This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TPAMI.2018.2874634, IEEE Transactions on Pattern Analysis and Machine Intelligence

Shallowing Deep Networks: Layer-wise Pruning based on Feature Representations

Shi Chen, Qi Zhao, *Member, IEEE*

Abstract—Recent surge of Convolutional Neural Networks (CNNs) has brought successes among various applications. However, these successes are accompanied by a significant increase in computational cost and the demand for computational resources, which critically hampers the utilization of complex CNNs on devices with limited computational power. In this work, we propose a feature representation based layer-wise pruning method that aims at reducing complex CNNs to more compact ones with equivalent performance. Different from previous parameter pruning methods that conduct connection-wise or filter-wise pruning based on weight information, our method determines redundant parameters by investigating the features learned in the convolutional layers and the pruning process is operated at a layer level. Experiments demonstrate that the proposed method is able to significantly reduce computational cost and the pruned models achieve equivalent or even better performance compared to the original models on various datasets.

Index Terms—Model Pruning, Compact Design, Convolutional Neural Networks.

1 Introduction

In recent years, Convolutional Neural Networks (CNNs) have been widely applied to various tasks including Image Classification [1, 2, 3, 4], Object Detection [5, 6], Semantic Segmentation [7, 8] and Image Captioning [9]. While these CNN based models are capable of achieving state-of-the-art performance, deploying them usually requires intensive computational resources. For example, AlexNet [1] with 8 layers and over 6×10^7 parameters requires approximately 7.3×10^8 FLOPs [1] for a single inference. High demand on computational power prohibits the utilization of such models on mobile devices and even most of the PCs, making them impractical for many domains. Furthermore, as the models grow much deeper to achieve better results (from 8-layer AlexNet to 152-layer ResNet [4]), the significantly increased computational cost becomes even more prohibitive for model deployment. Therefore, for wide applicability of CNN models especially in resource-limited scenarios, model compression and acceleration is essential.

To save computational cost of CNNs, several works propose to reduce network size via connection-wise pruning [10, 11], filter-wise pruning [12] and low-rank approximation [13]. Connection-wise pruning methods work by pruning connections with relatively small magnitudes of weights, however, it usually leads to non-structured connectivity in the pruned networks which can result in difficulties in acceleration on hardware platform. Furthermore, most of the reduction is achieved at fully-connected layers instead of convolutional layers that require more computational resources. Filter-wise pruning methods estimate the absolute sum of weights and remove filters together with their connections. Since the pruning process is operated on a filter level, the number of pruned parameters is relatively restricted. Low-rank approximation methods perform acceleration on CNNs by coordinating parameters in dense matrices and approximating the convolutional operations. The approximation is conducted within each layer while other layers are fixed during retraining. This process is iterative thus costly to obtain the optimal weight approximation.

Inspired by the observations from [14] which suggests that the effects and dynamics of different intermediate layers can be studied via estimating their corresponding feature representations with linear classifiers, we propose a feature representation based parameter pruning method that reduces CNNs by removing layers with small improvement on feature representations. Different from the aforementioned works, our method does not explicitly investigate the weights within the models but directly analyzes the features learned at different convolutional layers. Since the pruning procedure is conducted in a layer-wise fashion, more parameters can be pruned from the original models compared to [12], and no sparse connectivity is introduced. Besides, to boost the performance of the pruned models and make use of knowledge from the original models, we utilize knowledge distillation [15] for transferring knowledge from previous models to the current ones. Experimental results on different datasets show that our method is able to significantly reduce the number of parameters in convolutional layers and achieve comparable performance as original models with more compact networks. The main contributions of this paper can be summarized as follows:

1) We propose a layer-wise pruning method that identify and remove redundant convolutional layers within deep neural networks.

2) Different from previous methods which focus on investigating weight information to identify redundancy, our method analyzes the feature representations computed at different layers and locates those layers which provide minor contributions on boosting the performance of features.

3) To compensate the loss of performance caused by pruning, we introduce a knowledge transfer mechanism for parameter pruning that adopts information from the

1. number of float point operations

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TPAMI.2018.2874634, IEEE Transactions on Pattern Analysis and Machine Intelligence

original models via retraining with a knowledge distillation technique.

4) We studied the effects of various settings on layer-wise pruning method and analyzed the corresponding trade-off between performance and computational cost. Experimental results on two tasks and various datasets demonstrate that our method is able to significantly reduce the computational cost of deep neural networks while achieving comparable or even better performance compared to the original models.

2 Related Works

2.1 Parameter Pruning

In order to accelerate Deep Neural Networks and reduce the computational cost, various methods have been proposed on pruning redundancy within the networks and designing compact model.

Connection-wise Pruning. The computational cost of CNNs can be reduced by removing certain connections with minor contributions at different layers. In [10, 11], Han *et al.* introduce a connection-wise parameter pruning method that prunes the weights with small magnitudes and retrains the pruned model without hurting the overall performance. While the method is able to reduce the number of parameters in AlexNet [1] by 9× and VGG-16 [2] by 13×, most of the reduction is achieved at fully-connected layers and no significant reduction is observed among convolutional layers. Since convolutional layers usually lead to higher computational cost compared to fully-connected layers and many recent CNNs such as ResNet [4] are built with fewer fully-connected layers, it becomes crucial to reducing the computational cost in convolutional layers.

Filter-wise Pruning. Li *et al.* [12] propose a filter-wise pruning method that removes less effective filters in convolutional layers. For a specific layer, the method first computes the L1-norm of kernel weights for all of the filters within the layer and then sorts them based on corresponding values. Filters with smallest weights are pruned from the layer and the interactions between adjacent layers are reconstructed accordingly. Since the pruning process is operated at the filter level, the number of reduced parameters is relatively limited compared to layer-wise pruning.

Low-rank Approximation. A layer in the CNN model, either convolutional or fully-connected, can be decomposed into several layers with a smaller amount of parameters for more efficient computation and storage. Jaderberg *et al.* [16] propose two strategies for reducing the computational cost of CNNs by exploiting the cross-channel or filter redundancy and reconstructing each convolutional layer with two simpler layers with less computational cost. In [13], the convolutional layer is approximated with the sum of multiple rank-1 tensors while the fully-connected layer is approximated by the multiplication of two low-rank matrices. Wang *et al.* [17] adopt the tensor block-term decomposition method for speeding up the computation within convolutional layers.

Low-precision Representation. In most of the current Deep Learning platforms, the parameters within CNNs are represented using 32-bit float-point numbers, which leads to considerable amount of computation and storage for a complex model. In order to reduce the computational overheads and requirement on storage, several works adopt low-precision representations to construct the CNN models. In [18], a 16-bit fixed-point representation is utilized to train the Deep Neural Networks (DNNs) without significant loss of performance. Courbariaux *et al.* [19] present a binarized CNN that converts the multiplications to additions via binarizing the model parameters during both backward and forward propagation.

Compact Network. Instead of leveraging complex models to achieve satisfying performance, Iandola *et al.* [20] propose a compact CNN model that is capable of achieving comparable results as the AlexNet but reducing the number of parameters by 50 times.

Compared to aforementioned works, the proposed method focuses on reducing computational cost among convolutional layers and conducts parameter pruning in a layer-wise fashion. Moreover, our method is compatible with current Deep Learning platforms and does not require any changes on the general framework of CNNs.

2.2 Linear Classifier Probes

Alain *et al.* [14] propose a diagnosis method for analyzing intermediate layers in Deep Neural Networks (DNNs). By training a set of linear classifiers on features extracted from the intermediate layers of a DNN and estimating their performance on validation data, [14] aims at understanding the roles and dynamics of various layers within the network. Inspired by this work, in order to study the effects of different convolutional layers and locate layers with a small improvement on feature representations, we use linear classifiers consisting of a single fully-connected layer to analyze the corresponding role of each convolutional layer on lifting the performance of features.

2.3 Knowledge Distillation

Hinton *et al.* [15] propose knowledge distillation for training an efficient ensemble model with a set of specialist models. Instead of averaging the predictions of different models, knowledge distillation compresses the knowledge in an ensemble via introducing an additional term in the objective function. By training on an objective function with soft targets constructed on logits extracted from specialist models, the ensemble model is able to absorb knowledge transferred from specialist models. In this work, while our goal differs from the original objective of knowledge distillation, we utilize the knowledge transfer mechanism from knowledge distillation and improve the performance of a pruned model via absorbing knowledge from the original model.

3 Feature Representation based Parameter Pruning

We focus on pruning convolutional layers with relatively small contributions to increasing the performance of feature representations. We then construct compact networks without the pruned layers, which achieves equivalent performance as the original complex models with transfer learning. Section 3.1 presents our layer-wise pruning method which locates less effective layers by diagnosing features computed at different convolutional layers. In section 3.2,

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TPAMI.2018.2874634, IEEE Transactions on Pattern Analysis and Machine Intelligence

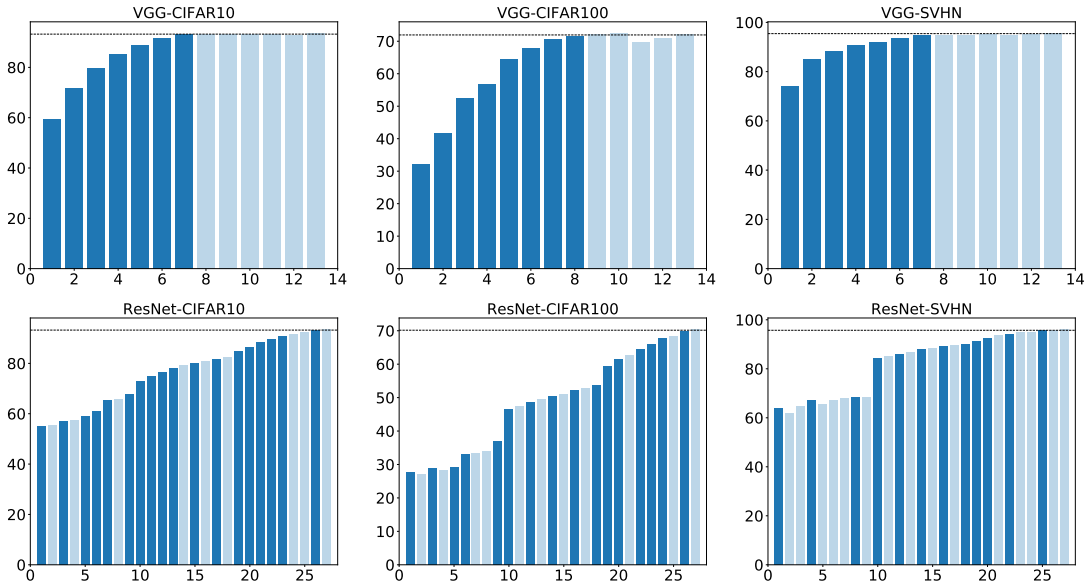

Fig. 1. Linear classifier probes on VGG-16 (first row) and ResNet-56 (second row). Blue bars are validation accuracy of the linear classifiers trained using features from intermediate layers, black dashed lines are the validation accuracy of the CNNs. The bars with transparent color represent the layers considered to have less contributions based on the predefined threshold.

we propose a retraining strategy that utilizes the knowledge distillation technique. We demonstrate the proposed method with two typical and state-of-the-art networks, *i.e.*, VGG-16 that consists of 13 convolutional layers which are connected in a feed-forward manner and residual networks (ResNet-56, ResNet-101) that connect their convolutional via both feed-forward connections and skip connections with identity mapping. Note that for ResNet, in order to preserve the characteristics of the network, we consider each residual block with several layers as an independent layer.

3.1 Layer-wise Pruning via Feature Diagnosis

CNNs, containing both feature extractor and classifier, learn reasonable feature representations specific to the current task with corresponding training data. While the features from the last convolutional layer of a CNN tend to provide the best discriminative power and are commonly utilized in various application as the input features, features from intermediate layers also contain important information related to the tasks and can be used to analyze the behaviors of the corresponding CNNs. In [14], the authors propose a method called Linear Classifier Probe to gain understandings on the behaviors of a DNN. Specifically, it trains a set of linear classifiers on features extracted at different layers within the network to explore the roles and dynamics of intermediate layers.

Inspired by the observations in [14], in this work we use a single fully-connected layer as the linear classifier to evaluate the effectiveness of a layer thus finding the ones to be pruned. By comparing the performance of classifiers trained on features computed at adjacent layers, layers that have minor improvement on the feature representations are identified based on a predefined threshold. As an example, in Figure 1 we visualize the results of feature diagnosis on various datasets with different models, *i.e.*, VGG-16 and ResNet-56 on CIFAR-10 [21], CIFAR-100 [21] and SVHN [22] datasets. Layers that provide insufficient contributions to feature representations are labeled with transparent color.

With the predefined threshold being set as 1.5% of the performance of original model (default threshold in our experiments, a discussion can be found in section 4.5), on CIFAR-10 and SVHN nearly half of the convolutional layers in ResNet-56 have limited contributions on improving the performance of feature representations. Furthermore, in all of the datasets, saturation is observed at the last several layers of VGG-16, indicating that these layers are not necessary here since the partial model with the first several layers are already capable of computing discriminative features for the task, *i.e.*, image classification.

Based on the above observations on the diagnosis results, we conduct the pruning process by directly removing the layers that provide minor influences on improving the feature representations. While pruning VGG-16 is straightforward as removing the sub-sequential layers does not affect the interactions within the previous layers, layer pruning on ResNet-56 is less intuitive due to the destruction of correlations between adjacent layers. However, according to [23], in residual networks, essential information can be well preserved via the skip connections and removing certain layers from the networks only results in minor influences instead of destructive impacts on the overall performance. Therefore, we argue that despite the destruction of interactions between adjacent layers, with proper retraining, the pruned residual network is still able to reconstruct the

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TPAMI.2018.2874634, IEEE Transactions on Pattern Analysis and Machine Intelligence

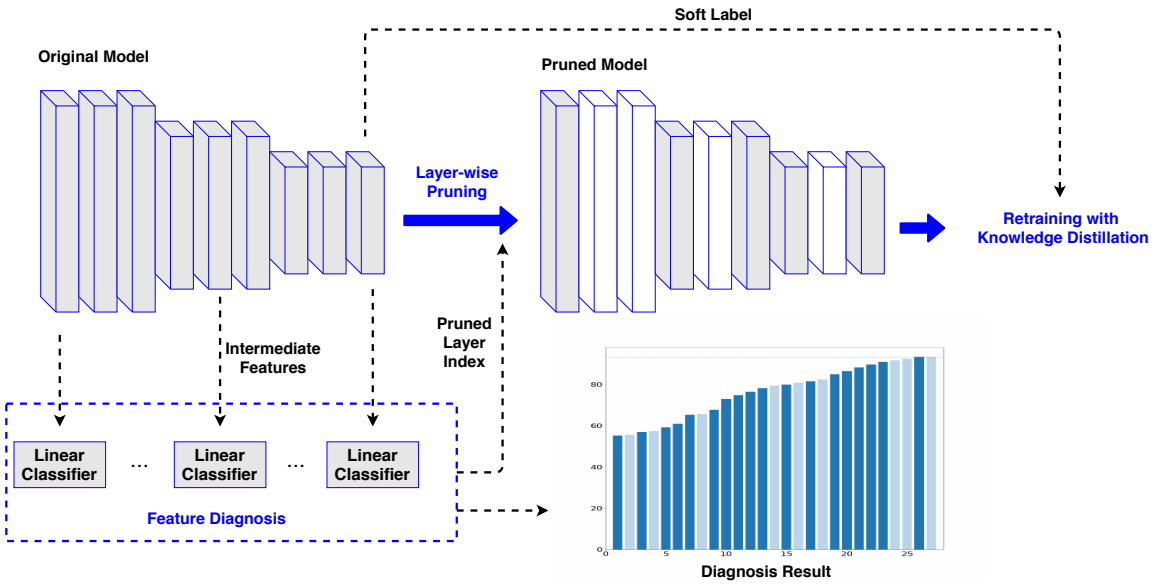

Fig. 2. Procedure of proposed layer-wise pruning.

interactions between its layers and compensate the loss of the performance. We demonstrate this claim via empirical results shown in Section 4.

3.2 Knowledge Transfer via Distillation

So far, we have obtained networks with a more compact architecture by reconstructing deep networks. While retraining the networks with pretrained weights until convergence is able to achieve reasonable results, the pruned model may not perform as well as the original network due to the modifications on the architecture. To compensate the performance loss with network reconstruction, we adopt knowledge distillation [15] to transfer knowledge from the original model to the pruned model for boosting its performance.

The principal idea of using knowledge distillation on parameter pruning is to introduce an additional term into the objective function and encourage the pruned model to mimic the predictions of the original model which is well trained previously. By constructing the distillation term based on the logits from the original model and the current one, the retraining objective becomes to generate similar predictions as the original model at a logit level while at the same time respecting the ground truth labels of the training samples. By utilizing knowledge distillation, we form a teacher-student network, where the original model as the teacher provides logit information for each training sample and the pruned model as the student learns from the teacher and ground truth labels simultaneously. For a general multi-class classification problem with softmax activation at the output layer, the objective function with knowledge distillation can be represented as follows:

$$L(y, \hat{y}) = -\sum_i y_i log(\hat{y}_i) - \alpha \cdot \sum_i p_i log(q_i) \qquad (1)$$

Here y_i and $\hat{y}_i$ are the ground truth probability and predicted probability of the i_{th} class, α is a balanced factor for controlling contributions between two terms. Instead of utilizing the hard targets, the regularization term for knowledge distillation is constructed based on soft targets which can be denoted as:

$$q_i = \frac{exp(z_i/T)}{\sum_j exp(z_j/T)} \quad \text{and} \quad p_i = \frac{exp(v_i/T)}{\sum_j exp(v_j/T)} \qquad (2)$$

where T is a hyper-parameter called temperature, z and v are logits computed by the pruned model and the original model respectively.

Suppose that we denote C as the second term in Equation 1 without the balanced factor α, the derivative of knowledge distillation with respect to logits can be computed as follows:

$$\frac{\partial C}{\partial z_i} = \frac{1}{T}(q_i - p_i) = \frac{1}{T}(\frac{exp(z_i/T)}{\sum_j exp(z_j/T)} - \frac{exp(v_i/T)}{\sum_j exp(v_j/T)}) \qquad (3)$$

According to the fact that $1 + x \leq exp(x)$, under high temperature environment the above derivative can be approximated as follows:

$$\frac{\partial C}{\partial z_i} \approx \frac{1}{T}(\frac{1 + z_i/T}{N + \sum_j z_j/T} - \frac{1 + v_i/T}{N + \sum_j v_j/T}) \qquad (4)$$

If we assume that the logits have zero mean for each sample, *i.e.* $\sum_j z_j = \sum_j v_j = 0$, the above function can be further simplified as:

$$\frac{\partial C}{\partial z_i} \approx \frac{1}{NT^2}(z_i - v_i) \qquad (5)$$

Therefore, the mechanism of knowledge distillation can be approximated as matching the logits between the original model and the pruned model that needs to be retrained, which can be implemented by introducing a companion objective before the softmax activation function. In this work, we empirically set the hyper-parameter $\frac{\alpha}{NT^2}$ in the knowledge distillation term as 0.03 during all of the experiments. The overall pruning process with knowledge distillation is shown in Figure 2.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TPAMI.2018.2874634, IEEE Transactions on Pattern Analysis and Machine Intelligence

	Original	Pruned	Pruned-KD	Pruned-R	Pruned-S	Parameter	FLOP
VGG-CIFAR10	**93.50**	93.40	93.47	-	93.00	87.9	38.9
VGG-CIFAR100	72.38	73.25	**73.43**	-	72.65	78.9	32.9
VGG-SVHN	**96.50**	95.98	96.36	-	95.93	87.9	38.9
ResNet-CIFAR10	93.03	93.09	**93.29**	92.86	92.32	42.3	34.8
ResNet-CIFAR100	**70.01**	69.77	69.78	69.01	67.89	36.1	38.3
ResNet-SVHN	96.70	96.49	**96.75**	96.43	96.46	54.6	52.2
VGG-CIFAR10 [12]	93.25	93.40	-	-	-	64.0	34.2
ResNet-CIFAR10 [12]	93.04	93.06	-	-	-	13.7	27.6

TABLE 1
Experimental results on the CIFAR10, CIFAR100 and SVHN datasets using VGG16 and ResNet-56. All of the scores are in percentage. The best scores for each dataset with a specific model are highlighted via bold text. Results above the horizontal line are our contributions.

4 EXPERIMENTS

To demonstrate the effectiveness of our parameter pruning method, we conduct experiments on the CIFAR-10, CIFAR-100, SVHN datasets for single-label classification and the MSCOCO [24] dataset for multi-label classification. Three different models are adopted for evaluation, including VGG-16, ResNet-56 and ResNet-101. We analyze the performance of models pruned under different settings, *i.e.*, the model pruned via the proposed method without using knowledge distillation (Pruned), the model pruned with the proposed method and knowledge distillation (Pruned-KD), the model that is randomly pruned but maintains the same amount of layers as the model pruned by our method (Pruned-R), the model pruned by the proposed method but trained from scratch without using the pretrained weights (Pruned-S). To quantify the reduction of computational cost, we select two popular evaluation metrics to estimate the efficiency of models, including the number of parameters (Parameter) and the number of float point operations (FLOP).

4.1 Implementation

Architecture and Objective Function. In all of our experiments, we utilize the model architectures proposed in the original papers except that: 1) For VGG-16, we adopt a slightly modified version from [12], which contains only 1 fully-connected layer. 2) For ResNet-101 on multi-label classification, instead of only predicting the existence for each MSCOCO category, we predict the two probabilities for both the existence and non-existence of each category (note that softmax activation is applied on the output for each category independently). As for the objective functions, for single-label classification we use the standard cross-entropy loss while for multi-label classification we use the multi-label binary cross-entropy loss.

Data Augmentation and Data Split. We adopt the same data augmentation techniques as [4] for model training: 4 pixels are padded on each side and a 32×32 crop is randomly sampled from the padded image or its horizontal flip. Besides, for the SVHN dataset we remove the flipping to preserve the structure of digits (a flipped '5' is not actually a digit), and for MSCOCO multi-label classification we do not use any data augmentation besides channel-wise normalization in order to utilize the pretrained weights trained on ImageNet [25] classification. To obtain the training set for model training, validation set for model selection and feature diagnosis, and test set for model evaluation, for CIFAR-10, CIFAR-100 and SVHN[2] we split the original training set into training and validation sets using the ratio of 0.9 to 0.1, and use the original test set for evaluation. For MSCOCO multi-label classification, due to the lack of the official test set, we follow the training and evaluation process from [26, 27, 28]: training on the complete training set and evaluating on the validation set.

Training. We use Stochastic Gradient Descent to train all models, where the batch sizes are set to be 128 for single-label classification with VGG-16 and ResNet-56, 50 for multi-label classification with ResNet-101. For the learning schedule of training original models, we use the following settings: 1) for ResNet-56 and VGG-16, we use the same schedule from [4], *i.e.*, learning rate is initialized as 0.1 and divided by 10 at the 32k and 48k iterations, and training is terminated at 64k iterations, 2) for ResNet-101, we use constant learning rate 2×10^{-3}. The training is terminated at 5 epochs and the best model is selected via performance on the validation set. The learning schedule for retraining the pruned models is the same as training the original models, except that for VGG-16 we initialize the learning rate as 0.01 instead of 0.1 for fast convergence at the beginning.

4.2 Result on Single-label Classification

In this section, we demonstrate the effectiveness of the proposed layer-wise pruning method for single-label classification, with experiments on the CIFAR-10, CIFAR-100, SVHN datasets using VGG-16 and ResNet-56. The pruned layers are highlighted in Figure 1 via the visualization of feature diagnosis, and in Table 1, we compare the test accuracy of the original models with models pruned under different settings. In addition to the model performance, we also show the ***reduction*** of computational cost in terms of model parameters and float point operations. For reference, we also report the pruning results of [12] with the same models.

Table 1 shows comparative results on CIFAR-10, CIFAR-100 and SVHN datasets. We see that the proposed pruning method is able to significantly reduce the computational cost and achieve equivalent or even better performance compared to the original deep models. For example, on CIFAR-100 with VGG-16, despite more than 87% parameters being pruned from the network, the model constructed with the proposed method is still able to achieve comparable result with nearly no loss of performance. On CIFAR-10 with

2. We do not use the extra training set from SVHN.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TPAMI.2018.2874634, IEEE Transactions on Pattern Analysis and Machine Intelligence

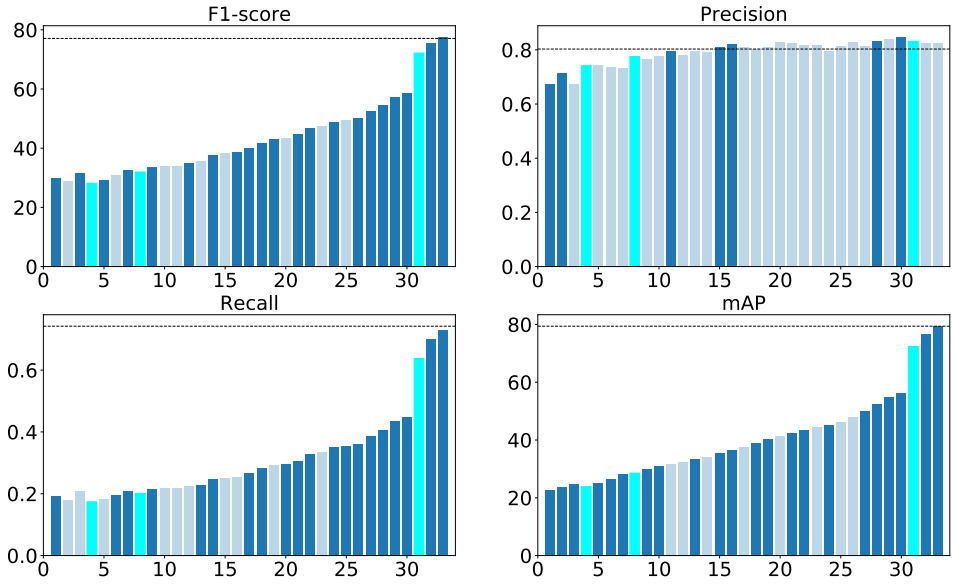

Fig. 3. Feature diagnosis results in terms of different metrics for MSCOCO multi-label classification. The x-axis denotes the layer (block) index while the y-axis represents the score for respective metrics. Black dotted lines indicate the performance of the original model. The layers to be pruned determined by each corresponding metric are highlighted with transparent colors. Layers with cyan color change the dimensionality (*i.e.*, number of filters) of features.

ResNet-56, our method is capable of achieving better results than the original model even though 42.3% of parameters have been removed. These observations demonstrate the effectiveness of the proposed layer-wise pruning method, *i.e.*, by removing the redundant layers based on feature representations and retraining the pruned model with proper settings, we are able to construct compact models that are powerful in both accuracy and efficiency in terms of computational demand. Furthermore, the comparison between our method and the method proposed in [12] also shows the advantages of the proposed method in both computation reduction and model performance.

By comparing the performance of the pruned models trained under different settings (third and fourth columns in Table 1), we can see the relative improvements achieved via knowledge distillation and the effectiveness of feature diagnosis. With the utilization of knowledge distillation, we are able to increase the performance of the pruned models on various datasets. Compared to the randomly pruned models with the same amount of parameters, models pruned based on the feature representations at corresponding layers can achieve significantly better performance, indicating that feature diagnosis plays an essential role on supervising the layer-wise pruning. Note that for VGG-16, since there is no way to randomly prune the models with the same amount of parameters and at the same time maintaining the interactions between layers, we do not include the results of randomly pruned models.

4.3 Result on Multi-label Classification

To further demonstrate the effectiveness of proposed layer-wise pruning and show its generalizability, next we conduct experiments on the more challenging MSCOCO multi-label classification task. In Figure 3, we visualize the feature diagnosis on ResNet-101.

	Precision	Recall	F1-score	mAP
Original	80.3	74.2	77.1	79.4
Pruned	80.6	71.4	75.7	78.6
Pruned-KD	81.6	72.5	76.8	78.9
Pruned-R	81.2	71.3	76.2	78.1
Pruned-S	81.1	68.8	74.5	75.7

TABLE 2
Experimental results on MSCOCO multi-label classification task with ResNet-101, pruning is determined by the performance on mAP. The reduction on Parameter and FLOP are 16.8% and 19.09% respectively.

From Figure 3, we observe that the multi-label classification model tends to have different characteristics compared to the models for single-label classification in our experiments. Specifically, 1) while the performance of feature representations increases in a relatively smooth manner on single-label classification, for multi-label classification, large performance gaps are observed in the last three layers of the network. This is likely because of the high demand on model capacity to process the data: for the complex data with abundant semantic attributes from MSCOCO, only with sufficient amount of trainable parameters (for ResNet-101, the last three layers contains more than 40% model parameters) can the performance of feature representations be significantly improved, 2) unlike single-label classification where significant improvements are observed at layers with dimesionality change, in multi-label classification these layers may have minor or even negative effects on boosting

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TPAMI.2018.2874634, IEEE Transactions on Pattern Analysis and Machine Intelligence

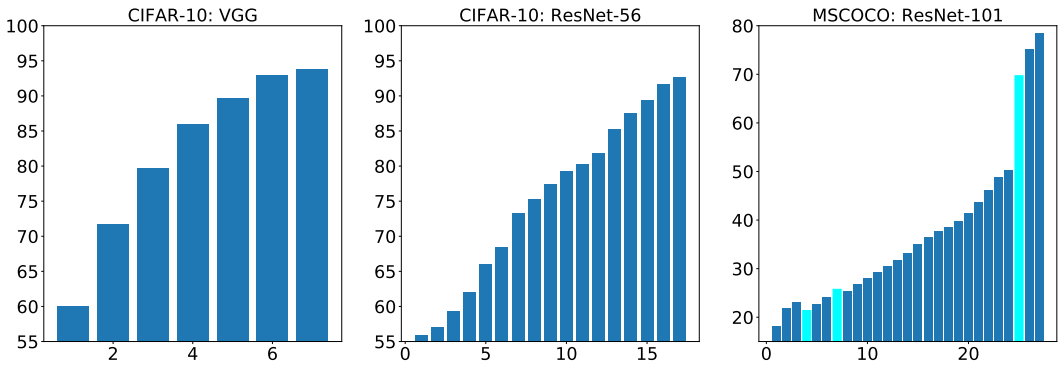

Fig. 4. Visualization of feature diagnosis on models after layer-wise pruning. Cyan color indicates the unpruned layers with dimensionality changes.

the performance of features. This could be a result of co-adaptation, *i.e.*, several adjacent layers including the one with dimensionality change corporate with each other and together they compensate the loss caused by sub-sampling and learning reasonable feature representations. Therefore, without interacting with sub-sequential layers, the layers with dimensionality changes themselves fail to provide sufficient improvement on the feature representations. While these layers can not be pruned due to mismatch on the number of filters after pruning and thus left unpruned, since there are only a few of them within modern deep neural networks, proposed layer-wise pruning still has sufficient flexibility to reduce computational cost on various models in spite of the aforementioned limitations.

Since multiple metrics are utilized for evaluating model performance of multi-label classification, including precision, recall, F1-score and mean Average Precision (mAP), we highlight the redundant layers with respect to each of the metrics. We then empirically select mAP as the metric to determine the layers to be pruned since it can well balance the trade off between computational reduction and model performance. Pruning based on F1-score is capable of reducing more layers but tends to result in significant loss of performance (2.5%). Precision and Recall are highly correlated with F1-score and thus not utilized. Table 2 reports the comparative results on proposed layer-wise pruning method based on mAP.

According to the quantitative results in table 2, model pruned by our method achieves equivalent results with respect to the original model (0.3% loss of performance) with a reduction by around 19% in computational cost. Furthermore, when comparing the performance of the model with and without knowledge distillation, the model pruned based on feature representation, and the randomly pruned model, we can see that knowledge distillation and feature diagnosis are able to consistently improve the overall performance. Based on the aforementioned experimental results, it is reasonable to conclude that our layer-wise pruning method is capable of constructing computationally efficient yet powerful network via pruning out redundant layers within a complex network and is generalized to various types of CNN models.

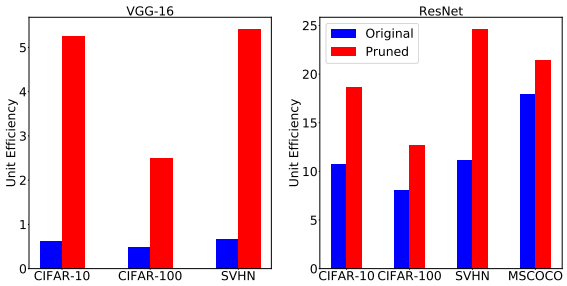

Fig. 5. Unit efficiency for various models on different datasets. Note that for ResNet, ResNet-101 is used on the MSCOCO dataset while ResNet-56 is used for experiments on other datasets.

4.4 Analysis of Model Efficiency

To evaluate the efficiency of models constructed based on our parameter pruning method, we first conduct feature diagnosis on the models after layer-wise pruning. As an example, Figure 4 shows the performance of feature representations for VGG-16, ResNet-56 on CIFAR-10 and ResNet-101 on MSCOCO. As the results show, compared to the original models in Figure 1 and Figure 3, the pruned networks evolve the features in a more efficient fashion with all of their layers providing sufficient contributions (larger than the predefined threshold) on improving the performance of features.

We then visualize the *unit efficiency*, defined as model performance per 100K parameters, in Figure 5 to show the models' efficiency on utilizing their parameters. Note that for visualization purpose, we enlarge the unit efficiency by 100 times for MSCOCO multi-label classification. For single-label classification, accuracy is utilized as the reference while for multi-label classification, mAP is used. As shown in Figure 5, since the proposed method only leads to minor or even no performance loss, the unit efficiency increases significantly across different datasets. For example, on the CIFAR-10 dataset we achieve 74% and 833% increase of unit efficiency using ResNet-56 and VGG-16 respectively. Despite the high demand on model capacity, our pruning method is still able to improve the unit efficiency by 17% on MSCOCO multi-label classification task.

	Accuracy(%)				Parameter Pruned(%)			
	0%	0.75%	1.5%	3%	0%	0.75%	1.5%	3%
CIFAR-10	93.03	93.46	93.29	92.61	0	10.2	42.3	48.7
CIFAR-100	70.01	70.49	69.78	69.16	0	11.7	36.1	50.0
SVHN	96.70	96.88	96.75	96.29	0	33.6	54.6	76.7

TABLE 3
Comparison between original ResNet-56 and models pruned under different thresholds.

4.5 Effects of Threshold Selection

In this paper, we present a layer-wise pruning method that improves the computational efficiency of a CNN by removing layers that fail to provide sufficient contributions on improving feature representations. While 1.5% of the performance of original model is used as the default parameter within our experiments, in this section we focus on exploring the influences of various values for this threshold. More specifically, we use ResNet-56 on single-label classification (CIFAR-10, CIFAR-100 and SVHN) as an example and report results with two pruning thresholds in addition to the default one, *i.e.*, 0.75% and 3% of the original performance. We report the corresponding experimental results in Table 3.

According to the comparison shown in Table 3, using 0.75% as the pruning threshold tends to provide the best model performance among different datasets but fails to significantly reduce the computational cost. On the other hand, pruning with 3% as the threshold is able to remove lots of parameters from the original model but usually leads to over-pruning and degraded model performance. Adopting 1.5% as the threshold achieves a reasonable tradeoff between performance and computational efficiency. Under various settings, pruning based on the 1.5% of the original performance reduces the model parameters by over 35% while resulting in equivalent or even slightly better performance compared to the original models.

5 Conclusion

This paper proposes a layer-wise parameter pruning method for training compact CNNs based on existing deep networks. By estimating the performance of feature representations extracted at different convolutional layers within the architecture, layers with relatively small contributions are located and removed from the original networks. To boost performance of the pruned models and help them efficiently regain accuracy, knowledge distillation is introduced in the retraining procedure for transferring information from the original deep models to the more compact ones. Extensive comparative experiments demonstrate that models constructed by the proposed method are able to achieve equivalent or even slightly better performance than the original models with high efficiency on parameter utilization.

References

[1] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 1097–1105. Curran Associates, Inc., 2012.

[2] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014.

[3] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Computer Vision and Pattern Recognition (CVPR)*, 2015.

[4] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, June 2016.

[5] Jifeng Dai, Yi Li, Kaiming He, and Jian Sun. R-fcn: Object detection via region-based fully convolutional networks. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems 29*, pages 379–387. Curran Associates, Inc., 2016.

[6] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28*, pages 91–99. Curran Associates, Inc., 2015.

[7] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR) 2015*, November 2015.

[8] K. He, G. Gkioxari, P. Dollr, and R. Girshick. Mask r-cnn. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 2980–2988, Oct 2017.

[9] Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhudinov, Rich Zemel, and Yoshua Bengio. Show, attend and tell: Neural image caption generation with visual attention. In David Blei and Francis Bach, editors, *Proceedings of the 32nd International Conference on Machine Learning (ICML-15)*, pages 2048–2057. JMLR Workshop and Conference Proceedings, 2015.

[10] Song Han, Jeff Pool, John Tran, and William Dally. Learning both weights and connections for efficient neural network. In *Advances in Neural Information Processing Systems (NIPS)*, pages 1135–1143, 2015.

[11] Song Han, Huizi Mao, and William J Dally. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. *International Conference on Learning Representations (ICLR)*, 2016.

[12] Hao Li, Asim Kadav, Igor Durdanovic, Hanan Samet,

and Hans Peter Graf. Pruning filters for efficient convnets. *International Conference on Learning Representations (ICLR)*, 2017.

[13] Emily L. Denton, Wojciech Zaremba, Joan Bruna, Yann Lecun, and Rob Fergus. Exploiting linear structure within convolutional networks for efficient evaluation. In Z. Ghahramani, M. Welling, C. Cortes, N.d. Lawrence, and K.q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 1269–1277. Curran Associates, Inc., 2014.

[14] Guillaume Alain and Yoshua Bengio. Understanding intermediate layers using linear classifier probes. *CoRR*, abs/1610.01644, 2016.

[15] Geoffrey E. Hinton, Oriol Vinyals, and Jeffrey Dean. Distilling the knowledge in a neural network. *CoRR*, abs/1503.02531, 2015.

[16] Max Jaderberg, Andrea Vedaldi, and Andrew Zisserman. Speeding up convolutional neural networks with low rank expansions. In *Proceedings of the British Machine Vision Conference*. BMVA Press, 2014.

[17] Peisong Wang and Jian Cheng. Accelerating convolutional neural networks for mobile applications. In *Proceedings of the 2016 ACM on Multimedia Conference*, MM '16, pages 541–545, New York, NY, USA, 2016. ACM.

[18] Suyog Gupta, Ankur Agrawal, Kailash Gopalakrishnan, and Pritish Narayanan. Deep learning with limited numerical precision. In *Proceedings of the 32Nd International Conference on International Conference on Machine Learning - Volume 37*, ICML'15, pages 1737–1746. JMLR.org, 2015.

[19] Matthieu Courbariaux, Yoshua Bengio, and Jean-Pierre David. BinaryConnect: Training deep neural networks with binary weights during propagations. In Corinna Cortes, Neil D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28 (NIPS 2015)*, pages 3123–3131. Curran Associates, Inc., 2015.

[20] Forrest N. Iandola, Matthew W. Moskewicz, Khalid Ashraf, Song Han, William J. Dally, and Kurt Keutzer. Squeezenet: Alexnet-level accuracy with 50x fewer parameters and <1mb model size. *CoRR*, abs/1602.07360, 2016.

[21] Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, 2009.

[22] Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y. Ng. Reading digits in natural images with unsupervised feature learning. In *NIPS Workshop on Deep Learning and Unsupervised Feature Learning 2011*, 2011.

[23] Andreas Veit, Michael J. Wilber, and Serge J. Belongie. Residual networks behave like ensembles of relatively shallow networks. In Daniel D. Lee, Masashi Sugiyama, Ulrike von Luxburg, Isabelle Guyon, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*, pages 550–558, 2016.

[24] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollr, and C. Lawrence Zitnick. Microsoft coco: Common objects in context. In *European Conference on Computer Vision (ECCV)*, Zrich, 2014. Oral.

[25] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR09*, 2009.

[26] F. Zhu, H. Li, W. Ouyang, N. Yu, and X. Wang. Learning spatial regularization with image-level supervisions for multi-label image classification. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2027–2036, July 2017.

[27] Z. Wang, T. Chen, G. Li, R. Xu, and L. Lin. Multi-label image recognition by recurrently discovering attentional regions. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 464–472, Oct 2017.

[28] M. Oquab, L. Bottou, I. Laptev, and J. Sivic. Is object localization for free? - weakly-supervised learning with convolutional neural networks. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 685–694, 2015.

Shi Chen received his B.E. degree in the school of computer science, Wuhan University, Wuhan, China, in 2015, and M.S. degree from University of Minnesota, Minneapolis, USA, in 2017. He is currently a Ph.D. student in the department of computer science, University of Minnesota, USA. His research interests include computer vision, pattern recognition and deep learning.

Qi Zhao is an assistant professor in the Department of Computer Science and Engineering at the University of Minnesota, Twin Cities. Her main research interests include computer vision, machine learning, cognitive neuroscience, and mental disorders. She received her Ph.D. in computer engineering from the University of California, Santa Cruz in 2009. She was a postdoctoral researcher in the Computation and Neural Systems, and Division of Biology at the California Institute of Technology from 2009 to 2011. Prior to joining the University of Minnesota, Qi was an assistant professor in the Department of Electrical and Computer Engineering and the Department of Ophthalmology at the National University of Singapore. She has published more than 40 journal and conference papers in top computer vision, machine learning, and cognitive neuroscience venues, and edited a book with Springer, titled Computational and Cognitive Neuroscience of Vision, that provides a systematic and comprehensive overview of vision from various perspectives, ranging from neuroscience to cognition, and from computational principles to engineering developments. She is a member of the IEEE.