

Accelerating Deep Unsupervised Domain Adaptation with Transfer Channel Pruning

Chaohui Yu^{*†}, Jindong Wang^{*†}, Yiqiang Chen^{*†✉}, Zijing Wu[‡]

^{*}Beijing Key Lab. of Mobile Computing and Pervasive Device, Inst. of Computing Technology, CAS, Beijing, China

[†]University of Chinese Academy of Sciences, Beijing, China

[‡]Columbia University, New York, NY 10024, USA

Email: {yuchaohui17s, wangjindong, yqchen}@ict.ac.cn, zw2442@columbia.edu

Abstract— Deep unsupervised domain adaptation (UDA) has recently received increasing attention from researchers. However, existing methods are computationally intensive due to the computation cost of Convolutional Neural Networks (CNN) adopted by most work. To date, there is no effective network compression method for accelerating these models. In this paper, we propose a unified *Transfer Channel Pruning (TCP)* approach for accelerating UDA models. TCP is capable of compressing the deep UDA model by pruning less important channels while simultaneously learning transferable features by reducing the cross-domain distribution divergence. Therefore, it reduces the impact of negative transfer and maintains competitive performance on the target task. To the best of our knowledge, TCP is the *first* approach that aims at accelerating deep UDA models. TCP is validated on two benchmark datasets Office-31 and ImageCLEF-DA with two common backbone networks VGG16 and ResNet50. Experimental results demonstrate that TCP achieves comparable or better classification accuracy than other comparison methods while significantly reducing the computational cost. To be more specific, in VGG16, we get even higher accuracy after pruning 26% floating point operations (FLOPs); in ResNet50, we also get higher accuracy on half of the tasks after pruning 12% FLOPs. We hope that TCP will open a new door for future research on accelerating transfer learning models.

I. INTRODUCTION

Deep neural networks have significantly improved the performance of diverse machine learning applications. However, in order to avoid overfitting and achieve better performance, a large amount of labeled data is needed to train a deep network. Since the manual labeling of massive training data is usually expensive in terms of money and time, it is urgent to develop effective algorithms to reduce the labeling workload on the domain to be learned (i.e. *target* domain).

A popular solution to solve the above problem is called *transfer learning*, or *domain adaptation* [1], which tries to transfer knowledge from well-labeled domains (i.e. *source* domains) to the target domain. Specifically, Unsupervised Domain Adaptation (UDA) is considered more challenging since the target domain has no labels. The key is to learn a discriminative model to reduce the distribution divergence between domains. In recent years, deep domain adaptation methods have produced competitive performance in various tasks [2], [3], [4]. This is because that they take advantages of CNN to learn more transferable representations [2], [3] compared to traditional methods. Popular CNN architectures such as AlexNet [5], VGGNet [6], and ResNet [7] are widely

adopted as the backbone networks for deep unsupervised domain adaptation methods. Then, knowledge can be transferred to the target domain by reducing the cross-domain distance such as Maximum Mean Discrepancy (MMD) [8] or KL divergence [9].

Unfortunately, it is still challenging to deploy these deep UDA models on resource constrained devices such as mobile phones since there is a huge computational cost required by these methods. In order to reduce resource requirement and accelerate the inference process, a common solution is *network compression*. Network compression methods mainly include network quantization [10], [11], weight pruning [12], [13], [14], and low-rank approximation [15], [16]. Especially channel pruning [13], [14], [17], which is a type of weight pruning and compared to other methods, it does not need special hardware or software implementations. And it can reduce *negative transfer* [1] by pruning some redundant channels, in which, negative transfer refers to the knowledge learned on the source domain that has a negative effect on the learning on the target domain. So it is a good choice for compressing deep UDA models.

However, it is not feasible to apply the above network compression methods directly to the UDA problems. The reasons are two folds. Firstly, these compression methods are proposed to solve *supervised* learning problems, which is not suitable for the UDA settings since there are no labels in the target domain. Secondly, even if we can acquire some labels manually, applying these compression methods directly to UDA will result in negative transfer, since they fail to consider the distribution discrepancy between the source and target domains. Currently, there is no effective network compression method for UDA.

In this paper, we propose a unified network compression method called **Transfer Channel Pruning (TCP)** for accelerating deep unsupervised domain adaptation models. The general framework of our method TCP is shown in Fig. 1. Starting from a deep unsupervised domain adaptation base model, TCP iteratively evaluates the importance of channels with the transfer channel evaluation module and remove less important channels for both source and target domains. TCP is capable of compressing the deep UDA model by pruning less important channels while simultaneously learning transferable features by reducing the cross-domain distribution divergence.

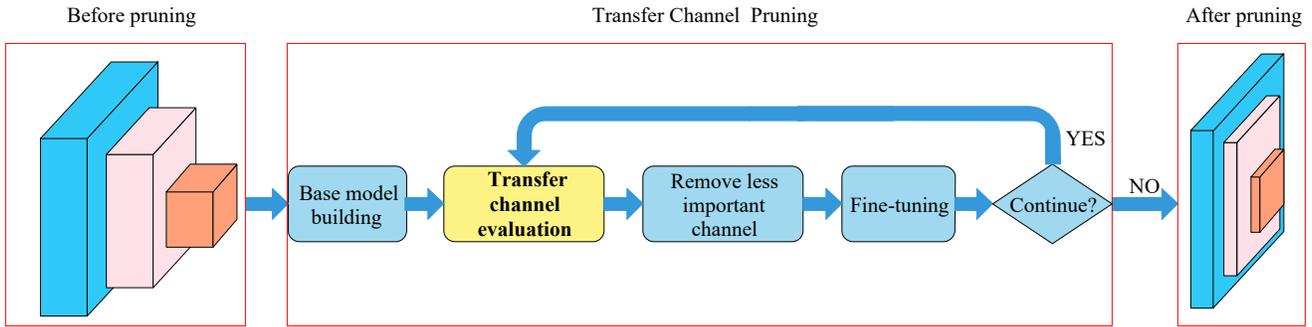


Fig. 1. The framework of the proposed Transfer Channel Pruning (TCP) approach.

Experimental results demonstrate that TCP achieves better classification accuracy than other comparison pruning methods while significantly reducing the computational cost. To the best of our knowledge, TCP is the *first* approach to accelerate the deep UDA models.

To summarize, the contributions of this paper are as follows:

1) We present TCP as a unified approach for accelerating deep unsupervised domain adaptation models. TCP is a generic, accurate, and efficient compression method that can be easily implemented by most deep learning libraries.

2) TCP is able to reduce negative transfer by considering the cross-domain distribution discrepancy using the proposed *Transfer Channel Evaluation* module.

3) Extensive experiments on two public UDA datasets demonstrate the significant superiority of our TCP method.

II. RELATED WORK

Our work is mainly related to unsupervised domain adaptation and network compression.

A. Unsupervised Domain Adaptation

UDA is a specific area of transfer learning [1], which is to learn a discriminative model in the presence of the domain-shifts between domains. The main problem of UDA is how to reduce the domain shift between the source and target domains. There are many methods to tackle this problem: traditional (shallow) learning and deep learning.

Traditional (shallow) learning methods have several aspects: 1) Subspace learning. Subspace Alignment (SA) [18] aligns the base vectors of both domains and Subspace Distribution Alignment (SDA) [19] extends SA by adding the subspace variance adaptation. CORAL [20] aligns subspaces in second-order statistics. 2) Distribution alignment. Joint Distribution Adaptation (JDA) [21] is proposed to match both distributions with equal weights. Later works extend JDA by adding structural consistency [22] and domain invariant clustering [23]. But these works treat the two distributions equally and fail to leverage the different importance of distributions. Recently, Wang *et al.* proposed the Manifold Embedded Distribution Alignment (MEDA) [24], [25] approach to dynamically evaluate the different effect of marginal and conditional distributions and achieved the state-of-the-art results on domain adaptation.

As for deep learning methods, CNN can learn nonlinear deep representations and capture underlying factors of variation between different tasks [26]. These deep representations can disentangle the factors of variation, which enables the transfer of knowledge between tasks [27]. Recent works on deep UDA embed domain-adaptation modules into deep networks to improve transfer performance [28], [29], [30], [31], where significant performance gains have been obtained. UDA has wide applications in computer vision [3], [32] and natural language processing [2] and is receiving increasing attention from researchers.

As far as we know, no previous UDA approach has focused on the acceleration of the network.

B. Network Compression

These years, for better accuracy, designing deeper and wider CNN models has become a general trend, such as VGGNet [6] and ResNet [7]. However, as the CNN grow bigger, it is harder to deploy these deep models on resource constrained devices. Network compression becomes an efficient way to solve this problem. Network compression methods mainly include network quantization, low-rank approximation and weight pruning. Network quantization is good at decreasing the presentation precision of parameters so as to reduce the storage space. Low-rank approximation reduces the storage space by low-rank matrix techniques, which is not efficient for point-wise convolution [33]. Weight pruning mainly includes two methods, neural pruning [12], [34] and channel pruning [13], [14], [17], [35].

Channel pruning methods prune the whole channel each time, so it is fast and efficient than neural pruning which removes a single neuron connection each time. It is a structured pruning method, compared to network quantization and low-rank approximation, it does not introduce sparsity to the original network structure and also does not require special software or hardware implementations. It has demonstrated superior performance compared to other methods and many works [13], [14], [35] have been proposed to perform channel pruning on pre-trained models with different kinds of criteria.

These above pruning methods mainly aim at supervised learning problems, by contrast, there have been few studies for compressing unsupervised domain adaptation models. As far as we know, we are the first to study how to do channel pruning for deep unsupervised domain adaptation.

TCP is primarily motivated by [14], while our work is different from it. TCP is presented for pruning unsupervised domain adaptation models. To be more specific, we take the discrepancy between the source and target domains into consideration so we can prune the less important channels not just for the source domain but also for the unlabeled target domain. We call this Transfer Channel Evaluation, which is highlighted in yellow in Fig. 1.

III. TRANSFER CHANNEL PRUNING

In this section, we introduce the proposed Transfer Channel Pruning (TCP) approach.

A. Problem Definition

In unsupervised domain adaptation, we are given a source domain $\mathcal{D}_s = \{(\mathbf{x}_i^s, y_i^s)\}_{i=1}^{n_s}$ of n_s labeled examples and a target domain $\mathcal{D}_t = \{\mathbf{x}_j^t\}_{j=1}^{n_t}$ of n_t unlabeled examples. \mathcal{D}_s and \mathcal{D}_t have the same label space, i.e. $\mathbf{x}_i, \mathbf{x}_j \in \mathbb{R}^d$ where d is the dimensionality. The marginal distributions between two domains are different, i.e. $P_s(\mathbf{x}_s) \neq P_t(\mathbf{x}_t)$. The goal of deep UDA is to design a deep neural network that enables learning of transfer classifiers $y = f_s(\mathbf{x})$ and $y = f_t(\mathbf{x})$ to close the source-target discrepancy and can achieve the best performance on the target dataset.

For a pre-trained deep UDA model, its parameters can be denoted as \mathbf{W} . Here we assume the l_{th} convolutional layer has an output activation tensor \mathbf{a}_l of size of $h_l \times w_l \times k_l$, where k_l represents the number of output channels of the l_{th} layer, and h_l and w_l stand for the height and width of feature maps of the l_{th} layer, respectively. Therefore, the goal of TCP is to prune a UDA model in order to accelerate it with comparable or even better performance on the target domain. In this way, we can obtain smaller models that require less computation complexity and memory consumption, which can be deployed on resource constrained devices.

B. Motivation

We compress the deep UDA model using model pruning methods for their efficiency. A straightforward model pruning technique is a *two-stage* method, which first prunes the model on the source domain with supervised learning and then fine-tunes the model on the target domain. However, negative transfer [1] is likely to happen during this pruning process since the discrepancy between the source and target domains is ignored.

In this work, we propose a unified **Transfer Channel Pruning (TCP)** approach to tackle such challenge. TCP is capable of compressing the deep UDA model by pruning less important channels while simultaneously learning transferable features by reducing the cross-domain distribution divergence. Therefore, TCP reduces the impact of negative transfer and maintains competitive performance on the target task. In short, TCP is a generic, accurate, and efficient compression method that can be easily implemented by most deep learning libraries.

To be more specific, Fig. 1 illustrates the main idea of TCP. There are mainly three steps. Firstly, TCP learns the base

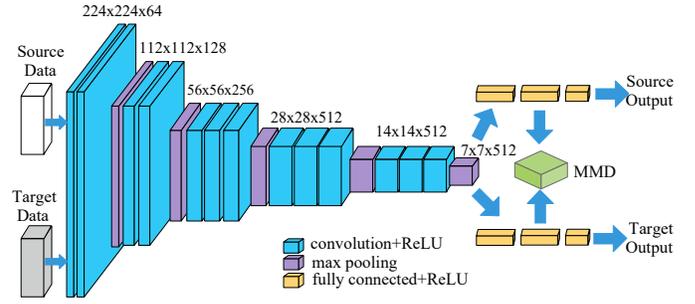


Fig. 2. The basic deep UDA architecture.

deep UDA model through *Base Model Building*. The base model is fine-tuned with the standard UDA criteria. Secondly, TCP evaluates the importance of channels of all layers with the *Transfer Channel Evaluation* and performs further fine-tuning. Specifically, the convolutional layers, which usually dominate the computation complexity, are pruned in this step. Thirdly, TCP *iteratively refines* the pruning results and stops after reaching the trade-off between accuracy and FLOPs (i.e. computational cost) or parameter size.

C. Base Model Building

In this step, we build the base UDA model with deep neural networks. Deep neural networks have been successfully used in UDA with state-of-the-art algorithms [30], [3], [31] in recent years. Previous studies [36], [31] have shown that the features extracted by deep networks are general at lower layers, while specific at the higher layers since they are more task-specific. Therefore, more transferable representations can be learned by transferring the features at lower layers and then fine-tune the task-specific layers. During fine-tuning, the cross-domain discrepancy can be reduced by certain adaptation distance. Since our main contribution is not designing new deep UDA networks, we adopt DAN [3] as the base architecture. DAN is a popular deep UDA approach and its variants have been widely adopted for UDA tasks. In the following, we will briefly introduce the main idea of DAN, and more details can be found in its original paper.

As shown in Fig. 2, we learn transferable features via several convolutional and pooling layers (the blue and purple blocks). Then, the classification task can be accomplished with the fully-connected layers (the yellow blocks). Maximum Mean Discrepancy (MMD) [9] is adopted as the adaptation loss in order to reduce domain shift. MMD has been proposed to provide the distribution difference between the source and target datasets. And it has been widely utilized in many UDA methods [3], [37], [31]. The MMD loss between two domains can be computed as

$$L_{mmd} = \left\| \frac{1}{n_s} \sum_{\mathbf{x}_i \in \mathcal{D}_s} \phi(\mathbf{x}_i) - \frac{1}{n_t} \sum_{\mathbf{x}_j \in \mathcal{D}_t} \phi(\mathbf{x}_j) \right\|_{\mathcal{H}}^2, \quad (1)$$

where \mathcal{H} denotes Reproducing Kernel Hilbert Space (RKHS) with gaussian kernel and $\phi(\cdot)$ denotes some feature map to map the original samples to RKHS.

Several popular architectures can serve as the backbone network of DAN, such as AlexNet [5], VGGNet [6], and ResNet [7]. After obtaining the base model, we can perform channel pruning to accelerate the model.

D. Transfer Channel Evaluation

The goal of transfer channel evaluation is to iteratively evaluate the importance of output channels of layers in order to prune the \mathcal{K} least important channels. Here \mathcal{K} is controlled by users. In the pruning process, we want to preserve and refine a set of parameters \mathbf{W}' , which represents those important parameters for both source and target domains. Let $L(\mathcal{D}_s, \mathcal{D}_t, \mathbf{W})$ be the cost function for UDA and $\mathbf{W}' = \mathbf{W}$ at the starting time. For a better set of parameters \mathbf{W}' , we want to minimize the loss change after pruning a channel $\mathbf{a}_{l,i}$. This can be considered as an optimization problem. Here we introduce the absolute difference of loss:

$$|\Delta L(\mathbf{a}_{l,i})| = |L(\mathcal{D}_s, \mathcal{D}_t, \mathbf{a}_{l,i}) - L(\mathcal{D}_s, \mathcal{D}_t, \mathbf{a}_{l,i} = 0)|, \quad (2)$$

which means the loss change after pruning the i_{th} channel of the l_{th} convolutional layer. And we want to minimize $|\Delta L(\mathbf{a}_{l,i})|$ by selecting the appropriate channel $\mathbf{a}_{l,i}$. Pruning will stop until a trade-off between accuracy and pruning object (FLOPs or parameter size) has been achieved.

However, it is hard to find a set of optimal parameters in one go, because the search space is $2^{|\mathbf{W}|}$ which is too huge to compute and try every combination. Inspired by [14], our TCP solves this problem with a greedy algorithm by iteratively removing the \mathcal{K} least important channels at each time.

1) **Criteria:** Criteria is the criterion for judging the importance of channels. Since the key to channel pruning is to select the least important channel, especially for UDA, we design the criteria of TCP carefully. There are many heuristic criteria, including the L_2 -norm of filter weights, the activation statistics of feature maps, mutual information between activations and predictions and Taylor expansion, etc. Here we choose the *first-order Taylor expansion* as the base criteria since its efficiency and performance has been verified in [14] for pruning supervised learning models. Compared with our TCP, we also take pruning as an optimization problem, however, the objective we want to optimize is the final performance on the unlabeled target dataset. So we design our criteria in a different way which is better for pruning deep UDA models.

According to Taylor's theorem, the Taylor expansion at point $x = a$ can be computed as:

$$f(x) = \sum_{p=0}^P \frac{f^{(p)}(a)}{p!} (x-a)^p + R_p(x), \quad (3)$$

where p denotes the p_{th} derivative of $f(x)$ at point $x = a$ and the last item $R_p(x)$ represents the p_{th} remainder. To approximate $|\Delta L(\mathbf{a}_{l,i})|$, we can use the first-order Taylor expansion near $\mathbf{a}_{l,i} = 0$ which means the loss change after removing $\mathbf{a}_{l,i}$, then we can get:

$$f(\mathbf{a}_{l,i} = 0) = f(\mathbf{a}_{l,i}) - f'(\mathbf{a}_{l,i}) \cdot \mathbf{a}_{l,i} + \frac{|\mathbf{a}_{l,i}|^2}{2} \cdot f''(\xi), \quad (4)$$

where ξ is a value between 0 and $\mathbf{a}_{l,i}$, and $\frac{|\mathbf{a}_{l,i}|^2}{2} \cdot f''(\xi)$ is a Lagrange form remainder which requires too much computation, so we abandon this item for accelerating the pruning process. Then back to Eq. (2), we can get:

$$L(\mathcal{D}_s, \mathcal{D}_t, \mathbf{a}_{l,i} = 0) = L(\mathcal{D}_s, \mathcal{D}_t, \mathbf{a}_{l,i}) - \frac{\partial L}{\partial \mathbf{a}_{l,i}} \cdot \mathbf{a}_{l,i}. \quad (5)$$

Then, we combine Eq. (2) and Eq. (5) and get the criteria G of TCP:

$$G(\mathbf{a}_{l,i}) = |\Delta L(\mathbf{a}_{l,i})| = \left| \frac{\partial L}{\partial \mathbf{a}_{l,i}} \cdot \mathbf{a}_{l,i} \right|, \quad (6)$$

which means the absolute value of product of the activation and the gradient of the cost function, and $\mathbf{a}_{l,i}$ can be calculated as:

$$\mathbf{a}_{l,i} = \frac{1}{N} \sum_{n=1}^N \frac{1}{h_l \times w_l} \sum_{p=1}^{h_l} \sum_{q=1}^{w_l} \mathbf{a}_{l,i}^{p,q}, \quad (7)$$

where N is the number of batch size and $\mathbf{a}_{l,i}^{p,q}$ is the value of the p_{th} row and the q_{th} column of the activated feature map $\mathbf{a}_{l,i}$.

2) **Loss Function of TCP:** To make TCP focus on pruning UDA models, we simultaneously take the source domain and the unlabeled target domain into consideration. The loss function of TCP consists of two parts, $L_{cls}(\mathcal{D}_s, \mathbf{W})$ and $L_{mmd}(\mathcal{D}_s, \mathcal{D}_t, \mathbf{W})$. Here, $L_{cls}(\mathcal{D}_s, \mathbf{W})$ is a cross-entropy loss which denotes the classification loss on source domain and can be computed as:

$$L_{cls}(\mathcal{D}_s, \mathbf{W}) = - \frac{1}{N} \sum_{i=1}^N \sum_{c=1}^C P_{i,c} \log(h_c(x_i^s)) \quad (8)$$

where C is the number of classes of source dataset, $P_{i,c}$ is the probability of x_i^s belonging to class c , and $h_c(x_i^s)$ denotes the probability that the model predicts x_i^s as class c . And $L_{mmd}(\mathcal{D}_s, \mathcal{D}_t, \mathbf{W})$ denotes the MMD loss between the source and target domains that presented in Eq. (1). The total loss function can be computed as:

$$L(\mathcal{D}_s, \mathcal{D}_t, \mathbf{W}) = L_{cls}(\mathcal{D}_s, \mathbf{W}) + \beta L_{mmd}(\mathcal{D}_s, \mathcal{D}_t, \mathbf{W}), \quad (9)$$

where

$$\beta = \frac{4}{1 + e^{-1 \cdot \frac{i}{ITER}}} - 2. \quad (10)$$

Here, β is a dynamic value which takes values in $(0, 1)$. $i \in (0, ITER)$ where $ITER$ is the number of pruning iterations. We design β in this way for two main reasons, on the one hand, during the early stage of pruning, the weights have not converged and keep unstable so the L_{mmd} is too large and makes the pruned model hard to converge. On the other hand, in the rest of the pruning process, the L_{mmd} becomes more important that can guide the pruned model to focus more on the target domain. So the criteria of TCP can be computed as:

$$G(\mathbf{a}_{l,i}) = \left| \frac{\partial L_{cls}(\mathcal{D}_s, \mathbf{W})}{\partial \mathbf{a}_{l,i}^s} \cdot \mathbf{a}_{l,i}^s + \beta \frac{\partial L_{mmd}(\mathcal{D}_s, \mathcal{D}_t, \mathbf{W})}{\partial \mathbf{a}_{l,i}^t} \cdot \mathbf{a}_{l,i}^t \right|, \quad (11)$$

Algorithm 1 TCP: Transfer Channel Pruning

Input: Source domain $\mathcal{D}_s = \{(\mathbf{x}_i^s, y_i^s)\}_{i=1}^{n_s}$, target domain $\mathcal{D}_t = \{\mathbf{x}_j^t\}_{j=1}^{n_t}$, the baseline \mathbf{W} .

Output: A pruned model \mathbf{W}' for deep unsupervised domain adaptation.

- 1: Fine-tune the unsupervised domain adaptation baseline until the best performance achieved on the unlabeled target dataset;
 - 2: **for** iteration i **do**
 - 3: Sort the importance of channels by criteria Eq. (11) and identify less significant channels;
 - 4: Remove the \mathcal{K} least important channels of the layers;
 - 5: Short-term fine-tune;
 - 6: **if** the trade-off between accuracy on the target domain and FLOPs or parameter size has achieved **then**
 - 7: *break*
 - 8: **end if**
 - 9: **end for**
 - 10: Long-term fine-tune;
 - 11: **return** pruned model \mathbf{W}' .
-

where $\mathbf{a}_{i,i}^s$ and $\mathbf{a}_{i,i}^t$ denote the activation with source data and target data respectively.

E. Iterative Refinement

After the transfer channel evaluation, each channel is sorted according to Eq. (11) and the \mathcal{K} least important channels are removed after each pruning iteration. Then, a short-term fine-tuning is adopted to the pruned model to help the model to converge and the pruning is done after a trade-off between accuracy and FLOPs or parameter size has been achieved. In which, the trade-off means both the computational complexity of the model and accuracy on the target domain are all acceptable. And since the target domain has no labels, so in practice, a small target domain test dataset is built by acquiring some labels manually. The learning procedure of TCP is described in Algorithm 1.

IV. EXPERIMENTAL ANALYSIS

In this section, we evaluate the performance of TCP via experiments on pruning deep unsupervised domain adaptation models. We evaluate our approaches for VGGNet [6] and ResNet [7] on two popular datasets Office-31 [38] and ImageCLEF-DA¹. All our methods are implemented based on the PyTorch [39] framework and the code will be released at github.com/jindongwang/transferlearning/code/deep/TCP.

A. Datasets

1) *Office-31*: This dataset is a standard and maybe the most popular benchmark for unsupervised domain adaptation. It consists of 4,110 images within 31 categories collected from everyday objects in an office environment. It consists of three domains: *Amazon* (A), which contains images downloaded

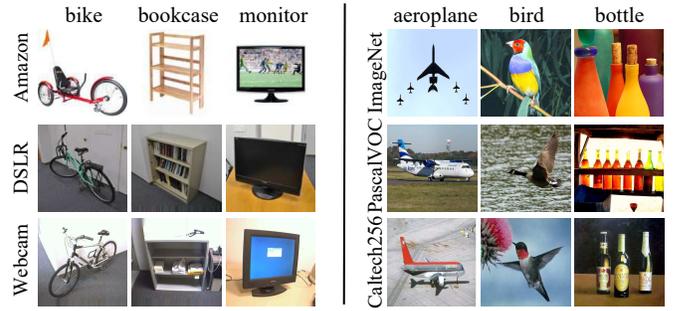


Fig. 3. A sample of the two benchmark datasets. Left: the Office-31 dataset; Right: the ImageCLEF-DA dataset.

from amazon.com, *Webcam* (W) and *DSLR* (D), which contain images respectively taken by web camera and digital SLR camera under different settings. A sample of the Office-31 dataset is shown in the left part of Fig. 3. We evaluate all our methods across six transfer tasks on all the three domains $\mathbf{A} \rightarrow \mathbf{W}$, $\mathbf{W} \rightarrow \mathbf{A}$, $\mathbf{A} \rightarrow \mathbf{D}$, $\mathbf{D} \rightarrow \mathbf{A}$, $\mathbf{D} \rightarrow \mathbf{W}$ and $\mathbf{W} \rightarrow \mathbf{D}$.

2) *ImageCLEF-DA*: This dataset is a benchmark dataset for ImageCLEF 2014 domain adaptation challenge, and it is collected by selecting the 12 common categories shared by the following public datasets and each of them is considered as a domain: *Caltech-256* (C), *ImageNet ILSVRC 2012* (I), *Pascal VOC 2012* (P) and *Bing* (B). There are 50 images in each category and 600 images in each domain. A sample of the ImageCLEF-DA dataset is shown in the right part of Fig. 3. We evaluate all methods across six transfer tasks following [3]: $\mathbf{I} \rightarrow \mathbf{P}$, $\mathbf{P} \rightarrow \mathbf{I}$, $\mathbf{I} \rightarrow \mathbf{C}$, $\mathbf{C} \rightarrow \mathbf{I}$, $\mathbf{P} \rightarrow \mathbf{C}$ and $\mathbf{C} \rightarrow \mathbf{P}$. Compared with Office-31, this dataset is more balanced and can be a good comparable dataset to Office-31.

B. Implementation Details

We mainly compare three methods: 1) **Two_stage**: which is the most straightforward method that applies channel pruning to the source domain task first, then fine-tune for the target domain task with the pruned model. 2) **TCP_w/o_DA**: Our TCP method without the MMD loss, here we call it domain adaptation (DA) loss. Which also means $\beta = 0$ all the time in Eq. (9). 3) **TCP**: Our full TCP method with DA loss.

We evaluate all the methods on two popular backbone networks: VGG16 [6] and ResNet50 [7]. As baselines, VGG16-based and ResNet50-based are the original models that are not pruned. As for VGG16-based model, it has 13 convolutional layers and 3 fully-connected layers. We prune all the convolutional layers and the first fully-connected layer and we only use the activations of the second fully-connected layer as image representation and build the MMD loss which is shown in Fig. 2. And as for ResNet50-based model, we use similar settings as VGG16-based model with a few differences. Because of the shortcut and residual branch structure, we only prune the inside convolutional layers of each bottleneck block. The MMD loss is built with the only fully-connected layer. Moreover, we also take the Batch Normalization (BN) [40] layers into consideration and reconstruct the whole model during pruning.

¹<http://imageclef.org/2014/adaptation>

TABLE I

The performance on Office-31 dataset (VGG16-based and ResNet50-based). Here, $FLOPs \downarrow$ and $Param \downarrow$ denote the decrement of FLOPs and parameter size compared with the baseline, Acc means the accuracy on target domain.

Models	FLOPs↓	A→W		D→W		W→D		A→D		D→A		W→A		Average	
		Acc	Param↓												
VGG16-base		74.0%		94.0%		97.5%		72.3%		54.1%		55.2%		74.5%	
Two_stage		69.4%	29.4%	94.5%	31.2%	99.0%	30.6%	69.8%	29.3%	42.7%	32.5%	47.2%	28.3%	70.4%	30.2%
TCP_w/o_DA	26%	73.0%	29.7%	95.5%	32.7%	99.3%	29.2%	75.8%	25.8%	45.9%	29.3%	50.4%	29.6%	73.3%	29.4%
TCP		76.1%	36.8%	96.1%	36.2%	99.8%	32.6%	76.2%	35.9%	47.9%	37.1%	51.2%	39.8%	74.5%	36.4%
Two_stage		57.1%	63.3%	88.1%	68.0%	96.3%	64.5%	55.0%	61.0%	31.8%	66.2%	32.7%	63.1%	60.2%	64.3%
TCP_w/o_DA	70%	53.5%	62.5%	89.2%	61.3%	97.9%	57.5%	61.8%	54.8%	35.3%	62.6%	34.5%	58.8%	62.0%	59.6%
TCP		74.1%	69.3%	89.5%	69.8%	98.8%	65.2%	65.9%	66.2%	35.6%	68.5%	38.5%	68.2%	67.1%	67.9%
ResNet50-base		80.3%		97.1%		99.2%		78.9%		64.3%		62.3%		80.3%	
Two_stage		75.8%	32.4%	96.7%	31.4%	99.5%	35.5%	76.0%	30.4%	48.0%	28.3%	50.1%	29.4%	74.4%	31.2%
TCP_w/o_DA	12%	79.8%	33.5%	97.0%	35.5%	100%	34.5%	77.1%	36.2%	47.8%	34.5%	52.6%	33.1%	75.7%	34.5%
TCP		81.8%	37.7%	98.2%	36.2%	99.8%	37.0%	77.9%	36.9%	50.0%	35.0%	55.5%	36.9%	77.2%	36.7%
Two_stage		65.5%	56.2%	93.0%	56.3%	98.7%	57.3%	64.9%	56.0%	34.0%	57.2%	38.9%	57.3%	65.8%	56.7%
TCP_w/o_DA	46%	75.1%	56.4%	95.8%	56.4%	99.2%	56.6%	70.8%	55.8%	34.2%	56.4%	41.5%	56.6%	69.4%	56.4%
TCP		77.4%	58.4%	96.3%	58.0%	100%	57.1%	72.0%	59.0%	36.1%	57.8%	46.3%	58.5%	71.3%	58.1%

TABLE II

The performance on ImageCLEF-DA dataset (VGG16-based and ResNet50-based).

Models	FLOPs↓	I→P		P→I		I→C		C→I		C→P		P→C		Average	
		Acc	Param↓												
VGG16-base		71.3%		80.0%		88.5%		77.0%		61.1%		87.2%		77.5%	
Two_stage		68.0%	29.0%	77.7%	29.9%	88.5%	27.5%	64.5%	29.0%	56.0%	29.3%	83.3%	26.9%	73.0%	28.6%
TCP_w/o_DA	26%	70.5%	33.6%	79.5%	34.2%	89.0%	33.1%	77.1%	34.5%	62.2%	35.0%	85.1%	33.1%	77.2%	33.9%
TCP		72.0%	39.0%	80.5%	32.2%	90.5%	35.1%	77.8%	36.3%	64.8%	36.6%	87.5%	34.5%	78.9%	35.6%
Two_stage		58.6%	65.8%	69.2%	62.9%	80.6%	64.3%	57.7%	57.0%	43.2%	67.8%	74.5%	61.5%	63.9%	63.2%
TCP_w/o_DA	70%	61.0%	55.4%	69.1%	65.7%	80.5%	66.5%	55.1%	63.3%	47.0%	66.5%	70.9%	65.8%	63.9%	63.9%
TCP		61.9%	66.7%	69.5%	66.0%	81.8%	65.7%	59.8%	68.7%	49.7%	68.8%	75.9%	67.2%	66.4%	67.1%
ResNet50-base		74.8%		82.2%		92.3%		83.3%		70.0%		89.8%		82.1%	
Two_stage		71.8%	29.4%	81.3%	31.5%	92.1%	34.4%	76.5%	32.4%	64.0%	29.2%	84.0%	30.8%	78.2%	31.2%
TCP_w/o_DA	12%	73.0%	33.5%	80.5%	34.6%	92.0%	33.8%	76.1%	31.2%	64.3%	30.1%	86.3%	36.3%	78.7%	33.2%
TCP		75.0%	37.5%	82.6%	36.5%	92.5%	35.5%	80.8%	36.7%	66.2%	36.6%	86.5%	37.6%	80.6%	36.7%
Two_stage		65.6%	53.2%	71.8%	56.5%	85.2%	54.2%	68.2%	54.0%	57.4%	51.5%	78.1%	54.0%	71.1%	53.9%
TCP_w/o_DA	46%	66.6%	55.4%	73.0%	57.4%	85.5%	55.4%	67.7%	55.5%	55.5%	53.6%	77.0%	57.1%	70.8%	55.7%
TCP		67.8%	57.2%	77.5%	58.0%	88.6%	56.2%	71.6%	58.5%	57.7%	55.7%	79.5%	58.2%	73.8%	57.3%

In practice, all the input images are cropped to a fixed size 224×224 and randomly sampled from the resized image with horizontal flip and mean-std normalization. At first, we fine-tune all the UDA models on each unsupervised domain adaptation tasks for 200 epochs with learning rate from 0.01 to 0.0001 and the batch size = 32. During pruning, we set $\mathcal{K} = 64$ which means 64 channels will be removed after each pruning iteration. After that, extra 5 epochs are adopted to help the pruned model to converge. And we follow [41] to prune the baseline with different compression rate and make the compression rate as different as possible. The VGG16-based baseline is pruned with 26% and 70% FLOPs reduced while the ResNet50-based baseline is pruned with 12% and 46% FLOPs reduced. ResNet50 has lower compression rate since the bottleneck structure stops some layers from being pruned.

We follow standard evaluation protocol for UDA and use all source examples with labels and all target examples without labels [42]. The labels for the target domain are only used for evaluation. We adopt *classification accuracy* on the target domain and *parameter reduction* as the evaluation metrics: higher accuracy and fewer parameters indicate better performance.

C. Results and Analysis

Firstly, we evaluate all the tasks on Office-31 dataset. The results are shown in TABLE I. As can be seen, our TCP

method outperforms other methods under the same compression rate (FLOPs reduction) and can reduce more parameters. And the FLOPs in convolutional layers is calculated by:

$$FLOPs = HWC_{in}K^2C_{out}, \quad (12)$$

where H, W, C_{out} is the height, width and channel number of output feature map, K is the kernel size, C_{in} refers to the number of input channels, and the bias item is ignored due to its small contribution. Moreover, It is important and interesting that TCP achieves even better performance than the baseline model (which is not pruned). This is probably because some redundant channels in the base model are removed thus negative transfer is reduced. Especially for the results of ResNet50-based models, our baseline is almost the same as the result of DAN in [3]. However, we can get better performance on half of the tasks and we even get 100% on task W→D after 46% FLOPs have been reduced.

Secondly, we evaluate our methods on ImageCLEF-DA dataset and the results are shown in TABLE II. We can draw the same conclusion that TCP performs better on all tasks on ImageCLEF-DA dataset. We get higher accuracy than the baseline on all the VGG16-based experiments after 26% FLOPs have been reduced, and we also get higher accuracy on the target dataset on half of the tasks on ResNet50-based experiments after 12% FLOPs have been reduced, compared with the baseline which is almost the same as DAN [3].

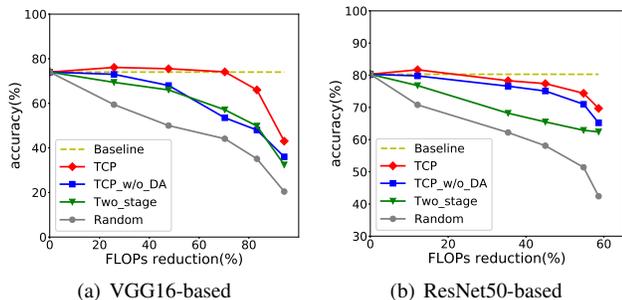


Fig. 4. The pruning result on task A→W with more compression rate.

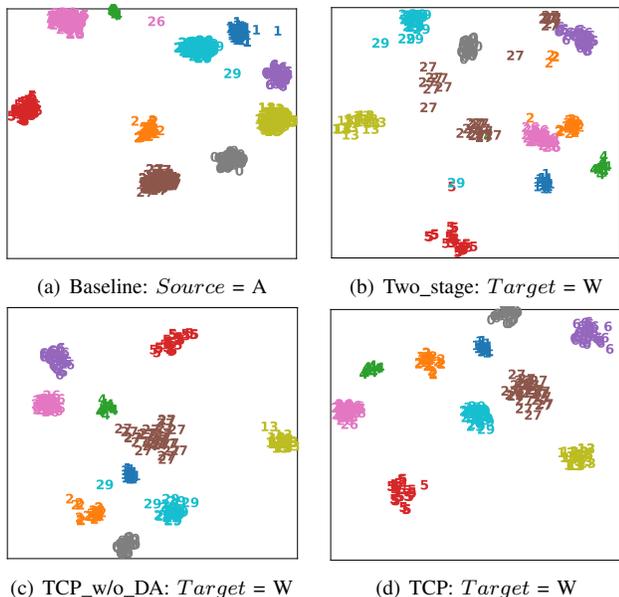
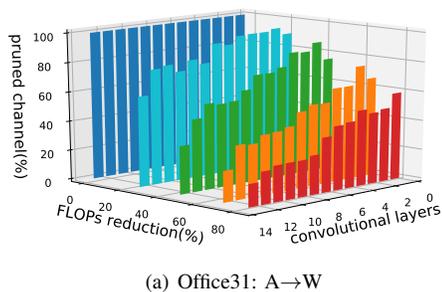
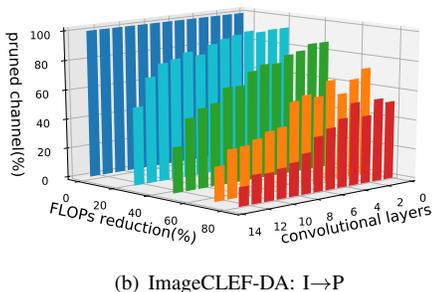


Fig. 5. The t-SNE visualization of network activations. (a) is generated by ResNet50-based baseline without pruning on source domain. (b)(c)(d) are generated by ResNet50-based (with 12% FLOPs pruned) with our three methods on target domain respectively. Best view in color.

Apart from TABLE I and TABLE II, Fig. 4 shows the comparison for all methods. And we also add a Random method which randomly removes a certain number of channels to achieve the same reduction of FLOPs. Combining these results, more conclusions can be made. 1) Compared with Two_stage, TCP is more efficient because it is a unified framework and treat the pruning as a single optimization problem, while Two_stage is a split method and it does not take the target domain into consideration while pruning. 2) Compared with TCP_w/o_DA, the full TCP uses the transfer channel evaluation to represent the discrepancy between the source and target domains. We try to remove those less important channels for both source and target domains and reduce negative transfer by reducing domain discrepancy. 3) As can be seen from Fig. 4, our TCP outperforms other methods on unlabeled target dataset under different compression rate. 4) This indicates that TCP is generic, accurate, and efficient, which can dramatically reduce the computational cost of a deep UDA model without sacrificing the performance.



(a) Office31: A→W



(b) ImageCLEF-DA: I→P

Fig. 6. The pruned structure of all the 13 convolutional layers of VGG16-based network on different dataset for deep unsupervised domain adaptation. Best view in color.

D. Effectiveness Analysis

Visualization analysis. To evaluate the effectiveness of TCP in reducing negative transfer, in Fig. 5, we follow [43] to visualize the model activations of task A→W pruned by different methods using t-SNE [43]. Fig. 5(a) shows the results of ResNet50-based baseline without pruning on the source domain. And Fig. 5(b), Fig. 5(c) and Fig. 5(d) denote the result of ResNet50-based models on the target domain, which have been pruned by 12% FLOPs with our three methods Two_stage, TCP_w/o_DA and TCP respectively. The colored digits represent the ground truth of the examples, so the number is from 0 to 30, which denotes target dataset has 31 categories. Here we randomly pick 10 categories to visualize. As can be seen, the target categories are discriminated more clearly with the model pruned by our TCP method. This suggests that our TCP method is effective in learning more transferable features by reducing the cross-domain divergence.

Pruned structure analysis. To explore if there is any pattern in the structure of the pruned models, we show the structure of pruned models on task A→W and I→P in Fig. 6 with TCP. As we can see, higher layers have more redundancy than lower layers in VGG16-based models, and our TCP prefer pruning the higher layers. This is reasonable for UDA because the lower layers usually encode common and important features for both source and target domains. Moreover, because there are more parameters in higher layers of CNN, especially the first fully-connected layer in VGG16, our TCP thus can prune more parameters under almost the same compression rate. The same result can be observed on ResNet50-based models.

V. CONCLUSION AND FUTURE WORK

In this paper, we propose a unified Transfer Channel Pruning (TCP) approach for accelerating deep unsupervised domain adaptation models. TCP is capable of compressing the deep UDA model by pruning less important channels while simultaneously learning transferable features by reducing the cross-domain distribution divergence. Therefore, it reduces the impact of negative transfer and maintains competitive performance on the target task. TCP is a generic, accurate, and efficient compression method that can be easily implemented by most deep learning libraries. Experiments on two public benchmark datasets demonstrate the significant superiority of our TCP method over other methods.

In the future, we plan to extend TCP in pruning adversarial networks and apply it to heterogeneous UDA problems.

VI. ACKNOWLEDGMENT

This work is supported in part by National Key R & D Plan of China (No.2017YFB1002800), NSFC (No.61572471), and Beijing Municipal Science & Technology Commission (No.Z171100000117017).

REFERENCES

- [1] S. J. Pan, Q. Yang *et al.*, “A survey on transfer learning,” *IEEE Transactions on knowledge and data engineering*, vol. 22, no. 10, pp. 1345–1359, 2010.
- [2] H. J. Zhao and J. Liu, “Finding answers from the word of god: Domain adaptation for neural networks in biblical question answering,” in *IJCNN*. IEEE, 2018, pp. 1–8.
- [3] M. Long, Y. Cao, J. Wang, and M. I. Jordan, “Learning transferable features with deep adaptation networks,” in *ICML*, 2015.
- [4] H. Venkateswara, J. Eusebio, S. Chakraborty, and S. Panchanathan, “Deep hashing network for unsupervised domain adaptation,” in *Proc. CVPR*, 2017, pp. 5018–5027.
- [5] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “Imagenet classification with deep convolutional neural networks,” in *NIPS*, 2012, pp. 1097–1105.
- [6] K. Simonyan and A. Zisserman, “Very deep convolutional networks for large-scale image recognition,” in *ICLR*, 2015.
- [7] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in *CVPR*, 2016, pp. 770–778.
- [8] J. Wang, Y. Chen, L. Hu, X. Peng, and S. Y. Philip, “Stratified transfer learning for cross-domain activity recognition,” in *2018 IEEE International Conference on Pervasive Computing and Communications (PerCom)*. IEEE, 2018, pp. 1–10.
- [9] S. J. Pan, I. W. Tsang, J. T. Kwok, and Q. Yang, “Domain adaptation via transfer component analysis,” *IEEE Transactions on Neural Networks*, vol. 22, no. 2, pp. 199–210, 2011.
- [10] M. Rastegari, V. Ordonez, J. Redmon, and A. Farhadi, “Xnor-net: Imagenet classification using binary convolutional neural networks,” in *ECCV*. Springer, 2016, pp. 525–542.
- [11] A. Zhou, A. Yao, Y. Guo, L. Xu, and Y. Chen, “Incremental network quantization: Towards lossless cnns with low-precision weights,” *arXiv preprint arXiv:1702.03044*, 2017.
- [12] S. Han, J. Pool, J. Tran, and W. Dally, “Learning both weights and connections for efficient neural network,” in *NIPS*, 2015, pp. 1135–1143.
- [13] Y. He, X. Zhang, and J. Sun, “Channel pruning for accelerating very deep neural networks,” in *ICCV*, vol. 2, no. 6, 2017.
- [14] P. Molchanov, S. Tyree, T. Karras, T. Aila, and J. Kautz, “Pruning convolutional neural networks for resource efficient inference,” in *ICLR*, 2017.
- [15] E. L. Denton, W. Zaremba, J. Bruna, Y. LeCun, and R. Fergus, “Exploiting linear structure within convolutional networks for efficient evaluation,” in *NIPS*, 2014, pp. 1269–1277.
- [16] S. Han, H. Mao, and W. J. Dally, “Deep compression: Compressing deep neural networks with pruning, trained quantization and Huffman coding,” *arXiv preprint arXiv:1510.00149*, 2015.
- [17] N. Yu, S. Qiu, X. Hu, and J. Li, “Accelerating convolutional neural networks by group-wise 2d-filter pruning,” in *IJCNN*. IEEE, 2017, pp. 2502–2509.
- [18] B. Fernando, A. Habrard, M. Sebban, and T. Tuytelaars, “Unsupervised visual domain adaptation using subspace alignment,” in *ICCV*, 2013, pp. 2960–2967.
- [19] B. Sun and K. Saenko, “Subspace distribution alignment for unsupervised domain adaptation,” in *BMVC*, 2015, pp. 24–1.
- [20] B. Sun, J. Feng, and K. Saenko, “Return of frustratingly easy domain adaptation,” in *AAAI*, vol. 6, no. 7, 2016, p. 8.
- [21] M. Long, J. Wang, G. Ding, J. Sun, and P. S. Yu, “Transfer feature learning with joint distribution adaptation,” in *ICCV*, 2013, pp. 2200–2207.
- [22] C.-A. Hou, Y.-H. H. Tsai, Y.-R. Yeh, and Y.-C. F. Wang, “Unsupervised domain adaptation with label and structural consistency,” *IEEE Transactions on Image Processing*, vol. 25, no. 12, pp. 5552–5562, 2016.
- [23] J. Tahmoresnezhad and S. Hashemi, “Visual domain adaptation via transfer feature learning,” *Knowledge and Information Systems*, vol. 50, no. 2, pp. 585–605, 2017.
- [24] J. Wang, W. Feng, Y. Chen, H. Yu, M. Huang, and P. S. Yu, “Visual domain adaptation with manifold embedded distribution alignment,” in *ACM MM*. ACM, 2018, pp. 402–410.
- [25] J. Wang, Y. Chen, S. Hao, W. Feng, and Z. Shen, “Balanced distribution adaptation for transfer learning,” in *ICDM*. IEEE, 2017, pp. 1129–1134.
- [26] Y. Bengio, A. Courville, and P. Vincent, “Representation learning: A review and new perspectives,” *IEEE transactions on pattern analysis and machine intelligence*, vol. 35, no. 8, pp. 1798–1828, 2013.
- [27] M. Oquab, L. Bottou, I. Laptev, and J. Sivic, “Learning and transferring mid-level image representations using convolutional neural networks,” in *CVPR*, 2014, pp. 1717–1724.
- [28] X. Glorot, A. Bordes, and Y. Bengio, “Domain adaptation for large-scale sentiment classification: A deep learning approach,” in *ICML*, 2011, pp. 513–520.
- [29] A. Kumar, P. Sattigeri, K. Wadhawan, L. Karlinsky, R. Feris, B. Freeman, and G. Wornell, “Co-regularized alignment for unsupervised domain adaptation,” in *NIPS*, 2018, pp. 9366–9377.
- [30] Y. Ganin, E. Ustinova, H. Ajakan, P. Germain, H. Larochelle, F. Laviolette, M. Marchand, and V. Lempitsky, “Domain-adversarial training of neural networks,” *The Journal of Machine Learning Research*, vol. 17, no. 1, pp. 2096–2030, 2016.
- [31] E. Tzeng, J. Hoffman, K. Saenko, and T. Darrell, “Adversarial discriminative domain adaptation,” in *CVPR*, vol. 1, no. 2, 2017, p. 4.
- [32] J. Hoffman, E. Tzeng, T. Park, J.-Y. Zhu, P. Isola, K. Saenko, A. A. Efros, and T. Darrell, “Cycada: Cycle-consistent adversarial domain adaptation,” in *ICML*, 2018.
- [33] F. Chollet, “Xception: Deep learning with depthwise separable convolutions,” *arXiv preprint*, pp. 1610–02357, 2017.
- [34] J. Lin, Y. Rao, J. Lu, and J. Zhou, “Runtime neural pruning,” in *Advances in Neural Information Processing Systems*, 2017, pp. 2181–2191.
- [35] J.-H. Luo, J. Wu, and W. Lin, “Thinet: A filter level pruning method for deep neural network compression,” *arXiv preprint arXiv:1707.06342*, 2017.
- [36] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson, “How transferable are features in deep neural networks?” in *NIPS*, 2014, pp. 3320–3328.
- [37] S. J. Pan, J. T. Kwok, and Q. Yang, “Transfer learning via dimensionality reduction,” in *AAAI*, vol. 8, 2008, pp. 677–682.
- [38] K. Saenko, B. Kulis, M. Fritz, and T. Darrell, “Adapting visual category models to new domains,” in *ECCV*. Springer, 2010, pp. 213–226.
- [39] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, “Automatic differentiation in pytorch,” 2017.
- [40] S. Ioffe and C. Szegedy, “Batch normalization: Accelerating deep network training by reducing internal covariate shift,” in *ICML*, 2015.
- [41] Y. Hu, S. Sun, J. Li, X. Wang, and Q. Gu, “A novel channel pruning method for deep neural network compression,” *arXiv preprint arXiv:1805.11394*, 2018.
- [42] B. Gong, K. Grauman, and F. Sha, “Connecting the dots with landmarks: Discriminatively learning domain-invariant features for unsupervised domain adaptation,” in *ICML*, 2013, pp. 222–230.
- [43] J. Donahue, Y. Jia, O. Vinyals, J. Hoffman, N. Zhang, E. Tzeng, and T. Darrell, “Decaf: A deep convolutional activation feature for generic visual recognition,” in *ICML*, 2014, pp. 647–655.