Neurocomputing 440 (2021) 197–206

Contents lists available at ScienceDirect

Neurocomputing

journal homepage: www.elsevier.com/locate/neucom

Brief papers

Progressive principle component analysis for compressing deep convolutional neural networks

Jing Zhou [a], Haobo Qi [b,*], Yu Chen [b], Hansheng Wang [b]

[a] Center for Applied Statistics, School of Statistics, Renmin University of China, Beijing 100872, China
[b] Guanghua School of Management, Peking University, Beijing 100871, China

ARTICLE INFO

Article history:
Received 21 May 2020
Revised 6 January 2021
Accepted 8 January 2021
Available online 18 January 2021
Communicated by Zidong Wang

Keywords:
CNN compression
model acceleration
progressive PCA
kernel-wise reduction

ABSTRACT

In this work, we propose a progressive principal component analysis (PPCA) method for compressing deep convolutional neural networks. The proposed method starts with a prespecified layer and progressively moves on to the final output layer. For each target layer, PPCA conducts kernel principal component analysis for the estimated kernel weights. This leads to a significant reduction in the number of kernels in the current layer. As a consequence, the channels used for the next layer are also reduced substantially. This is because the number of kernels used in the current layer determines the number of channels for the next layer. For convenience, we refer to this as a progressive effect. As a consequence, the entire model structure can be substantially compressed, and both the number of parameters and the inference costs can be substantially reduced. Meanwhile, the prediction accuracy remains very competitive with respect to that of the baseline model. The effectiveness of the proposed method is evaluated on a number of classical CNNs (AlexNet, VGGNet, ResNet and MobileNet) and benchmark datasets.

The empirical findings are very encouraging. The code is available at https://github.com/zhoujing89/ppca.

1. Introduction

In recent years, there has been a surge of interest in employing deep convolutional neural networks (CNNs) in various fields, such as image classification and object detection [1,2]. The state-of-the-art results of CNNs have attracted considerable attention from both academia and industry. However, the application of cumbersome CNNs often requires a vast number of parameters, which are expensive in terms of both computation and storage. This hinders the employment of CNNs on low-end devices, such as mobile phones. One particular issue is that mobile devices often have very limited storage, memory, and computation power resources. Therefore, it is essential to develop highly compressed deep CNNs that have relatively low computational costs and high speedup in real-world applications. This requirement has made deep CNN compression an important research topic.

In a recent review paper, [3] summarized approaches for model compression and acceleration into four categories: compact model, tensor decomposition, data quantization and network sparsification. Our research is mostly related to the category of tensor decomposition but with a clear difference. To better illustrate this idea, we start with a standard CNN model structure as shown in the top panel in Fig. 1. Assume that the input is a 3-D tensor with d channels. We next apply a convolution kernel with size $k \times k$ to this tensor. Because the input tensor has d channels, the corresponding kernel weights should be a tensor of size $k \times k \times d$. In most cases, one kernel is insufficient for the desired accuracy. It is thus typical to employ a large number of convolution kernels. Let n be the total number of kernels used. This makes the kernel weight tensor a 4-D tensor of size $n \times k \times k \times d$. This leads to a total of ndk^2 weight parameters. Consequently, a large number of weight parameters are needed if the number of kernels (i.e., n) is large for many layers.

In fact, the powerful and sustainable effect of a large n value goes beyond the current layer, progressively affecting the next layer as shown in the top panel in Fig. 1. By applying n kernels (with size $k \times k \times d$) to the input tensor, we can obtain an output feature map with a total of n channels. In other words, the output feature map is also a tensor but of depth n. If another kernel with size (for example) $k \times k$ needs to be applied to this output tensor, the kernel weight tensor should be of size $k \times k \times n$. This leads to a total of nk^2 weight parameters to be estimated. If, once again, a

* Corresponding author.
E-mail addresses: jing.zhou@ruc.edu.cn (J. Zhou), qihaobo_gsm@pku.edu.cn (H. Qi), yu.chen@pku.edu.cn (Y. Chen), hansheng@pku.edu.cn (H. Wang).

https://doi.org/10.1016/j.neucom.2021.01.035
0925-2312/

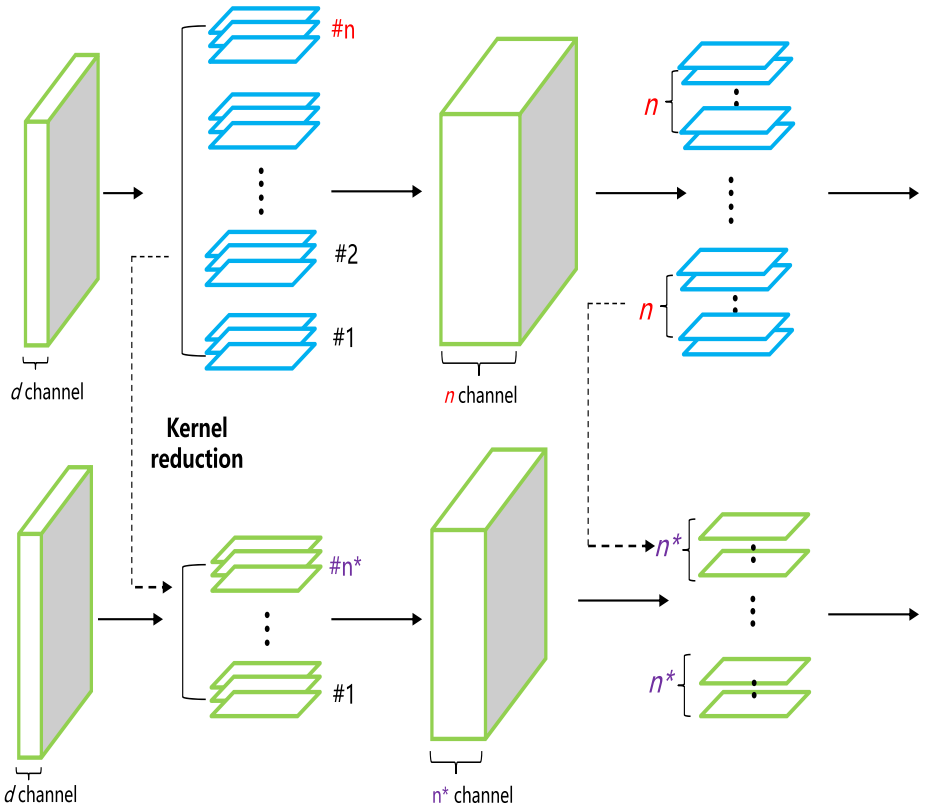


Fig. 1. Illustration of PPCA methods. Cubes represent feature maps, and stacked rectangles represent convolution kernels. Solid lines indicate convolution operations, and the dashed lines indicate compression operations. The top panel represents the baseline model, and the bottom panel represents the reduced model.

total of n kernels are employed, the total number of parameters needed by this output layer becomes n^2k^2, which is quadratic in n and could be very large.

The above discussion suggests that it would be of great practical interest to perform kernel reduction for every possible layer while sacrificing as little accuracy as possible. By doing so, the entire CNN model structure can be greatly compressed, as seen in the bottom panel in Fig. 1. This would lead to a significant reduction in terms of not only the number of parameters but also inference costs. In fact, such a possibility does exist. Ample empirical evidence suggests that the estimated kernel weights from the same CNN layer are often highly correlated with each other. This becomes the solid empirical foundation for model compression.

Specifically, we propose here a simple and effective PPCA solution, where PPCA stands for progressive principal component analysis. It starts with a designated CNN layer and then is applied to subsequent layers in a progressive manner. For a given CNN layer with n convolution kernels, the estimated kernel weights are likely to be heavily correlated. This is particularly true if the number of kernels (i.e., n) is relatively large. Therefore, most kernel weights can be well approximated by linear combinations of a few base weights. Accordingly, a novel kernel principal component analysis can be conducted. The resulting eigenvalues are then analyzed. In most cases, very few top eigenvalues can explain a significant portion of the total variability of the tensor weights. Let n^* be the minimal number of top eigenvalues such that their explained total variability exceeds a prespecified threshold value (e.g., 95%). This suggests that the original n weight tensors could be well approximated by only n^* base weights. Consequently, it might be unnecessary to keep a large number of kernels (n) for the target layer. Instead, a much reduced number, n^*, might be sufficient. This constitutes our kernel-PCA technique.

We then apply this kernel-PCA technique to every subsequent layer progressively up to the output layer. This leads to a reduced CNN that shares the same overall structure as the baseline CNN. More specifically, both the baseline and reduced CNNs share the same number of layers and kernel sizes, but the reduced CNN likely has a greatly reduced depth. The key difference between the two CNNs is that the number of kernels and thus channels used by the reduced CNN could be much smaller than that of the baseline CNN; see Fig. 1. This leads to a significant reduction in terms of parameters and inference costs. If necessary, the whole process can be further iterated a number of times so that the CNN can be further compressed.

Compared with previous model compression methods, PPCA has a number of unique features. First, for many previously published methods, model compression was conducted only for the input images [4–6], a single or a small number of convolutional layers [7,8], or only dense layers [9]. In contrast, PPCA can be performed for every possible layer. Second, unlike some previous methods [9,10], PPCA does not increase the number of layers in the CNN. Third, PPCA reduces the number of kernels and thus channels whenever possible, while most other methods keep the number of channels unchanged for the existing layer [11–15]. Lastly, PPCA can significantly reduce the complexity of running time, which could be seen as a non-marginal contribution to the existing practice and literature.

This paper is organized as follows. Section 2 reviews some of the related literature. Section 3 develops the PPCA method. Section 4 presents extensive experiments on the proposed method,

and their results are summarized in Section 5. Section 6 concludes the paper with a brief discussion for future research.

2. Related work

Many 2-D projection methods have been proposed in the past for image classification and face recognition. For example, [4] developed a 2-D PCA method for feature extraction. This method was soon generalized by [5]. They suggested that feature extraction worked not only for the row dimension but also for the column dimension of images. This led to a bidirectional 2-D PCA method. To preserve local image structure, [6] developed a 2-D locality-preserving projection (2-D LPP) method, while [16] proposed a 2-D neighborhood-preserving projection (2-D NPP) method. For a better prediction accuracy, [17] combined a 2-D PCA method with SVM. Ref. [18] proposed a sparse 2-D local discriminant projections (S2-D LDP) method. Ref. [19] also used the 2-D PCA method for filter analysis, which led to a 2-D PCA-Net method. This method was further improved by [14] using l_1 penalization. Ref. [15] utilized low-rank learning techniques for the 2-D NPP method with improved robustness and better discriminative ability. Undoubtedly, these pioneering methods are extremely useful for dimension reduction. However, they are only performed for the observed inputs (i.e., images, subimages or vectors).

Another set of literature focuses on low-rank approximation [12,7,8,20,13]. The common focus of these methods is layer decomposition. Ref. [20] proposed a 3-D decomposition method to compress all the convolutional layers. Ref. [13] presented a Tucker decomposition method for model compression. Ref. [11] proposed a model compression method for recurrent neural network (RNN) and long short-time memory (LSTM) models. Ref. [9] focused on layer pairs with dense connections in a DNN model. To implement the method, several extra layers were created for the baseline model. Ref. [10] proposed a rank-constraint dimension reduction method for the weight matrix between the input layer and the first hidden layer in a DNN model. Their method requires an intermediate layer to be inserted between layers. None of these methods reduces the number of kernels and thus channels for the existing layers.

For a more comprehensive literature survey in this regard, we refer to [3] for an excellent review. To summarize, we find that the existing research on model compression can be further improved from the following perspectives. First, we need a method that is sufficiently generalized so that it can be applied to every possible layer. Second, the method should not increase the number of layers. Lastly, the new method should reduce the number of kernels and thus channels as much as possible.

3. Methodology

3.1. The PPCA method

Consider a standard CNN with a total of T layers. For any layer t ($1 \leqslant t \leqslant T$), assume the depth of the feature map is d_t. Then, n_t kernels of size $k_t \times k_t$ are applied to this feature map. This leads to a weight tensor of size $n_t \times k_t \times k_t \times d_t$ with a total of $n_t d_t k_t^2$ parameters. Because the number of kernels used in the tth layer (i.e., n_t) determines the depth of the feature map for the next layer (i.e., d_{t+1}), we should have $n_t = d_{t+1}$. The corresponding kernel weight tensor should be of size $n_{t+1} \times k_t \times k_t \times n_t = n_{t+1} \times k_t \times k_t \times d_{t+1}$. A total of $n_{t+1} n_t k_t^2 = n_{t+1} d_{t+1} k_t^2$ parameters are involved. For the $(t+1)$ th layer, this number could be large if n_t and n_{t+1} are large. Therefore, the objective of PPCA is to reduce the number of kernels (i.e., n_t) and thus the number of channels (i.e., d_{t+1}) as much as possible. Of course, this should be done in a very careful way so that the predictive power is not substantially sacrificed.

Recall that the feature map of the tth layer is also a tensor but of depth d_t. A total of n_t kernels of size $k_t \times k_t$are applied to this feature map. This leads to a weight tensor of size $n_t \times k_t \times k_t \times d_t$. This tensor can be written as $W = (\omega_{i,x,y,j}) \in \mathbb{R}^{n_t \times k_t \times k_t \times d_t}$ with $1 \leqslant i \leqslant n_t, 1 \leqslant x, y \leqslant k_t$, and $1 \leqslant j \leqslant d_t$. To reduce the number of kernels used in this layer (i.e., n_t), a novel principal component analysis should be conducted for W. To this end, we reshape the 4-D tensor W into a standard 2-D matrix format as $V = (v_{s,i}) \in \mathbb{R}^{(k_t^2 d_t) \times n_t}$. In other words, V is a $(k_t^2 d_t) \times n_t$ matrix with $w_{i,x,y,j} = v_{s,i}$ and $s = (x-1)(k_t d_t) + (y-1)d_t + j$. Note that the bias terms are not discussed here for notation simplicity purposes. However, they should be taken into consideration for practical implementations.

We next centralize V by column so that its column mean is 0. We then construct the covariance matrix as $\Sigma = V^\top V/(k_t^2 d_t)$. Next, we perform SVD on Σ as

$$\Sigma = U \Lambda U^\top = \sum_{i=1}^{n_t} \lambda_i u_i u_i^\top, \tag{1}$$

where $U = (u_1, u_2, \cdots, u_{n_t})$ is an orthonormal matrix and $\Lambda = \mathrm{diag}(\lambda_1, \lambda_2, \cdots, \lambda_{n_t})$ with $\lambda_1 \geqslant \lambda_2, \cdots \geqslant \lambda_{n_t}$ is a diagonal matrix. Here, λ_i is an eigenvalue of Σ, and u_i is the corresponding eigenvector. Obviously, these quantities (i.e., $\Sigma, U, \Lambda, \lambda_i$, and u_i) are layer dependent. Thus, a subscript t should be used. However, for the sake of notation simplicity, the subscript t is omitted. To determine the appropriate number of principal components (e.g., rank), we use the cumulative variability contribution rate as a criterion. For any positive integer $1 \leqslant m \leqslant n_t$, the rate is defined as

$$r_m = \sum_{i=1}^{m} \lambda_i / \sum_{i=1}^{n_t} \lambda_i,$$

Then, the number of kernels that need to be retained is given by n_t^*, which is the smallest m such that $r_m > r_{\min}$ and $r_{\min}$ is a prespecified variability threshold value (e.g., 90%). For convenience purposes, we refer to the above process as the kernel-PCA technique.

Then, the PPCA method is extended by applying the above kernel-PCA technique progressively to every designated layer. Practically, it should start from a prespecified starting layer (e.g., the t_0th layer) and progressively move on to the final output layer. Accordingly, the layers earlier than t_0 (i.e., $t < t_0$) remain unchanged. We then define $n_t^* = n_t$ for $t < t_0$. Once the kernel-PCA operation is progressively accomplished for every designated layer, it can be restarted again from the prespecified starting layer for further model compression. This might lead to multiple iterations. Denote the total number of iterations by n_{iter}. This leads to the final output $\{n_t^* : 1 \leqslant t \leqslant T\}$, and a greatly reduced CNN is produced. Therefore, the reduced model should be retrained. As a result, the information contained in the training data can be further utilized.

The reduced CNN shares the same overall structure as the baseline CNN. Both the baseline and reduced CNNs share the same number of layers and kernel sizes (but the reduced CNN likely has a greatly reduced depth). The key difference is that the reduced CNN has a much smaller number of kernels and thus channels for many layers than the baseline model; see Fig. 1. Obviously, this leads to a significant reduction in terms of parameters and inference costs.

3.2. Tuning parameters

Even though the proposed PPCA method is conceptually simple, its practical implementation is not trivial. It involves three important tuning parameters: the variability threshold value r_m, the

starting layer t_0, and the total number of iterations n_{iter}. For a given CNN, these tuning parameters should be practically fine-tuned to achieve the best empirical performance.

The first tuning parameter is the variability threshold r_m. It largely determines the reduced number of kernels for the target layer. Generally, the smaller the r_m value is, the higher the reduction rate that can be achieved, and thus a smaller number of parameters is obtained. In our simulation experiments, values between 85% and 99% are studied.

The second tuning parameter is the starting layer t_0. Our experience suggests that the earlier PPCA is started, the higher the reduction rate that can be achieved. However, we should not start PPCA too early. Otherwise, the CNN could be overcompressed and the resulting prediction accuracy could be significantly degraded. In fact, for many benchmark CNNs (e.g., VGG), most parameters are incorporated in the later layers (i.e., those layers close to the final output layer). We thus suggest not starting PPCA too early.

The third tuning parameter is the number of iterations. As one can expect, a greater number of iterations leads to higher reduction ratios. Unfortunately, this often comes at a considerable cost of predictive power. Our empirical experience suggests that optimal performance is often achieved within the first two iterations.

3.3. Progressive dimension reduction effect

It is remarkable that a reduction in the number of kernels used in the current layer should progressively determine the number of channels for the next layer. Consequently, a reduction in the number of kernels for one layer can lead to parameter reduction for both the current and the next layers. We refer to this as the progressive effect.

The progressive effect leads to a significant reduction in parameters and running time complexity. To better explain this idea, consider, for example, a standard CNN as the baseline model. Assume the depth of its feature map for every layer to be the same, that is, $d_t = d$ for some $d > 0$. Further assume the number of kernels used by each layer is also the same, that is, $n_t = n$ for some $n > 0$. The number of kernels used by the current layer (i.e., n_t) determines the channel depth for the next layer (i.e., d_t). We should have $n = n_t = d_{t+1} = d$ for most ts. For simplicity, we assume $n = d = d_t = n_t$ for every $1 \leqslant t \leqslant T$. Last, assume that the kernel size used for each layer is also the same, $k_t \times k_t = k \times k$. Suppose the input feature map for layer t is of size $M_t \times M_t$. Then the size of output feature map for layer t is given by $(M_t - k + 1) \times (M_t - k + 1)$ (assume stride size equal to 1). Recall that there are a total of T layers. Then, we can easily calculate the total number of parameters and running time complexity needed by a baseline CNN. The total number of parameter is given by Tk^2n^2 and the total running time complexity is $\mathcal{O}(\sum_{t=1}^{T}(M_t - k + 1)^2k^2n^2)$. Remarkably, both of them are quadratic (not linear) functions in n. If there were no progressive effects, this would be a linear function in n. Assume that we can reduce the number of kernels used by each layer from n to approximately $n/2$ (i.e., by 50%). Then, the total number of parameters required by the reduced model becomes $Tk^2n^2/4$ and the total running time complexity becomes $\mathcal{O}(\sum_{t=1}^{T}(M_t - k + 1)^2k^2n^2/4)$. This represents a 75% reduction both in terms of parameters and running time complexity.

4. Experiments

4.1. CNN models and datasets

To demonstrate its empirical performance, the proposed PPCA method is evaluated on a number of classical CNNs and benchmark datasets. The classical CNNs are AlexNet (e.g., approximately 30 million parameters) [21], VGG (e.g., approximately 15 million parameters) [1], ResNet (e.g., approximately 0.47 million parameters) [2], and MobileNet (e.g., approximately 3 million parameters) [22]. The benchmark datasets are the CIFAR10, CIFAR100, SVHN [23,24] and CatDog dataset released by Kaggle in 2013.

4.2. Performance measures

Following the existing literature [25,26], we consider two metrics to gauge the empirical performances of different compression methods: the parameter reduction ratio (Prr) and the FLOP reduction ratio (Frr). Meanwhile, the out-of-sample prediction accuracy (Acc) is also monitored.

4.3. Competing methods

For comparison purposes, a total of three competing methods are studied, and a summary of their empirical performance as reported in the literature is provided: NS [25], SFP [27] and COP [26]. More specifically, NS [25] proposed a network slimming approach by enforcing channel-level sparsity. The key idea is to identify and prune unimportant layers using BN scaling factors. SFP [27] proposed a soft filter pruning (SFP) method for model acceleration. The key idea is to cut appropriately defined and unnecessary convolution kernels. COP [26] proposed an effective way to detect redundant filters in a network. The key idea is to evaluate the importance of a kernel by a novel correlation measure.

4.4. Implementation details of the CNNs

All classical CNNs (i.e., AlexNet, VGG, ResNet and MobileNet) are trained on each benchmark dataset (i.e., CIFAR10, CIFAR100, SVHN and CatDog). This leads to a total of $4 \times 4 = 16$ working models. All the working models are trained using the stochastic gradient descent (SGD) algorithm with a momentum effect of 0.9. The batch size is fixed to 128 for all working models. The weight decay rate is set to 0.0001 with an l_2-norm regularizer for AlexNet, ResNet, and MobileNet. The weight decay rate is set to 0.0005 with an l_2-norm regularizer for VGG. Different learning rate updating strategies are used for different training combinations. Finally, a total of 200 epochs are conducted for each working model. For each working model, we choose the epoch with the maximum prediction accuracy as the baseline model. All the experiments were run on a Tesla K80 GPU with 11 GB memory.

4.5. Tuning Parameter Specification for PPCA

As described earlier, a total of three tuning parameters are used by PPCA and are fine-tuned on real datasets for each CNN: the variability threshold value r_m, the starting layer t_0, and the total number of iterations n_{iter}. For the first tuning parameter (i.e., the variability threshold value), three different values are considered: 85%, 90%, and 95%.

The second parameter is the starting layer t_0. Different starting layers are used for different CNNs. For VGG, four different starting layers are considered, that is, $t_0 \in \{3, 4, 5, 6\}$. For AlexNet, three different starting layers are studied, that is, $t_0 \in \{2, 3, 4\}$. For MobileNet, four different starting layers are investigated, that is, $t_0 \in \{3, 4, 5, 6\}$.

The ResNet model contains a total of 15 residual blocks classified into three different groups. The first group contains 1–6 blocks with 16 convolution kernels. The second group contains 7–11 blocks with 32 convolution kernels. The last group contains 12–15 blocks with 64 convolution kernels. The starting layer for

ResNet is set to be the first convolutional layer in the 2nd or 3rd group.

The last tuning parameter is the number of iterations n_{iter}. Our empirical experience suggests that it cannot be too large. Otherwise, the CNN could be overcompressed, and the resulting prediction accuracy could be very poor. Accordingly, we set $n_{iter} \in \{1, 2\}$ for VGG, AlexNet and ResNet, and $n_{iter} = 1$ for MobileNet.

5. Results

5.1. Tuning parameter effects

In this subsection, we study the impact of the three tuning parameters (variability threshold r_m, starting layer t_0, and number of iterations n_{iter}). Three measures are used to gauge the finite sample performance: prediction accuracy (Acc), the parameter reduction ratio (Prr), and the FLOP reduction ratio (Frr). For illustration purposes, we use VGG with the CIFAR10 dataset as an example.

First, consider the variability threshold value r_m. For this experiment, three different values are studied: $r_m = 85\%,\ r_m = 90\%$, and $r_m = 95\%$. For each r_m, a total of 8 experiments are conducted. Each experiment corresponds to different combinations of t_0 and n_{iter}. Then, the prediction accuracy (Acc), parameter reduction ratio (Prr), and FLOP reduction ratio (Frr) for those 8 experiments are summarized.

The detailed results are given in Fig. 2. As seen in the left panel of Fig. 2, we find that as long as the variability threshold value is set to be no less than $r_m = 90\%$, little predictive power would be sacrificed. In fact, a closer look reveals that the resulting prediction accuracy (Acc) could be even higher than that of the baseline model. However, in the middle panel of Fig. 2, we find that the parameter reduction ratio (Prr) could be substantial. In the case of $r_m = 90\%$, the Prr is much larger than 80%. This suggests that the baseline model can be substantially compressed with little sacrifice in predictive power. Last, in the right panel of Fig. 2, we find that the reduction in FLOPs is also substantial. With $r_m = 90\%$, the resulting Frr is far greater than 80%. This suggests that the inference cost of the reduced model can also be significantly reduced.

Next, we study the second tuning parameter, that is, the starting layer t_0. In this case, we consider a total of 4 different starting layers: $t_0 = 3,\ t_0 = 4,\ t_0 = 5$, and $t_0 = 6$. For each fixed t_0, we conduct a total of 6 experiments. Each experiment corresponds to a different combination of r_m and n_{iter}. The detailed results are given in Fig. 3. In the middle and right panels of Fig. 3, we find that earlier starting layers result in greater parameter and FLOP reduction ratios. However, this might come at the cost of prediction accuracy (Acc). If the starting layer is set to be too early (e.g., $t_0 = 3$), the resulting Acc could be highly unstable; see the left panel of Fig. 3. In many cases, the accuracy is much worse than that of the baseline model. However, if the starting layer is appropriately specified (e.g., $t_0 = 5$), little prediction accuracy would be sacrificed. In fact, it is clearly better than that of the baseline model in most cases. Meanwhile, the parameter and FLOP reduction ratios remain substantial. Specifically, the median values of Prr and Frr are approximately 80% for $t_0 = 5$.

The last tuning parameter is the number of iterations n_{iter}. In this case, we investigate 2 different values: $n_{iter} = 1$ and $n_{iter} = 2$. For each fixed n_{iter}, a total of 12 experiments are conducted. Each experiment corresponds to a different combination of r_m and t_0. The detailed results are given in Fig. 4. As expected, a larger n_{iter} value leads to higher parameter and FLOP reduction ratios, as shown in the middle and right panels of Fig. 4. However, the left panel of Fig. 4 suggests that $n_{iter} = 2$ might not be a good choice for this particular example because it leads to an overcompressed model structure and thus a substantially degraded prediction accuracy. For this example, $n_{iter} = 1$ is clearly the optimal choice. The resulting median values of Prr and Frr remain close to 80%.

5.2. Prediction and compression trade-off

On the one hand, the proposed PPCA method aims to compress a CNN model as much as possible. On the other hand, an overcompressed CNN model might suffer from a significant loss of prediction accuracy. Thus, it is of great importance to understand the trade-off between the prediction accuracy and model compression. Obviously, they should be appropriately balanced.

To this end, we use VGG, ResNet and MobileNet together with CIFAR100 as examples. For each CNN model and dataset combination, 3 different variability thresholds ($r_m \in \{85\%, 90\%, 95\%\}$) are considered. The number of iterations is set to $n_{iter} \in \{1, 2\}$ for VGG and ResNet and $n_{iter} = 1$ for MobileNet. The starting layer is set to $t_0 \in \{3, 4, 5, 6\}$ for VGG and MobileNet and $t_0 \in \{3, 8\}$ for ResNet. This leads to a total of 24 combinations for VGG, 12 combinations for ResNet, and 12 combinations for MobileNet. For each experiment, we summarize its prediction accuracy (Acc) and the parameter reduction ratio (Prr). Their relationships are plotted in the top panel of Fig. 5. The red line represents the accuracy of the baseline model, and the blue line is 2% below the red line. Similar scatter plots are also provided for the FLOP reduction ratio (Frr). The detailed results are given in the bottom panel of Fig. 5.

From the top panel of Fig. 5, we can draw the following conclusions. For VGG and ResNet, we find that the accuracy gradually decreases as the parameter reduction ratio increases. Specifically,

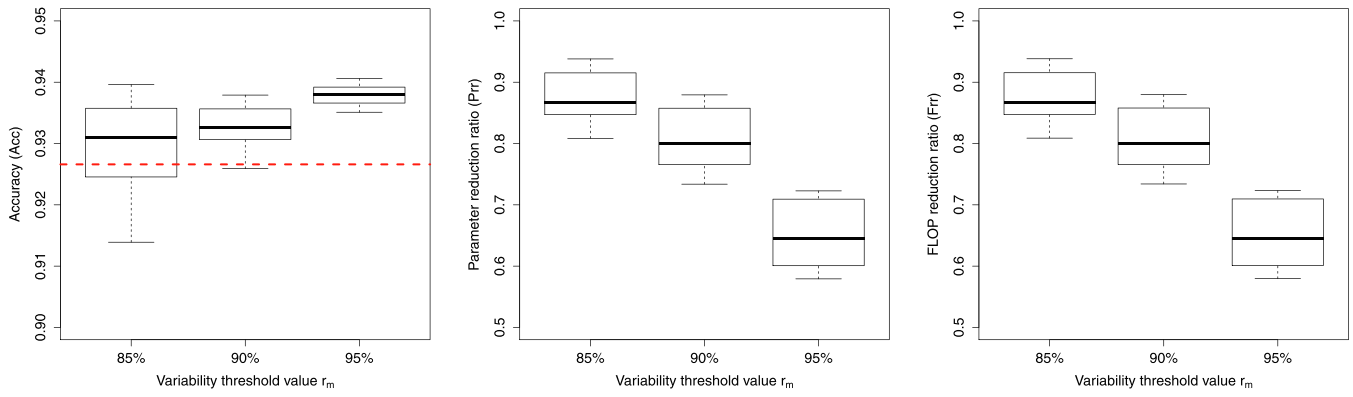


Fig. 2. Detailed experimental results for VGG on the CIFAR10 datasets. Three different values of the variability threshold are considered ($r_m = 85\%, 90\%, 95\%$). Three performance measures are summarized: accuracy (Acc), the parameter reduction ratio (Prr), and the FLOP reduction ratio (Frr). The red dashed line in the left panel represents the accuracy of the baseline model.

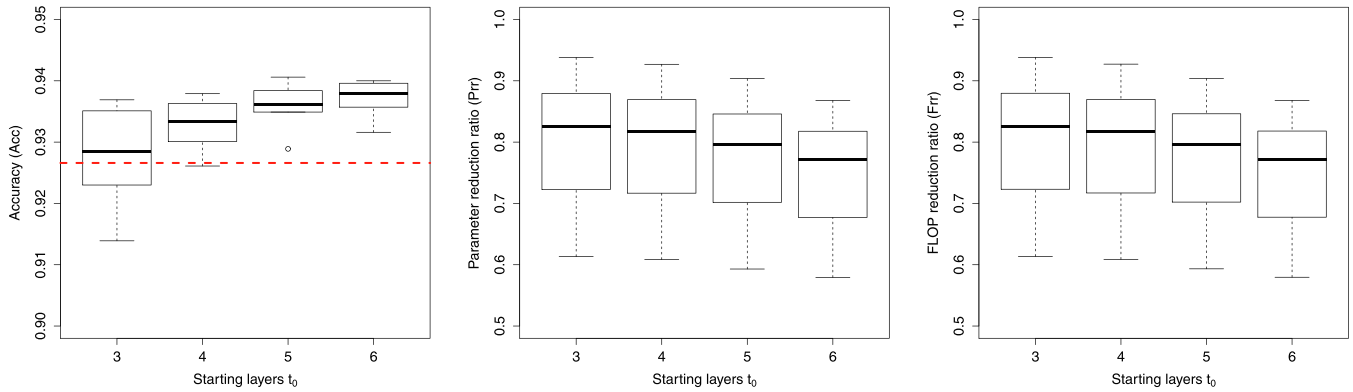


Fig. 3. Detailed experimental results for VGG on the CIFAR10 datasets. Four different starting layers are considered ($t_0 = 3, 4, 5, 6$). Three performance measures are summarized: accuracy (Acc), the parameter reduction ratio (Prr), and the FLOP reduction ratio (Frr). The red dashed line in the left panel represents the accuracy of the baseline model.

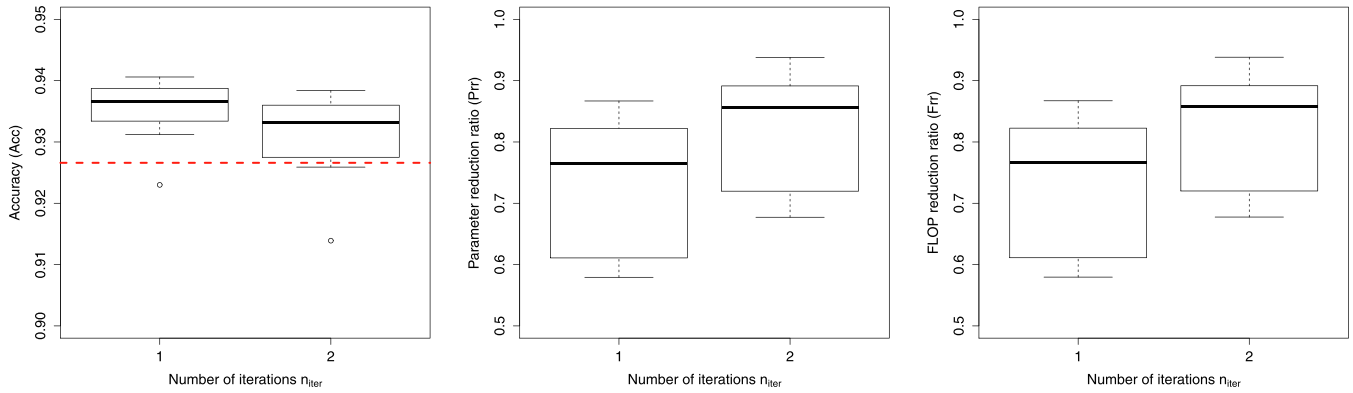


Fig. 4. Detailed experimental results for VGG on the CIFAR10 datasets. Two different numbers of iterations are considered (n_{iter} = 1,2). Three performance measures are summarized: accuracy (Acc), the parameter reduction ratio (Prr), and the FLOP reduction ratio (Frr). The red dashed line in the left panel represents the accuracy of the baseline model.

more than half of the reported VGG cases falling below the blue line. This indicates that an overreduced model may lose competitiveness in terms of accuracy. ResNet seems also very sensitive to the parameter reduction ratio, as there are eight cases that fall below the blue line. However, the story is somewhat different for MobileNet. It seems that MobileNet is not very sensitive to the parameter reduction ratio. Almost all the cases are around the red line and the blue line.

The bottom panel of Fig. 5 displays the relationship between the FLOP reduction ratio (Frr) and prediction accuracy (Acc). The patterns are qualitatively similar to those seen in the top panel.

5.3. Fine-tuned PPCA results

In this subsection, we report the fine-tuned PPCA results so that their optimal performance can be demonstrated. The entire study setup is nearly identical to that described above. However, for the best empirical performance, we try to extend the search scope for all the tuning parameters. As demonstrated in the previous subsection, it appears that increasing the number of iterations (e.g., $n_{iter} > 2$) can hardly be helpful. Similarly, the starting PPCA at too early a layer is unlikely to be wise either. Consequently, the only possibility left is to extend the search scope for r_m. Accordingly, every value in $\{85\%, 86\%, \cdots, 99\%\}$ is tested for r_m in this subsection. The best results in terms of prediction accuracy (Acc) are summarized in Table 1.

From Table 1, we can draw the following conclusions. First, for all cases, the baseline models can be compressed substantially using PPCA with little sacrifice of accuracy. For example, the value of Prr and Frr could be more than 85% in the case of MobileNet on SVHN with no sacrifice of accuracy. Second, we report that for many cases, the prediction accuracy of the reduced model is higher than that of the baseline model. For example, for the VGG model on the CIFAR10 dataset, the prediction accuracy of the baseline model is as high as 92.66%, while that of the reduced model is further improved to 94.06%. To summarize, we find that the proposed PPCA method works quite well on all CNNs and datasets under study.

5.4. Competing methods

Finally, we compare the performance of PPCA with a number of important competing methods, which were discussed in section IV. For comparison purposes, we include in Table 2 only those CNN and dataset combinations whose empirical results are reported in the existing literature. As a result, two classical CNNs (i.e., VGG and ResNet) and two benchmark datasets (CIFA10 and CIFA100) are included.

The results for VGG are summarized in the top panel of Table 2. We find that the results are very encouraging. PPCA demonstrates outstanding model compression with little sacrifice in predictive power. For example, for the CIFA10 database, the ΔAcc value (i.e. the difference in the Acc between the baseline model and the reduced model) of the PPCA method is only −0.05. This suggests that the drop in Acc from baseline model to the reduced model is as low as 0.05%. The corresponding parameter reduction ratio is

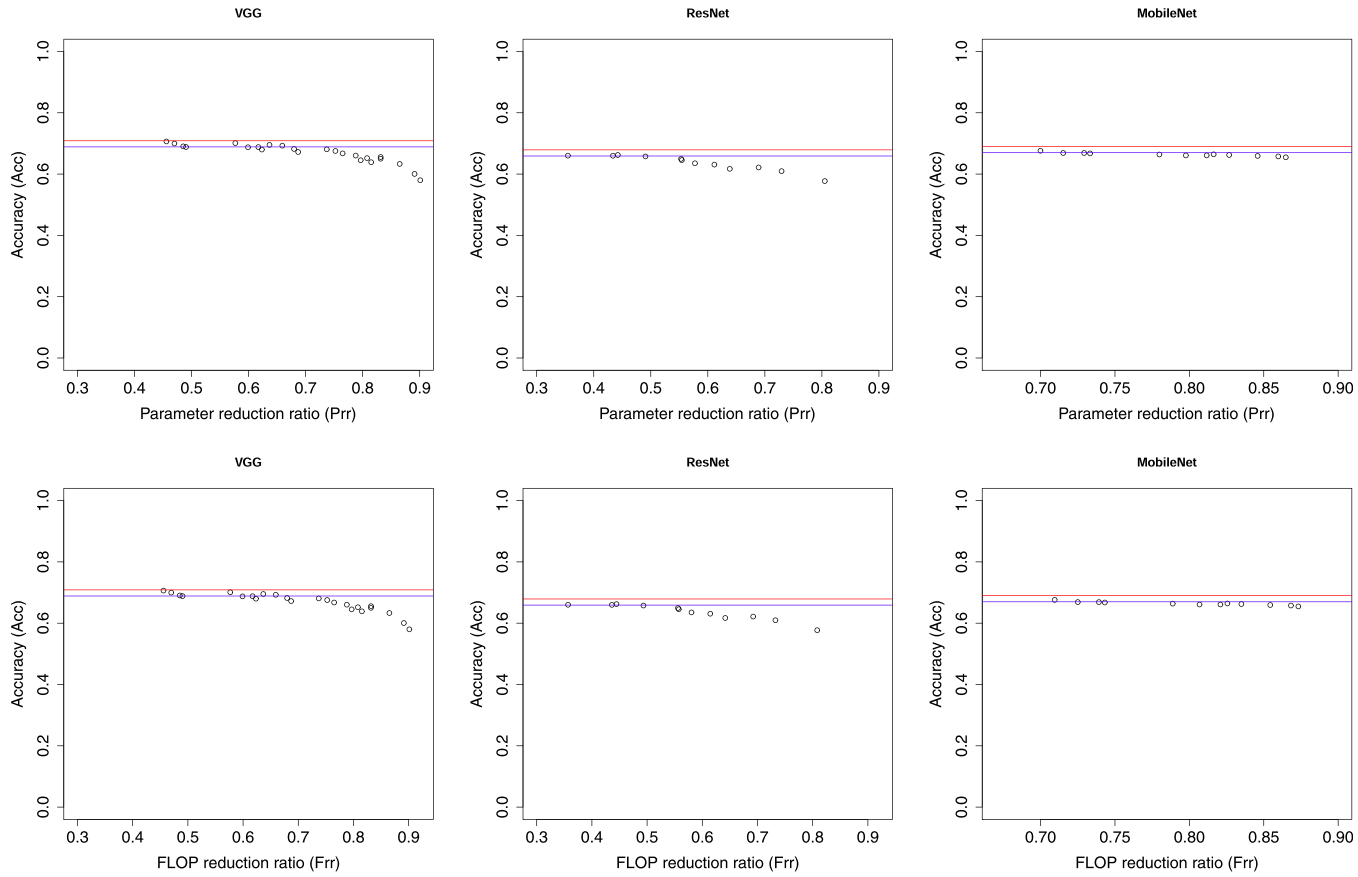


Fig. 5. Top panel: Parameter reduction rate (Prr) and prediction accuracy (Acc) for VGG, ResNet, and MobileNet. Bottom panel: FLOP reduction rate (Frr) and prediction accuracy (Acc) for the same CNNs. The red line is the prediction accuracy for the baseline model. The blue line is 2% below the red line.

Table 1
Fine tuned PPCA results for all CNNs and datasets combination. Alg is the algorithm name. Baseline Acc is the prediction accuracy of the baseline model. Reduced Acc is the prediction accuracy of the reduced model. ΔAcc is the difference between the baseline Acc and reduced Acc. Prr is the parameter reduction ratio, and Frr is the FLOP reduction ratio.

Datasets	Alg	Baseline Acc (%)	Reduced Acc (%)	ΔAcc (%)	Prr (%)	Frr (%)
CIFAR10	AlexNet	90.42	90.31	−0.11	63.7	52.9
	VGG	92.66	94.06	+1.40	59.3	59.3
	ResNet	92.96	92.67	−0.29	42.8	43.0
	MobileNet	91.85	91.11	−0.74	69.4	69.9
CIFAR100	AlexNet	65.27	63.29	−1.98	69.2	62.5
	VGG	70.87	70.64	−0.23	45.6	45.6
	ResNet	67.90	67.38	−0.52	30.7	30.8
	MobileNet	68.98	67.61	−1.37	70.0	70.9
SVHN	AlexNet	95.71	96.00	+0.29	82.2	71.8
	VGG	96.59	97.08	+0.49	45.2	45.3
	ResNet	96.74	96.76	+0.02	56.7	57.0
	MobileNet	95.69	95.71	+0.02	86.4	86.8
CatDog	AlexNet	87.40	87.59	+0.19	78.2	66.8
	VGG	90.28	90.43	+0.15	84.7	84.8
	ResNet	91.06	90.07	−0.99	37.2	37.4
	MobileNet	85.42	87.66	+2.24	67.3	67.7

as high as 92.7%, which ranks as the 2nd best parameter reduction ratio among the compared methods. Meanwhile, the resulting FLOP reduction ratio is the largest with Frr = 92.7%, and it is approximately 19.2% higher than that of the second best competing method. Similar results can be concluded from the CIFAR100 case.

The results for ResNet are summarized in the bottom panel of Table 2. The performance of PPCA remains to be competitive but no longer the best one. Such a result is expected. The key technique used by PPCA is kernel PPCA, which performs best for CNN models with very "thick" layers (i.e., layers with a large number of kernels and channels). Many classical CNNs are of this type (e.g., LeNet, AlexNet, VGG, etc.). However, ResNet is somewhat different; its network structure is already slimmed to some extent, in the sense that not many kernels are contained in each layer. For this case, both the NS and COP methods seem to be better choices. To summarize, PPCA seems to work well for all cases and has a superior performance for CNNs with a large number of kernels.

Table 2
Results for competing methods vs PPCA. Alg is the algorithm. Acc is the optimal predictive accuracy. ΔAcc is the accuracy change between the baseline model and the reduced model. Prr is the parameter reduction ratio, and Frr is the FLOP reduction ratio. The values of the different measures for the competing methods are from their original papers.

Datasets	Alg	Acc	ΔAcc (%)	Prr (%)	Frr (%)
VGG					
CIFA10	NS	93.59	+0.03	73.9	29.2
	SFP	92.99	−0.57	73.0	73.0
	COP	93.31	−0.25	92.8	73.5
	PPCA	92.61	−0.05	92.7	92.7
CIFA100	SFP	71.52	−1.07	42.3	42.2
	COP	71.77	−0.82	73.2	43.1
	PPCA	70.10	−0.77	57.5	57.7
ResNet					
CIFA10	NS	90.23	−2.41	35.3	53.4
	SFP	92.08	−0.56	39.2	41.0
	COP	91.97	−0.67	57.5	53.9
	PPCA	92.56	−0.40	47.8	48.1
CIFA100	SFP	67.83	−0.91	33.8	33.9
	COP	68.29	−0.45	35.2	34.2
	PPCA	67.38	−0.52	30.7	30.8

To further verify this, we use VGG (e.g., a classical network with "thick" layers) and ResNet (e.g., a classical network with "thin" layers) as two representative network structures. They were first trained on the CIFAR10 dataset. Next, we consider all the convolution layers before the output layer for the network under study. Singular value decomposition was conducted for the kernel weights for each layer. We then computed the total variability explained by different top portions of eigenvalues (e.g., 20%, 40%, 60%, 80%, and 100%) for each layer. They were then averaged over different layers. The detailed results are summarized in Fig. 6. We find that for the VGG model, the total variability explained by the different top portions of eigenvalues is always greater than that of ResNet. This suggests that the convolutional layers used by the VGG network should have a better chance than those in ResNet for model compression. As a result, VGG (i.e., the network with "thick" layers) should be more suitable for the proposed PPCA method than ResNet (i.e., the network with "thin" layers). This might explain the superior performance of PPCA on "thick" type networks.

5.5. Running time analysis

As we discussed before, the running time complexity could be significantly reduced by the proposed PPCA method. To verify this, we conduct some simulation studies. A number of classical networks (AlexNet, VGG, ResNet, and MobileNet) are trained on the CIFAR10 dataset as examples. For each example, the best-performing PPCA model is selected and then compared with the full model in terms of real running time. Specifically, for a given target model (i.e., the PPCA model or the full model), a total of 10,000 pictures from the CIFAR10 dataset were evaluated and recorded the running time. We then replicated the above procedure for 100 times. In this way, the averaged real running time is computed and reported in Fig. 7. We find that for all four networks, the average running time for the PPCA model is smaller than that for the benchmark model. In most cases, the reduction is very significant.

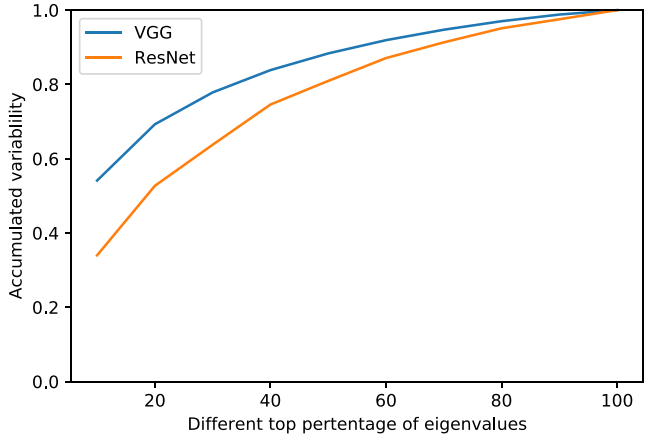


Fig. 6. Total variability explained by different top portion of eigenvalues. Blue line is for VGG and yellow line is for ResNet. Both are trained on CIFAR10 dataset.

6. Conclusions

In this paper, we present a PPCA method for CNN compression. The proposed method can be easily applied to both convolutional and fully connected layers. It starts with a prespecified layer and then conducts a novel kernel-PCA operation for subsequent layers. The entire process progressively moves on to the final output layer and could be iterated multiple times. The proposed method leads to a significant reduction in terms of parameters and inference costs. Meanwhile, the prediction power remains competitive. Extensive numerical studies are presented to demonstrate the empirical performance of the proposed method.

To conclude this article, we present here a number of interesting topics for future study. First, PPCA compresses a CNN structure by reducing the number of kernels. It does not change the size of the kernel for each layer. We suspect that the kernel size can also be learned automatically. Second, most important CNNs are rather complicated, with a very large number of parameters. This makes

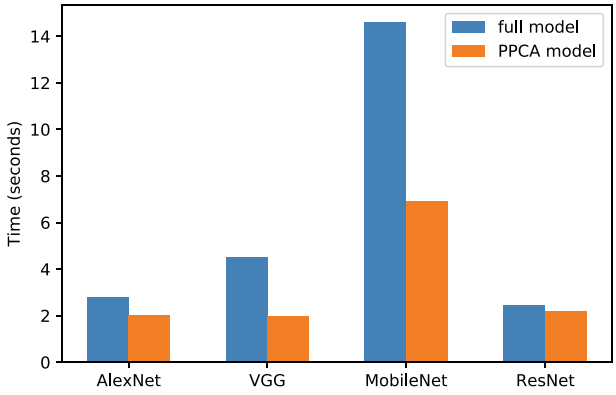


Fig. 7. Average running time on 10,000 pictures. Blue bar is for VGG and yellow bar is for ResNet. Both are trained on CIFAR10 dataset.

its numerical optimization very difficult (or at least time consuming). The development of a novel algorithm to speed-up the optimization would be another important topic for future study. Third, many state-of-the-art methods have been created in the field of tensor decomposition. Future studies should investigate ways to combine these methods together to further improve the empirical performance. Fourth, the proposed PPCA method only takes the weight matrix (e.g., W) into consideration, whereas the layer response x in the linear combination $Wx + b$ was not effectively used. This makes our method less sensitive to the data. This should be an interesting direction for future research. Lastly, the proposed PPCA method involves a large number of tuning parameters, and their optimal combination needs to be learned for different CNNs and datasets. This is another very time-consuming task. From a design-of-experiment (DOE) perspective, this is a computer experiment with many factors. It would thus be of great interest to design an optimization experiment so that the best tuning parameter combination can be detected with as few experiments as possible.

Declaration of Competing Interest

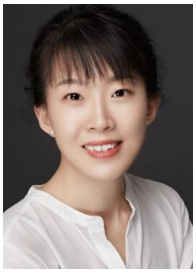
The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgment

Zhou's research is supported by the National Natural Science Foundation of China (Nos. 71702185, 71873137 and 11971504), Beijing Municipal Social Science Foundation (No. 19GLC052), National Statistical Science Research Project (No. 2020LZ38). Hansheng Wang's research is partially supported by the National Natural Science Foundation of China (Nos. 71532001, 11525101 and 71332006) and China's National Key Research Special Program (No. 2016YFC0207704). Consulting Research Project of Chinese Academy of Engineering (2020-XY-30).

References

[1] K. Simonyan, A. Zisserman, Very deep convolutional networks for large-scale image recognition, arXiv 1409.1556, 2014..
[2] K. He, X. Zhang, S. Ren, S. Jian, Deep residual learning for image recognition, in: 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016.
[3] B.L. Deng, G. Li, S. Han, L. Shi, Y. Xie, Model compression and hardware acceleration for neural networks: a comprehensive survey, Proceedings of the IEEE 108 (4) (2020) 485–532.
[4] J. Yang, D.D. Zhang, A.F. Frangi, J.Y. Yang, Two-dimensional pca: A new approach to appearance-based face representation and recognition, IEEE Transactions on Pattern Analysis and Machine Intelligence 26 (1) (2004) 131–137.
[5] D. Zhang, Z.H. Zhou, Two-directional two-dimensional pca for efficient face representation and recognition, Neurocomputing 69 (1/3) (2005) 224–231.
[6] D. Hu, G. Feng, Z. Zhou, Two-dimensional locality preserving projections (2dlpp) with its application to palmprint recognition, Pattern Recognition 40 (1) (2007) 339–342.
[7] E. Denton, W. Zaremba, J. Bruna, Y. Lecun, R. Fergus, Exploiting linear structure within convolutional networks for efficient evaluation, vol. 2, 2014..
[8] V. Lebedev, Y. Ganin, M. Rakhuba, I. Oseledets, V. Lempitsky, Speeding-up convolutional neural networks using fine-tuned cp-decomposition, 2014..
[9] J. Xue, J. Li, Y. Gong, Restructuring of deep neural network acoustic models with singular value decomposition, in: Interspeech, 2013..
[10] P. Nakkiran, R. Alvarez, R. Prabhavalkar, C. Parada, Compressing deep neural networks using a rank-constrained topology, 2015..
[11] R. Prabhavalkar, O. Alsharif, A. Bruguier, and . Mcgraw, On the compression of recurrent neural networks with an application to lvcsr acoustic modeling for embedded speech recognition, 2016..
[12] M. Jaderberg, A. Vedaldi, A. Zisserman, Speeding up convolutional neural networks with low rank expansions, in: BMVC 2014 – Proceedings of the British Machine Vision Conference 2014, 2014.
[13] Y.-D. Kim, E. Park, S. Yoo, T. Choi, L. Yang, D. Shin, Compression of deep convolutional neural networks for fast and low power mobile applications, 2016..
[14] Y.-K. Li, X.-J. Wu, J. Kittler, L1-2d 2pcanet: a deep learning network for face recognition, Journal of Electronic Imaging 28 (2) (2019) 023016.1–023016.9.
[15] Y. Lu, Z. Lai, X. Li, W.K. Wong, C. Yuan, D. Zhang, Low-rank 2-d neighborhood preserving projection for enhanced robust image representation, IEEE Transactions on Cybernetics 49 (5) (2019) 1859–1872.
[16] Z. Li, M. Du, 2d-npp: An extension of neighborhood preserving projection, in: 2007 International Conference on Computational Intelligence and Security (CIS 2007), 2007, pp. 410–414..
[17] T.H. Le, L. Bui, Face recognition based on svm and 2dpca, Computer Science abs/1110.5404 (2011).
[18] J.X. Zhang, Z.E. Zhang, J.Y. Liu, Palmprint recognition based on sparse two-dimensional local discriminant projections, Advanced Materials Research 998–999 (2014) 894–898.
[19] D. Yu, X.-J. Wu, 2dpcanet: a deep leaning network for face recognition, Multimedia Tools and Applications, 77 (10) (2018) 12919–12934..
[20] X. Zhang, J. Zou, K. He, J. Sun, Accelerating very deep convolutional networks for classification and detection, IEEE Transactions on Pattern Analysis and Machine Intelligence 38 (2015).
[21] A. Krizhevsky, I. Sutskever, G. Hinton, Imagenet classification with deep convolutional neural networks, in: Advances in Neural Information Processing Systems, vol. 25, no. 2, 2012..
[22] A.G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, H. Adam, Mobilenets: Efficient convolutional neural networks for mobile vision applications, 2017.
[23] A. Krizhevsky, G. Hinton, Learning multiple layers of features from tiny images, Computer Science Department, University of Toronto, Tech. Rep, vol. 1, 2009..
[24] Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, A. Ng, Reading digits in natural images with unsupervised feature learning, in: NIPS, 2011.
[25] Z. Liu, J. Li, Z. Shen, G. Huang, S. Yan, C. Zhang, Learning efficient convolutional networks through network slimming, 2017..
[26] W. Wang, C. Fu, J. Guo, D. Cai, X. He, Cop: Customized deep model compression via regularized correlation-based filter-level pruning, 2019, pp. 3785–3791..
[27] Y. He, G. Kang, X. Dong, Y. Fu, Y. Yang, Soft filter pruning for accelerating deep convolutional neural networks, 2018, pp. 2234–2240..



Jing Zhou received a B.S. in management from the Central University of Finance and Economics, Beijing, China in 2012 and a Ph.D. in Business Administration (quantitative marketing modeling direction) from Peking University, Beijing, China in 2016. She is currently an Associate Professor at the School of Statistics, Renmin University of China. Her research interests focus on network modeling, spatial econometrics and deep learning with business applications. Her previous research has been published in international journals, including the Journal of Business and Economic Statistics, Electronic Commerce Research and Applications, Statistica Sinica, Statistics and its Interface and Science China Mathematics.

Haobo Qi received a B.S. in statistics from Beijing Normal University, Beijing, China in 2019. He is currently pursuing his Ph.D. in Statistics from Peking University, Beijing, China. His research interests focus on statistics and deep learning.

Yu Chen received a B.S. degree in statistics from Renmin University of China, Beijing, China in 2015 and a Ph.D. degree in Statistics from Peking University, Beijing, China in 2020. His research interests focus on deep reinforcement learning and its applications on games.

Hansheng Wang received a B.S. in Statistics from Peking University, Beijing, China in 1998 and a Ph.D. in Statistics from the University of Wisconsin-Madison, USA in 2001. He is currently a professor at the Department of Business Statistics and Econometrics, Guanghua School of Management, Peking University, Beijing, China. He is the author of three books and more than 100 articles. His research interests include ultrahigh dimensional data analysis, network data analysis and statistical applications in e-commerce. He is an Associate Editor of the journals JASA, JBES, Statistica Sinica and Statistics and its Interface.