Figure 1. Illustration of histograms or distributions of the last convolutional activations given an input image (a). We show, in some channels, histograms and fitting distributions of the last convolution activations in CNN models trained with global average pooling (GAP) [17] (black bar) and covariance pooling [28] (red line), respectively. Obviously, both of them have long tails and do not obey strictly Gaussian distributions.

1. Introduction

Deep convolutional neural networks (CNNs) have attracted a great amount of attention in computer vision community, and demonstrated enormous advantages in many tasks, especially in large-scale visual classification [26, 34, 35, 17, 21, 20]. However, most deep CNN architectures summarize the last convolutional activations only using simple global first-order pooling methods, severely limiting the representation ability of deep CNNs. To handle this issue, researchers integrate some global covariance pooling (GCP) methods with deep CNNs in an end-to-end manner, which show great effectiveness to improve the classification performance of deep CNNs [23, 32, 37, 28, 31, 27].

Among these GCP networks, Ionescu *et al.* [23] first integrate a second-order pooling (O_2P) [4] layer into deep CNNs, namely DeepO_2P, and develop a matrix backpropagation theory for end-to-end training. A parallel work is bilinear CNN (B-CNN) [32], which concerns the outer product of the last convolutional activations extracted from two CNN models, followed by element-wise power normalization and ℓ_2-normalization. When two CNN models are identical, B-CNN reduces to a global second-order pooling of convolutional activations. Wang *et al.* [37] and Li *et al.* [28] insert respectively a global Gaussian distribution

*Qinghua Hu is the corresponding author. We thank NVIDIA corporation for donating GPU.

and matrix power normalized covariance pooling after the last convolution layer of deep CNNs, which obtain better performance by considering geometry of Gaussian distribution and robust estimation of covariance. Subsequently, improved B-CNN [31] shows empirically matrix square root normalization can improve B-CNN, and uses the modified Denman-Beavers iteration [19] to speed up inference of the networks. Li *et al*. [27] develop forward and backward propagation methods based on Newton-Schulz iteration [19] to accelerate both training and inference of matrix square root normalized covariance pooling networks [28].

Although previous deep GCP networks [23, 32, 37, 28, 31, 27] have proven to improve the representation ability of deep CNNs, these methods compute covariance pooling with invariably assuming that convolutional activations follow a Gaussian distribution. Such an assumption does not always hold true in real scenarios. To verify it, we randomly choose some images from ImageNet-1K [9], and compute histograms of the last convolutional activations extracted from pre-trained CNN model with global average pooling (GAP) [17], as well as fitting distributions of ones extracted from pre-trained GCP network [27]. Figure 1 plots the histograms and distributions of activations for one example (remaining others share similarity) in some channels. We observe that either histograms of activations extracted from GAP-based CNN or distributions of the ones extracted from GCP network are long tailed, not strictly obeying Gaussian distributions. Since Gaussian models fail to characterize long-tailed distributions, covariance of Gaussian has limited capability to capture characteristics of activations.

Compared with Gaussian models, multivariate generalized Gaussian models can better characterize complex distributions, especially long-tailed ones [2]. In terms of the considerations above, this paper proposes a deep global generalized Gaussian networks (3G-Net). The core of 3G-Net is to summarize the statistics of the last convolutional activations by computing a global covariance of generalized Gaussian, which can capture more precise characteristics of convolutional activations. However, different from covariance estimation in Gaussian setting, there exists no analytic solution for covariance estimation of generalized Gaussian, making our 3G-Net challenging. To circumvent this difficulty, inspired by [38], we first present a regularized maximum likelihood estimator, which allows us to robustly estimate covariance of generalized Gaussian distribution with a modified iterative re-weighted method. According to this estimator, we propose an unrolling re-weighted module and a square root covariance layer for computing the covariance of generalized Gaussian. The unrolling re-weighted module is designed to iteratively estimate weight of each activation in an approximative yet efficient manner. The square root covariance layer is used to compute matrix square root of the estimated covariance, which is resulted by our modified iterative re-weighted method. In this way, our 3G-Net can be flexibly trained in an end-to-end learning manner.

The overview of our proposed 3G-Net is illustrated in Figure 2. The experiments are conducted on two large-scale image benchmarks, i.e., ImageNet-1K [9] and Places365 [44]. The contributions of this paper can be concluded as follows. (1) We propose a novel deep global generalized Gaussian network (3G-Net) by estimating a global covariance of generalized Gaussian to summarize the statistics of the last convolutional activations, aiming at capturing precise characteristics of activations and further improving representation ability of deep CNNs. (2) To our best knowledge, we make the first attempt to robustly estimate covariance of generalized Gaussian distribution under deep CNN architectures. From a point of implementation view, we introduce an unrolling re-weighted module and a square root covariance layer based on the proposed robust estimator. (3) The experimental results on large-scale ImageNet-1K and Places365 demonstrate the proposed 3G-Net outperforms its counterparts under ResNet architectures [17], and achieves state-of-the-art performance.

2. Related Work

Recently, integration of preferable pooling or encoding methods into deep CNNs has shown effectiveness in improving classification performance. In contrary to aforementioned deep GCP networks [23, 32, 37, 28, 31, 27], some researchers study to approximate covariance pooling for obtaining lower-dimensional representations. Among them, Gao *et al*. [13] and Kong *et al*. [25] propose compact B-CNN and low-rank B-CNN methods, respectively. Dai *et al*. [8] fuse additional first-order (mean) information into the compact B-CNN [13] by simply concatenating them. Gou *et al*. [15] approximate [37] by introducing homogeneous mapping and sub-matrix square-root layers followed by compact B-CNN. Kernel pooling [7] extends the idea of [13] to higher-order (number of order > 2) pooling for fine-grained visual recognition. Cai *et al*. [3] suggest a compact higher-order pooling based on polynomial kernel approximation and rank-1 tensor decomposition [24]. Meanwhile, some works [40, 30] incorporate local approximated second-order statistics into convolution or fully-connected layers of deep CNNs to increase the nonlinearity of networks. Additionally, the classical Bag-of-Words models are also embedded into deep CNNs [1, 29]. Different from above methods, our 3G-Net incorporates a global covariance of generalized Gaussian into deep CNNs, obtaining better representations and higher classification accuracy.

Our covariance of generalized Gaussian layer involves an unrolling re-weighted module for computing iteratively weights of activations. It shares the similarity with the recently proposed self-attention mechanisms in deep CNNs [20, 39, 11, 41, 12], among which Hu *et al*. [20] introduce

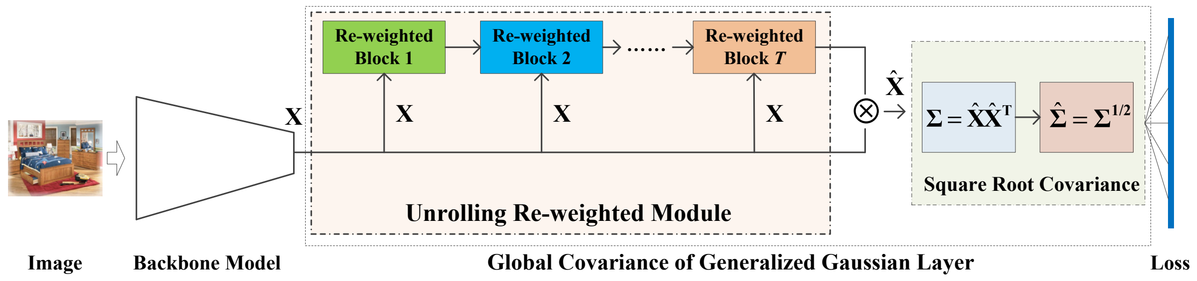

Figure 2. Overview of the proposed deep global generalized Gaussian networks (3G-Net), where a global covariance of generalized Gaussian is inserted after the last convolution block to summarize the statistics of activations. Based on the proposed robust covariance estimator, our global covariance of generalized Gaussian layer consists of an unrolling re-weighted module and a square root covariance layer.

a Squeeze-and-Excitation module into convolution blocks, performing channel-wise weighting on outputs of each convolution block. Going a step further, CBAM [41] extends [20] by introducing an additional spatial attention module. Wang *et al*. [39] and Du *et al*. [11] propose non-local blocks and interaction-aware attention module to obtain better attention maps, respectively. Fu *et al*. [12] propose an attention proposal sub-network to iteratively generates multi-scale region attention for obtaining representations. Many recent works are concerned with integration of attention modules into deep CNNs, and a comprehensive review on these methods is beyond the scope of this paper. Different from these methods, our unrolling re-weighted module is proposed, based on a modified iterative re-weighted method, for estimating covariance of generalized Gaussian.

3. Proposed Method

In this section, we will introduce our proposed 3G-Net. Firstly, we briefly recall definition of multivariate generalized Gaussian distribution and its parameter estimation. Then, we construct the trainable covariance of generalized Gaussian layer. Finally, we describe implementation of 3G-Net based on the covariance of generalized Gaussian layer.

3.1. Multivariate Generalized Gaussian Distribution

To summarize the statistics of the last convolutional activations $\mathbf{X} \in \mathbb{R}^{d\times N} = \{\mathbf{x}_1, \dots, \mathbf{x}_N\}$ with N samples of d-dimension, we propose to employ a multivariate generalized Gaussian distribution (MGGD) with zero mean [33] (i.e., covariance of MGGD), which takes the following form:

$$p(\mathbf{x}_i; \mathbf{\Sigma}; \beta; \delta) = \frac{\Gamma(d/2)}{\pi^{d/2}\Gamma(d/2\beta)2^{d/2\beta}} \frac{\beta}{\delta^{d/2}|\mathbf{\Sigma}|^{1/2}} \exp\left(-\frac{1}{2\delta^{\beta}}(\mathbf{x}_i\mathbf{\Sigma}^{-1}\mathbf{x}_i^T)^{\beta}\right), \tag{1}$$

where β and δ are shape and scale parameters of MGGD, respectively; $\mathbf{\Sigma}$ is covariance matrix of MGGD, and Γ is a Gamma function.

Note that Eqn. (1) will reduce respectively to the Gaussian and Laplacian distributions when $\beta = 1$ and $\beta = 0.5$. Clearly, MGGD is able to characterize more complex distributions, in comparison to Gaussian models. Moreover, MGGD has the ability to model long-tailed distributions, more suitable for convolutional activations (as shown in Figure 1). However, there is no closed-form solution for parameter estimation of MGGD. As shown in [43], maximum likelihood estimation (MLE) for $\mathbf{\Sigma}$ of MGGD is defined by

$$\arg\min_{\mathbf{\Sigma}} \sum_{n=1}^{N} (\mathbf{x}_n^T\mathbf{\Sigma}^{-1}\mathbf{x}_n)^{\beta} + N\log|\mathbf{\Sigma}|. \tag{2}$$

Based on above MLE (2), covariance $\mathbf{\Sigma}$ of MGGD can be estimated by a fixed point algorithm (or iterative reweighed method) [33, 43]. Correspondingly, estimation of $\mathbf{\Sigma}$ in t-iteration is described as:

$$\mathbf{\Sigma}_t = \frac{1}{N}\sum_{n=1}^{N} \frac{Nd}{y_n^t + (y_n^t)^{1-\beta}\sum_{j\neq n}(y_j^t)^{\beta}} \cdot \mathbf{x}_n\mathbf{x}_n^T, \tag{3}$$

where $y_n^t = \mathbf{x}_n^T\mathbf{\Sigma}_{t-1}^{-1}\mathbf{x}_n$, $\delta = \left(\frac{\beta}{dN}\sum_{j\neq n}(y_j^t)^{\beta}\right)^{\frac{1}{\beta}}$ and β can be estimated by Newton-Raphson procedure [33], i.e.,

$$\beta_t = \beta_{t-1} - f(\beta_{t-1})/f^{'}(\beta_{t-1}). \tag{4}$$

Here $f^{'}(\beta_{t-1})$ is the partial derivative of $f(\beta_{t-1})$, and $f(\beta_{t-1})$ is a function associated with y^t, β_{t-1} and a digamma function (refer to [33, Eqn. (13)] for details). As proven in [33], parameter estimation of MGGD based on Eqns. (3) and (4) can converge to a stationary point.

3.2. Trainable Covariance of Generalized Gaussian Layer

To construct our trainable covariance of generalized Gaussian layer, we first introduce a regularized MLE for robustly estimating covariance of generalized Gaussian. Then, we achieve this estimator by developing an unrolling re-weighted module and a square root covariance layer.

3.2.1 Robust Covariance Estimation of Generalized Gaussian

Recent works [38, 28] show classical MLE for covariance of Gaussian is not robust under deep architectures, while robust estimator helps to improve performance. So we extend similar idea to estimate covariance of generalized Gaussian. As suggested in [38], we introduce the von Neumann divergence [10] between $\mathbf{\Sigma}$ and identity matrix $\mathbf{I}$ as a regularizer into the MLE (2). After some manipulations, we have

$$\arg\min_{\mathbf{\Sigma}} \frac{1}{N}\sum_{n=1}^{N}(y_n)^{\beta} + \log|\mathbf{\Sigma}| + \lambda \mathrm{tr}(\mathbf{\Sigma} - \log(\mathbf{\Sigma})), \quad (5)$$

where $y_n = \mathbf{x}_n^T\mathbf{\Sigma}^{-1}\mathbf{x}_n$ and λ is a regularizing constant. Analogous to [38], Eqn. (5) allows a robust estimator for covariance of generalized Gaussian under deep architectures.

However, the objective function in Eqn. (5) has no analytic solution. For its optimization, we develop a modified iterative re-weighted method so that we can obtain the analytic expression of unique optimal solution for each iteration. Let $\mathbf{\Sigma}_t$ be $\frac{1}{N}\sum_{n=1}^{N} w(\mathbf{x}_n, \mathbf{\Sigma}_{t-1}) \cdot \mathbf{x}_n\mathbf{x}_n^T$, where $w(\mathbf{x}_n, \mathbf{\Sigma}_{t-1})$ indicate weights of $\mathbf{x}_n$ in Eqn. (3) and $\mathbf{\Sigma}_{t-1}$ is estimated covariance in (t-1)-th iteration. The solution of Eqn. (5) can be concluded in the following theorem.

Theorem 1 *Let $\mathbf{\Sigma}_t = \mathbf{U}\mathrm{Diag}(\sigma_d)\mathbf{U}^T$ be the singular value decomposition (SVD) of $\mathbf{\Sigma}_t$, where $\mathrm{Diag}(\sigma_d)$ and $\mathbf{U}$ are diagonal and orthogonal matrices consisting of singular values σ_d and eigenvectors, respectively. Then, objective function (5) can be optimized iteratively as*

$$\mathbf{\Sigma}_t = \frac{1}{N}\sum_{n=1}^{N} w(\mathbf{x}_n, \mathbf{\Sigma}_{t-1}) \cdot \mathbf{x}_n\mathbf{x}_n^T, \quad (6)$$

$$\widehat{\mathbf{\Sigma}}_t = \mathbf{U}\mathrm{Diag}\left(\sqrt{\left(\frac{1-\lambda}{\lambda}\right)^2 + \frac{\sigma_d}{\lambda}} - \frac{1-\lambda}{\lambda}\right)\mathbf{U}^T,$$

which is the unique optimal solution in t-th iteration.

Note that we set λ to 1 throughout all the experiments, as $\lambda = 0.5 \sim 1$ achieve the similar performance, and $\lambda = 1$ is easier to be implemented. According to Theorem 1, our regularized iterative re-weighted method robustly estimates covariance of generalized Gaussian with iteratively re-weighting activations and computing matrix square root of covariance. Due to page limit, complete proof of Theorem 1 is given in the supplementary material.

3.2.2 Unrolling Re-weighted Module

According to Eqn. (6), our robust estimator needs to compute the weights of activations in each iteration. Specifically, the weight associated with each activation $\mathbf{x}_n$ in t-th iteration can be computed as

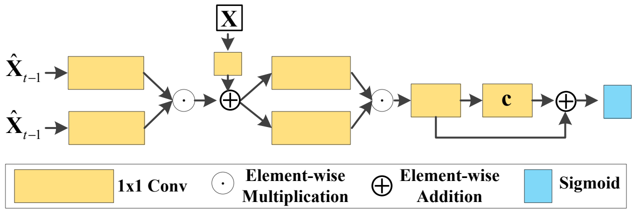

Figure 3. Diagram of a re-weighted block in our unrolling re-weighted module.

$$w(\mathbf{x}_n, \mathbf{\Sigma}_{t-1}) = \frac{Nd}{y_n^t + (y_n^t)^{1-\beta}\sum_{j\neq n}(y_j^t)^{\beta}}, \quad (7)$$

where $y_n^t = \mathbf{x}_n^T\mathbf{\Sigma}_{t-1}^{-1}\mathbf{x}_n$, $\mathbf{\Sigma}_{t-1} = \widehat{\mathbf{X}}_{t-1}\widehat{\mathbf{X}}_{t-1}^T$, and $\widehat{\mathbf{X}}_{t-1}$ indicates the weighted activations in (t-1)-th iteration. Since $\sum_{j\neq n}(y_j^t)^{\beta}$ is independent of $\mathbf{x}_n$, we can rewrite Eqn. (7) as

$$w(\mathbf{x}_n, \mathbf{\Sigma}_{t-1}) = \frac{Nd}{y_n^t + c_n(y_n^t)^{1-\beta}}. \quad (8)$$

Here c_n is a postive constant [33, Remark II.3]. Given a set of activations $\mathbf{X} = \{\mathbf{x}_1, \cdots, \mathbf{x}_N\}$, we can compute weights of $\mathbf{X}$ as:

$$\mathbf{w}_t = Nd/\left(\left[\Lambda(\mathbf{Y}_{t-1})\right] + \mathbf{c}\odot\left[\Lambda(\mathbf{Y}_{t-1})\right]^{1-\beta}\right), \quad (9)$$

where $\mathbf{Y}_{t-1} = \mathbf{X}^T\mathbf{\Sigma}_{t-1}^{-1}\mathbf{X}$, $\mathbf{c} = [c_1, \ldots, c_N]$, and $\Lambda(\mathbf{Y}_{t-1})$ extracts the diagonal elements of matrix $\mathbf{Y}_{t-1}$. $\odot$ and $/$ indicate element-wise multiplication and division, respectively.

However, Eqn. (9) involves matrix inversion, which is not suitable for GPU parallel implementation, slowing down the training of network. To handle this issue, we employ Newton-Schulz Iteration [19] and the idea of tensor approximation [24] to develop a re-weighted block, as illustrated in Figure 3, which can approximate Eqn. (9) in an efficient manner. Firstly, to avoid computing inversion of $\mathbf{\Sigma}_{t-1}$, we decompose $\mathbf{\Sigma}_{t-1}^{-1}$ into $\mathbf{\Sigma}_{t-1}^{-1/2}\mathbf{\Sigma}_{t-1}^{-1/2}$. Then $\mathbf{Y}_{t-1}$ can be computed by $\left(\mathbf{\Sigma}_{t-1}^{-1/2}\mathbf{X}\right)^T\left(\mathbf{\Sigma}_{t-1}^{-1/2}\mathbf{X}\right)$. Given $\mathbf{Q}_0 = \mathbf{\Sigma}_{t-1}$ and $\mathbf{P}_0 = \mathbf{I}$, $\mathbf{\Sigma}_{t-1}^{-1/2}$ can be computed with the following Newton-Schulz iteration method [19]:

$$\mathbf{Q}_k = \frac{1}{2}\mathbf{Q}_{k-1}(3\mathbf{I} - \mathbf{P}_{k-1}\mathbf{Q}_{k-1}),$$
$$\mathbf{P}_k = \frac{1}{2}(3\mathbf{I} - \mathbf{P}_{k-1}\mathbf{Q}_{k-1})\mathbf{P}_{k-1}. \quad (10)$$

After K iterations, $\mathbf{Q}_K$ and $\mathbf{P}_K$ will converge to $\mathbf{\Sigma}_{t-1}^{1/2}$ and $\mathbf{\Sigma}_{t-1}^{-1/2}$, respectively. Previous works [31, 27] have demonstrated Eqn. (10) can achieve satisfying performance with only one iteration. It motivates us to approximate $\mathbf{\Sigma}_{t-1}^{-1/2}$ with one-step Newton-Schulz iteration, i.e., $\mathbf{\Sigma}_{t-1}^{-1/2} \approx \mathbf{P}_1 = \frac{1}{2}(3\mathbf{I} - \mathbf{\Sigma}_{t-1})$. So $\mathbf{Y}_{t-1}$ is approximated as

$$\mathbf{Y}_{t-1} \approx \frac{1}{4}\mathbf{Z}^T\mathbf{Z}, \;\; \mathbf{Z} = (-\widehat{\mathbf{X}}_{t-1}\widehat{\mathbf{X}}_{t-1}^T + 3\mathbf{I})\mathbf{X}. \tag{11}$$

So far, we compute $\Lambda(\mathbf{Y}_{t-1})$ in Eqn. (9) requiring $\Lambda(\mathbf{Z}^T\mathbf{Z})$ and $\widehat{\mathbf{X}}_{t-1}\widehat{\mathbf{X}}_{t-1}^T$, both of which are second-order tensors. Following the idea of tensor approximation [30, 3, 40], we approximate them using learnable 1×1 convolutions followed by element-wise product. Given a $W \times H \times d$ tensor $\widehat{\mathcal{X}}_{t-1}$, which can be reshaped to $\widehat{\mathbf{X}}_{t-1}$ with $N = W \times H$, we can efficiently implement $\Lambda(\mathbf{Y}_{t-1})$ as follows:

$$\begin{aligned}\Lambda(\mathbf{Y}_{t-1}) &\approx \widehat{conv}_{1\times}\big(conv_{1\times}(\mathcal{Z}_{t-1}) \odot conv_{1\times}(\mathcal{Z}_{t-1})\big), \\ \mathcal{Z}_{t-1} &\approx [conv_{1\times}(\widehat{\mathcal{X}}_{t-1}) \odot conv_{1\times}(\widehat{\mathcal{X}}_{t-1})] \oplus conv_{1\times}(\mathcal{X}),\end{aligned} \tag{12}$$

where $conv_{1\times}$ and $\widehat{conv}_{1\times}$ denotes one and a group of 1×1 convolutions, respectively; $\oplus$ means element-wise addition.

Given $\Lambda(\mathbf{Y}_{t-1})$, the weights $\mathbf{w}_t$ are computed with $Nd/\big(\Lambda(\mathbf{Y}_{t-1}) + \mathbf{c} \odot \Lambda(\mathbf{Y}_{t-1})^{1-\beta}\big)$. Here, $Nd/(\cdot)$ can be regarded as a normalization on estimated weights and $\mathbf{c} \odot \Lambda(\mathbf{Y}_{t-1})$ is a Hadamard product, where we use a $Sigmoid$ function and one 1×1 convolution for implementation, respectively. Finally, we compute $\mathbf{w}_t$ using the following formulation:

$$\mathbf{w}_{t-1} \approx \phi\big(\Lambda(\mathbf{Y}_{t-1}) \oplus conv_{1\times1}((\Lambda(\mathbf{Y}_{t-1}))^{1-\beta})\big), \tag{13}$$

where ϕ is a $Sigmoid$ function. In terms of Eqns. (12) and (13), we can implement our re-weighted block using basic 1×1 convolutions, element-wise operations and $Sigmoid$ function, endowing its efficiency and straightforward backpropagation. To estimate covariance of generalized Gaussian, we need to compute Eqn. (9) repeatedly. To this end, as illustrated in Figure 2, we propose an unrolling re-weighted module. It consists of multiple consecutive re-weighted blocks, each of which aims at implementing Eqn. (9). By stacking multiple re-weighted blocks, we can flexibly construct our unrolling re-weighted module.

3.2.3 Square Root Covariance Layer

As shown in Eqn. (6), for optimizing the regularized MLE (5), we need to compute matrix square root of covariance once the weights are estimated. Here we construct a square root covariance layer to achieve it. Let weighted activations be $\widehat{\mathbf{X}}$, we can compute square root covariance of $\widehat{\mathbf{X}}$ as

$$\widehat{\mathbf{\Sigma}} = (\widehat{\mathbf{X}}\widehat{\mathbf{X}}^T)^{1/2} = \mathbf{\Sigma}^{1/2} = \widehat{\mathbf{U}}\widehat{\mathbf{\Lambda}}^{\frac{1}{2}}\widehat{\mathbf{U}}^T, \tag{14}$$

where $\mathbf{\Sigma} = \widehat{\mathbf{U}}\widehat{\mathbf{\Lambda}}\widehat{\mathbf{U}}^T$ is SVD of $\mathbf{\Sigma}$. $\widehat{\mathbf{\Lambda}}$ and $\widehat{\mathbf{U}}$ are the eigenvalues and eigenvectors of $\mathbf{\Sigma}$, respectively. $\widehat{\mathbf{\Lambda}}^{\frac{1}{2}}$ indicates element-wise square root of the eigenvalues. It is easy to see that computation of Eqn. (14) heavily dependents on SVD or eigenvalues decomposition (EIG).

However, SVD or EIG is limitedly supported on GPU, slowing down the training speed of square root covariance layer [27]. Eqn.(10) gives the form of Newton-Schulz Iteration [19], which shows $\widehat{\mathbf{\Sigma}}$ can be approximated by $\mathbf{Q}_K$ with K iterations and initialization of $\mathbf{Q}_0 = \mathbf{\Sigma}$ and $\mathbf{P}_0 = \mathbf{I}$. Compared with matrix square root via SVD (14), Eqn. (10) only involves matrix multiplication, suitable for GPU implementation. Here we employ the recently proposed trainable iterative method [27] based on Eqn. (10) to make better use of multi-GPU and accelerate the training of network. As suggested in [27], additional pre-normalization (i.e., $\mathbf{Q}_0 = \frac{1}{\mathrm{tr}(\mathbf{\Sigma})}\mathbf{\Sigma}$) and post-compensation (i.e., $\widehat{\mathbf{\Sigma}} = \sqrt{\mathrm{tr}(\mathbf{\Sigma})}\mathbf{Q}_K$) are employed. Thus, the partial derivative of loss function l with respect to $\widehat{\mathbf{X}}$ can be derived based on matrix backpropagation [23].

Specifically, backpropagation formula of post-compensation takes the following form:

$$\frac{\partial l}{\partial \mathbf{Q}_K} = \sqrt{\mathrm{tr}(\mathbf{\Sigma})}\frac{\partial l}{\partial \widehat{\mathbf{\Sigma}}}; \;\; \frac{\partial l}{\partial \mathbf{\Sigma}}\Big|_{\mathrm{p}} = \frac{\mathrm{tr}\Big(\big(\partial l/\partial \widehat{\mathbf{\Sigma}}\big)^T\mathbf{Q}_K\Big)}{2\sqrt{\mathrm{tr}(\mathbf{\Sigma})}}\mathbf{I}. \tag{15}$$

According to Eqn. (10), the gradients of k-th iteration are

$$\begin{aligned}\frac{\partial l}{\partial \mathbf{Q}_{k-1}} =& \frac{1}{2}\Big(\frac{\partial l}{\partial \mathbf{Q}_k}\Big(3\mathbf{I} - \mathbf{Q}_{k-1}\mathbf{P}_{k-1}\Big) - \mathbf{P}_{k-1}\frac{\partial l}{\partial \mathbf{P}_k}\mathbf{P}_{k-1} \\ &- \mathbf{P}_{k-1}\mathbf{Q}_{k-1}\frac{\partial l}{\partial \mathbf{Q}_k}\Big), \\ \frac{\partial l}{\partial \mathbf{P}_{k-1}} =& \frac{1}{2}\Big(\Big(3\mathbf{I} - \mathbf{Q}_{k-1}\mathbf{P}_{k-1}\Big)\frac{\partial l}{\partial \mathbf{P}_k} - \mathbf{Q}_{k-1}\frac{\partial l}{\partial \mathbf{Q}_k}\mathbf{Q}_{k-1} \\ &- \frac{\partial l}{\partial \mathbf{P}_k}\mathbf{P}_{k-1}\mathbf{Q}_{k-1}\Big), \\ \frac{\partial l}{\partial \mathbf{Q}_0} =& \frac{1}{2}\Big(\frac{\partial l}{\partial \mathbf{Q}_1}\Big(3\mathbf{I} - \mathbf{Q}_0\Big) - \frac{\partial l}{\partial \mathbf{P}_1} - \mathbf{Q}_0\frac{\partial l}{\partial \mathbf{Q}_1}\Big).\end{aligned} \tag{16}$$

Considering Eqn. (15), the gradient of l with respect to $\mathbf{\Sigma}$ can be computed as

$$\frac{\partial l}{\partial \mathbf{\Sigma}} = \frac{\mathrm{tr}\Big(\big(\partial l/\partial \mathbf{Q}_0\big)^T\mathbf{\Sigma}\Big)}{-(\mathrm{tr}(\mathbf{\Sigma}))^2}\mathbf{I} + \frac{\partial l/\partial \mathbf{Q}_0}{\mathrm{tr}(\mathbf{\Sigma})} + \frac{\partial l}{\partial \mathbf{\Sigma}}\Big|_{\mathrm{p}}. \tag{17}$$

Finally, the partial derivative of l with respect to $\widehat{\mathbf{X}}$ is

$$\frac{\partial l}{\partial \widehat{\mathbf{X}}} = \widehat{\mathbf{X}}\Big(\frac{\partial l}{\partial \mathbf{\Sigma}} + \Big(\frac{\partial l}{\partial \mathbf{\Sigma}}\Big)^T\Big), \tag{18}$$

Given $\partial l/\partial \widehat{\mathbf{X}}$ in Eqn. (18), we can complete backpropagation of square root covariance layer.

3.3. Deep Global Generalized Gaussian Networks

As suggested in previous methods [23, 32, 37, 28], we construct our deep global generalized Gaussian networks (3G-Net) by inserting the proposed covariance of generalized Gaussian layer after the last convolution block. In this paper, we employ ResNet-50 and ResNet-101 [17] as backbone models. Following the settings in [28], we add one 1×1 convolution with 256 filters between the last convolution block and the proposed covariance of generalized Gaussian layer, and remove downsampling in the last stage. As such, dimension of last convolutional activations is reduced from 2048 to 256, while their size increases from 7×7 to 14×14, balancing efficiency and effectiveness.

To accomplish our unrolling re-weighted module, channel numbers of input and output of $conv_{1\times}$ in the Eqn. (12) both are set to 256. $\widehat{conv}_{1\times}$ is composed of two consecutive 1×1 convolutions, where the channel numbers of input and output in the first convolution are respectively set to 256 and 64, while ones of the second convolution are set to 64 and 1, respectively. We discard the element-wise power operation of Eqn. (13) in the re-weighted block, as the experimental results show it has little effect on performance. For guaranteeing efficiency of our 3G-Net, we run respectively the Newton-Schulz Iteration (10) within one iteration and five iterations in unrolling re-weighted module and the final square root covariance layer, albeit more iterations may bring further improvement. By performing matrix triangulation, our covariance of generalized Gaussian layer outputs a $256(256 + 1)/2$-dimensional vector for final prediction. Note that, our 3G-Net will bring about additional $0.34 \times T$ M parameters (T indicates number of iterations), comparing with existing deep global covariance pooling networks in Gaussian setting [23, 32, 28, 27].

4. Experiments

To evaluate the effectiveness of our proposed 3G-Net, we conduct experiments on widely used ImageNet-1K [9] and Places365 [44] datasets. We first describe training details of our 3G-Net, and make ablation study to analyze effects of key components using ImageNet-1K. Finally, we compare with state-of-the-arts on both ImageNet-1K and Places365.

4.1. Training Details

To train our 3G-Net, we adopt the same data augmentation strategy with [34, 17, 28]. Specifically, all training images with mean subtraction and standard color augmentation are resized with their shorter side randomly sampled on [256, 512], and a 224×224 patch is randomly cropped from each resized image. The random horizontal flip is used. Following the settings in [17], we optimize the network parameters using SGD with a mini-batch size of 256, a momentum of 0.9 and a weight decay of 0.0001. The learning rate is initialized to 0.1, and is divided by 10 every 30 epochs. We adopt respectively single center crop and 10-crop predictions on ImageNet-1K and Places365, and report top-1 and top-5 error rates on validation sets for comparison. All programs are implemented using PyTorch package[1], and run on a PC equipped with four Titan Xp GPUs and 64G RAM.

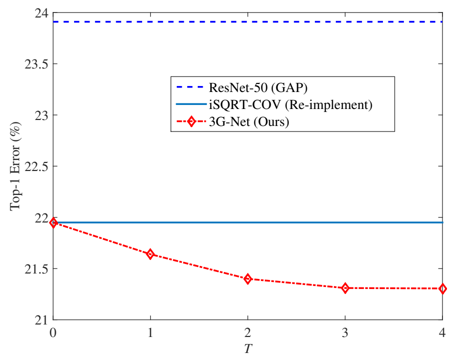

Figure 4. Effect of number of iterations (T) on the proposed 3G-Net with ResNet-50 architecture on ImageNet-1K.

4.2. Ablation Study on ImageNet-1K

Impact of Iteration Number Number of iterations (T) of our modified iterative re-weighted method (i.e., number of blocks in unrolling re-weighted module) plays a key role in covariance estimation of generalized Gaussian. To assess the effect of parameter T on our 3G-Net, we employ ResNet-50 as a backbone model and conduct experiments on ImageNet-1K dataset. It contains about 1.28M training images and 50K validation images, collected from 1,000 object categories. Top-1 errors of our 3G-Net with different numbers of iterations are shown in Figure 4, where we also compare with baseline methods, i.e., ResNet-50 with GAP [17] and iSQRT-COV [27]. Note that ResNet-50 with iSQRT-COV [27] can be regarded as a special case of our 3G-Net without re-weighting activations, i.e., estimating covariance in Gaussian setting. Increasing number of iterations can achieve lower classification errors, and the performance of our 3G-Net saturates with $T = 3$ (21.31% in Top-1 error). The larger number of iterations brings negligible gain but more computational and memory costs. Compared with baseline methods, our 3G-Net outperforms ResNet-50 based on GAP and iSQRT-COV over 2.6% and 0.64% in Top-1 error, respectively. Based on the above results, we set T to 3 throughout the following experiments for balancing effectiveness and efficiency.

Effectiveness of Re-weighted Module To verify the effectiveness of our unrolling re-weighted module, we compare with its two variants. The first one only employs $\widehat{conv}_{1\times}$ in Eq. (12) (i.e., two consecutive 1×1 convolu-

[1]The source code and network models will be available at `https://github.com/csqlwang/3G-Net`

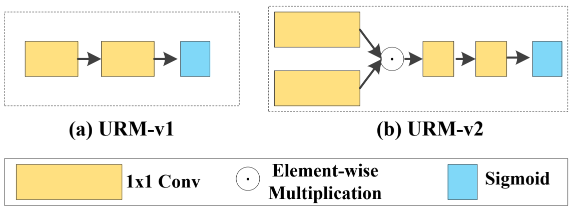

Figure 5. Illustration of two variants of our unrolling re-weighted module, i.e., (a) URM-v1 and (b) URM-v2.

Method	Top-1 error	Top-5 error
None	21.95	6.17
URM-v1	21.97	6.17
URM-v2	21.89	6.14
3G-Net with $T = 1$ (Ours)	21.64	5.81
3G-Net with $T = 3$ (Ours)	**21.31**	**5.61**

Table 1. Results (in %) of 3G-Net with different re-weighted modules on ImageNet-1K.

Method	Top-1 err.	Top-5 err.
GAP [17]	24.6	7.7
GAP (Re-implement)	23.91	7.15
Plain COV	26.41	9.09
B-CNN [32]	23.18	6.96
G^2DeNet [37]	22.77	6.55
MPN-COV [28]	22.73	6.54
iSQRT-COV [27]	22.14	6.22
iSQRT-COV (Re-implement)	21.95	6.17
3G-Net w/o Estimator (5)	25.17	8.14
3G-Net (Ours)	**21.31**	**5.61**

Table 2. Results (in %) of different global pooling methods under ResNet-50 architecture on ImageNet-1K.

tions), which is a commonly used method to generate spatial attention maps [41]. The second one introduces an additional $conv_{1\times}(\widehat{\mathbf{X}}_{t-1}) \odot conv_{1\times}(\widehat{\mathbf{X}}_{t-1})$ into the first variant of module. These two variants can be seen as two subdivisions of our unrolling re-weighted module, namely URM-v1 and URM-v2, respectively. The illustrations of URM-v1 and URM-v2 are shown in Figure 5. Table 1 lists the results of our method with different re-weighted modules, from it we can see that both URM-v1 and URM-v2 achieve no or negligible gain, while our single re-weighted block outperforms URM-v1 and URM-v2 by 0.36% and 0.33% in Top-5 error, respectively. The unrolling re-weighted module consisting of 3 blocks achieves further improvement, showing the effectiveness of our unrolling re-weighted module.

Robust Estimator We evaluate effect of robust estimation (5) on our 3G-Net. It brings no extra parameters, and is on par with non-robust one in space/time complexity. As compared in Tables 2 and 3, 3G-Net without (w/o) robust estimation (5) is superior to Plain COV, but is clearly inferior to the one with robust estimator. Additionally, 3G-Net under ResNet-50 with or w/o robust estimation obtain 43.07/13.34 *vs.* 45.47/15.00 on Places365. Above results clearly show the significance of our robust estimation.

Comparison of Various Global Pooling We compare

Method	Backbone Models	Top-1	Top-5
FBN [30]	ResNet-50	24.0	7.1
SORT [40]		23.82	6.27
ResNeXt [42]		22.11	5.90
SE [20]		23.29	6.62
CBAM [41]		22.66	6.31
A^2-Nets [6]		23.0	6.5
DropBlock [14]		21.87	5.98
iSQRT-COV [27]		22.14	6.22
3G-Net (Ours)		**21.31**	**5.61**
GAP [17]	ResNet-101	23.6	7.1
ResNeXt [42]		21.2	5.6
SE [20]		22.38	6.07
CBAM [41]		21.51	5.69
iSQRT-COV [27]		21.21	5.68
3G-Net w/o Estimator (5)		24.23	7.58
3G-Net (Ours)		**20.37**	**5.17**
ResNet-152 [17]		23.0	6.7
ResNet-152 + SE [20]		21.57	5.73
ResNet-200 [18]*		21.7	5.8
PyramidNet-200 [16]*		20.1	5.4
DenseNet-264 [21]		22.15	6.12

Table 3. Comparison of errors (in %) with state-of-the-art methods on ImageNet-1K. All methods employ single 224×224 crop prediction, and the competing results are duplicated from the original papers. *The results are copied from [20].

our 3G-Net with several existing global pooling methods using ResNet-50, including the original GAP [17], Plain covariance (COV) pooling (i.e., $\mathbf{X}\mathbf{X}^T$), B-CNN [32], MPN-COV [28], G^2DeNet [37] and iSQRT-COV [27]. The results of GAP, MPN-COV and iSQRT-COV are copied from the original papers. We implement Plain COV, B-CNN and G^2DeNet by ourselves, and we also re-implement GAP and iSQRT-COV. For fair comparison, we adopt the same settings of network and hyperparameters for all competitors. Note that we insert a BN layer [22] after B-CNN model for stable and rapid convergence. The results of different methods on ImageNet-1K are given in Table 2. All second-order pooling methods except Plain COV outperform the original GAP. Plain COV achieves unsatisfactory result in this case. G^2DeNet and MPN-COV obtain similar results, which are superior to B-CNN. Our 3G-Net achieves the best performance, demonstrating covariance of generalized Gaussian is more effective than the ones based on Gaussian [32, 28, 27, 37]. Our 3G-Net outperforms iSQRT-COV by 0.56% in Top-5 error, which is a non-trivial improvement, since iSQRT-COV is a very strong baseline while iSQRT-COV under ResNet-101 with much more parameters just achieves 0.54% gain over iSQRT-COV with ResNet-50.

4.3. Comparison on ImageNet-1K

Here we compare our 3G-Net, under both ResNet-50 and ResNet-101 architectures, with several state-of-the-art methods on ImageNet-1K. The top-1 and top-5 errors of different methods are listed in Table 3, where the results of ResNet-200 [18], PyramidNet-200 [16] and remaining competing methods are duplicated from [20] and the original papers, respectively. As shown in Table 3, our 3G-Net

Method	ResNet-50 [17] △	ResNet-50+B-CNN [32]	ResNet-50+iSQRT-COV [27]	ResNet-50+3G-Net (Ours)	ResNet-101+3G-Net (Ours)
Top-1 err.	44.82	44.24	43.68	**43.07**	**42.77**
Top-5 err.	14.71	14.27	13.73	**13.34**	**13.12**
Method	GoogLeNet [35]△	ResNet-152 [17] △	ResNet-101 [17]	ResNeXt-101 [42]♣	CRU-Net-116 [5]♣
Top-1 err.	46.37	45.26	44.09	43.79	43.40
Top-5 err.	16.12	14.92	13.93	13.75	13.45

Table 4. Results (in %) of different methods with 10-crop prediction on Places365. △The results are duplicated from `https://github.com/CSAILVision/places365`. ♣The results are copied from [5].

obtains the best performance among all competing methods under ResNet-50 architecture. Compared with deep local second-order statistics networks, i.e., FBN [30] and SORT [40], our 3G-Net achieves a clear improvement. Meanwhile, 3G-Net is superior to ResNeXt [42], which employs much wider convolution filters. Compared with deep CNNs based on various advanced self-attention methods [20, 41, 6], 3G-Net obtains 1.98%, 1.35%, 1.69% and 1.01%, 0.7%, 0.89% gains in top-1 and top-5, respectively. Our 3G-Net obtains 0.83% in top-1 (0.61% in top-5) gains over the top deep covariance pooling network [27].

When ResNet-101 is used as a backbone model, the proposed 3G-Net improves the original ResNet-101 with GAP by a large margin. In like manner, our 3G-Net is superior to ResNeXt-101 over 0.89% in top-1 (0.43% in top-5). Meanwhile, it outperforms respectively SE-Net and CBAM [20, 41] about 2.01% and 1.14% in top-1 (0.9% and 0.52% in top-5). The proposed 3G-Net improves iSQRT-COV over 0.84% and 0.51% in top-1 and top-5, respectively. Note that our 3G-Net based on ResNet-50 is slightly superior to iSQRT-COV with ResNet-101 in top-5 error, while our 50-layer 3G-Net outperforms 152-layer ResNet [17] and 152-layer ResNet with SE module [20]. Furthermore, 50-layer 3G-Net performs better than 200-layer ResNet [18] while 101-layer 3G-Net is slightly superior to 200-layer pyramidal ResNet [16] in top-5 error. Our 50-layer 3G-Net also outperforms DenseNet [21] of 264 layers. Above results clearly suggest the competitiveness of our 3G-Net.

4.4. Comparison on Places365

Finally, we evaluate our 3G-Net on standard Places365 dataset, which includes about 1.8M and 36.5K images of 365 scene classes for training and validation, respectively. Compared with ImageNet-1K, each sample image in Places365 involves of more objects, leading more ambiguity. Using ResNet-50 and ResNet-101 as backbone models, we compare with three global pooling methods (i.e., GAP [17], B-CNN [32] and iSQRT-COV [27]) and four deep CNN architectures (i.e., GoogLeNet [35], ResNet-152 [17], ResNeXt-101 [42] and CRU-Net-116 [5]). We adopt 10-crop prediction in comparison to the existing results.

The results of different methods are given in Table 4, where we implement B-CNN by ourselves, and implement iSQRT-COV using the source code released by the authors. We adopt exactly the same parameter settings for fair comparison. Compared with other three global pooling methods under ResNet-50 architectures, our 3G-Net obtains the lower classification error. The proposed 3G-Net is significantly better than the original GAP, and outperforms B-CNN and iSQRT-COV by 1.17% and 0.61% in top-1 error, respectively. Our 3G-Net can achieve further improvement using ResNet-101, and obtains the best results. It demonstrates the effectiveness of our global covariance of generalized Gaussian layer. Compared with the advanced deep CNN architectures, our 3G-Net clearly outperforms GoogLeNet and ResNet-152, while achieving better results than CRU-Net-116 and ResNeXt-101, although they are much deeper and wider.

5. Conclusion

In this paper, we proposed a novel 3G-Net, which robustly estimates a global covariance of generalized Gaussian distribution to summarize the last convolutional activations, since distributions of convolutional activations are complex and have long tails, which can not be fully characterized by Gaussian models. Our 3G-Net assumes distribution of convolutional activations obey a generalized Gaussian model, capturing characteristic of activations more precisely. The experimental results on large-scale ImageNet-1K and Places365 datasets demonstrated our 3G-Net can achieve higher classification accuracy than deep CNNs with either GAP or global covariance of Gaussian. The effectiveness of 3G-Net suggests more precise characterization of convolutional activations is helpful to improve performance of deep CNNs. In future, we will apply the proposed 3G-Net to action or video classification, and investigate integration of more diverse distributions (e.g., exponential families [36]) into deep CNNs for further improvements.

Acknowledgments

The work was supported by the National Natural Science Foundation of China (Grant No. 61806140, 61471082, 61671182, 61876127, 61732011), Natural Science Foundation of Tianjin Under Grants 17JCZDJC30800, and Young Elite Scientists Sponsorship Program by Tianjin. Qilong Wang was supported by China Post-doctoral Programme Foundation for Innovative Talent.

References

[1] R. Arandjelovic, P. Gronat, A. Torii, T. Pajdla, and J. Sivic. NetVLAD: CNN architecture for weakly supervised place recognition. In *CVPR*, 2016.

[2] G. E. P. Box and G. C. Tiao. *Bayesian Inference in Statistical Analysis*. Wiley, New York, USA, 1992.

[3] S. Cai, W. Zuo, and L. Zhang. Higher-order integration of hierarchical convolutional activations for fine-grained visual categorization. In *ICCV*, 2017.

[4] J. Carreira, R. Caseiro, J. Batista, and C. Sminchisescu. Free-form region description with second-order pooling. *IEEE TPAMI*, 37(6):1177–1189, 2015.

[5] Y. Chen, X. Jin, B. Kang, J. Feng, and S. Yan. Sharing residual units through collective tensor factorization to improve deep neural networks. In *IJCAI*, 2018.

[6] Y. Chen, Y. Kalantidis, J. Li, S. Yan, and J. Feng. A^2-Nets: Double attention networks. In *NIPS*, 2018.

[7] Y. Cui, F. Zhou, J. Wang, X. Liu, Y. Lin, and S. Belongie. Kernel pooling for convolutional neural networks. In *CVPR*, 2017.

[8] X. Dai, J. Yue-Hei Ng, and L. S. Davis. FASON: First and second order information fusion network for texture recognition. In *CVPR*, 2017.

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

[10] I. S. Dhillon and J. A. Tropp. Matrix nearness problems with Bregman divergences. *SIAM J. MAP*, 29(4):1120–1146, 2008.

[11] Y. Du, C. Yuan, B. Li, L. Zhao, Y. Li, and W. Hu. Interaction-aware spatio-temporal pyramid attention networks for action classification. In *ECCV*, 2018.

[12] J. Fu, H. Zheng, and T. Mei. Look closer to see better: Recurrent attention convolutional neural network for fine-grained image recognition. In *CVPR*, 2017.

[13] Y. Gao, O. Beijbom, N. Zhang, and T. Darrell. Compact bilinear pooling. In *CVPR*, 2016.

[14] G. Ghiasi, T.-Y. Lin, and Q. V. Le. DropBlock: A regularization method for convolutional networks. In *NIPS*, 2018.

[15] M. Gou, F. Xiong, O. Camps, and M. Sznaier. MoNet: Moments embedding network. In *CVPR*, 2018.

[16] D. Han, J. Kim, and J. Kim. Deep pyramidal residual networks. In *CVPR*, 2017.

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

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

[19] N. J. Higham. *Functions of Matrices: Theory and Computation*. SIAM, Philadelphia, PA, USA, 2008.

[20] J. Hu, L. Shen, and G. Sun. Squeeze-and-excitation networks. In *CVPR*, 2018.

[21] G. Huang, Z. Liu, L. van der Maaten, and K. Q. Weinberger. Densely connected convolutional networks. In *CVPR*, 2017.

[22] S. Ioffe and C. Szegedy. Batch Normalization: Accelerating deep network training by reducing internal covariate shift. In *ICML*, 2015.

[23] C. Ionescu, O. Vantzos, and C. Sminchisescu. Matrix backpropagation for deep networks with structured layers. In *ICCV*, 2015.

[24] T. G. Kolda and B. W. Bader. Tensor decompositions and applications. *SIAM Review*, 51(3):455–500, 2009.

[25] S. Kong and C. Fowlkes. Low-rank bilinear pooling for fine-grained classification. In *CVPR*, 2017.

[26] A. Krizhevsky, I. Sutskever, and G. E. Hinton. ImageNet classification with deep convolutional neural networks. In *NIPS*, 2012.

[27] P. Li, J. Xie, Q. Wang, and Z. Gao. Towards faster training of global covariance pooling networks by iterative matrix square root normalization. In *CVPR*, 2018.

[28] P. Li, J. Xie, Q. Wang, and W. Zuo. Is second-order information helpful for large-scale visual recognition? In *ICCV*, 2017.

[29] Y. Li, M. Dixit, and N. Vasconcelos. Deep scene image classification with the MFAFVNet. In *ICCV*, 2017.

[30] Y. Li, N. Wang, J. Liu, and X. Hou. Factorized bilinear models for image recognition. In *ICCV*, 2017.

[31] T.-Y. Lin and S. Maji. Improved bilinear pooling with CNNs. In *BMVC*, 2017.

[32] T.-Y. Lin, A. RoyChowdhury, and S. Maji. Bilinear CNN models for fine-grained visual recognition. In *ICCV*, 2015.

[33] F. Pascal, L. Bombrun, J. Tourneret, and Y. Berthoumieu. Parameter estimation for multivariate generalized Gaussian distributions. *IEEE TSP*, 61(23):5960–5971, 2013.

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

[35] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *CVPR*, 2015.

[36] M. J. Wainwright and M. I. Jordan. Graphical models, exponential families, and variational inference. *Foundations and Trends in Machine Learning*, 1(1-2):1–305, 2008.

[37] Q. Wang, P. Li, and L. Zhang. G^2DeNet: Global Gaussian distribution embedding network and its application to visual recognition. In *CVPR*, 2017.

[38] Q. Wang, P. Li, W. Zuo, and L. Zhang. RAID-G: Robust estimation of approximate infinite dimensional Gaussian with application to material recognition. In *CVPR*, 2016.

[39] X. Wang, R. Girshick, A. Gupta, and K. He. Non-local neural networks. In *CVPR*, 2018.

[40] Y. Wang, L. Xie, C. Liu, S. Qiao, Y. Zhang, W. Zhang, Q. Tian, and A. Yuille. SORT: Second-order response transform for visual recognition. In *ICCV*, 2017.

[41] C. Woo, J. Park, J.-Y. Lee, and I. S. Kweon. BAM: Convolutional block attention module. In *ECCV*, 2018.

[42] S. Xie, R. B. Girshick, P. Dollár, Z. Tu, and K. He. Aggregated residual transformations for deep neural networks. In *CVPR*, 2017.

[43] T. Zhang, A. Wiesel, and M. S. Greco. Multivariate generalized Gaussian distribution: Convexity and graphical models. *IEEE TSP*, 61(16):4141–4148, 2013.

[44] B. Zhou, À. Lapedriza, A. Khosla, A. Oliva, and A. Torralba. Places: A 10 million image database for scene recognition. *IEEE TPAMI*, 40(6):1452–1464, 2018.