Slimmable Domain Adaptation

Rang Meng[2], Weijie Chen[1,2,†], Shicai Yang[2], Jie Song[1], Luojun Lin[3], Di Xie[2]
Shiliang Pu[2], Xinchao Wang[4], Mingli Song[1], Yueting Zhuang[1,†]

[1]Zhejiang University, [2]Hikvision Research Institute, [3]Fuzhou University, [4]National University of Singapore

{mengrang, chenweijie5,yangshicai,xiedi,pushiliang.hri}@hikvision.com

{sjie,songml,yzhuang}@zju.edu.cn, linluojun2009@126.com, xinchao@nus.edu.sg

Abstract

Vanilla unsupervised domain adaptation methods tend to optimize the model with fixed neural architecture, which is not very practical in real-world scenarios since the target data is usually processed by different resource-limited devices. It is therefore of great necessity to facilitate architecture adaptation across various devices. In this paper, we introduce a simple framework, Slimmable Domain Adaptation, to improve cross-domain generalization with a weight-sharing model bank, from which models of different capacities can be sampled to accommodate different accuracy-efficiency trade-offs. The main challenge in this framework lies in simultaneously boosting the adaptation performance of numerous models in the model bank. To tackle this problem, we develop a Stochastic EnsEmble Distillation method to fully exploit the complementary knowledge in the model bank for inter-model interaction. Nevertheless, considering the optimization conflict between inter-model interaction and intra-model adaptation, we augment the existing bi-classifier domain confusion architecture into an Optimization-Separated Tri-Classifier counterpart. After optimizing the model bank, architecture adaptation is leveraged via our proposed Unsupervised Performance Evaluation Metric. Under various resource constraints, our framework surpasses other competing approaches by a very large margin on multiple benchmarks. It is also worth emphasizing that our framework can preserve the performance improvement against the source-only model even when the computing complexity is reduced to $1/64$. Code will be available at https://github.com/HIK-LAB/SlimDA.

1. Introduction

Deep neural networks are usually trained on the offline-collected images (labeled source data) and then embedded
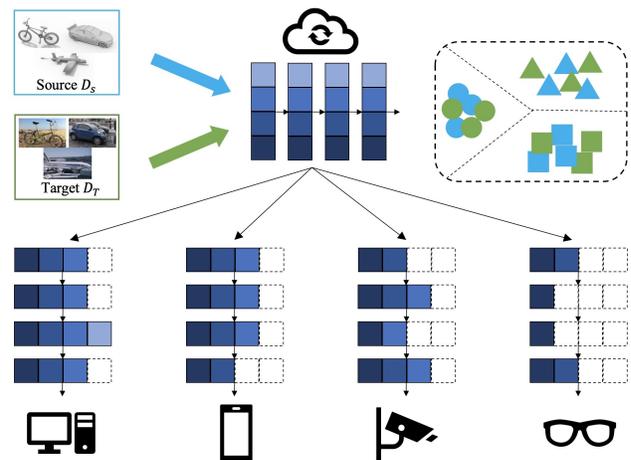
†Corresponding author

Figure 1. SlimDA: We only adapt once on cloud computing center but can flexibly sample models with diverse capacities to distribute to different resource-limited edge devices.

in edge devices to test the images sampled from new scenarios (unlabeled target data). This paradigm, in practice, degrades the network performance due to the domain shift. Recently, more and more researchers have delved into unsupervised domain adaptation (UDA) to address this problem.

Vanilla UDA aims to align source data and target data into a joint representation space so that the model trained on source data can be well generalized to target data [7,11,20,26,29,38,44]. Unfortunately, there is still a gap between academic studies and industrial needs: most existing UDA methods only perform weight adaptation with fixed neural architecture yet cannot fit the requirements of various devices in the real-world applications efficiently. Taking the example of a widely-used application scenario as shown in Fig.1, a domain adaptive model trained on a powerful cloud computing center is urged to be distributed to different resource-limited edge devices like laptops, smart mobile phones, and smartwatches, for real-time processing. In this scenario, vanilla UDA methods have to train a series of models with different capacities and architectures time-and-again to fit the requirements of devices with dif-

Methods	Data Type	Teacher	Student	Model
CKD	Labeled	Single	Single	Fixed
SEED	Unlabeled	Multiple	Multiple	Stochastic

Table 1. Conventional Knowledge Distillation (CKD) vs. Stochastic EnsEmble Distillation (SEED).

ferent computation budgets, which is expensive and time-consuming. To remedy the aforementioned issue, we propose Slimmable Domain Adaptation (SlimDA), in which we only train our model once so that the customized models with different capacities and architectures can be sampled flexibly from it to supply the demand of devices with different computation budgets.

Although slimmable neural networks [54–56] had been studied in the supervised tasks, in which models with different layer widths (*i.e.*, channel number) can be coupled into a weight-sharing model bank for optimization, there remain two challenges when slimmable neural networks meet unsupervised domain adaptation: 1) *Weight adaptation*: How to simultaneously boost the adaptation performance of all models in the model bank? 2) *Architecture adaptation*: Given a specific computational budget, how to search an appropriate model on the unlabeled target data?

For the first challenge, there is a straightforward baseline in which UDA methods are directly applied to each model sampled from the model bank. However, this paradigm neglects to exploit the complementary knowledge among tremendous neural architectures in the model bank. To remedy this issue, we propose Stochastic EnsEmble Distillation (SEED) to interact the models in the model bank so as to suppress the uncertainty of intra-model adaptation on the unlabeled target data. SEED is a curriculum mutual learning framework in which the expectation of the predictions from stochastically-sampled models are exploited to assist domain adaptation of the model bank. The differences between SEED and the conventional knowledge distillation are shown in Table 1. As for intra-model adaptation, we borrow the solution from the state-of-the-art bi-classifier-based domain confusion method (such as SymNet [57] and MCD [38]). Nevertheless, we analyze that there exists an optimization conflict between inter-model interaction and intra-model adaptation, which motivates us to augment an Optimization-Separated Tri-Classifier (OSTC) to modulate the optimization between them.

For the second challenge, it is intuitive to search models with optimal adaptation performance under different computational budgets after training the model bank. However, unlike performance evaluation in the supervised tasks, none of the labeled target data is available. To be compatible with the unlabeled target data, we exploit the model with the largest capacity as an anchor to guide performance ranking in the model bank, since the larger models tend to be

more accurate as empirically proven in [52]. In this way, we propose an Unsupervised Performance Evaluation Metric which is eased into the output discrepancy between the candidate model and the anchor model. The smaller the metric is, the better the performance is assumed to be.

Extensive ablation studies and experiments are carried out on three popular UDA benchmarks, *i.e.*, ImageCLEF-DA [27], Office-31 [36], and Office-Home [45], which demonstrate the effectiveness of the proposed framework. Our method can achieve state-of-the-art results compared with other competing methods. It is worth emphasizing that our method can preserve the performance improvement against the source-only model even when the computing complexity is reduced to $1/64\times$. To summarize, our main contributions are listed as follows:

- We propose SlimDA, a "once-for-all" framework to jointly accommodate the adaptation performance and the computation budgets for resource-limited devices.

- We propose SEED to simultaneously boost the adaptation performance of all models in the model bank. In particular, we design an Optimization-Separated Tri-Classifier to modulate the optimization between intra-model adaptation and inter-model interaction.

- We propose an Unsupervised Performance Evaluation Metric to facilitate architecture adaptation.

- Extensive experiments verify the effectiveness of our proposed SlimDA framework, which can surpass other state-of-the-art methods by a large margin.

2. Related Work

2.1. Unsupervised Domain Adaptation

Existing UDA methods aim to improve the model performance on the unlabeled target domain. In the past few years, discrepancy-based methods [14, 26, 41] and adversarial optimization methods [1, 11, 18, 25, 38] are proposed to solve this problem via domain alignment. Specifically, SymNet [57] develops a bi-classifier architecture to facilitate category-level domain confusion. Recently, Li *et.al.* [21] attempts to learn optimal architectures to further boost the performance on the target domain, which proves the significance of network architecture for UDA. These UDA methods focus on achieving a specific model with better performance on the target domain.

2.2. Neural Architecture Search

Neural Architecture Search (NAS) methods aim to search for optimal architectures automatically through reinforcement learning [4, 42, 43, 58, 59], evolution methods [9, 24, 34, 35], gradient-based methods [23, 30, 40, 47, 49] and so on. Recently, one-shot methods [2, 3, 13, 33, 49, 54] are very popular since only one super-network is required to

train, and numerous weight-sharing sub-networks of various architectures are optimized simultaneously. In this way, the optimal network architecture can be searched from the model bank. In this paper, we highlight that UDA is an unnoticed yet significant scenario for NAS, since they can be cooperated to optimize a scene-specific lightweight architecture in an unsupervised way.

2.3. Cross-domain Network Compression

Chen *et.al.* [6] proposes a cross-domain unstructured pruning method. Yu *et.al.* [53] adopts MMD [26] to minimize domain discrepancy and prunes filters in a Taylor-based strategy, and Yang *et.al.* [50,51] focuses on compressing graph neural networks. Feng *et.al.* [10] conducts adversarial training between the channel-pruned network and the full-size network. However, the performances of the existing methods still have a great improvement space. Moreover, their methods are not flexible enough to obtain numerous optimal models under diverse resource constraints.

3. Preliminary

3.1. Bi-Classifier Based Domain Confusion

3.1.1 Notation

A labeled source data $\mathcal{D}_s = \{(x_i^s, y_i^s)\}_{i=1}^{n_s}$ and an unlabeled target data $\mathcal{D}_t = \{(x_i^t)\}_{i=1}^{n_t}$ are provided for training. SymNet [57] is composed of a feature extractor F and two task classifiers C^s and C^t. A novel design in SymNet is to construct a new classifier C^{st} which shares the neurons with C^s and C^t. C^{st} is designed for domain discrimination and domain confusion without an explicit domain discriminator. The probability outputs of C^s, C^t and C^{st} are $\mathbf{g}(x; F, C^s) \in [0,1]^K$, $\mathbf{g}(x; F, C^t) \in [0,1]^K$ and $\mathbf{g}(x; F, C^{st}) \in [0,1]^{2K}$ respectively, where K is the class number of the task. The kth element of the probability output can be written as $\mathbf{g}_k^s(x)$, $\mathbf{g}_k^t(x)$ and $\mathbf{g}_k^{st}(x)$, respectively.

3.1.2 Task and Domain Discrimination

The training objective of task discrimination for $C^s \& C^t$ is:

$$\min_{C^s, C^t} -\frac{1}{n_s} \sum_{i=1}^{n_s} \log(\mathbf{g}_{y_i^s}^s(x_i^s)) - \frac{1}{n_s} \sum_{i=1}^{n_s} \log(\mathbf{g}_{y_i^s}^t(x_i^s)) \quad (1)$$

The training objective of domain discrimination for C^{st} is:

$$\min_{C^{st}} -\frac{1}{n_s} \sum_{i=1}^{n_s} \log \sum_{k=1}^{K} \mathbf{g}_k^{st}(x_i^s) - \frac{1}{n_t} \sum_{i=1}^{n_t} \log \sum_{k=1}^{K} \mathbf{g}_{k+K}^{st}(x_i^t) \quad (2)$$

3.1.3 Category-Level Domain Confusion

The training objective of category-level confusion is:

$$\min_{F} -\frac{1}{2n_s} \sum_{i=1}^{n_s} \log(\mathbf{g}_{y_i^s}^{st}(x_i^s)) - \frac{1}{2n_s} \sum_{i=1}^{n_s} \log(\mathbf{g}_{y_i^s+K}^{st}(x_i^s)) \quad (3)$$

The training objective of domain-level confusion is:

$$\min_{F} -\frac{1}{2n_t} \sum_{i=1}^{n_t} \log(\sum_{k=1}^{K} \mathbf{g}_k^{st}(x_i^t)) - \frac{1}{2n_t} \sum_{i=1}^{n_t} \log(\sum_{k=1}^{K} \mathbf{g}_{k+K}^{st}(x_i^t)) \quad (4)$$

Besides, an entropy minimization loss is conducted on $\mathcal{D}_t$ to optimize F. For more detailed technical illustration, we recommend referring to the original paper.

4. Method

4.1. Straightforward Baseline

It has been proven in slimmable neural networks that numerous networks with different widths (i.e., layer channel) can be coupled into a weight-sharing model bank and be optimized simultaneously. We begin with a baseline in which SymNet is straightforwardly merged with the slimmable neural networks. The overall objective of SymNet is unified as $\mathcal{L}_{dc}$ for simplicity. In each training iteration, several models can be stochastically sampled from the model bank $\{(F_j, C_j^s, C_j^t)\}_{j=1}^m \in (F, C^s, C^t)$, named as model batch, where m represents the model batch size. Here (F, C^s, C^t) can be viewed as the largest model, and the remaining models can be sampled from it in a weight-sharing manner. To make sure the model bank can be fully trained, the largest and the smallest models[*] should be sampled and constituted as a part of model batch in each training iteration. (Note that each model should re-calculate the statistical parameters of BN layers before deploying).

$$\left(\frac{\partial \mathcal{L}_{dc}}{\partial C^s}, \frac{\partial \mathcal{L}_{dc}}{\partial C^t}\right) = \left(\frac{1}{m} \sum_{j=1}^m \frac{\partial \mathcal{L}_{dc}}{\partial C_j^s}, \frac{1}{m} \sum_{j=1}^m \frac{\partial \mathcal{L}_{dc}}{\partial C_j^t}\right) \quad (5)$$

$$\frac{\partial \mathcal{L}_{dc}}{\partial F} = \frac{1}{m} \sum_{j=1}^m \frac{\partial \mathcal{L}_{dc}}{\partial F_j} \quad (6)$$

This baseline can be viewed as two alternated processes of Eqn.5 and Eqn.6 to optimize the model bank. To encourage inter-model interaction in the above baseline, we propose our SlimDA framework as shown in Fig. 2.

4.2. Stochastic EnsEmble Distillation

Stochastic Ensemble: It is intuitive that different models in the model bank can learn complementary knowledge about the unlabeled target data. Inspired by Bayesian learning with model perturbation, we exploit the models in the model bank via Monte Carlo Sampling to suppress the uncertainty from unlabeled target data. The expected prediction $\mathbf{g}_{seed}(x_i^t)$ can be approximated by taking the expectation of $\{\mathbf{g}(x_i; F_j, C_j^s, C_j^t)\}_{j=1}^m$ with respect to the model confidence $\{\mathbf{g}(F_j, C_j^s, C_j^t)\}_{j=1}^m$ [†]:

$$\mathbf{g}_{seed}(x_i^t) = \mathbb{E}_{\mathbf{g}(F_j, C_j^s, C_j^t)}\left[\mathbf{g}(x_i^t; F_j, C_j^s, C_j^t)\right] \quad (7)$$

---

[*]The smallest model corresponds to $1/64\times$ FLOPs model ($1/8\times$ channels) by default in this paper.

[†]$\mathbf{g}(F_j, C_j^s, C_j^t)$ is short for $\mathbf{g}(F_j, C_j^s, C_j^t | \mathcal{D})$ where $\mathcal{D}$ denotes the training data. The model confidence, ranging [0,1], can be interpreted to measure the relative accuracy among the models in the model bank.
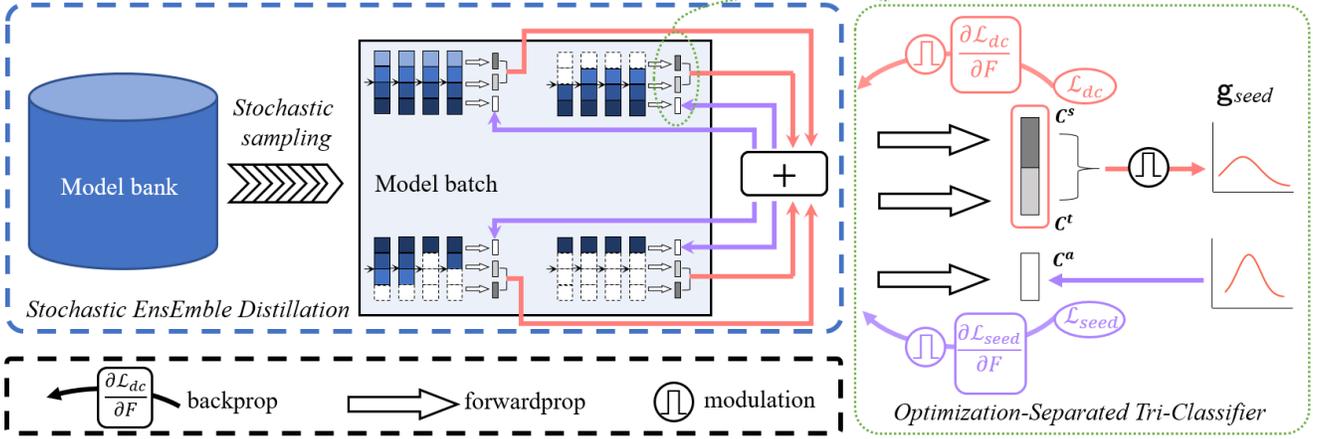
Figure 2. The training details of our proposed SlimDA framework. Our framework is composed of Stochastic EnsEmble Distillation (SEED) and an Optimization-Separated Tri-Classifier (OSTC) design. The SEED is designed to exploit the complementary knowledge in the model bank for multi-model interactions. The red arrows across C^s and C^t classifiers denote domain confusion training $\mathcal{L}_{dc}$ and knowledge aggregation in the model bank. The purple arrow across C^a classifier denotes SEED optimization $\mathcal{L}_{seed}$.

$$\mathbf{g}(x_i^t; F_j, C_j^s, C_j^t) = \frac{1}{2}(\mathbf{g}(x_i^t; F_j, C_j^s) + \mathbf{g}(x_i^t; F_j, C_j^t)) \quad (8)$$

where $\mathbb{E}$ is a weighted average function with $j=\{1, ..., m\}$, and the subscript of $\mathbb{E}$ denotes the weight.

Assumption 4.1 *As demonstrated by extensive empirical results in in-domain generalization work [52] and out-of-domain generalization work [5], the models with larger capacity[‡] perform more accurate than those with smaller capacity statistically. Thus, it is reasonable to assume that $\mathbf{g}(F_1, C_1^s, C_1^t) \geq \mathbf{g}(F_2, C_2^s, C_2^t) \geq ... \geq \mathbf{g}(F_m, C_m^s, C_m^t)$ where the index denotes the order of model capacity from large to small.*

In this work, we empirically define the model confidence in a hard way:

$$r_j = \frac{\mathcal{M}(F_j, C_j^s, C_j^t)}{\mathcal{M}(F, C^s, C^t)}$$
$$\Omega = \{(F_j, C_j^s, C_j^t) \text{ whose } r_j \geq \lambda\} \quad (9)$$
$$\mathbf{g}(F_j, C_j^s, C_j^t) = \begin{cases} 1, \text{ if } (F_j, C_j^s, C_j^t) \in \Omega \\ 0, \text{ otherwise} \end{cases}$$

where λ is set 0.5 by default, and $\mathcal{M}(\cdot)$ represents the model capacity. As the prediction tends to be uncertain on the unlabeled target data, we aim to produce lower-entropy predictions to boost the discrimination [12]. In this work, we apply a sharpening function to $\mathbf{g}_{seed}(x_i^t)$ to induce implicit entropy minimization during SEED training:

$$\mathbf{g}_{seed,k}(x_i^t) = \mathbf{g}_{seed,k}(x_i^t)^{\frac{1}{\tau}} / \sum_{k'=1}^{K} \mathbf{g}_{seed,k'}(x_i^t)^{\frac{1}{\tau}} \quad (10)$$

[‡]We use FLOPs as a metric to measure model capacity in this paper.

where τ is a temperature parameter for sharpening, and is set 0.5 by default in this paper. $\mathbf{g}_{seed}(x_i^t)$ is used to refine the model batch, along with domain confusion training in a curriculum mutual learning manner.

Distillation Bridged by Optimization-Separated Tri-Classifier: We cannot directly feed $\mathbf{g}_{seed}(x_i^t)$ back to the original bi-classifier for distillation since there exists optimization conflict between intra-model adaptation (Eqn.1-4) and inter-model interaction (distillation by $\mathbf{g}_{seed}(x_i^t)$) in the model bank, which are two asynchronous tasks. Specifically, in the q iteration, domain confusion bi-classifier of multi-models provides two-part information, including task discrimination and domain-confusion, and the two information are aggregated in $\mathbf{g}_{seed}^q(x_i^t)$. In the next $q+1$ iteration, the above information can be further updated via the bi-classifier training. However, if we transfer $\mathbf{g}_{seed}^q(x_i^t)$ back to the bi-classifier, $\mathbf{g}_{seed}^q(x_i^t)$ will offset the gains of the two information in $\mathbf{g}_{seed}^{q+1}(x_i^t)$ and hinder the refinement of $\mathbf{g}_{seed}(x_i^t)$. Thus, the curriculum learning in our SlimDA framework will be destroyed.

To this end, we introduce an Optimization-Separated Tri-Classifier (OSTC) $\{(C_j^s, C_j^t, C_j^a)\}_{j=1}^m \in (C^s, C^t, C^a)$, where the former two are preserved for domain confusion training, and the last one is designed to receive the stochastically-aggregated knowledge for distillation. The distillation loss is formulated as:

$$\mathcal{L}_{seed} = -\frac{1}{m \times n_t} \sum_{j=1}^{m} \sum_{i=1}^{n_t} \mathbf{g}_{seed}(x_i^t) \log(\mathbf{g}(x_i^t; F_j, C_j^a))$$
$$-\frac{1}{m \times n_s} \sum_{j=1}^{m} \sum_{i=1}^{n_s} \mathbf{1}_{y_i^s} \log(\mathbf{g}(x_i^s; F_j, C_j^a)) \quad (11)$$

Model batch size	I→P	P→I	I→C	C→I	C→P	P→C	Avg.
2 (w/ CKD)	71.8	83.3	93.0	82.2	67.3	89.2	81.1
2	73.5	88.3	92.2	87.1	69.3	91.5	83.7
4	74.3	89.0	92.6	87.5	69.8	92.0	84.2
6	75.3	90.2	94.1	88.3	71.7	94.2	85.6
8	76.1	89.9	94.9	88.1	71.7	94.2	85.5
10	78.3	90.7	95.8	88.3	71.8	94.8	86.6

Table 2. Two ablation studies on ImageCLEF-DA: 1) The second row represents the results from conventional knowledge distillation (CKD). 2) Comparison among different model batch sizes in SlimDA. We report the above results for $1/64\times$ models.

Baseline	SEED	OSTC	1×	1/2×	1/4×	1/8×	1/16×	1/32×	1/64×
✓			88.6	88.0	86.9	86.0	84.1	82.0	81.7
✓	✓		87.9	87.6	87.3	86.9	86.5	86.1	85.9
✓	✓	✓	88.9	88.7	88.8	88.4	88.3	87.2	86.6

Table 3. The ablation studies for the components in SlimDA on ImageCLEF-DA dataset.

Methods	1×	1/2×	1/4×	1/8×	1/16×	1/32×	1/64×
SymNet w/o SlimDA	78.9	77.0	76.7	74.8	71.2	68.2	69.3
SymNet w/ SlimDA	79.2	79.0	79.0	78.7	78.8	78.2	78.3
improvement	**0.3↑**	**2.0↑**	**2.3↑**	**3.9↑**	**7.6↑**	**10.0↑**	**9.0↑**
MCD w/o SlimDA	77.2	75.0	75.0	72.3	70.3	68.7	69.6
MCD w/ SlimDA	78.5	78.2	78.1	78.0	77.7	77.6	77.7
improvement	**1.3↑**	**3.2↑**	**3.1↑**	**5.7↑**	**7.4↑**	**8.9↑**	**8.1↑**
STAR w/o SlimDA	76.9	74.0	69.7	68.1	65.6	62.9	64.7
STAR w/ SlimDA	77.8	77.5	77.2	77.2	77.0	76.9	77.0
improvement	**0.9↑**	**3.5↑**	**7.5↑**	**9.1↑**	**11.4↑**	**14.0↑**	**12.3↑**

Table 4. Two ablation studies on ImageCLEF-DA for I→P adaptation task: 1) Comparison with different UDA methods injected into SlimDA, which indicates the universality of our framework. 2) Comparison with stand-alone networks (aka "w/o SlimDA"), which have the same architectures as our searched models but are trained individually outsides the model bank.

We optimize (C^s, C^t, C^a) with $\mathcal{L}_{dc}$ and $\mathcal{L}_{seed}$ losses:

$$\left(\frac{\partial \mathcal{L}_{dc}}{\partial C^s}, \frac{\partial \mathcal{L}_{dc}}{\partial C^t} \right) = \left(\frac{1}{m} \sum_{j=1}^{m} \frac{\partial \mathcal{L}_{dc}}{\partial C_j^s}, \frac{1}{m} \sum_{j=1}^{m} \frac{\partial \mathcal{L}_{dc}}{\partial C_j^t} \right)$$
$$\frac{\partial \mathcal{L}_{seed}}{\partial C^a} = \frac{1}{m} \sum_{j=1}^{m} \frac{\partial \mathcal{L}_{seed}}{\partial C_j^a} \quad (12)$$

As to optimize F, we use the model confidence in Eqn.9 to modulate the training objectives of $\mathcal{L}_{dc}$ and $\mathcal{L}_{seed}$:

$$\frac{\partial \mathcal{L}_{total}}{\partial F} = \mathbb{E}_{\mathbf{g}(F_j, C_j^s, C_j^t)} \left[\frac{\partial \mathcal{L}_{dc}}{\partial F_j} \right]$$
$$+ \mathbb{E}_{1 - \mathbf{g}(F_j, C_j^s, C_j^t)} \left[\frac{\partial \mathcal{L}_{seed}}{\partial F_j} \right] \quad (13)$$

To summarize, the OSTC in Eqn.12 and the feature extractor in Eqn.13 are optimized in an alternated way in each training iteration. Once finishing training, (C^s, C^t) are discarded and only C^a is retained to deploy more efficiently.

4.3. Unsupervised Performance Evaluation Metric

In the context of UDA, one of the challenging points is to evaluate the performance ranking of models on the unlabeled target data rather than the searching methods. According to the *Triangle Inequality Theorem*, we can obtain the relationship among the predictions of the candidate

Methods	#Params	FLOPs	I→P	P→I	I→C	C→I	C→P	P→C	Avg.
ShuffleNetV2 [32]	2.3M	146M	63.2	65.4	82.1	81.0	65.2	78.9	72.6
MobileNetV3 [19]	5.4M	219M	72.8	85.3	93.2	80.3	65.0	91.7	81.4
GhostNet [15]	5.2M	141M	75.8	89.5	95.5	86.1	70.2	94.0	85.2
MobileNetV2 [39]	3.5M	300M	76.0	90.6	95.1	87.0	69.1	95.1	85.5
EfficientNet_B0 [43]	5.3M	390M	76.5	88.5	96.5	87.3	71.3	94.0	85.6
SlimDA (1/8×ResNet-50)	4.0M	517M	78.7	91.7	97.2	90.5	75.8	96.2	88.4
SlimDA (1/64×ResNet-50)	1.6M	64M	78.3	90.7	95.8	88.3	71.8	94.8	86.6

Table 5. Performance comparison with different state-of-the-art lightweight networks on ImageCLEF-DA dataset.

model (F_j, C_j^a), the largest model (F, C^a), as well as the ground-truth label:

$$\| \mathbf{g}(\mathcal{D}_t; F_j, C_j^a) - GT(\mathcal{D}_t) \|_2^2 < \| \mathbf{g}(\mathcal{D}_t; F, C^a) - GT(\mathcal{D}_t) \|_2^2$$
$$+ \| \mathbf{g}(\mathcal{D}_t; F_j, C_j^a) - \mathbf{g}(\mathcal{D}_t; F, C^a) \|_2^2 \quad (14)$$

where $GT(\mathcal{D}_t)$ denotes the ground-truth label of the target data. According to the assumption 4.1, the model with the largest capacity tends to be the most accurate in the model bank, which means:

$$\| \mathbf{g}(\mathcal{D}_t; F_j, C_j^a) - GT(\mathcal{D}_t) \|_2^2 > \| \mathbf{g}(\mathcal{D}_t; F, C^a) - GT(\mathcal{D}_t) \|_2^2 \quad (15)$$

Combining Eqn.14 and Eqn.15, we can take the model with the largest capacity as an anchor to compare the performance of candidate models on the unlabeled target data. The Unsupervised Performance Evaluation Metric (UPEM) for each model can be written as:

$$\Delta_j = \| \mathbf{g}(\mathcal{D}^t; F_j, C_j^a) - \mathbf{g}(\mathcal{D}^t; F, C^a) \|_2^2 \quad (16)$$

where Δ_j is the $\mathcal{L}2$ distance between outputs of the candidate model and the anchor model. With the UPEM, we can use a greedy search method [33, 54] for neural architecture search (Note that we can also use other search methods, but this is not the deciding point in this paper).

5. Experiments

5.1. Dataset

ImageCLEF-DA [26] consists of 1,800 images with 12 categories over three domains: Caltech-256 (C), ImageNet ILSVRC 2012 (I), and Pascal VOC 2012 (P).
Office-31 [37] is a popular benchmark with about 4,110 images sharing 31 categories of daily objects from 3 domains: Amazon (A), Webcam (W) and DSLR (D).
Office-Home [46] contains 15,500 images sharing 65 categories of daily objects from 4 different domains: Art (Ar), Clipart (Cl), Product (Pr), and Real-World (Rw).

5.2. Model Bank Configurations

Following the exiting methods [6, 10, 53], we select ResNet-50 [16] as the main network to conduct the following experiments. Unlike these methods, the ResNet-50 adopted in this paper is a super-network that couples numerous models with different layer widths to form the model bank. Identical to these methods, the super-network should be firstly pre-trained on ImageNet and then fine-tuned on the downstream tasks.

Methods	#Params	FLOPs	I → P	P → I	I → C	C → I	C → P	P → C	Avg.	Δ
Source Only [57]	1×	1×	74.8	83.9	91.5	78.0	65.5	91.2	80.7	–
DAN [26]	1×	1×	74.5	82.2	92.8	86.3	69.2	89.8	82.5	–
RevGrad [11]	1×	1×	75.0	86.0	96.2	87.0	74.3	91.5	85.0	–
MCD [38] (impl.)	1×	1×	77.2	87.2	93.8	87.7	71.8	92.5	85.0	–
STAR [29] (impl.)	1×	1×	76.9	87.7	93.8	87.6	72.1	92.7	85.1	–
CDAN+E [28]	1×	1×	77.7	90.7	97.7	91.3	74.2	94.3	87.7	–
SymNets [57]	1×	1×	80.2	93.6	97.0	93.4	78.7	96.4	89.9	–
SymNets (impl.)	1×	1×	78.8	92.2	96.7	91.0	76.0	96.2	88.5	–
TCP [53]	1/1.7×	–	75.0	82.6	92.5	80.8	66.2	86.5	80.6	–
	1/2.5×	–	67.8	77.5	88.6	71.6	57.7	79.5	73.8	–
ADMP [10]	1/1.7×	–	77.3	90.2	90.2	95.8	88.9	73.7	86.3	–
	1/2.5×	–	77.0	89.5	95.5	88.9	72.3	91.2	85.7	–
SlimDA	1×	1×	79.2	92.3	97.5	91.2	76.7	96.5	88.9	–
	1/1.9×	1/2×	79.0	92.3	97.3	90.8	76.8	96.2	88.7	0.2↓
	1/3.9×	1/4×	79.0	92.2	97.3	90.8	77.2	96.3	88.8	0.1↓
	1/9.4×	1/8×	78.7	91.7	97.2	90.5	75.8	96.2	88.4	0.5↓
	1/12.8×	1/16×	78.8	91.5	97.3	90.2	76.0	96.2	88.3	0.6↓
	1/28.8×	1/32×	78.2	90.5	96.7	89.3	72.2	96.0	87.2	1.7↓
	1/64×	1/64×	78.3	90.7	95.8	88.3	71.8	94.8	86.6	2.3↓

Table 6. Performance on the ImageCLEF-DA dataset. "–" means that the results are not reported in the original paper. "impl." denotes our re-implementation using the released code. "Δ" indicates the performance gap between the searched model and ResNet-50 based model for each UDA method. TCP and ADMP are two related cross-domain network compression methods. We adapt the architectures for six adaptation tasks under seven computational constraints (FLOPs). Since the model architectures adapted for different tasks are different even with the same FLOPs, we calculate the parameter reduction (#Params) by averaging models in 6 adaptation tasks.

Methods	#Params	FLOPs	A → W	D→W	W→D	A→D	D→A	W→A	Avg.	Δ
Source Only [57]	1×	1×	79.9	96.8	99.5	84.1	64.5	66.4	81.9	–
Domain Confusion [17]	1×	1×	83.0	98.5	99.8	83.9	66.9	66.4	83.1	–
Domain Confusion+Em [17]	1×	1×	89.8	99.0	100.0	90.1	73.9	69.0	87.0	–
BNM [8]	1×	1×	91.5	98.9	100.0	90.3	70.9	71.6	87.1	–
DMP [31]	1×	1×	93.0	99.0	100.0	91.0	71.4	70.2	87.4	–
DMRL [48]	1×	1×	90.8	99.0	100.0	93.4	73.0	71.2	87.9	–
SymNets [57]	1×	1×	90.8	98.8	100.0	93.9	74.6	72.5	88.4	–
SymNets (impl.)	1×	1×	91.0	98.4	99.6	89.7	72.2	72.5	87.2	–
TCP [53]	1/1.7×	–	81.8	98.2	99.8	77.9	50.0	55.5	77.2	–
	1/2.5×	–	77.4	96.3	100.0	72.0	36.1	46.3	71.3	–
ADMP [10]	1/1.7×	–	83.3	98.9	99.9	83.1	63.2	64.2	82.0	–
	1/2.5×	–	82.1	98.6	99.9	81.5	63.0	63.2	81.3	–
SlimDA	1×	1×	90.7	91.2	91.1	99.8	73.7	71.0	87.6	–
	1/2×	1/2×	90.7	99.1	100.0	91.8	73.3	71.1	87.6	0.0↓
	1/4×	1/4×	90.5	98.1	99.8	91.9	73.1	71.2	87.4	0.2↓
	1/10×	1/8×	90.6	98.8	100.0	91.6	72.9	71.1	87.5	0.1↓
	1/14×	1/16×	90.5	98.7	99.8	91.4	73.1	71.0	87.4	0.2↓
	1/20×	1/32×	90.8	97.7	99.5	91.5	71.8	70.8	87.0	0.6↓
	1/64×	1/64×	91.2	97.2	98.9	91.2	71.3	68.9	86.8	0.8↓

Table 7. Performance on the Office-31 dataset. As for other illustrations, please refer to the caption of Table 6.

5.3. Implementation Details

An SGD optimizer with momentum of 0.9 is adopted to train all UDA tasks in this paper. Following [57], the learning rate is adjusted by $l = l_0/(1 + \alpha p)^{\beta}$, where $l_0 = 0.01$, $\alpha = 10$, $\beta = 0.75$, and p varies from 0 to 1 linearly with the training epochs. The training epoch is set 40. The training and testing image resolution is 224×224. An important technical detail is that, before performance evaluation, the models sampled in the model bank should update the statistic of their BN layers on the target domain via AdaBN [22]. We mainly take the computational complexity (FLOPs) as resource constraint for architecture adaptation, and set 1/64× FLOPs as the smallest model by default.

5.4. Ablation Studies

5.4.1 Analysis for SEED

Comparison with Conventional Knowledge Distillation: As shown in Table 2, we can observe that our SEED with different model batch size can outperform the conventional knowledge distillation by a large margin even under 1/64× FLOPs of ResNet-50 on ImageCLEF-DA.

Comparison among different model batch sizes: Model batch size is a vital hyper-parameter of our framework. Intuitively, a larger model batch is more sufficient to approximate the knowledge aggregation in the model bank. As shown in Table 2, a large model batch size is beneficial for

Methods	FLOPs	Ar→Cl	Ar→Pr	Ar→Rw	Cl→Ar	Cl→Pr	Cl→Rw	Pr→Ar	Pr→Cl	Pr→Rw	Rw→Ar	Rw→Cl	Rw→Pr	Avg.	Δ
Source Only	1×	34.9	50.0	58.0	37.4	41.9	46.2	38.5	31.2	60.4	53.9	41.2	59.9	46.1	–
DAN	1×	43.6	57.0	67.9	45.8	56.5	60.4	44.0	43.6	67.7	63.1	51.5	74.3	56.3	–
RevGrad	1×	45.6	59.3	70.1	47.0	58.5	60.9	46.1	43.7	68.5	63.2	51.8	76.8	57.6	–
CDAN-E	1×	50.7	70.6	76.0	57.6	70.0	70.0	57.4	50.9	77.3	70.9	56.7	81.6	65.8	–
SymNet	1×	47.7	72.9	78.5	64.2	71.3	74.2	64.2	48.8	79.5	74.5	52.6	82.7	67.6	–
BNM	1×	52.3	73.9	80.0	63.3	72.9	74.9	61.7	49.5	79.7	70.5	53.6	82.2	67.9	–
SlimDA	1×	52.8	72.3	77.2	63.5	72.3	73.3	64.8	52.2	79.7	72.4	57.8	82.8	68.4	–
	1/2×	52.4	72.0	77.2	62.8	72.0	73.0	65.1	53.2	79.4	72.1	55.3	82.0	68.0	0.4↓
	1/4×	51.9	71.8	77.1	62.4	71.9	72.3	64.8	53.1	78.8	71.9	55.1	82.1	67.8	0.6↓
	1/8×	51.6	71.4	76.6	62.5	71.0	71.0	65.0	53.0	78.4	71.5	55.2	81.6	67.4	1.0↓
	1/16×	51.0	71.0	75.9	61.3	70.6	70.0	64.4	52.3	77.7	70.2	54.6	81.2	66.7	1.7↓
	1/32×	50.0	70.6	74.0	57.7	70.3	68.9	60.1	51.6	76.3	67.5	51.7	80.9	65.0	3.4↓
	1/64×	49.7	70.1	72.9	56.6	70.0	66.3	56.5	48.3	75.9	65.9	55.5	80.9	64.0	4.4↓

Table 8. Performance on the Office-Home dataset. As for other illustrations, please refer to the caption of Table 6.
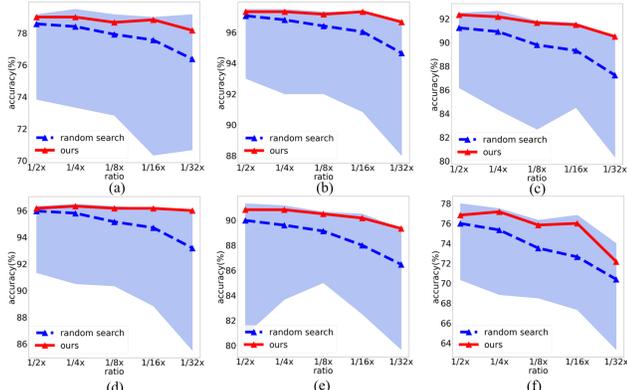

Figure 3. Comparison with randomly-searched models on six adaptation tasks on ImageCLEF-DA. One hundred models are randomly-searched under each FLOPs. The blue-filled area represents the gap between the maximum and minimum accuracy among random-searched models. The dotted blue dotted line represents the averaged values of random-searched models, and the solid red line represents the accuracies of our searched models.

our optimization method. We set the model batch size as 10 by default in our following experiments.

Comparison with stand-alone training: As shown in Table 4, compared with stand-alone training with the same network configuration, SEED can improve the overall adaptation performance by a large margin. Here "stand-alone" means that the models from $1\times$ to $1/64\times$ with the same topological configuration to the SlimDA counterparts are adapted individually outside the model bank. Specifically, not only the tiny models (from $1/2\times$ to $1/64\times$), but also the $1\times$ model trained with SEED outperform the corresponding stand-alone ones, which can be supported by more results in Table 6 and Table 7 (comparing the performance of $1\times$ models and our re-implemented ones).

Effectiveness of each component in our SlimDA: We conduct ablation studies to investigate the effectiveness of the components in our SlimDA framework. As shown in Table 3, the second row "Baseline" denotes the approach merging SymNet and slimmable neural network straightforwardly. We can observe that the SEED has a significant impact on the performance with fewer FLOPs, such as $1/4\times$,
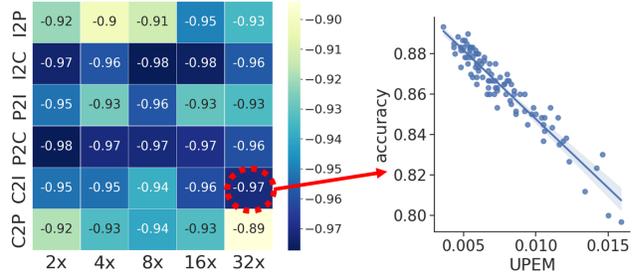

Figure 4. Pearson correlation coefficient between the unsupervised performance evaluation metric (UPEM) and the accuracy using ground-truth labels. It is performed on six adaptation tasks (ImageCLEF-DA) under five different computation constraints. In each grid, we sample 100 models to calculate the Pearson correlation coefficient. If it gets close to -1, it means that our metric is identical to use the ground-truth labels to measure the performance of each model.

$1/8\times$, $1/16\times$, $1/32\times$, and $1/64\times$, but the performances of $1\times$ and $1/2 \times$ models with SEED fall 0.7% and 0.4%, respectively, compared with the baseline, which is attributed to the optimization conflict between intra-model domain confusion and inter-model SEED. The last row shows that our proposed OSTC provides an impressive improvement on the performance of large models ($1\times$ and $1/2\times$) compared with both the SEED and baseline. Moreover, our proposed OSTC can further improve the performance of other models with fewer FLOPs. Overall, each component in SlimDA contributes to the performance-boosting of tiny models (from $1/64\times$ to $1/4\times$), the SEED w/o OSTC indeed brings negative transferring for models with larger capacity. However, our proposed OSTC can remedy the negative transferring issue and provide additional boosting under six FLOPs settings. The results in the ablation study also demonstrate the process of solving the challenges in our framework to accomplish the combination between UDA training and weight-sharing model bank.

5.4.2 Analysis for Architecture Adaptation

Analysis for UPEM: To validate the effectiveness of our proposed UPEM, we conduct 30 experiments which are composed of six adaptation tasks and five computational
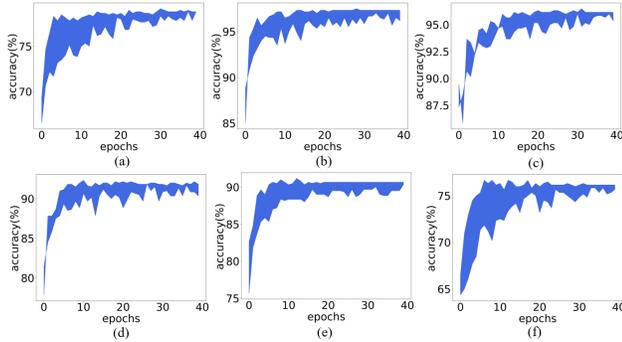
Figure 5. Convergence performance of the model bank. Six subfigures correspond to six adaptation tasks on ImageCLEF-DA. The results in each epoch are from 10 randomly-sampled models.

constraints. In each experiment, we sample 100 models in the model bank to evaluate the correlation between UPEM and the supervised accuracy with the ground-truth labels on target data. As shown in Fig.4, the Pearson correlation coefficients almost get close to -1, which means the smaller the UPEM is, the higher the accuracy is.

Comparison with Random Search: As shown in Fig.3, the necessity of architecture adaptation is highlighted from the significant performance gap among random searched models under each computational budget. Meanwhile, the architectures searched for different tasks are different. Moreover, given the same computational budgets, models greedily searched from the model bank with our UPEM are the top series in their actual performance sorting.

5.4.3 Compatibility with Different UDA Methods

Our SlimDA can also be cooperated with other UDA approaches, such as MCD [38] and STAR [29]. To present the universality of our framework, we carry out additional experiments of SlimDA with MCD and STAR, as well as their stand-alone counterparts. As shown in Table 4, the results from our SlimDA with MCD and STAR are still superior to the corresponding stand-alone counterparts by a large margin, which further validates the effectiveness of our SlimDA framework from another perspective.

5.4.4 Convergence Performance

We sample different models for performance evaluation during model bank training. As shown in Fig.5, the models coupled in the model bank are optimized simultaneously. The performances are steadily improved as learning goes.

5.5. Comparisons with State-of-The-Arts

Comparison with UDA methods: We conduct extensive experiments on three popular domain adaptation datasets, including ImageCLEF-DA, Office-31, and Office-Home. Detailed results and performance comparison with SOTA

UDA methods are presented in Table 6, Table 7 and Table 8. The benefits of our proposed SlimDA can be further pronounced when we reduce FLOPs extremely. Our method can preserve the accuracy improvement against the source-only model even when reducing FLOPs up to $1/64\times$. As we can see in Table 6, Table 7 and Table 8, when we reduce FLOPs even up to $1/64\times$, the accuracy on ImageCLEF-DA, Office-31 and Office-Home datasets only drop about 2.3%, 0.8% and 4.4%, respectively.

Comparision with cross-domain network compression methods: TCP [53] and ADMP [10] are two main cross-domain network compression methods for channel numbers, our method can significantly outperform them by a very large margin and achieve new SOTA results.

Comparison with lightweight networks: As shown in Table 5, not only human-designed lightweight networks like MobileNet series [19, 39], ShuffleNet [32] and Ghost-Net [15], but also auto-designed architectures like EfficientNet [43] is over-passed by our SlimDA with fewer FLOPs/parameters.

6. Conclusions

In this paper, we propose a simple yet effective SlimDA framework to facilitate weight and architecture joint-adaptation. In SlimDA, the proposed SEED exploits architecture diversity in a weight-sharing model bank to suppress prediction uncertainty on the unlabeled target data, and the proposed OSTC modulates the optimization conflict between intra-model adaptation and inter-model interaction. In this way, we can flexibly distribute resource-satisfactory models via a retrain-free sampling manner to various devices on target domain. Moreover, we propose UPEM to select the optimal cross-domain model under each computational budget. Extensive ablation studies and experiments are carried out to validate the effectiveness of SlimDA. Our SlimDA can also be extended to other visual computing tasks, such as object detection and semantic segmentation, which we leave as our future work. In short, our work provides a practical UDA framework towards real-world scenario, and we hope it can bring new inspirations to the widespread application of UDA.

Limitations. The assumption 4.1 is an essential prerequisite in this work, which will hold at least to a reasonable extent. However, it is not guaranteed theoretically, which may invalidate our method in some unknown circumstances.

Acknowledgements

References

[1] Konstantinos Bousmalis, Nathan Silberman, David Dohan, Dumitru Erhan, and Dilip Krishnan. Unsupervised pixel-level domain adaptation with generative adversarial networks. In *CVPR*, pages 3722–3731, 2017. 2

[2] Andrew Brock, Theodore Lim, James M Ritchie, and Nick Weston. Smash: one-shot model architecture search through hypernetworks. *arXiv preprint arXiv:1708.05344*, 2017. 2

[3] Han Cai, Chuang Gan, and S. Han. Once for all: Train one network and specialize it for efficient deployment. In *ICLR*, 2020. 2

[4] Han Cai, Jiacheng Yang, Weinan Zhang, Song Han, and Yong Yu. Path-level network transformation for efficient architecture search. *arXiv preprint arXiv:1806.02639*, 2018. 2

[5] P. Chattopadhyay, Y. Balaji, and J. Hoffman. Learning to balance specificity and invariance for in and out of domain generalization. 2020. 4

[6] Shangyu Chen, Wenya Wang, and Sinno Jialin Pan. Cooperative pruning in cross-domain deep neural network compression. In *IJCAI*, pages 2102–2108, 7 2019. 3, 5

[7] Weijie Chen, Luojun Lin, Shicai Yang, Di Xie, Shiliang Pu, Yueting Zhuang, and Wenqi Ren. Self-supervised noisy label learning for source-free unsupervised domain adaptation. *ArXiv*, abs/2102.11614, 2021. 1

[8] Shuhao Cui, Shuhui Wang, Junbao Zhuo, Liang Li, Qingming Huang, and Qi Tian. Towards discriminability and diversity: Batch nuclear-norm maximization under label insufficient situations. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3941–3950, 2020. 6

[9] Thomas Elsken, Jan Hendrik Metzen, and Frank Hutter. Efficient multi-objective neural architecture search via lamarckian evolution. *arXiv preprint arXiv:1804.09081*, 2018. 2

[10] Xiaoyu Feng, Z. Yuan, G. Wang, and Yongpan Liu. Admp: An adversarial double masks based pruning framework for unsupervised cross-domain compression. *ArXiv*, abs/2006.04127, 2020. 3, 5, 6, 8

[11] Yaroslav Ganin and Victor Lempitsky. Unsupervised domain adaptation by backpropagation. In *ICML*, pages 1180–1189, 2015. 1, 2, 6

[12] Yves Grandvalet, Yoshua Bengio, et al. Semi-supervised learning by entropy minimization. *CAP*, 367:281–296, 2005. 4

[13] Zichao Guo, X. Zhang, H. Mu, Wen Heng, Z. Liu, Y. Wei, and Jian Sun. Single path one-shot neural architecture search with uniform sampling. In *ECCV*, 2020. 2

[14] Philip Haeusser, Thomas Frerix, Alexander Mordvintsev, and Daniel Cremers. Associative domain adaptation. In *ICCV*, pages 2765–2773, 2017. 2

[15] Kai Han, Yunhe Wang, Qi Tian, Jianyuan Guo, Chunjing Xu, and Chang Xu. Ghostnet: More features from cheap operations. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1580–1589, 2020. 5, 8

[16] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016. 5

[17] Judy Hoffman, E. Tzeng, Trevor Darrell, and Kate Saenko. Simultaneous deep transfer across domains and tasks. *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 4068–4076, 2015. 6

[18] Judy Hoffman, Eric Tzeng, Taesung Park, Jun-Yan Zhu, Phillip Isola, Kate Saenko, Alexei Efros, and Trevor Darrell. Cycada: Cycle-consistent adversarial domain adaptation. In *ICML*, pages 1989–1998, 2018. 2

[19] Andrew Howard, Mark Sandler, Grace Chu, Liang-Chieh Chen, Bo Chen, Mingxing Tan, Weijun Wang, Yukun Zhu, Ruoming Pang, Vijay Vasudevan, et al. Searching for mobilenetv3. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1314–1324, 2019. 5, 8

[20] Xianfeng Li, Weijie Chen, Di Xie, Shicai Yang, and Yueting Zhuang. A free lunch for unsupervised domain adaptive object detection without source data. In *AAAI*, 2020. 1

[21] Yichen Li and Xingchao Peng. Network architecture search for domain adaptation. 2020. 2

[22] Yanghao Li, Naiyan Wang, Jianping Shi, Xiaodi Hou, and Jiaying Liu. Adaptive batch normalization for practical domain adaptation. *Pattern Recognition*, 80:109–117, 2018. 6

[23] Chenxi Liu, Liang-Chieh Chen, Florian Schroff, Hartwig Adam, Wei Hua, Alan L Yuille, and Li Fei-Fei. Auto-deeplab: Hierarchical neural architecture search for semantic image segmentation. In *CVPR*, pages 82–92, 2019. 2

[24] Hanxiao Liu, Karen Simonyan, Oriol Vinyals, Chrisantha Fernando, and Koray Kavukcuoglu. Hierarchical representations for efficient architecture search. *arXiv preprint arXiv:1711.00436*, 2017. 2

[25] Ming-Yu Liu and Oncel Tuzel. Coupled generative adversarial networks. In *NeurIPS*, pages 469–477, 2016. 2

[26] Mingsheng Long, Yue Cao, Jianmin Wang, and Michael Jordan. Learning transferable features with deep adaptation networks. In *ICML*, pages 97–105, 2015. 1, 2, 3, 5, 6

[27] Mingsheng Long, Han Zhu, Jianmin Wang, and Michael I Jordan. Deep transfer learning with joint adaptation networks. In *International conference on machine learning*, pages 2208–2217. PMLR, 2017. 2

[28] Mingsheng Long†, Zhangjie Cao†, Jianmin Wang†, and Michael I. Jordan. Conditional adversarial domain adaptation. 2017. 6

[29] Zhihe Lu, Yongxin Yang, Xiatian Zhu, Cong Liu, Yi-Zhe Song, and Tao Xiang. Stochastic classifiers for unsupervised domain adaptation. In *CVPR*, pages 9111–9120, 2020. 1, 6, 8

[30] Renqian Luo, Fei Tian, Tao Qin, Enhong Chen, and Tie-Yan Liu. Neural architecture optimization. In *NeurIPS*, pages 7816–7827, 2018. 2

[31] You-Wei Luo, Chuan-Xian Ren, D. Dai, and H. Yan. Unsupervised domain adaptation via discriminative manifold propagation. *IEEE transactions on pattern analysis and machine intelligence*, PP, 2020. 6

[32] Ningning Ma, Xiangyu Zhang, Hai-Tao Zheng, and Jian Sun. Shufflenet v2: Practical guidelines for efficient cnn architecture design. In *Proceedings of the European conference on computer vision (ECCV)*, pages 116–131, 2018. 5, 8

[33] Rang Meng, Weijie Chen, Di Xie, Yuan Zhang, and Sshiliang Pu. Neural inheritance relation guided one-shot layer assignment search. In *AAAI*, 2020. 2, 5

[34] Juan-Manuel Pérez-Rúa, Valentin Vielzeuf, Stéphane Pateux, Moez Baccouche, and Frédéric Jurie. Mfas: Multimodal fusion architecture search. In *CVPR*, pages 6966–6975, 2019. 2

[35] Esteban Real, Alok Aggarwal, Yanping Huang, and Quoc V Le. Regularized evolution for image classifier architecture search. In *AAAI*, volume 33, pages 4780–4789, 2019. 2

[36] K. Saenko, B. Kulis, M. Fritz, and T. Darrell. Adapting visual category models to new domains. In *ECCV*, 2010. 2

[37] Kate Saenko, Brian Kulis, Mario Fritz, and Trevor Darrell. Adapting visual category models to new domains. In *European conference on computer vision*, pages 213–226. Springer, 2010. 5

[38] Kuniaki Saito, Kohei Watanabe, Yoshitaka Ushiku, and Tatsuya Harada. Maximum classifier discrepancy for unsupervised domain adaptation. In *CVPR*, pages 3723–3732, 2018. 1, 2, 6, 8

[39] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4510–4520, 2018. 5, 8

[40] Richard Shin, Charles Packer, and Dawn Song. Differentiable neural network architecture search. 2018. 2

[41] Baochen Sun and Kate Saenko. Deep coral: Correlation alignment for deep domain adaptation. In *ECCV*, pages 443–450, 2016. 2

[42] Mingxing Tan, Bo Chen, Ruoming Pang, Vijay Vasudevan, Mark Sandler, Andrew Howard, and Quoc V Le. Mnasnet: Platform-aware neural architecture search for mobile. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2820–2828, 2019. 2

[43] Mingxing Tan and Quoc Le. Efficientnet: Rethinking model scaling for convolutional neural networks. In *International Conference on Machine Learning*, pages 6105–6114. PMLR, 2019. 2, 5, 8

[44] Eric Tzeng, Judy Hoffman, Kate Saenko, and Trevor Darrell. Adversarial discriminative domain adaptation. In *CVPR*, pages 7167–7176, 2017. 1

[45] H. Venkateswara, J. Eusebio, S. Chakraborty, and S. Panchanathan. Deep hashing network for unsupervised domain adaptation. In *CVPR*, 2017. 2

[46] Hemanth Venkateswara, Jose Eusebio, Shayok Chakraborty, and Sethuraman Panchanathan. Deep hashing network for unsupervised domain adaptation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5018–5027, 2017. 5

[47] Bichen Wu, Xiaoliang Dai, Peizhao Zhang, Yanghan Wang, Fei Sun, Yiming Wu, Yuandong Tian, Peter Vajda, Yangqing Jia, and Kurt Keutzer. Fbnet: Hardware-aware efficient convnet design via differentiable neural architecture search. In *CVPR*, pages 10734–10742, 2019. 2

[48] Yuan Wu, D. Inkpen, and A. El-Roby. Dual mixup regularized learning for adversarial domain adaptation. In *ECCV*, 2020. 6

[49] Sirui Xie, Hehui Zheng, Chunxiao Liu, and Liang Lin. Snas: stochastic neural architecture search. *arXiv preprint arXiv:1812.09926*, 2018. 2

[50] Yiding Yang, Zunlei Feng, Mingli Song, and Xinchao Wang. Factorizable graph convolutional networks. In *Neural Information Processing Systems*, 2020. 3

[51] Yiding Yang, Jiayan Qiu, Mingli Song, Dacheng Tao, and Xinchao Wang. Distilling knowledge from graph convolutional networks. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2020. 3

[52] Chris Ying, Aaron Klein, Eric Christiansen, Esteban Real, Kevin Murphy, and Frank Hutter. NAS-bench-101: Towards reproducible neural architecture search. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 7105–7114, Long Beach, California, USA, 09–15 Jun 2019. PMLR. 2, 4

[53] Chaohui Yu, J. Wang, Y. Chen, and Zijing Wu. Accelerating deep unsupervised domain adaptation with transfer channel pruning. pages 1–8, 2019. 3, 5, 6, 8

[54] J. Yu and T. Huang. Autoslim: Towards one-shot architecture search for channel numbers. *ArXiv*, abs/1903.11728, 2019. 2, 5

[55] J. Yu and T. Huang. Universally slimmable networks and improved training techniques. In *ICCV*, 2019. 2

[56] J. Yu, L. Yang, N. Xu, J. Yang, and T. Huang. Slimmable neural networks. 2018. 2

[57] Yabin Zhang, Hui Tang, Kui Jia, and Mingkui Tan. Domain-symmetric networks for adversarial domain adaptation. In *CVPR*, 2019. 2, 3, 6

[58] Barret Zoph and Quoc V Le. Neural architecture search with reinforcement learning. *arXiv preprint arXiv:1611.01578*, 2016. 2

[59] Barret Zoph, Vijay Vasudevan, Jonathon Shlens, and Quoc V Le. Learning transferable architectures for scalable image recognition. In *CVPR*, pages 8697–8710, 2018. 2