On Compressing Deep Models by Low Rank and Sparse Decomposition

Xiyu Yu[1] Tongliang Liu[1] Xinchao Wang[2] Dacheng Tao[1]
[1]UBTech Sydney AI Institute and SIT, FEIT, The University of Sydney
[2]IFP, Beckman Institute, The University of Illinois Urbana-Champaign (UIUC)
yuxiyu88@gmail.com tliang.liu@gmail.com xinchao@illinois.edu dacheng.tao@sydney.edu.au

Abstract

Deep compression refers to removing the redundancy of parameters and feature maps for deep learning models. Low-rank approximation and pruning for sparse structures play a vital role in many compression works. However, weight filters tend to be both low-rank and sparse. Neglecting either part of these structure information in previous methods results in iteratively retraining, compromising accuracy, and low compression rates. Here we propose a unified framework integrating the low-rank and sparse decomposition of weight matrices with the feature map reconstructions. Our model includes methods like pruning connections as special cases, and is optimized by a fast SVD-free algorithm. It has been theoretically proven that, with a small sample, due to its generalizability, our model can well reconstruct the feature maps on both training and test data, which results in less compromising accuracy prior to the subsequent retraining. With such a "warm start" to retrain, the compression method always possesses several merits: (a) higher compression rates, (b) little loss of accuracy, and (c) fewer rounds to compress deep models. The experimental results on several popular models such as AlexNet, VGG-16, and GoogLeNet show that our model can significantly reduce the parameters for both convolutional and fully-connected layers. As a result, our model reduces the size of VGG-16 by $15\times$*, better than other recent compression methods that use a single strategy.*

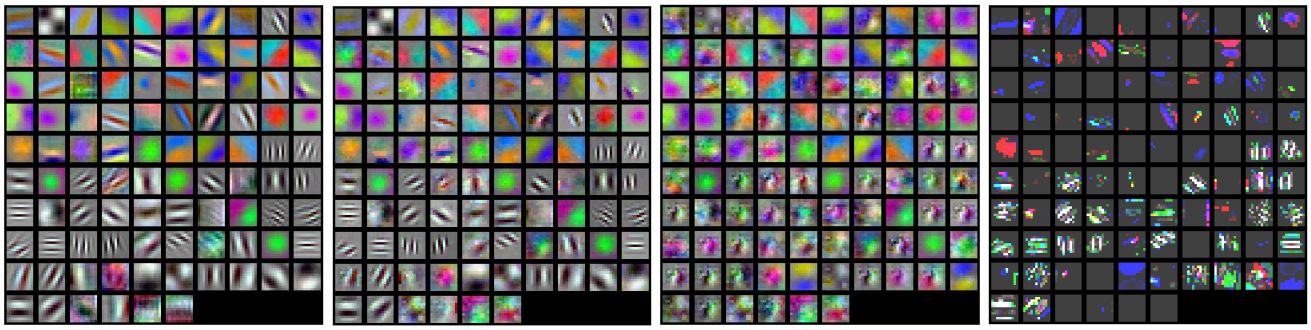
1. Introduction

Deep learning has delivered significant improvements in several fields including image classification [15, 23, 25, 9] and object detection [21]. However, the intensive computational and memory requirements of most deep models limit their deployment on daily use devices with low storages and computing capabilities such as cellphones and embedded devices. This limitation has motivated researchers to exploit the intrinsic redundancy found in parameters and feature maps in deep models. Generally, this redundancy is reflected in the structured nature of the weight matrices and feature maps [3, 24]. By removing the redundancy, resources can be saved without affecting the capacity and generalizability of most deep models.

Sparsity and low-rankness independently acted as vital structure assumptions in the previous work for redundancy removal. First, pruning is a straightforward strategy to remove correlated parameters and co-adapted neurons, and to obtain sparse structures. For example, LeCun et al. [16] used the second derivative information to guide the removal of unimportant weights. Han et al. [8] repeatedly retrained a sparsified model with unimportant weights removed using a hard thresholding method. He et al. [10] used measures like l_1 norm of out-linked weights to identify unimportant neurons. Mariet and Sra [19] sampled the most uncorrelated neurons by the determinant point processes, and removed other highly correlated neurons. Another structure assumption is the low-rankness. Weights in the convolutional and fully connected layers can be reduced by approximating low-rank filters [4, 27, 14]. Zhang et al. [30] estimated a low-rank subspace for the feature vectors that resulted in weight matrix decomposition, parameter reduction, and faster testing time.

However, we have observed that independently applying these assumptions is not sufficient and appropriate. Most previous work suffers from iteratively retraining, compromising accuracy, and low compression rates. Deep compression should thus process much richer structural information. We note that the weight filters usually share smooth components in a low-rank subspace, and also remember some important information represented by weights that are sparsely scattered outside the low-rank subspace. The resulting feature maps also contain the smooth components [30] and the spiky changes that represent the uniqueness of each feature.

Fig.1 well explains our intuition. We show the filters in the first layer of AlexNet. The $11\times11\times3\times96$ filters are represented by 96 3-channel images, rescaling the value range of the weights to [0,255]. The colors can reflect the patterns in the filters. As shown in Fig.1(a) and Fig.1(b), using the



(a) (b) (c) (d)

Figure 1. (a) Filters in the first layer of AlexNet (from Caffe Model Zoo). (b) Low-rank and sparse approximation using the proposed method. Here, rank is 12 (for L) and ratio of non-zero entries is 12.5% (for S). We reduce the number of parameters by $4\times$. (c) Low-rank components of the approximated filters. (d) Sparse components of the approximated filters. Here, for better visualization, we add a constant upon the whole sparse matrix.

low-rank and sparse decomposition, the resulting filters can well approximate the original counterpart when the number of paramters is reduced by $4\times$. Specifically, the low-rank component retains the smooth patterns in the filters, and the sparse component keeps some important patterns, such as directionality of the filters.

Therefore, we propose a single strategy to decompose weight matrices to their low-rank and sparse components. The novelty is a unified framework which regards the low-rank approximation and pruning connections as special cases, and reveals more insights about the richer structure information of parameters. With such proper structure assumptions, this model alleviates the compromising accuracy prior to retraining and achieves higher compression rate without loss of accuracy.

We integrate an asymmetric data reconstruction term into the low-rank and sparse decomposition. Further, we theoretically prove that, due to its generalizability, the proposed model guarantees the reconstructed feature maps of training and test data are not bad even when only a reasonable small sample is provided. Small reconstruction errors mean high accuracies for very deep compression which provides good initializations for retraining. Therefore, Stochastic Gradient Descent (SGD) solvers do not get stacked in bad local minima and retain the original[1] accuracy even with high compression rate. This can reduce the number of rounds[2] to compress the entire network and improve compression.

Motivated by the greedy bilateral smoothing, or briefly GreBsmo [32], for the low-rank and sparse decomposition, we develop the greedy bilateral decomposition (GreBdec) for deep model compression. Specifically, GreBdec uses only QR decompositions [5] and random projections, leading to a lower computational complexity compared with the traditional generalized Singular Value Decomposition (SVD) method [28].

Our experimental results demonstrate that our approach delivers higher compression rates in representative models compared to [8], the state-of-the-art compression method. The low sparsity rates and very low-rankness of the components result in a model that has a high potential for acceleration. Specifically, the sparse rates are lower than most previous work, which enables more efficient usage of the sparse matrix-vector multiplication operators.

2. Related Work

Compression by pruning. Han et al. [8] proposed a simple but effective pruning method that used a hard threshold determined by the standard deviation of the weights multiplied by a scalar. This threshold helps to remove the least important weights with small absolute values. To get sparser weight matrices, iterative hard thresholding was also applied. A special case of our model is that when we ignore the low-rank and data reconstruction terms, our method reduces to the simple hard thresholding method as in [8].

However, our approach has two main advantages over [8]. First, because of our feature map reconstruction term, our model provides a better initialization for retraining. We can usually compress all the convolutional layers in models like AlexNet and VGG-16 in one round rather than iteratively pruning and retraining. Second, by exploiting both the smooth components and important weights, we further reduce the weights in the convolutional layers, and the resulting components are much sparser than those in [8]. This is of potential values for accelerating the inference stages in deep models.

Compression by low-rank approximation. Reducing

[1] In this paper, the word "original" refers to the well-trained reference model prior to compression. For example, the original accuracy means the accuracy of the uncompressed deep model.

[2] A deep model usually needs to be repeatedly compressed in several rounds. Weight matrix decompositions for selected layers followed by a retraining process is one round.

parameter dimensions by low-rank approximation saves storage and simultaneously reduces time complexity during training and testing. Most methods [4, 12] approximate a tensor by minimizing the reconstruction error of the original parameters. However, these approaches tend to accumulate errors when multiple layers are compressed sequentially, and the output feature maps deviate far from the original values with the increase of compressed layers.

Here we regard this low-rank decomposition as a special case of our proposed model. Viewing the output features as a linear matrix product of the weight matrix and input features, we can treat the convolutional layer and the fully connected layer in the same way. Furthermore, the proposed model is integrated with a feature map reconstruction term which helps to alleviate the accumulated errors and obtain a better initialization for the subsequent retraining. Although Zhang et al. [30] also incorporate the feature map reconstruction, we are completely different in the following two aspects. First, Zhang et al. [30] minimize the reconstruction error to find an approximate low-rank subspace for the feature vectors while our model exploits the reconstruction error to estimate the low-rank and sparse components for weight matrices. Second, Zhang et al. [30] are mainly concerned with acceleration. As shown in their experiment results, the computational complexity based rank selection criteria often leads to large ranks for layers with more parameters, which results in less compression. By contrast, we focus more on deep compression for saving the storage. Due to the high sparsity and the low-rank approximation, the proposed model also possesses the potential for acceleration, which is achievable with further careful studies in the convolution with very sparse kernels.

Compression by other strategies. Weight sharing and quantization methods assume that many weights have similar values, and can thus be grouped in order to reduce the number of free parameters. Grouping methods include hashing [2], k-means and vector quantization [7, 6], and BinaryNets [20]. [29] presented an effective CNN compression approach in the frequency domain using the discrete cosine transform and quantization strategies. A different view is knowledge distillation that uses large and mature networks to teach a small model to learn good representations. For example, Hinton et al. [11] proposed using the output soft distributions as the knowledge of teacher networks. Romero et al. [22] also utilized the intermediate representations as hints to enrich the knowledge of the teacher. Luo et al. [18] claimed that neurons usually occupy more compact information to guide face model compression.

In this paper, we emphasize that the proposed method is orthogonal to compression techniques such as weight sharing , quantization, and Hoffman coding [7]. Consequently, our model can be combined with these approaches for further compression.

3. Deep Compression Model

Deep models are usually over-parameterized [3], with redundancy often resulting in huge storage and computational resource demands. But this redundancy also provides a opportunity to compress deep model without compromising accuracy provided that the deep network redundancy is properly removed. The weight matrix components usually reside in low-rank subspaces but some important entries are sparsely scattered in the weight matrices and mark the uniqueness of different filters. Therefore, we present a unified framework for deep compression by the low-rank and sparse decomposition. Our approach enjoys less information loss and produces better reconstructions for feature maps compared to SVD and pruning. After compression using low-rank and sparse decompositions, the model can be retrained to retain the original accuracy.

In deep model, the output response of a convolutional or fully connected layer can be obtained by

$$y = Wx,$$

where $x \in \mathbf{R}^k$ is the input feature vector, $W \in \mathbf{R}^{m\times k}$ is the weight matrix, and $y \in \mathbf{R}^m$ is the response. To explore a low-rank subspace combined with a sparse structure for the weight matrix W, we assume that $W \approx L + S$, where L is a low-rank component and S is a sparse matrix. Then, to compress the weight matrix, we have the following model:

$$\begin{aligned} \min_{L,S} \quad & \frac{1}{2}\|W - L - S\|_F^2, \\ s.t. \quad & \text{rank}(L) \le r, \\ & \text{card}(S) \le c, \end{aligned}$$

where $\text{rank}(L)$ denotes the rank of L and $\text{card}(S)$ denotes the cardinality of matrix S. This problem can be efficiently solved by "GoDec" [31]. We did not use "GreBsmo [32] because the parameter of soft thresholding is very difficult to be tuned to obtain the desired sparsity rate. Now suppose $L = UV$ where $U \in \mathbf{R}^{m\times r}$ and $V \in \mathbf{R}^{r\times k}$. We find that the total number of parameters drops from mk to $(m + k)r + c$. If r and c are small enough, many parameters can be reduced.

This decomposition is easily implemented. Take the convolutional layer as an example, we first concatenate two convolutional layers to implement the low-rank part. V represents convolutional filters, which fix the kernel sizes while changing the number of output feature maps to r. U indicates convolutional filters whose kernel sizes are equal to 1 with the input channel number equal to r. To obtain the final result, we sum the result of the low-rank part and the sparse convolutional layer where we add a mask to the original filters to help discard unmasked gradients during back-propagation.

However, a problem exists in this naïve decomposition method. If we sequentially and independently apply this naïve decomposition to several layers prior to a retraining, the approximation error of each layer will be accumulated. Thus, to alleviate this, an asymmetric data reconstruction term is incorporated. We consider a layer whose input feature map is not precise due to the approximation of the previous layer/layers. To abuse the notation, we still denote the approximate input as $X = [x_1, x_2, \cdots, x_n]$. Then, we use this approximate input to reconstruct the original output feature vectors $Y = [y_1, y_2, \cdots, y_n]$. Our refined model is:

$$\begin{aligned} \min_{L,S} \quad & \frac{1}{2n}\|Y - (L+S)X\|_F^2, \\ s.t. \quad & \frac{1}{2}\|W - L - S\|_F^2 \le \gamma, \\ & \text{rank}(L) \le r, \\ & \text{card}(S) \le c. \end{aligned} \tag{1}$$

This "asymmetric" feature map reconstruction process ensures the optimal approximation of weight matrices and feature maps in the current layer and reduces accumulated errors due to the approximate input.

Thanks to this asymmetric data reconstruction approach, the compressed model only suffers from a small drop in accuracy without retraining in some cases. For example, a $5\times$ compression rate for all convolutional layers with kernel size larger than 1 in GoogLeNet [25] leads to only a 3.7% decrease in testing accuracy (Top-5). We do not suggest that this process results in absolutely no loss of accuracy; no method can guarantee this when compression rates are very high. But, we can retrain the models, which finally enjoy high compression rates without loss of accuracy.

A special case. We consider a simplified model without the low-rank component:

$$\begin{aligned} \min_{L,S} \quad & \frac{1}{2n}\|Y - SX\|_F^2, \\ s.t. \quad & \frac{1}{2}\|W - S\|_F^2 \le \gamma, \\ & \text{card}(S) \le c. \end{aligned} \tag{2}$$

Finding a solution to this model is equivalent to minimizing $\frac{1}{2n}\|Y - SX\|_2^2 + \frac{\lambda}{2}\|W - S\|_F^2$, where λ is a Lagrange multiplier. If we ignore the reconstruction term, the hard thresholding method can be applied to find c entries with the largest absolute values, similar to [8]. Otherwise, the iterative hard thresholding [1] method can be used to find a solution.

Dimension reduction layers, for example, the convolutional layers with kernel size 1×1 in GoogLeNet, are integrated into many state-of-the-art deep models [25, 26]. In these layers, another low-rank component is not necessary to be found. Therefore, in our method, we apply the simplified model shown above to prune unimportant weights and obtain their sparse structures.

4. Optimization

In this section, we explore and exploit the "GreBsmo", a fast SVD-free "greedy bilateral" scheme, for the proposed model. Normally, problem (1) can be solved with an alternating optimization strategy on the equivalent objective function $\frac{1}{2n}\|Y - (L+S)X\|_2^2 + \frac{\lambda}{2}\|W - L - S\|_F^2$; that is:

$$\begin{cases} L_i = \text{TruncatedGSVD}(B_i A^\dagger, r); \\ S_i = P_\Omega(M), \text{ and } M = S_{i-1} - \eta(AS_{i-1} - C_i), \end{cases} \tag{3}$$

where $A = \lambda I + \frac{1}{n}XX^\top$; $A^\dagger$ is the Moore-Penrose Pseudoinverse of A; $B_i = \lambda(W - S_{i-1}) + \frac{1}{n}(YX^\top - S_{i-1}XX^\top)$; and $C_i = \lambda(W - L_i) + \frac{1}{n}(YX^\top - L_iXX^\top)$; $\Omega : |M_{p,q\in\Omega}| \neq 0$ and $\ge |M_{p,q\in\hat{\Omega}}|, |\Omega| \le c$, and $\hat{\Omega}$ is the complement of Ω. TruncatedGSVD$(\cdot, r)$ refers to the Truncated Generalized Singular Value Decomposition in which only r left/right singular vectors with the largest singular values are calculated [28].

However, the generalized SVD involves several SVDs in each iteration, which takes a lot of time when approximating large weight matrices, which are common in deep models. Therefore, developing a fast algorithm is non-trivial.

Greedy Bilateral Decomposition (GreBdec). In this paper, to solve the low-rank and sparse decomposition with a feature map reconstruction term, the "greedy bilateral" scheme [32] is explored and exploited. This greedy scheme uses only QR decompositions, random projections, and matrix multiplications, which reduces computational complexity and is very efficient.

To develop a SVD-free algorithm, we first modify model (1) using the bilateral factorization form of L; that is, letting $L = UV$. Then we have

$$\begin{aligned} \min_{U,V,S} \quad & \frac{1}{2n}\|Y - (UV+S)X\|_F^2 + \frac{\lambda}{2}\|W - UV - S\|_F^2, \\ s.t. \quad & \text{card}(S) \le c, \end{aligned} \tag{4}$$

where $U \in \mathbf{R}^{m\times r}, V \in \mathbf{R}^{r\times k}$, and $S \in \mathbf{R}^{m\times k}$.

Alternately optimizing U, V and S yields the following updating rules:

$$\begin{cases} U_i = B_i V_{i-1}^\top (V_{i-1} A V_{i-1}^\top)^\dagger; \\ V_i = (U_i^\top U_i)^\dagger U_i^\top (B_i A^\dagger); \\ S_i = P_\Omega(M), \text{ and } M = S_{i-1} - \eta(AS_{i-1} - C_i), \end{cases} \tag{5}$$

where $B_i = \lambda(W - S_{i-1}) + \frac{1}{n}(YX^\top - S_{i-1}XX^\top)$, $C_i = \lambda(W - U_iV_i) + \frac{1}{n}(YX^\top - U_iV_iXX^\top)$. Here, A is full-rank if a proper λ is given.

Updating rules in (5) have two shortcomings. First, a lot of matrix inversions and multiplications are involved. Second, r columns (rows) need to be updated in all iterations. These issues lead to a great computational complexity.

Since only the product UV determines the objective function, to avoid the mentioned problems, we can find a pair of (U, V) that have the same product as (U_i, V_i) in (5) but can be computed more efficiently. According to (5), we have

$$U_iV_i = U_i(U_i^\top U_i)^\dagger U_i^\top (B_iA^\dagger). \tag{6}$$

This implies that the product U_iV_i equals the orthogonal projection of $B_iA^\dagger$ onto the column space of U_i. According to (5), the column space of U_i can be represented by arbitrary orthogonal bases for the columns of $BV_{i-1}^\top$ because of the full rank of matrix A. By QR decomposition, we have $B_iV^\top = QR$, then the product $U_iV_i = \mathcal{P}_Q(B_iA^\dagger) = QQ^\top(B_iA^\dagger)$, where $\mathcal{P}_Q\cdot$ denotes the orthogonal projection of a matrix on the column space of Q. Therefore, if we replace U_i and V_i with Q and $Q^\top(B_iA^\dagger)$, we get a faster updating rule; that is,

$$\begin{cases} U_i = Q, QR(B_iV^\top) = QR; \\ V_i = Q^\top(B_iA^\dagger). \end{cases} \tag{7}$$

Here updating S remains the same with that in (5). For a better result, we can repeat (7) several times before updating S.

Instead of updating r columns (rows) at all iterations, we apply the greedy method in [32] to update the U and V. We start from a small rank (for example, rank 1). Then, at each iteration, we select extra Δr rows and concatenate them into V. These Δr rows are chosen to maximize the decrease in the objective value. We have

$$\frac{\partial \mathcal{L}}{\partial UV} = AUV + B, \tag{8}$$

where $B = \lambda(W - S) + \frac{1}{n}(YX^\top - SXX^\top)$. Thus, the Δr rows are the top Δr right singular vectors of the matrix $U^\top \frac{\partial \mathcal{L}}{\partial UV}$, which can be approximated by a fast random projection $R^\top U^\top \frac{\partial \mathcal{L}}{\partial UV}$, where $R \in \mathbf{R}^{r\times \Delta r}$ is a random matrix.

This greedy selection enables a "warm start" for higher-rank optimization and ensures faster computation compared to updating r columns (rows) at all iterations, not to mention updating L by generalized SVD. We summarize our approach in Algorithm 1.

Algorithm 1 Greedy Bilateral Decomposition (GreBdec)

Input: X, W, Y, target rank r, rank step size Δr, objective function f, and power K. Initial rank r_0 and initial $V = V_0$.

Output: U, V and S.

```
while unconvergence do
    for i = 0 to K do
        Update U and V using equation (7).
    end for
    Update S using last equation in (5).
    Calculate the top Δr right singular vectors v or random projections of U^⊤ ∂L/∂UV.
    Set V := [V, v].
end while
```

5. Theoretical Analysis

Given an input feature x, we have the reconstruction error of its output feature vector y:

$$f_{U,V,S}(x, y) = \|y - (UV + S)x\|_2^2,$$

where $\hat{y} = (UV + S)x$ is the reconstructed feature vector of the compressed model. During compression, we hope to find a decomposition so that for any pair $(y, \hat{y})$ in the training and test datasets, this reconstruction error can be small, which ensures that the accuracy will not drop too much. To this end, we need to minimize the expected reconstruction error, which is defined as:

$$R(U, V, S) = \mathbb{E}_{x,y}[f_{U,V,S}(x, y)].$$

However, in practice, we cannot access the test data. The worse case is that, the training dataset, such as ImageNet, is often so large that using all training data to feed the reconstruction term is not permitted by the limited computational resources. Thus, we try to find a solution by minimizing the empirical reconstruction error on a finite number of examples; that is,

$$R_n(U, V, S) = \frac{1}{n}\sum_{i=1}^{n} f_{U,V,S}(x_i, y_i),$$

where $x_1, \cdots, x_n \in \mathbf{R}^k$ and $y_1, \cdots, y_n \in \mathbf{R}^m$ correspond to the input and output feature vectors, respectively, and n is the sample size.

If the empirical reconstruction error can quickly converge to the expected one when increasing the sample size n, then the proposed model has good generalizability. After minimizing the empirical reconstruction error, we can expect that the feature map reconstruction on the unseen data is also good. Therefore, we provide a worst case analysis for the expected reconstruction error by upper bounding the gap between the empirical and expected reconstruction error.

We denote $\mathcal{T} = \{\{U, V, S\} | U \in \mathbf{R}^{m\times r}; V \in \mathbf{R}^{r\times k}; S \in \mathbf{R}^{m\times k}, \text{card}(S) \le c; \|W - UV - S\|_F \le \gamma\}$ as the set of all possible decompositions, $\{U_n, V_n, S_n\} = \arg\min_{\{U,V,S\}\in\mathcal{T}} R_n(U, V, S)$, and $\{U^*, V^*, S^*\} = \arg\min_{\{U,V,S\}\in\mathcal{T}} R(U, V, S)$, where

$R(U^*, V^*, S^*)$ is the optimal expected reconstruction error. By minimizing the empirical reconstruction error, we try to search for a $\{U_n, V_n, S_n\}$ in $\mathcal{T}$ such that $R(U_n, V_n, S_n)$ is close to the expected reconstruction error $R(U^*, V^*, S^*)$. The proposed model is claimed to be consistent if the defect $R(U_n, V_n, S_n) - R(U^*, V^*, S^*)$ converges to zero when sufficient examples are given. Here we define the *generalization error* of this model as:

$$\sup_{\{U,V,S\}\in\mathcal{T}} |R(U, V, S) - R_n(U, V, S)|.$$

Similar to the analysis method in [17], the error bounds are derived as follows:

Theorem 1. *Suppose that the reconstruction error function $f_{U,V,S}$ for any $\{U, V, S\} \in \mathcal{T}$ has a range contained in $[0, b]$. There exist constants $c_1 \geq 1$, $\alpha \geq 0$ and $s \geq 0$ such that, for any $\delta \in (0, 1)$, with probability at least $1 - \delta$, we have*

$$\begin{aligned} &R(U_n, V_n, S_n) - R(U^*, V^*, S^*) \\ &\leq 2 \sup_{\{U,V,S\}\in\mathcal{T}} |R(U, V, S) - R_n(U, V, S)| \\ &\leq 2b\sqrt{\frac{(mk + (m + k)r) \ln (4nc_1\beta) + \ln 1/\delta}{2n}} + \frac{4}{n}, \end{aligned}$$

where $\beta = [(\sqrt{kr} + \sqrt{mr})c_1 s + \sqrt{2}cs + \sqrt{kr}\alpha c_1 + \sqrt{c}c_1^2\alpha(\sqrt{mr} + \sqrt{kr} + \sqrt{2}) + \sqrt{2}cc_1\alpha]\alpha$.

Remark 1. More discussions about the universal constants c_1, α, and s can be found in the Supplementary Material. They are introduced to bound $\{U, V, S\}$, input and output feature vectors.

Remark 2. According to Theorem 1, we upper bound the generalization error with the order of $O(\sqrt{\frac{\ln n}{n}})$. Therefore, as the sample size n increases, the gap between these two reconstruction errors converges to 0, and $\{U_n, V_n, S_n\}$ converges to $\{U^*, V^*, S^*\}$. This result implies we only require a reasonable small sample to solve the proposed model, which is supported by our empirical experiments that only 3,000 training images are employed. In practice, by minimizing the empirical reconstruction error using GreBdec, the obtained empirical reconstruction error of feature maps is often small. This indicates that the feature map reconstruction error of the unseen data in the training and test datasets is also not large according to our theorem.

6. Experiments

In this section, we compress several frequently used convolutional neural networks (CNNs). Test accuracies (Top-1 and Top-5) on ILSVRC2012 validation dataset are reported in order to evaluate the performances of deep compression methods. We first show the overall parameters and accuracies before and after compression (Table 1). Then, we provide further details about how we compress each network, and by applying low-rank and sparse structure assumptions, we reduce the size of several networks by $4.5\times$ to $15\times$, which surpasses many recent compression methods. Finally, the compression results on the convolutional layers are analyzed. Compared to representative state-of-the-art methods, our approach shows significant potential for compressing convolutional layers, which is crucial for the most advanced CNNs such as inception models [25, 26].

Table 1. Compression rates for deep networks. O: Reference network. C: Compressed network. R: Compression rate. $\#W$: Total number of weights in the networks.

Network	Top-1	Top-5	#W	R
AlexNet(O)	57.22%	80.27%	61M	10
AlexNet(C)	57.26%	80.31%	6M	
VGG-16(O)	68.50%	88.68%	138M	15
VGG-16(C)	68.75%	89.06%	9.7M	
GoogLeNet(O)	68.70%	88.90%	7M	4.5
GoogLeNet(C)	67.30%	88.11%	1.5M	

All our experiments are implemented on Caffe [13]. To implement the sparse layers, we add masks on original weight matrices to discard the parameters (resp. gradients) during the inference (resp. training) stage. Our reference models are from the Caffe Model Zoo, and all accuracies are measured without data augmentation.

In our experiments, we fixed the parameter $\eta = 10^{-3}$ and, empirically, we let $\lambda = 10^t \sigma_{\max}(\frac{1}{n}(XX^\top))$, where $\sigma_{\max}(\cdot)$ denotes the largest singular value; we tune t to obtain the best result. We randomly sampled 3,000 training images to optimize the proposed model, which is sufficient to reconstruct the feature maps with only small errors. This is in accordance with our theoretical analysis.

6.1. Analysis on the Whole Networks

AlexNet and VGG-16 on ImageNet. We first examine two popular deep networks, AlexNet [15] and VGG-16 [23]. To compress them, we follow the strategy in [8] and choose a three-phase scheme: first, we compress all the convolutional layers and retrain them with fixed parameters in the fully connected layers. Second, we do the opposite. Third, when the accuracy stops increasing, we retrain the entire model with small learning rates 10^{-5} or 10^{-6}. In the first phase, we do not need iteratively compressing and retraining as in [8]. All convolutional layers are compressed to the desired compression rates in one round.

Both AlexNet and VGG-16 have 3 fully connected layers, which occupy the most storage space. The proposed model compresses these layers as well as most state-of-the-art methods. Furthermore, our method achieves higher compression rates for the convolutional layers compared with methods like [8]. The results are shown in Table 2

Table 2. Compression statistics for AlexNet. L: Low-rank. S: Sparse. R: Compression rate.

Layer	#W	#L/#W	#S/#W	R
conv1	35K	47%	38%	85%
conv2	307K	11%	10%	21%
conv3	885K	12%	10%	22%
conv4	663K	12%	10%	22%
conv5	442K	11%	10%	21%
fc6	38M	0%	8%	8%
fc7	17M	0%	9%	9%
fc8	4M	0%	24%	24%
Total	61M	–	–	9.9% (10×)

Table 3. Compression statistics for VGG-16. L: Low-rank. S: Sparse. R: Compression rate.

Layer	#W	#L/#W	#S/#W	R
conv1_1	2K	0%	100%	100%
conv1_2	37K	10%	10%	20%
conv2_1	74K	12%	11%	23%
conv2_2	148K	11%	10%	23%
conv3_1	295K	12%	12%	24%
conv3_2	590K	11%	11%	22%
conv3_3	590K	11%	11%	22%
conv4_1	1M	12%	12%	24%
conv4_2	2M	11%	11%	22%
conv4_3	2M	11%	11%	22%
conv5_1	2M	11%	11%	22%
conv5_2	2M	11%	11%	22%
conv5_3	2M	11%	11%	22%
fc6	103M	0%	4%	4%
fc7	17M	0%	4%	4%
fc8	4M	0%	20%	20%
Total	138M	–	–	6.9% (15×)

and 3, where # represents the number of parameters and W is the original weight matrix.

GoogLeNet on ImageNet. Our method can powerfully compress traditional networks with fully connected layers, which usually dominate the model size. The most recent networks such as inception models, however, tend to replace the fully connected layer with a convolutional layer or global average pooling layer to save storage. Thus, how to compress the most common convolutional layers in these models becomes the critical problem. Testing our method on GoogLeNet, it shows that our model is good at removing redundancy in convolutional layers (Table 4). The number of parameters is reduced by a factor of $4.5\times$ with only a small decrease of accuracy. Here we first compress the convolutional layers with kernel size larger than 1 in the first round. Then, we iteratively reduce the parameters in other layers using the simplified model.

Comparison of compression methods. To verify the overall performance of our method, our algorithm is compared to network pruning method [8], low-rank tensor decomposition method [27], and Trucker decomposition method [14]. These works represent the start-of-the-art compression methods which use a single structure assumption; that is, sparsity [8] or low-rankness [27, 14]. Shown in Table 5, by appropriately applying both the low-rank and sparse structures, the proposed method achieves higher compression rates for the entire networks.

Table 4. Compression statistics for GoogLeNet. L: Low-rank. S: Sparse. R: Compression rate.

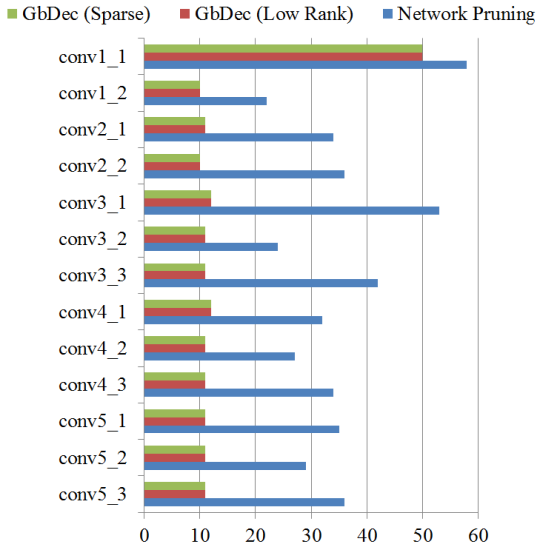
Module	#W	#L/#W	#S/#W	R
conv1_1	9K	0%	100%	100%
conv2	115K	57%	26%	83%
inception_3a	164K	8%	13%	21%
inception_3b	389K	9%	13%	22%
inception_4a	376K	6%	16%	22%
inception_4b	449K	7%	15%	22%
inception_4c	510K	8%	15%	23%
inception_4d	605K	8%	14%	22%
inception_4e	868K	8%	15%	23%
inception_5a	1M	6%	16%	22%
inception_5b	1M	7%	15%	22%
fc	1M	0%	20%	20%
Total	7M	–	–	22% (4.5×)

Table 5. Comparison of overall compression rates. O: Reference model. R: Compression rate. #W: The total number of weights in the networks.

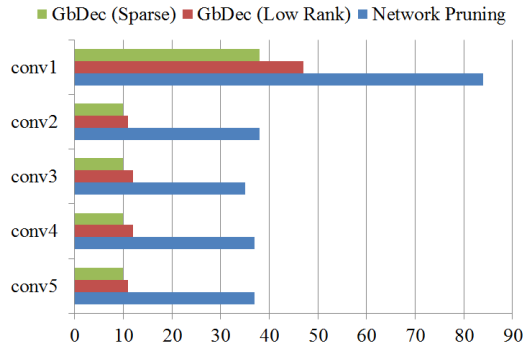
Network	Top-1	Top-5	#W	R
AlexNet (O)	57.22%	80.27%	61M	1×
Tai et al. [27]	–	79.66%	12.2M	5×
Kim et al. [14]	–	78.33%	11M	5.46×
Han et al. [8]	57.23%	80.33%	6.7M	9×
GreBdec	57.26%	80.31%	6M	**10**×
VGG-16 (O)	68.50%	88.68%	138M	1×
Tai et al. [27]	–	90.31%	50.2M	2.75×
Kim et al. [14]	–	89.40%	127M	1.09×
Han et al. [8]	68.66%	89.12%	10.3M	13×
GreBdec	68.75%	89.06%	9.7M	**15**×
GoogLeNet (O)	68.70%	88.90%	6.9M	1×
Tai et al. [27]	–	91.79%	2.4M	2.84×
Kim et al. [14]	–	88.66%	4.7M	1.28×
GreBdec	67.30%	88.11%	1.5M	**4.5**×

6.2. Analysis on the Convolutional Layers

Comparsion of compression methods. To reduce the number of parameters and exploit the spatial structure of representations, the state-of-the-art deep learning models such as GoogLeNet, Inception V3, and ResNet tend to replace the fully connected layers by some other types of layers, such as the convolutional layers. Thus, compressing the convolutional layers is crucial. By incorporating the proper structures of weight matrices, our method significantly reduce the parameters for the convolutional layers. We address this by analyzing the compression results of the convolutional layers. As shown in Fig.2, our method al-



(a) Comparison on VGG-16

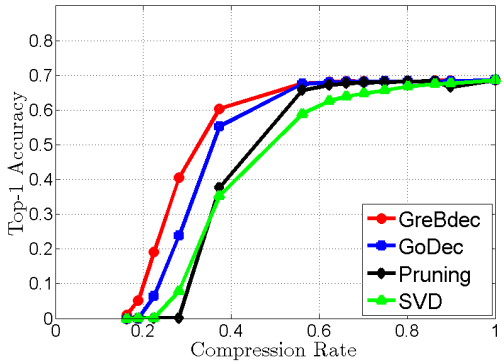


(b) Comparison on AlexNet

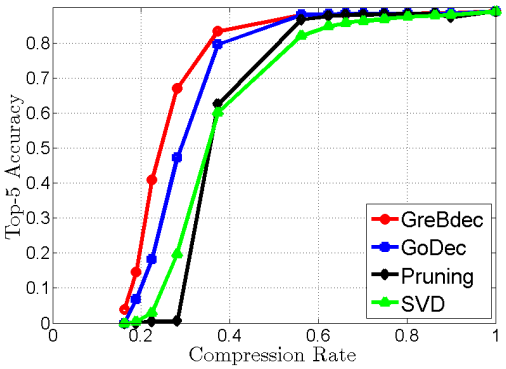
Figure 2. Comparison of compression rates on the convolutional layers on AlexNet and VGG-16. x-axis is the percentage of the total number of original weights.

most doubles the compression rates on many convolutional layers compared to the state-of-the-art pruning method [8]. The following discussion also provides some insights about the advantages of the low-rank and sparse decomposition.

Discussion. Low-rank and sparse decompositions usually result in smaller reconstruction errors of output features compared with SVD and pruning. This also motivates the idea that weight matrices in deep models can be better represented by its low-rank and sparse components, and we can compress deep models by low-rank and sparse decompositions. We verify this by evaluating the performances of compressed models prior to retraining. This experiment is based on the VGG-16 model [23]. We compress all the convolutional layers by different compression rates, and analyze the corresponding test accuracies prior to retraining are shown in Fig.3. It can be seen that low-rank and sparse decompositions ("GreBdec" and "GoDec") result in larger gains in test accuracies when the compression rate is high.



(a) Top-1 Accuracy



(b) Top-5 Accuracy

Figure 3. Testing accuracy (without retraining) comparison between the proposed method "GreBdec", "GoDec" [31], SVD and pruning [8]. x-axis is the percentage of the total number of parameters in the convolutional layers.

The effectiveness of the feature map reconstruction term is also highlighted by the higher accuracies of our model compared to "GoDec".

7. Conclusion

Here we propose a unified deep compression framework that decomposes weight matrices into their low-rank and sparse components. Compared to traditional SVDs and pruning methods, the proposed model significantly improves the performance prior to retraining, especially when feature map reconstructions are integrated into the framework. This high performance provides a better initialization for the subsequent retraining, which helps the proposed model to achieve high compression rates without loss of accuracy for many popular models. We can save at most $15\times$ storage space, which beats many recent methods using a single strategy.

Acknowledgement

This research was supported by Australian Research Council Projects FT-130101457, DP-140102164, LP-150100671.

References

[1] S. Bahmani, B. Raj, and P. T. Boufounos. Greedy sparsity-constrained optimization. *Journal of Machine Learning Research*, 14(Mar):807–841, 2013. 4

[2] W. Chen, J. T. Wilson, S. Tyree, K. Q. Weinberger, and Y. Chen. Compressing neural networks with the hashing trick. 2015. 3

[3] M. Denil, B. Shakibi, L. Dinh, N. de Freitas, et al. Predicting parameters in deep learning. In *Advances in Neural Information Processing Systems (NIPS)*, pages 2148–2156, 2013. 1, 3

[4] E. L. Denton, W. Zaremba, J. Bruna, Y. LeCun, and R. Fergus. Exploiting linear structure within convolutional networks for efficient evaluation. In *Advances in Neural Information Processing Systems (NIPS)*, pages 1269–1277, 2014. 1, 3

[5] G. H. Golub and C. F. Van Loan. *Matrix computations*, volume 3. JHU Press, 2012. 2

[6] Y. Gong, L. Liu, M. Yang, and L. Bourdev. Compressing deep convolutional networks using vector quantization. *arXiv preprint arXiv:1412.6115*, 2014. 3

[7] S. Han, H. Mao, and W. J. Dally. Deep compression: Compressing deep neural network with pruning, trained quantization and huffman coding. *International Conference on Learning Representations (ICLR)*, 2016. 3

[8] S. Han, J. Pool, J. Tran, and W. Dally. Learning both weights and connections for efficient neural network. In *Advances in Neural Information Processing Systems (NIPS)*, pages 1135–1143, 2015. 1, 2, 4, 6, 7, 8

[9] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. *arXiv preprint arXiv:1512.03385*, 2015. 1

[10] T. He, Y. Fan, Y. Qian, T. Tan, and K. Yu. Reshaping deep neural network for fast decoding by node-pruning. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 245–249. IEEE, 2014. 1

[11] G. Hinton, O. Vinyals, and J. Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015. 3

[12] M. Jaderberg, A. Vedaldi, and A. Zisserman. Speeding up convolutional neural networks with low rank expansions. *arXiv preprint arXiv:1405.3866*, 2014. 3

[13] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell. Caffe: Convolutional architecture for fast feature embedding. In *The 22nd ACM international conference on Multimedia (ACMMM)*, pages 675–678. ACM, 2014. 6

[14] Y.-D. Kim, E. Park, S. Yoo, T. Choi, L. Yang, and D. Shin. Compression of deep convolutional neural networks for fast and low power mobile applications. *arXiv preprint arXiv:1511.06530*, 2015. 1, 7

[15] A. Krizhevsky, I. Sutskever, and G. E. Hinton. ImageNet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems (NIPS)*, pages 1097–1105, 2012. 1, 6

[16] Y. LeCun, J. S. Denker, S. A. Solla, R. E. Howard, and L. D. Jackel. Optimal brain damage. In *Advances in Neural Information Processing Systems (NIPS)*, volume 2, pages 598–605, 1989. 1

[17] T. Liu, D. Tao, and D. Xu. Dimensionality-dependent generalization bounds for k-dimensional coding schemes. *Neural Computation*, 28(10):2213–2249, 2016. 6

[18] P. Luo, Z. Zhu, Z. Liu, X. Wang, and X. Tang. Face model compression by distilling knowledge from neurons. In *Thirtieth AAAI Conference on Artificial Intelligence (AAAI)*, 2016. 3

[19] Z. Mariet and S. Sra. Diversity networks. *arXiv preprint arXiv:1511.05077*, 2015. 1

[20] M. Rastegari, V. Ordonez, J. Redmon, and A. Farhadi. XNOR-Net: ImageNet classification using binary convolutional neural networks. *arXiv preprint arXiv:1603.05279*, 2016. 3

[21] S. Ren, K. He, R. Girshick, and J. Sun. Faster R-CNN: Towards real-time object detection with region proposal networks. In *Advances in Neural Information Processing Systems (NIPS)*, pages 91–99, 2015. 1

[22] A. Romero, N. Ballas, S. E. Kahou, A. Chassang, C. Gatta, and Y. Bengio. FitNets: Hints for thin deep nets. *arXiv preprint arXiv:1412.6550*, 2014. 3

[23] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *International Conference on Learning Representations (ICLR)*, 2015. 1, 6, 8

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

[25] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1–9, 2015. 1, 4, 6

[26] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna. Rethinking the inception architecture for computer vision. *arXiv preprint arXiv:1512.00567*, 2015. 4, 6

[27] C. Tai, T. Xiao, X. Wang, et al. Convolutional neural networks with low-rank regularization. *arXiv preprint arXiv:1511.06067*, 2015. 1, 7

[28] Y. Takane and M. A. Hunter. Constrained principal component analysis: a comprehensive theory. *Applicable Algebra in Engineering, Communication and Computing*, 12(5):391–419, 2001. 2, 4

[29] Y. Wang, C. Xu, S. You, D. Tao, and C. Xu. CNNpack: Packing convolutional neural networks in the frequency domain. In *Advances In Neural Information Processing Systems (NIPS)*, pages 253–261, 2016. 3

[30] X. Zhang, J. Zou, K. He, and J. Sun. Accelerating very deep convolutional networks for classification and detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 2015. 1, 3

[31] T. Zhou and D. Tao. Godec: Randomized low-rank & sparse matrix decomposition in noisy case. In *The 28th Interna-*

tional Conference on Machine Learning (ICML), pages 33–40, 2011. 3, 8

[32] T. Zhou and D. Tao. Greedy bilateral sketch, completion & smoothing. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 650–658, 2013. 2, 3, 4, 5