AdaFilter: Adaptive Filter Fine-tuning for Deep Transfer Learning

Yunhui Guo [†] **Yandong Li** [‡] **Liqiang Wang**[‡] **Tajana Rosing** [†]
University of California, San Diego, CA [†] University of Central Florida, Orlando, FL [‡]
yug185@eng.ucsd.edu, lyndon.leeseu@outlook.com, lwang@cs.ucf.edu, tajana@ucsd.edu

Abstract

There is an increasing number of pre-trained deep neural network models. However, it is still unclear how to effectively use these models for a new task. *Transfer learning*, which aims to transfer knowledge from source tasks to a target task, is an effective solution to this problem. Fine-tuning is a popular transfer learning technique for deep neural networks where a few rounds of training are applied to the parameters of a pre-trained model to adapt them to a new task. Despite its popularity, in this paper we show that fine-tuning suffers from several drawbacks. We propose an adaptive fine-tuning approach, called AdaFilter, which selects only a part of the convolutional filters in the pre-trained model to optimize on a *per-example* basis. We use a recurrent gated network to selectively fine-tune convolutional filters based on the activations of the previous layer. We experiment with 7 public image classification datasets and the results show that AdaFilter can reduce the average classification error of the standard fine-tuning by 2.54%.

Introduction

Inductive transfer learning (Pan, Yang, and others 2010) is an important research topic in traditional machine learning that attempts to develop algorithms to transfer knowledge from *source tasks* to enhance learning in a related *target task*. While transfer learning for traditional machine learning algorithms has been extensively studied (Pan, Yang, and others 2010), how to effectively conduct transfer learning using deep neural networks is still not fully exploited. The widely adopted approach, called *fine-tuning*, is to continue the training of a pre-trained model on a given target task. Fine-tuning assumes the source and target tasks are related. Thus, we would expect the pre-trained parameters from the source task to be close to the optimal parameters for the target target.

In the standard fine-tuning, we either optimize all the pre-trained parameters or freeze certain layers (often the initial layers) of the pre-trained model and optimize the rest of the layers towards the target task. This widely adopted approach has two potential issues. First, the implicit assumption behind fine-tuning is that all the images in the target dataset should follow the same fine-tuning policy (i.e., all the images should fine-tune all the parameters of the pre-trained model), which may not be true in general. Recent work (Shazeer et al. 2017) points out that activating parts of the network on a *per-example* basis can better capture the diversity of the dataset to provide better classification performance. In transfer learning, some classes in the target dataset may have overlap with the source task, so directly reusing the pre-trained convolutional filters without fine-tuning can benefit the learning process due to the *knowledge transfer*. The second issue is that deep neural networks often have millions of parameters, so when the pre-trained model is being applied on a relatively small target dataset, there is a danger of *overfitting* since the model is *overparameterized* for the target dataset. Solving either of the issues is of great practical importance due to the wide use of fine-tuning for a variety of applications. In a recent work (Guo et al. 2019b), the authors proposed an adaptive fine-tuning method to generate fine-tuning policy for a new task on a per-example basis. However, the generated fine-tuning policy can be only used to decide which layers to be fine-tuned.

In this paper, we propose a deep transfer learning model, called AdaFilter, which automatically selects reusable *filters* from the pre-trained model on a *per-example* basis. This is achieved by using a recurrent neural network (RNN) gate (Graves, Mohamed, and Hinton 2013), which is conditioned on the activations of the previous layer, to layerwisely decide which filter should be reused and which filter should be further fine-tuned for each example in the target dataset. In AdaFilter, different examples in the target dataset can fine-tune or reuse different convolutional filters in the pre-trained model. AdaFilter mitigates the overfitting issue by reducing the number of trainable parameters for each example in the target dataset via reusing pre-trained filters. We experiment on 7 publicly available image classification datasets. The results show that the proposed AdaFilter outperforms the standard fine-tuning on all the datasets.

The contributions of this paper can be summarized as follows,

- We propose AdaFilter, a deep transfer learning algorithm which aims to improve the performance of the widely used fine-tuning method. We propose *filter selec-*

tion, *layer-wise recurrent gated network* and *gated batch normalization* techniques to allow different images in the target dataset to fine-tune different convolutional filters in the pre-trained model.

- We experiment with 7 publicly available datasets and the results show that the proposed method can reduce the average classification error by 2.54% compared with the standard fine-tuning.
- We also show that AdaFilter can consistently perform better than the standard fine-tuning during the training process due to more efficient *knowledge transfer*.

Related Work

Transfer learning addresses the problem of how to transfer the knowledge from source tasks to a target task (Pan, Yang, and others 2010). It is closely related to life-long learning (Parisi et al. 2019), multi-domain learning (Guo et al. 2019a) and multi-task learning (Ruder 2017). Transfer learning has achieved great success in a variety of areas, such as computer vision (Raina et al. 2007), recommender systems (Pan, Yang, and others 2010) and natural language processing (Min, Seo, and Hajishirzi 2017). Traditional transfer learning approaches include subspace alignment (Gong et al. 2012), instance weighting (Dudík, Phillips, and Schapire 2006) and model adaptation (Duan et al. 2009).

Recently, a slew of works (Kumar et al. 2018; Ge and Yu 2017; Li et al. 2018; 2019) are focusing on transfer learning with deep neural networks. The most widely used approach, called fine-tuning, is to slightly adjust the parameters of a pre-trained model to allow *knowledge transfer* from the source task to the target task. Commonly used fine-tuning strategies include fine-tuning all the pre-trained parameters and fine-tuning the last few layers (Long et al. 2015). In (Yosinski et al. 2014), the authors conduct extensive experiments to investigate the transferability of the features learned by deep neural networks. They showed that transferring features is better than random features even if the target task is distant from the source task. Kornblith et al. (Kornblith, Shlens, and Le 2018) studied the problem that whether better ImageNet models transfer better. Howard et al. (Howard and Ruder 2018) extended the fine-tuning method to natural language processing (NLP) tasks and achieved the state-of-the-art results on six NLP tasks. In (Ge and Yu 2017), the authors proposed a selective joint fine-tuning scheme for improving the performance of deep learning tasks with insufficient training data. In order to get the parameters of the fine-tuned model close to the original pre-trained model, (Li, Grandvalet, and Davoine 2018) explicitly added regularization terms to the loss function. In (Guo et al. 2019b), the authors proposed a method to decide which layers to be fine-tuned for a new task. The proposed AdaFilter is more fine-grained as the fine-tuning policy is filter-level rather than layer-level.

AdaFilter

In this work, we propose a deep transfer learning method which finds the convolutional filters in a pre-trained model that are reusable for each example in the target dataset. Figure 1 shows the overview of the proposed approach. We first use a *filter selection* method to achieve per-example fine-tuning scheme. We therefore leverage a *recurrent gated network* which is conditioned on the activations of the previous layer to layerwisely decide the fine-tuning policy for each example. Finally, we propose *gated batch normalization* to consider the different statistics of the output channels produced by the pre-trained layer and the fine-tuned layer.

To this end, we first introduce the filter selection method in Sec. 3.1. Then we introduce our recurrent gated network in Sec. 3.2. Finally we present the proposed gated batch normalization method in Sec. 3.3. In Sec. 3.4, we discuss how the proposed method can achieve better performance by mitigating the issues of the standard fine-tuning.

Filter Selection

In convolutional neural network (CNN), convolutional filters are used for detecting the presence of specific features or patterns in the original images. The filters in the initial layers of CNN are used for detecting low level features such as edges or textures while the filters at the end of the network are used for detecting shapes or objects (Zeiler and Fergus 2014). When convolutional neural networks are used for transfer learning, the pre-trained filters can be reused to detect similar patterns on the images in the target dataset. For those images in the target dataset that are similar to the images in the source dataset, the pre-trained filters should not be fine-tuned to prevent from being destructed.

The proposed *filter selection* method allows different images in the target dataset to fine-tune different pre-trained convolutional filters. Consider the i-th layer in a convolutional neural network with input feature map $x_i \in \mathbb{R}^{n_i \times w_i \times h_i}$, where n_i the number of input channels, w_i is the width of the feature map and h_i is the height of the feature map. Given x_i, the convolutional filters in the layer i produce an output $x_{i+1} \in \mathbb{R}^{n_{i+1} \times w_{i+1} \times h_{i+1}}$. This is achieved by applying n_{i+1} convolutional filter $\mathcal{F} \in \mathbb{R}^{n_i \times k \times k}$ on the input feature map. Each filter $\mathcal{F} \in \mathbb{R}^{n_i \times k \times k}$ is applied on x_i to generate one channel of the output. All the n_{i+1} filters in the i-th convolutional layer can be stacked together as a $4D$ tensor.

We denote the $4D$ convolutional filters in the i-th layer as F_i. Given $x_i \in \mathbb{R}^{n_i \times w_i \times h_i}$, $F_i(x_i)$ is the output $x_{i+1} \in \mathbb{R}^{n_{i+1} \times w_{i+1} \times h_{i+1}}$. To allow different images to fine-tune different filters, we initialize a new $4D$ convolutional filter S_i from F_i and *freeze* S_i during training. We use a binary vector $G_i(x_i) \in \{0, 1\}^{n_{i+1}}$, called the fine-tuning policy, which is conditioned on the input feature map x_i to decide which filters should be reused and which filters should be fine-tuned. With $G_i(x_i)$, the output of the layer i can be calculated as,

$$x_{i+1} = G_i(x_i) \circ F_i(x_i) + (1 - G_i(x_i)) \circ S_i(x_i) \qquad (1)$$

where $\circ$ is a Hadamard product. Each element of $G_i(x_i) \in \{0, 1\}^{n_{i+1}}$ is multiplied with the corresponding channel of $F_i(x_i) \in \mathbb{R}^{n_{i+1} \times w_{i+1} \times h_{i+1}}$ and $S_i(x_i) \in \mathbb{R}^{n_{i+1} \times w_{i+1} \times h_{i+1}}$. Essentially, the fine-tuning policy $G_i(x_i)$ selects each channel of x_{i+1} either from the output produced

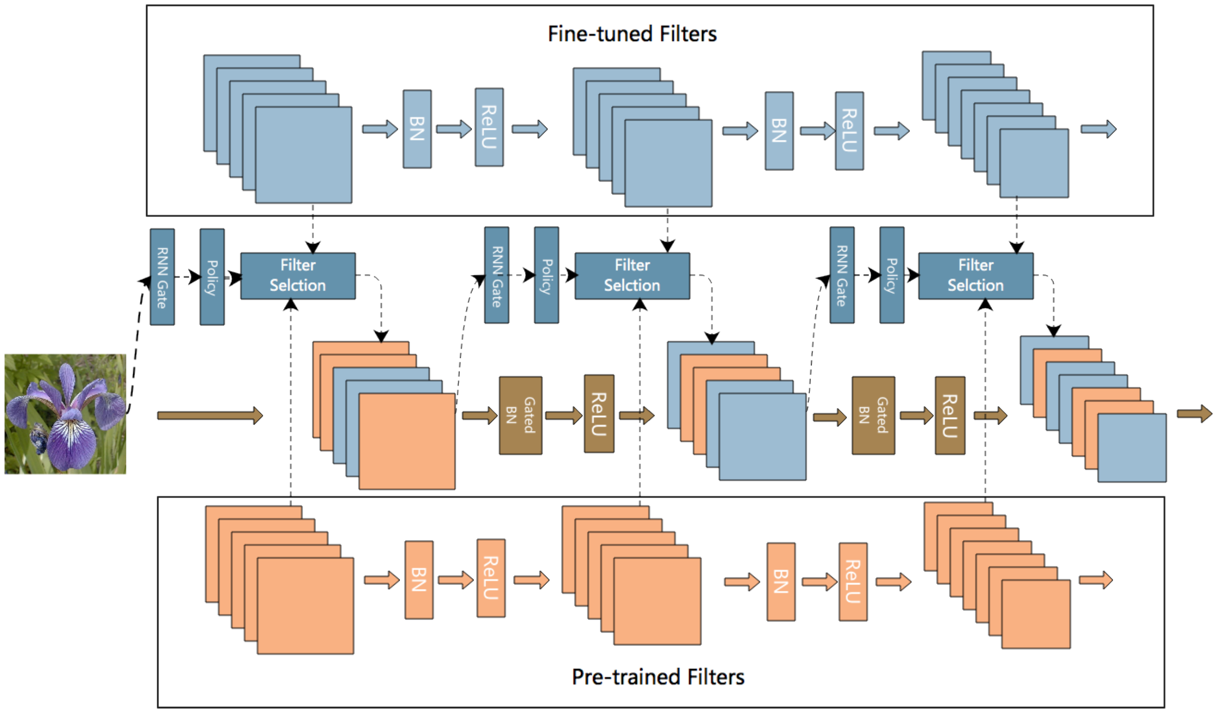

Figure 1: The overview of AdaFilter for deep transfer learning. Layerwise recurrent gated network (the middle row) decides how to select the convolutional filters from the fine-tuned layer (the first row) and the pre-trained layer (the last row) conditioned on activations of the previous layer. Note that the RNN gate is shared across all the layers.

by the pre-trained layer S_i (if the corresponding element is 0) or the fine-tuned layer F_i (if the corresponding element is 1). Since $G_i(x_i)$ is conditioned on x_i, different examples in the target dataset can fine-tune different pre-trained convolutional filters in each layer.

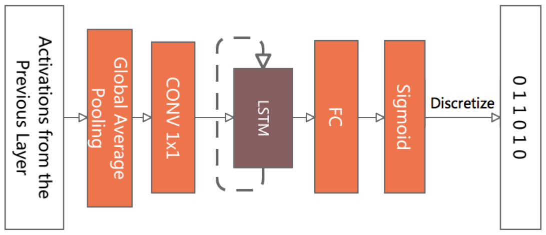

Figure 2: The proposed recurrent gated network.

Layerwise Recurrent Gated Network

There are many possible choices to generate the fine-tuning policy $G_i(x_i)$. We adopt a recurrent gated network to both consider the dependencies between different layers and the model size. Figure 2 illustrates the proposed recurrent gated network which takes activations from the previous layer as input and discretizes the output of sigmoid function as the fine-tuning policy.

Recurrent neural network (RNN) (Graves, Mohamed, and Hinton 2013) is a powerful tool for modelling sequential data. The hidden states of the recurrent neural network can remember the correlations between different timestamps. In order to apply the RNN gate, we need to map the input feature map x_i into a low-dimensional space. We first apply a global average pooling on the input feature map x_i and then use a 1×1 convolution which translates the $3D$ input feature map into a one-dimensional embedding vector. The embedding vector is used as the input of the RNN gate to generate the layer-dependent fine-tune policy $G_i(x_i)$. We translate the output of the RNN gate using a linear layer followed by a sigmoid function. To obtain the binary fine-tuned policy $G_i(x_i)$, we use a hard threshold function to discretize the output of the sigmoid function.

The discreteness of $G_i(x_i)$ makes it hard to optimize the network using gradient-based algorithm. To mitigate this problem, we use the *straight-through estimator* which is a widely used technique for training binarized neural network in the field of neural network quantization (Guo 2018). In the *straight-through estimator*, during the forward pass we discretize the sigmoid function using a threshold and during backward we compute the gradients with respect to the input of the sigmoid function,

$$\begin{aligned} &\textbf{Forward:} \quad x_b = \begin{cases} 1, & \text{sigmoid}(x) \geq 0.5, \\ 0, & \text{otherwise} \end{cases} \\ &\textbf{Backward:} \quad \frac{\partial E}{\partial x} = \frac{\partial E}{\partial x_b} \end{aligned} \tag{2}$$

where E is the loss function. The adoption of the *straight-through estimator* allows us to back-propagate through the discrete output and directly use gradient-based algorithms to end-to-end optimize the network. In the experimental section, we use Long Short-Term Memory (LSTM) (Hochreiter and Schmidhuber 1997) which has shown to be useful for different sequential tasks. We also compare the proposed recurrent gated network with a CNN-based gated network. The experimental results show that we can achieve higher classification accuracy by explicitly modelling the cross-layer correlations.

Gated Batch Normalization

Batch normalization (BN) (Ioffe and Szegedy 2015) layer is designed to alleviate the issue of internal covariate shifting of training deep neural networks. In BN layer, we first standardize each feature in a mini-batch, then scale and shift the standardized feature. Let $X_{p,n}$ denote a mini-batch of data, where p is the batch size and n is the feature dimension. BN layer normalizes a feature dimension x_j as below,

$$\hat{x}_j = \frac{x_j - E[X_{.j}]}{\sqrt{Var[X_{.j}]}} \tag{3}$$

$$y_j = \gamma_j \hat{x}_j + \beta_j \tag{4}$$

The scale parameter γ_j and shift parameter β_j are trained jointly with the network parameters. In convolutional neural networks, we use BN layer after each convolutional layer and apply batch normalization over each channel (Ioffe and Szegedy 2015). Each channel has its own scale and shift parameters.

In the standard BN layer, we compute the mean and variance of a particular channel across all the examples in the mini-batch. In AdaFilter, some examples in the mini-batch use the channel produced by the pre-trained filter while the others use the channel produced by the fine-tuned filter. The statistics of the channels produced by the pre-trained filters and the fine-tuned filters are different due to *domain shift*. To consider this fact, we maintain two BN layers, called Gated Batch normalization, which normalize the channels produced by the pre-trained filters and the fine-tuned filters separately. To achieve this, we apply the fine-tuning policy learned by the RNN gate on the output of the BN layers to select the normalized channels,

$$x_{i+1} = G_i(x_i) \circ BN_1(x_{i+1}) + (1 - G_i(x_i)) \circ BN_2(x_{i+1}) \tag{5}$$

where BN_1 and BN_2 denote the BN layer for the pre-trained filters and the fine-tuned filters separately and $\circ$ is the Hadamard product. In this way, we can deal with the case when the target dataset and the source dataset have very different domain distribution by adjusting the corresponding shift and scale parameters.

Discussion

The design of the proposed AdaFilter mitigates two issues brought by the standard fine-tuning. Since the fine-tuning policy is conditioned on the activations of the previous layer, different images can fine-tune different pre-trained filters. The images in the target dataset which are similar to the source dataset can reuse more filters from the pre-trained model to allow better *knowledge transfer*. On the other hand, while the number of parameters compared with standard fine-tuning increase by a factor of 2.2x with AdaFilter, the trainable parameters for a particular image are much fewer than the standard fine-tuning due to the reuse of the pre-trained filters. This alleviates the issue of *overfitting* which is critical if the target dataset is much smaller than the source dataset.

All the proposed modules are differentiable which allows us to use gradient-based algorithm to end-to-end optimize the network. During test time, the effective number of filters for a particular test example is equal to a standard fine-tuned model, thus AdaFilter has similar test time compared with the standard fine-tuning.

Dataset	**Training**	**Evaluation**	**Classes**
Stanford Dogs	12000	8580	120
UCF-101	7629	1908	101
Aircraft	3334	3333	100
Caltech 256 - 30	7680	5120	256
Caltech 256 - 60	15360	5120	256
MIT Indoors	5360	1340	67
Omniglot	19476	6492	1623

Table 1: Datasets used to evaluate AdaFilter against other fine-tuning baselines.

Experimental Settings

Datasets

We compare the proposed AdaFilter method with other fine-tuning and regularization methods on 7 public image classification datasets coming from different domains: Stanford dogs (Khosla et al. 2011), Aircraft (Maji et al. 2013), MIT Indoors (Quattoni and Torralba 2009), UCF-101 (Bilen et al. 2016), Omniglot (Lake, Salakhutdinov, and Tenenbaum 2015), Caltech 256 - 30 and Caltech 256 - 60 (Griffin, Holub, and Perona 2007). For Caltech 256 - x (x = 30 or 60), there are x training examples for each class. The statistics of the datasets are listed in Table 1. Performance is measured by classification accuracy on the evaluation set.

Baselines

We consider the following fine-tuning variants and regularization techniques for fine-tuning in the experiments,

- Standard Fine-tuning: this is the standard fine-tuning method which fine-tunes all the parameters of the pre-trained model.
- Fine-tuning half: only fine-tune second half of the layers of the pre-trained model and freeze the first half of layers.
- Random Policy: use AdaFilter with a random fine-tuning policy. This shows the effectiveness of fine-tuning policy learned by the recurrent gated network.
- L2-SP (Li, Grandvalet, and Davoine 2018): this is a recently proposed regularization method for fine-tuning which explicitly adds regularization terms in the loss

Method	Stanford-Dogs	UCF-101	Aircraft	Caltech256-30	Caltech256-60	MIT Indoors	Omniglot
Standard Fine-tuning	77.47%	73.10%	52.59%	78.09%	82.25%	76.42%	87.06%
Fine-tuning half	79.61%	76.43%	53.61%	78.86%	82.55%	76.94%	87.29%
Random Policy	81.84%	75.15%	54.15%	79.90%	83.35%	76.71%	85.78%
L2-SP	79.69%	74.33%	**56.52%**	79.33%	82.89%	76.41%	86.92%
AdaFilter	**82.44%**	**76.99%**	55.41%	**80.62%**	**84.31%**	**77.53%**	**87.46%**

Table 2: The results of AdaFilter and all the baselines.

function to encourage the fine-tuned model to be similar to the pre-trained model.

Pretrained Model

To compare AdaFilter with each baseline. We use ResNet-50 which is pre-trained on ImageNet. The ResNet-50 starts with a convolutional layer followed by 16 blocks with residual connection. Each block contains three convolutional layers and are distributed into 4 macro blocks (i.e, [3, 4, 6, 3]) with downsampling layers in between. The ResNet-50 ends with an average pooling layer followed by a fully connected layer. For a fair comparison with each baseline, we use the pre-trained model from Pytorch which has a classification accuracy of 75.15% on ImageNet.

Implementation Details

Our implementation is based on Pytorch. All methods are trained on 2 NVIDIA Titan Xp GPUs. We use SGD with momentum as the optimizer. The initial learning rate is 0.01 for the classification network and the initial learning rate for the recurrent gated network is 0.1. The momentum rate is 0.9 for both classification network and recurrent gated network. The batch size is 64. We train the network with a total of 110 epochs. The learning rate decays three times at the 30th, 60th and 90th epoch respectively.

Results and Analysis

Quantitative Results

AdaFilter vs Baselines We show the results of AdaFilter and all the baselines in Table 2. AdaFilter achieves the best results on 6 out of 7 datasets. It outperforms the standard fine-tuning on all the datasets. Compared with the standard fine-tuning, AdaFilter can reduce the classification error by up to 5%. This validates our claim that by exploiting the idea of per-example filter fine-tuning, we can greatly boost the performance of the standard fine-tuning method by mitigating its drawbacks. While fine-tuning half of the layers generally performs better than the standard fine-tuning, it still performs worse than AdaFilter since it still applies the same fine-tuning policy for all the images which ignores the similarity between the target task and the source task.

Compared with Random policy and L2-SP, AdaFilter obtains higher accuracy by learning optimal fine-tuning policy for each image in the target dataset via the recurrent gated network. The results reveal that by carefully choosing different fine-tuning for different images in the target dataset, we can achieve better transfer learning results. With AdaFilter, we can automatically specialize the fine-tuning policy for each test example which cannot be done manually due to the huge search space.

Test accuracy curve We show the test accuracy curve on four benchmark datasets in Figure 3. We can clearly see that the proposed AdaFilter consistently achieves higher accuracy than the standard fine-tune method across all the datasets. For example, after training for one epoch, AdaFilter reaches a test accuracy of 71.96% on the Stanford Dogs dataset while the standard fine-tuning method only achieves 54.69%. Similar behavior is also observed on other datasets. The fact that AdaFilter can reach the same accuracy level as standard fine-tuning with much fewer epochs is of great practical importance since it can reduce the training time on new tasks.

Qualitative Results

Visualization of Policies In this section, we show the fine-tuning policies learned by the recurrent gated network on Caltech256-30 and Caltech256-60 in Figure 4. The x-axis denotes the layers in the ResNet-50. The y-axis denotes the percentage of images in the evaluation set that use the fine-tuned filters in the corresponding layer. As we can see, there is a strong tendency for images to use the pre-trained filters in the initial layers while fine-tuning more filters at the higher layers of the network. This is intuitive since the filters in the initial layers can be reused on the target dataset to extract visual patterns (e.g., edges and corners). The higher layers are mostly task-specific which need to be adjusted further for the target task (Lee, Ekanadham, and Ng 2008). We also note that the policy distribution is varied across different datasets, this suggests that for different datasets it is preferable to design different fine-tuning strategies.

Ablation Study

Gated BN vs Standard BN In this section, an ablation study is performed to demonstrate the effectiveness of the proposed *gated batch normalization*. We compare gated batch normalization (Gated BN) against the standard batch normalization (Standard BN). In Gated BN, we normalize the channels produced by the pre-trained filters and fine-tuned filters separately as in Equation 5. In standard batch normalization, we use one batch normalization layer to normalize each channel across a mini-batch,

$$x_{i+1} = BN(x_{i+1}) \tag{6}$$

Table 3 shows the results of the Gated BN and the standard BN on all the datasets. Clearly, Gated BN can achieve higher accuracy by normalizing the channels produced by

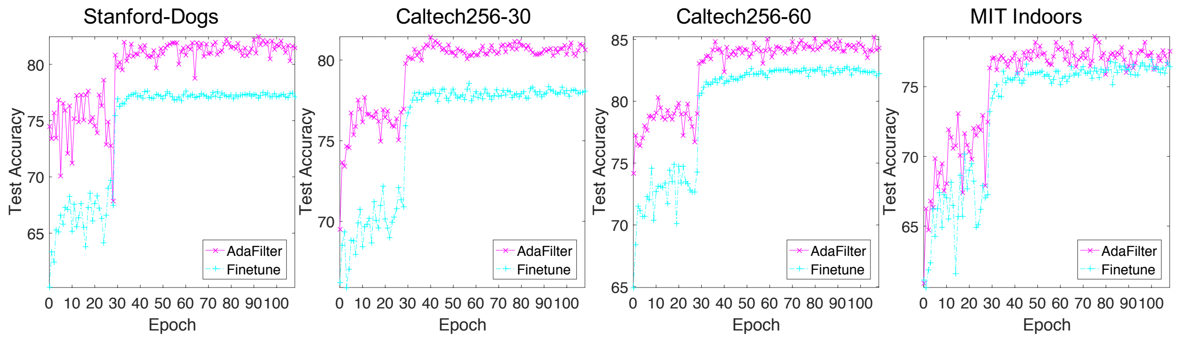

Figure 3: The test accuracy curve of AdaFilter and the standard fine-tuning on Stanford-Dogs, Caltech256-30, Caltech256-60 and MIT Indoors.

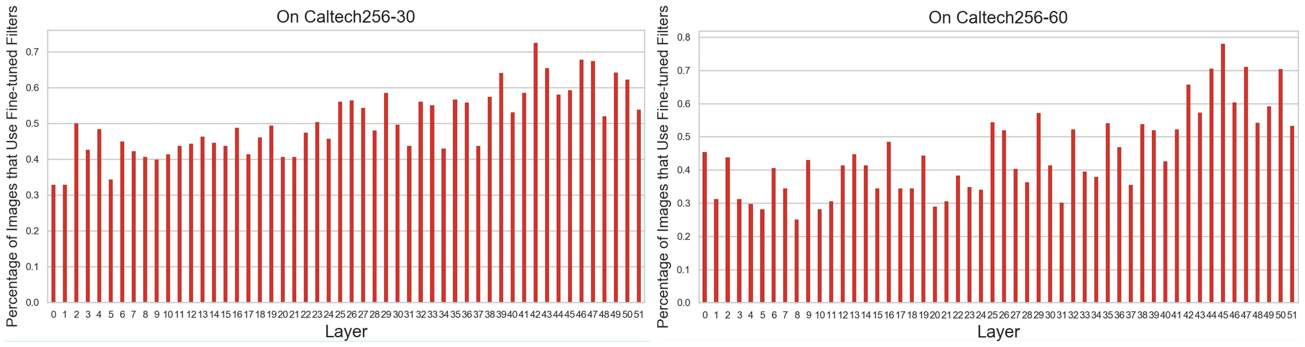

Figure 4: The visualization of fine-tuning policies on Caltech256-30 and Caltech256-60.

Dataset	Stanford-Dogs	Aircraft	Omniglot	UCF-101	MIT Indoors	Caltech-30	Caltech-60
Gated BN	82.44%	55.41%	87.46%	76.99%	77.53%	80.06%	84.31%
Standard BN	82.02%	54.33%	87.27%	76.02%	77.01%	79.84%	83.84%

Table 3: Comparison of Gated BN and the standard BN

Dataset	Stanford-Dogs	Aircraft	Omniglot	UCF-101	MIT Indoors	Caltech-30	Caltech-60
RNN-based	82.44%	55.41%	87.46%	76.99%	77.53%	80.06%	84.31%
CNN-based	83.05%	54.63%	87.04%	76.33%	77.46%	80.25%	83.41%

Table 4: Comparison of recurrent gated network and a CNN-based policy network. The "RNN-based" means the recurrent gated network.

the pre-trained filters and fine-tuned filters separately. This suggests that although we can reuse the pre-trained filters on the target dataset, it is still important to consider the difference of the domain distributions between the target task and the source task.

Recurrent Gated Network vs CNN-based Gated Network In this section, we perform an ablation study to show the effectiveness of the proposed recurrent gated network. We compare the recurrent gated network against a CNN-based policy network. The CNN-based policy network is based on ResNet-18 which receives images as input and predicts the fine-tuning policy for all the filters at once. In the CNN-based model, the input image is directly used as the input for the CNN. The output of the CNN is a list of fully connected layers (one for each output feature map in the original backbone network) followed by sigmoid activation function. We show the results of the recurrent gated network and CNN-based policy network in Table 4. Recurrent gated network performs better than the CNN-based policy network on most of the datasets by explicitly considering the dependency between layers. More importantly, predicting the policy layerwisely and reusing the hidden states of the recurrent gated network can greatly reduce the number of parameters. The lightweight design of the recurrent gated network is also faster to train than the CNN-based alternative.

Conclusion

In this paper, we propose a deep transfer learning method, called AdaFilter, which adaptively fine-tunes the convolutional filters in a pre-trained model. With the proposed *filter selection*, *recurrent gated network* and *gated batch normalization* techniques, AdaFilter allows different images in the target dataset to fine-tune different pre-trained filters to enable better knowledge transfer. We validate our methods on seven publicly available datasets and show that AdaFilter outperforms the standard fine-tuning on all the datasets. The proposed method can also be extended to lifelong learning (Yoon et al. 2017) by modelling the tasks sequentially.

Acknowledgment

This work is supported in part by CRISP, one of six centers in JUMP, an SRC program sponsored by DARPA. This work is also supported by NSF CHASE-CI #1730158, NSF #1704309 and Cyber Florida Collaborative Seed Award.

References

Bilen, H.; Fernando, B.; Gavves, E.; Vedaldi, A.; and Gould, S. 2016. Dynamic image networks for action recognition. In *CVPR*.

Duan, L.; Tsang, I. W.; Xu, D.; and Maybank, S. J. 2009. Domain transfer SVM for video concept detection. In *CVPR*.

Dudík, M.; Phillips, S. J.; and Schapire, R. E. 2006. Correcting sample selection bias in maximum entropy density estimation. In *NIPS*.

Ge, W., and Yu, Y. 2017. Borrowing treasures from the wealthy: Deep transfer learning through selective joint fine-tuning. In *CVPR*.

Gong, B.; Shi, Y.; Sha, F.; and Grauman, K. 2012. Geodesic flow kernel for unsupervised domain adaptation. In *CVPR*.

Graves, A.; Mohamed, A.-r.; and Hinton, G. 2013. Speech recognition with deep recurrent neural networks. In *Acoustics, speech and signal processing (icassp), 2013 ieee international conference on*, 6645–6649. IEEE.

Griffin, G.; Holub, A.; and Perona, P. 2007. Caltech-256 object category dataset.

Guo, Y.; Li, Y.; Feris, R.; Wang, L.; and Rosing, T. 2019a. Depthwise convolution is all you need for learning multiple visual domains. *arXiv preprint arXiv:1902.00927*.

Guo, Y.; Shi, H.; Kumar, A.; Grauman, K.; Rosing, T.; and Feris, R. 2019b. Spottune: transfer learning through adaptive fine-tuning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 4805–4814.

Guo, Y. 2018. A survey on methods and theories of quantized neural networks. *arXiv preprint arXiv:1808.04752*.

Hochreiter, S., and Schmidhuber, J. 1997. Long short-term memory. *Neural computation* 9(8):1735–1780.

Howard, J., and Ruder, S. 2018. Universal language model fine-tuning for text classification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, 328–339.

Ioffe, S., and Szegedy, C. 2015. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*.

Khosla, A.; Jayadevaprakash, N.; Yao, B.; and Fei-Fei, L. 2011. Novel dataset for fine-grained image categorization. In *First Workshop on Fine-Grained Visual Categorization, IEEE Conference on Computer Vision and Pattern Recognition*.

Kornblith, S.; Shlens, J.; and Le, Q. V. 2018. Do better imagenet models transfer better? *arXiv preprint arXiv:1805.08974*.

Kumar, A.; Sattigeri, P.; Wadhawan, K.; Karlinsky, L.; Feris, R. S.; Freeman, W. T.; and Wornell, G. 2018. Co-regularized alignment for unsupervised domain adaptation. In *NIPS*.

Lake, B. M.; Salakhutdinov, R.; and Tenenbaum, J. B. 2015. Human-level concept learning through probabilistic program induction. *Science* 350(6266):1332–1338.

Lee, H.; Ekanadham, C.; and Ng, A. Y. 2008. Sparse deep belief net model for visual area v2. In *Advances in neural information processing systems*, 873–880.

Li, Z.; Wei, Y.; Zhang, Y.; and Yang, Q. 2018. Hierarchical attention transfer network for cross-domain sentiment classification. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

Li, Z.; Li, X.; Wei, Y.; Bing, L.; Zhang, Y.; and Yang, Q. 2019. Transferable end-to-end aspect-based sentiment analysis with selective adversarial learning. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, 4582–4592. Hong Kong, China: Association for Computational Linguistics.

Li, X.; Grandvalet, Y.; and Davoine, F. 2018. Explicit inductive bias for transfer learning with convolutional networks. In *ICML*.

Long, M.; Cao, Y.; Wang, J.; and Jordan, M. I. 2015. Learning transferable features with deep adaptation networks. In *ICML*.

Maji, S.; Rahtu, E.; Kannala, J.; Blaschko, M.; and Vedaldi, A. 2013. Fine-grained visual classification of aircraft. *arXiv preprint arXiv:1306.5151*.

Min, S.; Seo, M.; and Hajishirzi, H. 2017. Question answering through transfer learning from large fine-grained supervision data. *arXiv preprint arXiv:1702.02171*.

Pan, S. J.; Yang, Q.; et al. 2010. A survey on transfer learning. *IEEE Transactions on knowledge and data engineering* 22(10):1345–1359.

Parisi, G. I.; Kemker, R.; Part, J. L.; Kanan, C.; and Wermter, S. 2019. Continual lifelong learning with neural networks: A review. *Neural Networks*.

Quattoni, A., and Torralba, A. 2009. Recognizing indoor scenes. In *CVPR*, 413–420. IEEE.

Raina, R.; Battle, A.; Lee, H.; Packer, B.; and Ng, A. Y. 2007. Self-taught learning: transfer learning from unlabeled data. In *Proceedings of the 24th international conference on Machine learning*, 759–766. ACM.

Ruder, S. 2017. An overview of multi-task learning in deep neural networks. *arXiv preprint arXiv:1706.05098*.

Shazeer, N.; Mirhoseini, A.; Maziarz, K.; Davis, A.; Le, Q.; Hinton, G.; and Dean, J. 2017. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. *arXiv preprint arXiv:1701.06538*.

Yoon, J.; Yang, E.; Lee, J.; and Hwang, S. J. 2017. Lifelong learning with dynamically expandable networks. *arXiv preprint arXiv:1708.01547*.

Yosinski, J.; Clune, J.; Bengio, Y.; and Lipson, H. 2014. How transferable are features in deep neural networks? In *NIPS*.

Zeiler, M. D., and Fergus, R. 2014. Visualizing and understanding convolutional networks. In *European conference on computer vision*, 818–833. Springer.