Target Aware Network Adaptation for Efficient Representation Learning

Yang Zhong[1,✉], Vladimir Li[1], Ryuzo Okada[2], and Atsuto Maki[1]

[1] KTH Royal Institute of Technology, Stockholm, Sweden
{yzhong, vlali, atsuto}@kth.se
[2] Toshiba Corporate Research and Development Center, Kawasaki, Japan
ryuzo.okada@toshiba.co.jp

Abstract. This paper presents an automatic network adaptation method that finds a ConvNet structure well-suited to a given target task, e.g. image classification, for efficiency as well as accuracy in transfer learning. We call the concept target-aware transfer learning. Given only small-scale labeled data, and starting from an ImageNet pre-trained network, we exploit a scheme of removing its potential redundancy for the target task through iterative operations of filter-wise pruning and network optimization. The basic motivation is that compact networks are on one hand more efficient and should also be more tolerant, being less complex, against the risk of overfitting which would hinder the generalization of learned representations in the context of transfer learning. Further, unlike existing methods involving network simplification, we also let the scheme identify redundant portions across the entire network, which automatically results in a network structure adapted to the task at hand. We achieve this with a few novel ideas: (i) cumulative sum of activation statistics for each layer, and (ii) a priority evaluation of pruning across multiple layers. Experimental results by the method on five datasets (Flower102, CUB200-2011, Dog120, MIT67, and Stanford40) show favorable accuracies over the related state-of-the-art techniques while enhancing the computational and storage efficiency of the transferred model.

Keywords: Target-aware · Network adaptation · Model Compaction · Transfer Learning.

1 Introduction

The methodology of constructing feature representations has been recently advanced from a well-known hand-crafted manner to a learning-based one. Conventional hand-crafted features are typically designed by leveraging the domain knowledge of human experts [1, 4, 21]. The learning based approaches often generate discriminative image representations using large-scale labeled datasets, such as ImageNet [6], Places [38], MS COCO [18], and CelebA [19], with deep and complex convolutional neural networks (ConvNets). Thanks to the generic transferability, these learned deep representations from the ConvNets can also be utilized for other unseen tasks by means of transfer learning [3, 7, 27, 35, 36]:

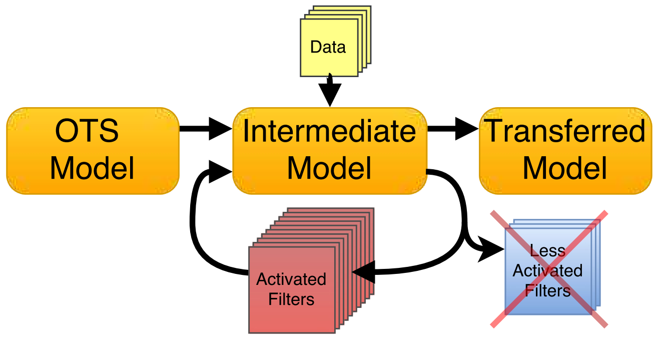

Fig. 1. A schematic of network adaptation proposed in this paper. An off-the-shelf (OTS) model is iteratively structured during the process of transfer learning where in each iteration, less significant filters in terms of activation statistics are discarded while the rest of the filters are reused in the next iteration.

one way is to directly use the pre-trained ConvNet to map images to the learned feature space with or without selecting more discriminative features [37]; a more common practice, however, is to transfer a large off-the-shelf (OTS) model to a target task by mildly tuning the ConvNet parameters to make it more specific to a new target task [3, 19, 35].

Despite the successful applications of ConvNets along the scenario of transfer learning, a basic question with respect to the model structure is still unaddressed: a question about whether the off-the-shelf network is sufficiently complex or more than ample to model the target task. That is, most of the existing approaches to ConvNet transfer learning take a predefined network architecture as given and optimize the parameters therein, without seeking for a better suited network structure even though the given task can be much simpler and therefore requiring a less complex model. Indeed, it has been shown that a model built for a source task could be less activated on target tasks [23]. This suggests that ConvNets can be made more compact to derive discriminative feature representations more efficiently in transfer learning scenarios. Moreover, learning representations by reusing a model designed for a very large-scale problem (such as ImageNet classification [6]) for smaller target tasks would risk models to get overfitted to the target domain. Tailoring a network suitable for a target task helps to enhance regularization when learning the target task representations.

As a natural progression for the transferred model to be more effectively adapted to target task, which we call *target-aware transfer learning*, an intuitive next step is to automatically remove possible redundancy from the transferred off-the-shelf model. Besides an obvious benefit in improving computational efficiency, one could expect increased accuracy on a target task if it were possible to find a less complex model that is better structured to target data. Thus, in this paper, we propose an automatic network adaptation method for target-aware transfer learning. To this end, we exploit cumulative sum of activation statistics for each ConvNet layer to determine the priority of filters to prune across the network while fine-tuning the parameters in an iterative pipeline.

Although there are approaches to simplifying ConvNets in general [16] or learning a sparse network structure [2, 33, 39], to the best of our knowledge this is the first work that addresses an automatic network adaptation for transfer learning. The work presented in this paper is close to [25] in that they deploy a framework consisting of iterative filter pruning and fine-tuning, and [25] also provides a comparative study on criteria for network pruning. As a development orthogonal to those criteria, our method has a functionality to modify and adapt the network structure according to the target task while also avoiding a greedy search to select parts of the network to be removed. The contributions of this paper are summarized as follows:

– We propose a target-aware feature learning framework, *Network Adaptation*, which efficiently generates useful image representations.
– The Network Adaptation automatically tailors an off-the-shelf network (the weights and architecture) to the target task data at hand, so that the learned representations are more favorable and more efficient on the target tasks than the ones from simply fine-tuned off-the-shelf models.
– The results highlight that the generalization of the features gets enhanced through the model compaction by our Network Adaptation for transferring given models to new target tasks.

In Section 2, we introduce recent related work. The Network Adaptation method is described in detail in Section 3. Comparative experimental results are demonstrated in Section 4.2 and Section 4.3 with ablation studies of different pruning strategies in Section 4.4. Section 5 concludes the paper.

2 Related Work

Transfer learning with deep ConvNets addresses a question of how to exploit a trained network for the sake of another task that has some commonality (to some unknown extent) to the source problem which the model has been trained on. For instance, one may use an ImageNet trained model and further train it to classify a subset of ImageNet classes, or even perform a rather different task [13] which has less abundant data. The goal of transfer learning is to construct feature representations for a target task.

A commonly adopted way of transferring a ConvNet is to fine-tune a source task model for a target task. A small learning rate is often used to optimize the target-task orientated objective function so that the learned representations still preserve the generalization learned from the data in the source domain. Feature selection could also be performed before fine-tuning, as different levels of features have different utility for the target task [7, 27, 36]. One can see feature selection in this context as a plain data-dependent network compaction approach since some high-level layers may be skipped in a transferred network. In our work, network compaction is considered and applied on all across the network based on the activation statistics.

Although ConvNets transfer learning has demonstrated higher performance than the conventional approaches in solving many computer vision problems [3], the enormous computational cost and memory footprint for using ConvNets have hindered applications in some practical scenarios. To alleviate demanding hardware requirements, attentions have been paid to network compaction [9, 12, 22, 23, 25]. Most of the existing compaction methods consider pruning models through coding [9], sparsity [10, 33], matrix decomposition [23], or norm of filter weights [16].

Although it is known that small-scale target data often provides a poor sampling of the target domain, which causes overfitting with complex models, it is still very seldom that network compaction is explicitly employed to counteract overfitting to further help transfer learning. Very recently, Pavlo et al. [25] proposed an iterative pruning method to optimize a fine-tuned networks for a target task. In every iteration, all the filters in the network are evaluated and one filter is pruned at a time based on saliency. The iterative loop needs to be carried out until a reasonable trade-off between accuracy and efficiency is reached. In our method, it is sufficient to perform the network adaptation for much fewer iterations and it prunes many insignificant filters along the network in every single iteration.

On another aspect, to improve regularization for fine-tuning, it has been found that optimizing multiple helpful objectives leads to better model effectiveness compared to plain fine-tuning which only utilizes a cross-entropy loss [8, 17]. One can attempt to reduce the domain variance through certain metrics (i.e. perform domain adaptation) as in [20, 31]. It is also viable to perform multi-task learning explicitly. In [8], a data selection approach was developed to select (source-domain) data similar in low-level features to perform multi-task learning. Rather than relying on the availability of foreign data as in the aforementioned work, in [17], the predictions of target domain images given by the source model were recorded before training. They were used later as replacements of extra training data. Then, the current predictions were compared to the recorded ones to compose an additional objective in the loss function. In this way, the network was able to be optimized on the target tasks, but at the same time be able to "remember" how to perform well on the source task. In this work, our approach also achieved better regularization without dependence on the availability of any extra data other than the target task at hand.

3 Target-aware Network Adaptation

As the intermediate representations along the network are likely to be redundant (over-completed) for a target task, it is reasonable to question if an off-the-shelf architecture is unnecessarily over-structured when being transferred to a target task. By rethinking this commonly adopted fine-tuning procedure, in this section, we propose a network adaptation method to prune and structure an off-the-shelf network for a given target task in Section 3.1.

3.1 Network Adaptation

Algorithm 1: Iterative Network Adaptation process.

```
  Source: 1) An off-the-shelf network, 2) Labeled data in target domain.
1 Step 0: Fine-tuning the off-the-shelf network on the target task.
2 for iteration $i$ do
3     Break from the loop when $i$ reaches a certain value.
4     Step 1: Prune filters in the network optimized in the previous step
      according to the activation statistics for the training data:
      (a) On each layer, identify less significant filters in terms of cumulative sum of
          average activations;
      (b) Among all the network layers, prioritize the need for pruning identified
          filters by the global priority.
      Step 2: Fine-tune the pruned network on the target task, with the same
      objective function as in Step 0.
```

The details of our network adaptation procedure are described in Algorithm 1. First, it takes an off-the-shelf network as a starting point and performs fine-tuning on the target task. After that, in Step 1, it first collects activation statistics and prunes the trained network. Specifically, in Step 1(a), the average intensity of the activation maps is calculated on every layer after feeding in the entire training set.

Let us assume a convolutional layer which has an output activation tensor A, with a size of $H \times W \times K$ (where K represents the number of output channels, and H and W stand for the hight and width of feature maps, respectively). The channel-wise activation, $\mathbf{a}^{(k)}(k = 1, ..., K)$, is simply calculated as:

$$\mathbf{a}^{(k)} = \frac{1}{W \times H} \sum_{w=1}^{W} \sum_{h=1}^{H} A_{w,h}^{(k)}. \tag{1}$$

After feeding N training images, $\mathbf{a}^{(k)}$ is averaged over N instances:

$$\bar{\mathbf{a}} = \{\bar{a}_k\}_1^K, \bar{a}_k = \sum_{n=1}^{N} (a_n^{(k)})/N. \tag{2}$$

The channel-wise mean activation is then normalized by its $L1$ norm:

$$\hat{\mathbf{a}} = \frac{\bar{\mathbf{a}}}{\|\bar{\mathbf{a}}\|_1}. \tag{3}$$

Then, we compute the cumulative sum of descendingly sorted normalized mean activation $\hat{\mathbf{a}}$, which is denoted by

$$\mathbf{c} = cumsum(\hat{\mathbf{a}}') \mid \hat{\mathbf{a}}' = sort(\hat{\mathbf{a}}). \tag{4}$$

Figure 2 illustrates the curve $\mathbf{c}$. Based on the cumulative sum, the filters corresponding to a cumulative sum value higher than a threshold r can be identified, i.e.,

$$h = \arg\min_k \mid c_k - r \mid, \tag{5}$$

$$\mathbf{m}'_p = \{m'_k\}_1^K,\ m'_k = \begin{cases} 1, k \leq h \\ 0, k > h \end{cases}, \tag{6}$$

$$\mathbf{m}_p = sort^{-1}(\mathbf{m}'_p), \tag{7}$$

where c_k is an individual element in $\mathbf{c}$ and $\mathbf{m}_p$ is a binary vector that indicates the indexes of potential filters to keep on a certain layer. The ratio between h and K is illustrated in Figure 2 as the vertical broken line. The filters corresponding to the right side of that line are considered for pruning.

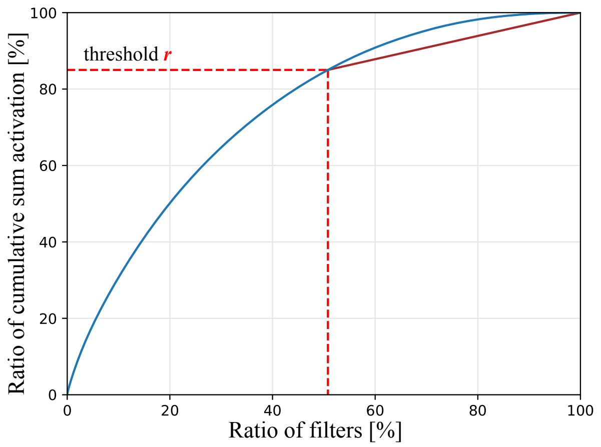

Fig. 2. Cumulative sum of activations: an example from FC6 layer of the VGG-16 network based on the Flower102 dataset. The horizontal red broken line indicates the cumulative threshold ratio r, which is 85% in this case. Filters corresponding to the right side of the vertical broken line are considered for pruning. The slope of the brown line represents the priority in Equation 8.

Next, in Step 1 (b), in order to further encourage the nature of data-dependent network adaptation, we additionally develop a way to calculate the priority. It helps us to decide whether to perform pruning or not to potential filters on a certain layer. The priority s for layer l is:

$$s^{(l)} = \frac{1-r}{1-h^{(l)}/K^{(l)}} \tag{8}$$

Finally, the filters are pruned on the layers that have a priority lower than the mean of priority of all layers guided by $\mathbf{m}_p$; pruning is therefore dynamically performed on the layers that are globally less important. The pruned network is thereafter optimized in Step 2 and the network adaptation is performed iteratively. The network adaptation can be terminated after a number of iterations when the validation accuracy starts to drop or the model size falls below a certain ratio.

4 Experiments

In this section, we first describe the datasets we used for the experiments and the details of how we trained the networks in Section 4.1. We then demonstrate the improvement brought about by our method in both accuracy and efficiency by comparing to the recent work addressing transfer learning and network compaction in an identical or a similar context in Section 4.2. Section 4.3 and Section 4.4 reveal how the Network Adaptation impacts the network structure and computational efficiency. In addition, we also performed an ablation study which demonstrates the significance of activation-based network adaptation versus random filter pruning methods in Section 4.5.

4.1 Experimental Setup

In order to provide comprehensive results, our method is comparatively evaluated on five datasets: the Flower102 [26], the CUB200-2011 [32], the Dog120 [14], the MIT67 [28], and the Stanford40 [34], which represent classification tasks in different scenarios. These datasets are among the most common benchmarks employed by the recent transfer learning related work.

Flower102 has 8189 images of 102 flower classes. The training set and the validation set both have ten images respectively for each class. The other images form the test set. In our experiments, we faithfully followed the training and test protocol, i.e. training only employs the 1020 images from the training set without any mixture with the validation instances. **Dog120** dataset contains 120 dog classes where each of the dog class has 100 training images and the test set contains 8580 images. **CUB200-2011** (denoted by "CUB200" in the following) dataset has 6000 training images and 5700 test images of 200 bird species. **MIT67** has 80 training images and 20 test images per indoor scene class. **Stanford40** dataset contains images of human actions of 40 classes, each of which contains 180 to 300 images. On the Dog120, CUB200, MIT67, and Stanford40, 10% training images were separated to form the validation set for model training.

To augment the training images, we employed random jittering and subtle scaling and rotation perturbations to the training images. We resized images of all involved datasets to 250×250 and the aspect ratio of the images was retained by applying zero padding all the time. During test time, we averaged over the network responses from the target-task classifiers over ten crops which were sampled from the corners and the centers of originals and the flipped counterparts.

There are a few well-known off-the-shelf networks that can be used in this study such as the AlexNet [15], the InceptionNet [30], the ResNet [11], and the VGGNet [29]. We choose the VGG-16 architecture in our experiments as it is a well studied and widely utilized structure for different kinds of computer vision problems. More importantly, it facilitates comparisons to be as fair as possible since the VGG-16 performs neutrally comparing to the "main-stream" architectures. The off-the-shelf VGG-16 model used in our experiments was pre-trained on the ImageNet [6] and is publicly accessible [24].

For training networks, we used a batch size of 32 with a conservative learning rate of 10^{-4}, which was helpful to achieve stable convergence, on all the datasets. The learning rate was dropped by a factor of 10 once the validation loss stopped decreasing. On most of the datasets, the learning rate was reduced to 10^{-5} within 20 epochs. The model training was terminated when the validation loss stopped decreasing. The model snapshots which performed best on the validation sets during the training were used for performance evaluation on the corresponding test sets. We set weight decay to 0.0005 and Dropout ratio to 0.5.

4.2 Comparative Performance Evaluations

Our experiments in this section focus on comparing our Network Adaptation approach with the standard fine-tuning on the selected target tasks. One straightforward way to apply the Network Adaptation to an off-the-shelf architecture is to perform filter-wise pruning along the entire network, i.e. pruning filters from Conv1_1 until FC7 in the VGG-16 architecture. Considering the "blessing of dimensionality" [5], however, another rational choice is to apply the Network Adaptation excluding the FC7 layer. In this case, the dimensionality of the high-level feature is maintained, which could be helpful to ensure the discrimination power, although at a cost of marginal computational overhead. In the following experiments, we evaluate both options by running independent Network Adaptation processes from the same fine-tuned starting point (i.e. from an identical model at "Step 0" in Algorithm 1 on each dataset).

Specifically, in our experiments, we first performed fine-tuning on each dataset and then performed Network Adaptation iteratively. At each iteration, we used the best performing model on the validation set for performance evaluations. It was then used as the starting model standpoint for the next step. For the Network Adaptation, it is easy to see that the validation accuracy could fluctuate in the first few iterations and decrease eventually. With this regard, we ran 20 Network Adaptation iterations and selected the best performing model (on the validation sets) for the performance evaluation and compared them to the corresponding fine-tuning baselines.

Table 1 shows how the Network Adaptation performs over the standard fine-tuning. First, it can be seen that by keeping the dimensionality of FC7 (NwA w/o FC), it outperforms the standard fine-tuning except on Flower102 and Dog120

Table 1. Comparing the test accuracy of fine-tuning with Network Adaptation (NwA) of different design options. The test accuracy of Network Adaptation given here was determined by the best validation accuracy on the corresponding dataset. The Network Adaptation iteration (shortened to "Iter.") number when the best validation accuracy occurred is listed in the parenthesis after test accuracy. Threshold ratio r was set to 2%.

	Fine-tune	NwA w/FC7	NwA w/o FC7
CUB200	75.84%	76.74% (@Iter. 1)	**77.49**% (@Iter. 2)
Dog120	82.79%	**82.88**% (@Iter. 1)	82.73% (@Iter. 2)
Flower102	84.96%	**85.14**% (@Iter. 1)	83.51% (@Iter. 9)
MIT67	70.30%	**71.34**% (@Iter. 5)	71.34% (@Iter. 4)
Stanford40	76.23%	**77.19**% (@Iter. 5)	77.01% (@Iter. 5)

dataset. When the dimensionality of the FC7 layer is reduced, Network Adaptation outperforms the standard fine-tuning on all the datasets by an error rate reduction of 3.72% on CUB200, 0.52% on Dog120, 1.20% on Flower102, 3.50% on MIT67, and 4.03% on Stanford40. Remember that such a performance gain was achieved by using more compact networks and only the target task training data. It suggests that how to restructure the network structure to achieve better model effectiveness could also be taken into consideration when transferring an off-the-shelf model to a new target task.

Second, by applying the Network Adaptation along the entire VGG-16 architecture, even better test accuracy can be achieved on almost all datasets except CUB200. This means that restructuring the entire network not only results in even lower computational cost but also promises better performance margin on average. In other words, keeping the dimensionality of the high-level abstractions may not be as important in general. In the following experiments, we perform the Network Adaptation along the entire network unless otherwise stated.

To comprehensively evaluate the Network Adaptation, we focus on the recent approaches which employed network compaction based methods (but brought about accuracy gains) to address transfer learning problems. Given that regularization may be improved by building compact networks for target tasks, a recent state-of-the-art method [17] that explicitly enhanced regularization through multi-task learning is also compared to our approach. These approaches may not employ exactly the same network architecture and other experimental setups (e.g. [23] used a deeper VGG-19 network) as ours, which consequently resulted in slightly different fine-tuning baseline accuracy. To handle the discrepancies, a reasonable comparative evaluation is to compare the relative accuracy gain of these methods, as shown in Table 2.

On the effectiveness of the learned representations, it is clear that the Network Adaptation contributed to superior accuracy than other methods except for the case when it was not applied to the FC7 layer of a VGG-16 network on Flower102. We argue that this is an obvious advantage to some existing network pruning methods which have to make compromises to the performance

Table 2. Comparing the gain of the test accuracy on various datasets with the recent related approaches. Each entry is the accuracy gain compared to the corresponding fine-tuning baseline. Threshold ratio r set to 2% in our Network Adaptation.

	CUB200	Dog120	Flower102	MIT67	Stanford40
Best of Deep Compression [23]	0.12%	—	0.1%	—	0.79%
Best of Efficient Inference [25]	0.5%	—	—	—	—
LwF [17]	-0.6%	—	—	0.3%	—
Ours, w/o FC7	1.65%	-0.04%	-1.45%	1.04%	0.78%
Ours, w/ FC7	0.9%	0.09%	0.18%	1.04%	0.96%

Note that our total compression ratio is on-par with that in [23], see Figure 4. The best gain on CUB200 from [25] was manually retrieved from Figure 4 in [25].

on target tasks. The accuracy gain, in general, is attributed to the improved regularization (which was achieved in different ways in these methods). It can be seen that making networks more compact is potentially a useful means to improve regularization owing to the positive gains. Our Network Adaptation indeed achieved better regularization than others given the more favorable performance improvements. In addition, we noticed that the network compaction process is quite efficient with the Network Adaptation approach as it does not depend on exhaustive search as in [25]; the compaction is also more effective for reducing the computational cost than in [23] as filters across the entire network were simultaneously considered for pruning. The network compaction effects of our approach are demonstrated with details in the following sections.

4.3 Evolution of Network Size during Network Adaptation

The Network Adaptation, on the one hand, can be used to improve the model effectiveness on the target tasks, on the other hand, it also enforces target data-driven model compaction which effectively reduces the size of ConvNets. In this way, one can make a trade-off between the network size and the model accuracy on the target task with our Network Adaptation.

To demonstrate how the discriminability of the learned representations varies on the target tasks along the Network Adaptation process, we show the validation accuracy and the test accuracy along 20 Network Adaptation iterations in Figure 3. One can find that after fine-tuning on the target tasks (Iteration 0) the Network Adaptation is able to boost the discrimination power of the learned representations, which was reflected on the validation and test accuracy in the first few iterations. It is clear that the improvement in the test accuracy is data dependent. We can expect around or more than 1% accuracy increment on MIT67, CUB200, and Stanford40, which have moderately large training sets. But the improvement was less significant when training data was too sparse or much too large. As on the Flower102, it had only 10 training images per class, while on Dog120 each class had around 100 images for training for each class, which was 2 to 3 times larger than other datasets. The accuracies dropped at

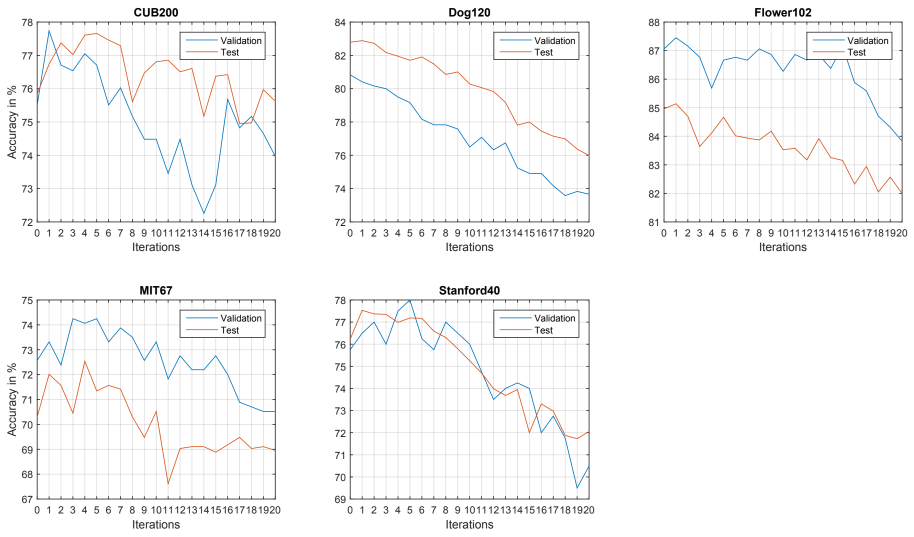

Fig. 3. Validation accuracy versus test accuracy on the datasets during 20 iterations of Network Adaptation.

later iterations when the networks lost more capacity. As our threshold r was only 2%, the test accuracies were not significantly decreased even at Iteration 20.

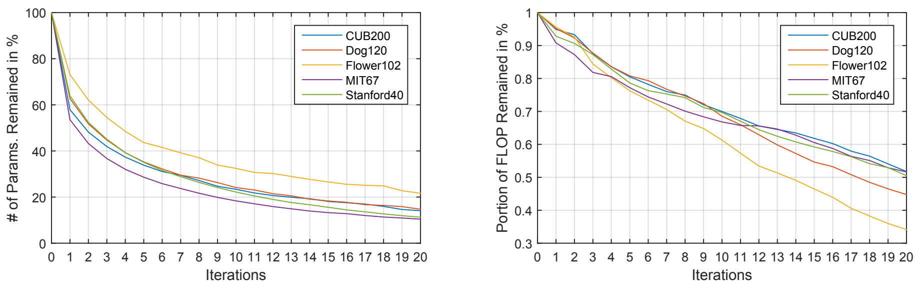

Fig. 4. Normalized reduction of the number of network coefficients (left) and the corresponding normalized reduction of FLOP (right) with a VGG-16 architecture achieved by the Network Adaptation on different datasets.

The reduction of the total number of parameters and the corresponding computational costs at each Network Adaptation iteration are shown in Figure 4. First, it can be seen that the total number of network coefficient was reduced in an exponential trend. The network size can be halved in the first five iterations on all the tasks even with an insignificant threshold value. The network size got linearly reduced after Iteration 8. Second, the Network Adaptation resulted in roughly a linear decrementing trend in the computational cost in terms of Floating-point Operation (FLOP). For both of the decreasing curves of network size and FLOP, a similar data-dependent character can be observed; on the tasks with more abundant data, they shared a common tendency which was deviated on the task with the least data.

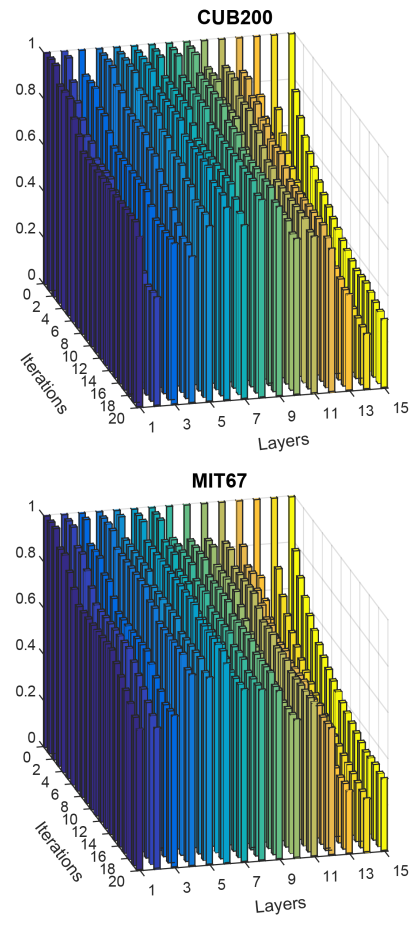

Fig. 5. Relative change of layer width at each Network Adaptation iteration. The "Layers" scale, from 1 to 15, represents the learnable layers from "Conv1_1" till "FC7" in a VGG-16 structure.

The detailed sizes of each layer width (number of filters) for the 20 Network Adaptation iterations on the selected datasets are visualized in Figure 5. It is easy to see that a VGG-16 architecture was (re-)shaped differently by our Network Adaptation on different tasks. An interesting phenomenon is that the lower level filters in the vicinity of Conv2_1 layer were pruned relatively more than other layers, but on MIT67 the pruning rate was not as large as on the others. This suggests a large but task-specific redundancy in the lower layers of the off-the-shelf model. The same situation can be observed on higher-level convolutional layers. It can, therefore, be inferred that the learned high-level image representation at the FC7 layer indeed depends on different combinations of the convolutional features: the indoor scene classification MIT67 relies more on lower level features than other tasks; CUB200 and Stanford40 require more descriptive mid-level representations; Flower102 more or less relies on higher-level convolutional features.

4.4 Impact of Threshold Ratio

In this section, we evaluate the impact of threshold ratio (r) to the effectiveness of Network Adaptation. Specifically, we perform ten iterations of Network Adaptation with 2%, 5%, and 10% threshold ratios and evaluate the best models indicated by the validation sets to explore the performance of the Network Adaptation with different setups.

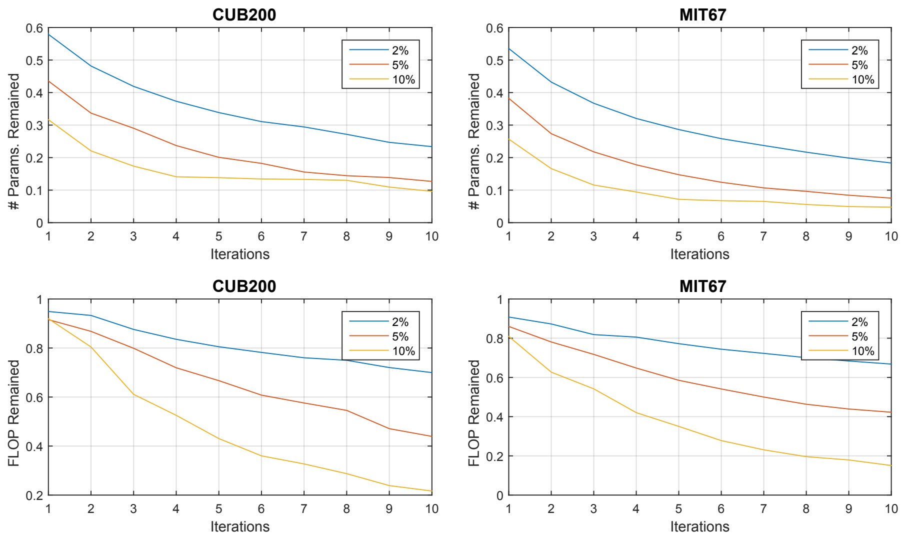

Fig. 6. Comparing pruning effects by using different threshold ratios in Network Adaptation. The x-axis represents the iterations of the Network Adaptation from 1 to 10, and the relative residual portion of network parameters and computational costs of a VGG-16 architecture is shown in y-axis.

As shown in Figure 6, by employing different threshold ratios in Network Adaptation, a target VGG-16 model can be compacted to different extents; the higher the rate is, the more the models get compact in general. For a single threshold rate, however, a VGG-16 model can become compact to different levels as well. This is due to fact that the Network Adaptation is totally a target data guided method. E.g., for a threshold ratio of 10% at Iteration 10, more than 10% of the coefficients were retained on the CUB200, but on the MIT67 the remaining portion was only around 6%. MIT67 data yielded stronger pruning effect in the Network Adaptation process.

Table 3. Test accuracy on the CUB200 and the MIT67 with different threshold ratios. The Network Adaptation iteration which gave the best performing model (on validation set) are given in parenthesis.

	Threshold Ratios		
	2%	5%	10%
CUB200	77.49% (@Iter. 2)	76.19% (@Iter. 4)	76.64% (@Iter. 1)
MIT67	71.34% (@Iter. 4)	71.49% (@Iter. 2)	69.48% (@Iter. 2)

Table 3 lists the test accuracy values on the CUB200 and the MIT67 under three threshold ratios. It reveals a rough relation between the number of iterations and the threshold ratios with respect to the best test accuracy; a mild pruning rate may require more steps to achieve the best accuracy and vice versa. From these results, we may infer that a threshold ratio between 5% and 10% would be a practical choice which balances between the training speed and the model accuracy.

4.5 Significance of Least Activation Pruning

In our Network Adaptation, we prune filters which correspond to the least activations of a given dataset. As shown in Figure 5, it shapes the model architecture based on the target data in a way that it removes different numbers of filters on different layers. To highlight the importance of this pruning strategy, we evaluate two alternative filter pruning methods which can be potentially used by our Network Adaptation framework in two stages.

In the first experiment, we compared the performance discrepancy between removing a random 10% of filters versus pruning 10% of the least activated filters in each layer along the whole network. In both ways, the models were uniformly pruned along the structure but these pruning strategies cannot (re-)shape the model structure. Due to the randomness, we ran five independent experiments for one iteration with both strategies and compared the average accuracy.

The second experiment intended to further verify that randomly pruning filters performs less effectively than filters selected based on activation. This is done by directly comparing random pruning versus activation based pruning in the Network Adaptation procedure, where random pruning removes an equal amount of filters as by the Network Adaptation in each layer in each iteration (we also ran 20 iterations with the threshold ratio r set to 2%). The same starting models at "Step 0" as in the previous experiments were used in these

Table 4. Comparing other pruning strategies to the activation based approach we used in the Network Adaptation. "Least Act." indicates "Least Activation". "Random" is short for random pruning strategies for each experiment.

	Experiment 1		Experiment 2	
	Random 10%	Least Act., 10%	Random	NwA
CUB200	76.02%	**76.46**%	75.96% (@Iter. 1)	**77.03**% (@Iter. 7)
MIT67	65.33%	**67.60**%	68.73% (@Iter. 4)	**71.34**% (@Iter. 5)

experiments but the Network Adaptation iterations were run independently for the comparative studies here. The results are listed in Table 4.

The results of Experiment 1 show that uniformly pruning a network performs inferior to the Network Adaptation. Comparing the accuracy of the random pruning strategy in both experiments to that of the Network Adaptation, one can see that in general random filter pruning performs less effectively than activation based as used in our method, i.e. activation based methods offer a more reasonable pruning.

5 Conclusions

We proposed an iterative network adaptation approach to learn useful representations for transfer learning problems which have access to only small-scale target data. Our approach automatically structures model architecture by pruning less important filters, in which a data dependent method is developed to identify less important filters. Being adapted in the proposed pipeline, the transferred models are shown to be more computationally efficient and demonstrate at least comparable performance to the widely used "fine-tuning" practice and even the related state-of-the-art approaches.

In addition to the experimental results, the current development of our Network Adaptation approach left us some interesting open questions. First, for transfer learning, how to enhance regularization with limited labeled data is not fully studied yet; we demonstrated that model compaction would be a promising candidate, but it will be also interesting to evaluate other alternatives. Second, for network adaptation, it is ideal to have equally favorable performance to all target tasks but the classification accuracies by our approach slightly decreased in some situations. This hints us that it is also helpful to consider partly expanding the network architecture according to data to complement the capacity of the transferred model.

Acknowledgements

We acknowledge fruitful discussions and comments to this work from colleagues of Toshiba Corporate Research and Development Center. We thank NVIDIA Corporation for their generous donation of NVIDIA GPUs.

Bibliography

[1] Ahonen, T., Hadid, A., Pietikinen, M.: Face recognition with local binary patterns. In: Computer Vision - ECCV 2004, Lecture Notes in Computer Science, vol. 3021, pp. 469–481 (2004)

[2] Alvarez, J.M., Salzmann, M.: Learning the number of neurons in deep networks. In: Advances in Neural Information Processing Systems 29, pp. 2270–2278 (2016)

[3] Azizpour, H., Razavian, A., Sullivan, J., Maki, A., Carlsson, S.: Factors of transferability for a generic convnet representation. IEEE Transactions on Pattern Analysis and Machine Intelligence **38**(9), 1790–1802 (2016)

[4] Bay, H., Tuytelaars, T., Van Gool, L.: Surf: Speeded up robust features. In: Computer Vision – ECCV 2006. pp. 404–417 (2006)

[5] Chen, D., Cao, X., Wen, F., Sun, J.: Blessing of dimensionality: High-dimensional feature and its efficient compression for face verification. In: 2013 IEEE Conference on Computer Vision and Pattern Recognition. pp. 3025–3032 (2013). https://doi.org/10.1109/CVPR.2013.389

[6] Deng, J., Dong, W., Socher, R., Li, L.J., Li, K., Fei-Fei, L.: ImageNet: A Large-Scale Hierarchical Image Database. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (June 2009)

[7] Donahue, J., Jia, Y., Vinyals, O., Hoffman, J., Zhang, N., Tzeng, E., Darrell, T.: Decaf: A deep convolutional activation feature for generic visual recognition. In: Proceedings of the 31st International Conference on Machine Learning. vol. 32, pp. 647–655 (2014)

[8] Ge, W., Yu, Y.: Borrowing treasures from the wealthy: Deep transfer learning through selective joint fine-tuning. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (July 2017)

[9] Han, S., Mao, H., Dally, W.J.: Deep compression: Compressing deep neural network with pruning, trained quantization and huffman coding. CoRR **abs/1510.00149** (2015), `http://arxiv.org/abs/1510.00149`

[10] Han, S., Pool, J., Tran, J., Dally, W.: Learning both weights and connections for efficient neural network. In: Advances in Neural Information Processing Systems 28, pp. 1135–1143 (2015)

[11] He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (June 2016)

[12] Hinton, G.E., Vinyals, O., Dean, J.: Distilling the knowledge in a neural network

[13] Kendall, A., Grimes, M., Cipolla, R.: PoseNet: A convolutional network for real-time 6-dof camera relocalization. In: Proceedings of the International Conference on Computer Vision (ICCV) (2015)

[14] Khosla, A., Jayadevaprakash, N., Yao, B., Fei-Fei, L.: Novel dataset for fine-grained image categorization. In: First Workshop on Fine-Grained Visual

Categorization, IEEE Conference on Computer Vision and Pattern Recognition (June 2011)

[15] Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. In: Advances in Neural Information Processing Systems 25, pp. 1097–1105 (2012)

[16] Li, H., Kadav, A., Durdanovic, I., Samet, H., Peter Graf, H.: Pruning filters for efficient convnets. In: Proceedings of the International Conference on Learning Representations (ICLR) (Apr 2017)

[17] Li, Z., Hoiem, D.: Learning without forgetting. In: European Conference on Computer Vision, Part IV. pp. 614–629 (2016)

[18] Lin, T.Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., Zitnick, C.L.: Microsoft COCO: Common objects in context. In: European Conference on Computer Vision, Part V. pp. 740–755 (2014)

[19] Liu, Z., Luo, P., Wang, X., Tang, X.: Deep learning face attributes in the wild. In: The IEEE International Conference on Computer Vision (ICCV). pp. 3730–3738 (2015)

[20] Long, M., Cao, Y., Wang, J., Jordan, M.: Learning transferable features with deep adaptation networks. In: Proceedings of the 32nd International Conference on Machine Learning. vol. 37, pp. 97–105 (2015)

[21] Lowe, D.G.: Object recognition from local scale-invariant features. In: Proceedings of the Seventh IEEE International Conference on Computer Vision. vol. 2, pp. 1150–1157 (1999)

[22] Luo, J., Wu, J., Lin, W.: ThiNet: A filter level pruning method for deep neural network compression. CoRR **abs/1707.06342** (2017), `http://arxiv.org/abs/1707.06342`

[23] Masana, M., van de Weijer, J., Herranz, L., Bagdanov, A.D., Alvarez, J.M.: Domain-adaptive deep network compression. In: The IEEE International Conference on Computer Vision (ICCV) (Oct 2017)

[24] ModelZoo: Pretrained VGG-16 model for TensorFlow, available at: `http://download.tensorflow.org/models/vgg_16_2016_08_28.tar.gz`, accessed in Aug 2017.

[25] Molchanov, P., Tyree, S., Karras, T., Aila, T., Kautz, J.: Pruning convolutional neural networks for resource efficient inference. In: Proceedings of the International Conference on Learning Representations (ICLR) (Apr 2017)

[26] Nilsback, M.E., Zisserman, A.: Automated flower classification over a large number of classes. In: Proceedings of the Indian Conference on Computer Vision, Graphics and Image Processing (Dec 2008)

[27] Oquab, M., Bottou, L., Laptev, I., Sivic, J.: Learning and transferring mid-level image representations using convolutional neural networks. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2014)

[28] Quattoni, A., Torralba, A.: Recognizing indoor scenes. 2009 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops (CVPR Workshops) pp. 413–420 (2009)

[29] Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. CoRR **abs/1409.1556** (2014)

[30] Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., Rabinovich, A.: Going deeper with convolutions. In: The IEEE International Conference on Computer Vision (ICCV) (2014)

[31] Tzeng, E., Hoffman, J., Darrell, T., Saenko, K.: Simultaneous deep transfer across domains and tasks. In: The IEEE International Conference on Computer Vision (ICCV) (December 2015)

[32] Wah, C., Branson, S., Welinder, P., Perona, P., Belongie, S.: The Caltech-UCSD Birds-200-2011 Dataset. Tech. rep. (2011)

[33] Wen, W., Wu, C., Wang, Y., Chen, Y., Li, H.: Learning structured sparsity in deep neural networks. In: Advances in Neural Information Processing Systems 29, pp. 2074–2082 (2016)

[34] Yao, B., Jiang, X., Khosla, A., Lin, A.L., Guibas, L., Fei-Fei, L.: Human action recognition by learning bases of action attributes and parts. In: The IEEE International Conference on Computer Vision (ICCV). pp. 1331–1338 (2011)

[35] Yosinski, J., Clune, J., Bengio, Y., Lipson, H.: How transferable are features in deep neural networks? In: Advances in Neural Information Processing Systems 27, pp. 3320–3328 (2014)

[36] Zeiler, M.D., Fergus, R.: Visualizing and understanding convolutional networks. In: European Conference on Computer Vision, Part I. pp. 818–833 (2014)

[37] Zhong, Y., Sullivan, J., Li, H.: Transferring from face recognition to face attribute prediction through adaptive selection of off-the-shelf cnn representations. In: 23rd International Conference on Pattern Recognition (ICPR). pp. 2264–2269 (2016)

[38] Zhou, B., Lapedriza, A., Xiao, J., Torralba, A., Oliva, A.: Learning deep features for scene recognition using places database. In: Advances in Neural Information Processing Systems 27, pp. 487–495 (2014)

[39] Zhou, H., Alvarez, J.M., Porikli, F.: Less is more: Towards compact cnns. In: European Conference on Computer Vision 2016, Part IV. pp. 662–677 (2016)