Slimmable Domain Adaptation (Supplementary Materials)

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

A. Implementation Details

A.1. Details for SEED

The overall SEED schema is summarized in Algorithm 1. For each data batch, we sample a model batch stochastically and then utilize SEED to train each model batch. We first calculate model confidence in each model batch, which has two-fold utilities in the SEED: 1) weighting the predictions from different models to generate more confident ensemble prediction; 2) weighting the gradients from intra-model adaptation and inter-model interaction losses with respect to the feature extractors. After SEED training, we can obtain a "once-for-all" domain adaptive model bank.

A.2. Details for Neural Architecture Search

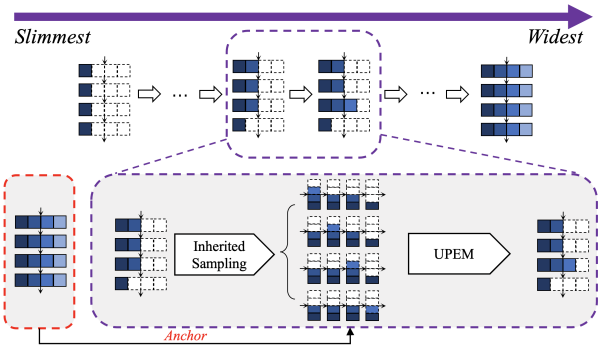

Figure 1. Inherited Greedy Search for channel configurations.

Although many neural architecture search methods (*e.g.*, enumerate method [1] and genetic algorithms [4]) can be coupled with our proposed UPEM to search optimal architectures on the unlabeled target data, we would like to provide more technique details for the efficient search method we used in the main text of the paper. As shown in Fig. 1, we present the search method for channel configurations inspired by [12], which we dub as "Inherited Greedy Search".

Algorithm 1: Stochastic EnsEmble Distillation

input : labeled source data $\mathcal{D}_s = \{(x_i^s, y_i^s)\}_{i=1}^{n_s}$;
unlabeled target data $\mathcal{D}_t = \{(x_i^t)\}_{i=1}^{n_t}$;
model bank (F, C^s, C^t, C^a);
learning rate l;
model batch size m;
domain confusion loss $\mathcal{L}_{dc}$;
cross-entropy loss $\mathcal{L}_{ce}$;
output: model bank (F, C^a)

for x_i^s, x_i^t, y_i^s *in* $\mathcal{D}_s \cup \mathcal{D}_t$ **do**
  Stochastically Sample a Model Batch: $\{(F_j, C_j^s, C_j^t, C_j^a)\}_{j=1}^m$;
  Calculate Model Confidence: $\{\mathbf{g}(F_j, C_j^s, C_j^t)\}_{j=1}^m$;
  Calculate $\{\frac{\partial \mathcal{L}_{dc}}{\partial F_j}, \frac{\partial \mathcal{L}_{dc}}{\partial C_j^s}, \frac{\partial \mathcal{L}_{dc}}{\partial C_j^t}\}_{j=1}^m$;
  Generate $\mathbf{g}_{seed}$:
  $\mathbf{g}_{seed} = \mathbb{E}_{\mathbf{g}(F_j, C_j^s, C_j^t)}[\mathbf{g}(x_i^t; F_j, C_j^s, C_j^t)]$;
  Calculate $\mathcal{L}_{seed}$:
  $\mathcal{L}_{seed} = \mathcal{L}_{ce}(\mathbf{g}(x_i^t; F_j, C_j^a), \mathbf{g}_{seed}) + \mathcal{L}_{ce}(\mathbf{g}(x_i^s; F_j, C_j^a), y_i^s)$;
  Calculate $\{\frac{\partial \mathcal{L}_{seed}}{\partial F_j}, \frac{\partial \mathcal{L}_{seed}}{\partial C_j^a}\}_{j=1}^m$;
  Update $\{C^s, C^t, C^a\}$:
  $C^s \leftarrow C^s - l \cdot \frac{1}{m} \sum_{j=1}^m \frac{\partial \mathcal{L}_{dc}}{\partial C_j^s}$;
  $C^t \leftarrow C^t - l \cdot \frac{1}{m} \sum_{j=1}^m \frac{\partial \mathcal{L}_{dc}}{\partial C_j^t}$;
  $C^a \leftarrow C^a - l \cdot \frac{1}{m} \sum_{j=1}^m \frac{\partial \mathcal{L}_{seed}}{\partial C_j^a}$;
  Update F:
  $F \leftarrow F - l \cdot \mathbb{E}_{\mathbf{g}(F_j, C_j^s, C_j^t)}[\frac{\partial \mathcal{L}_{dc}}{\partial F_j}] - l \cdot \mathbb{E}_{1-\mathbf{g}(F_j, C_j^s, C_j^t)}[\frac{\partial \mathcal{L}_{seed}}{\partial F_j}]$;
end

In the Inherited Greedy Search, the larger optimal models are assumed to be developed from the smaller optimal ones. Under this guidance, we obtain optimal models under different computational budgets from the slimmest to the widest models in a trip.

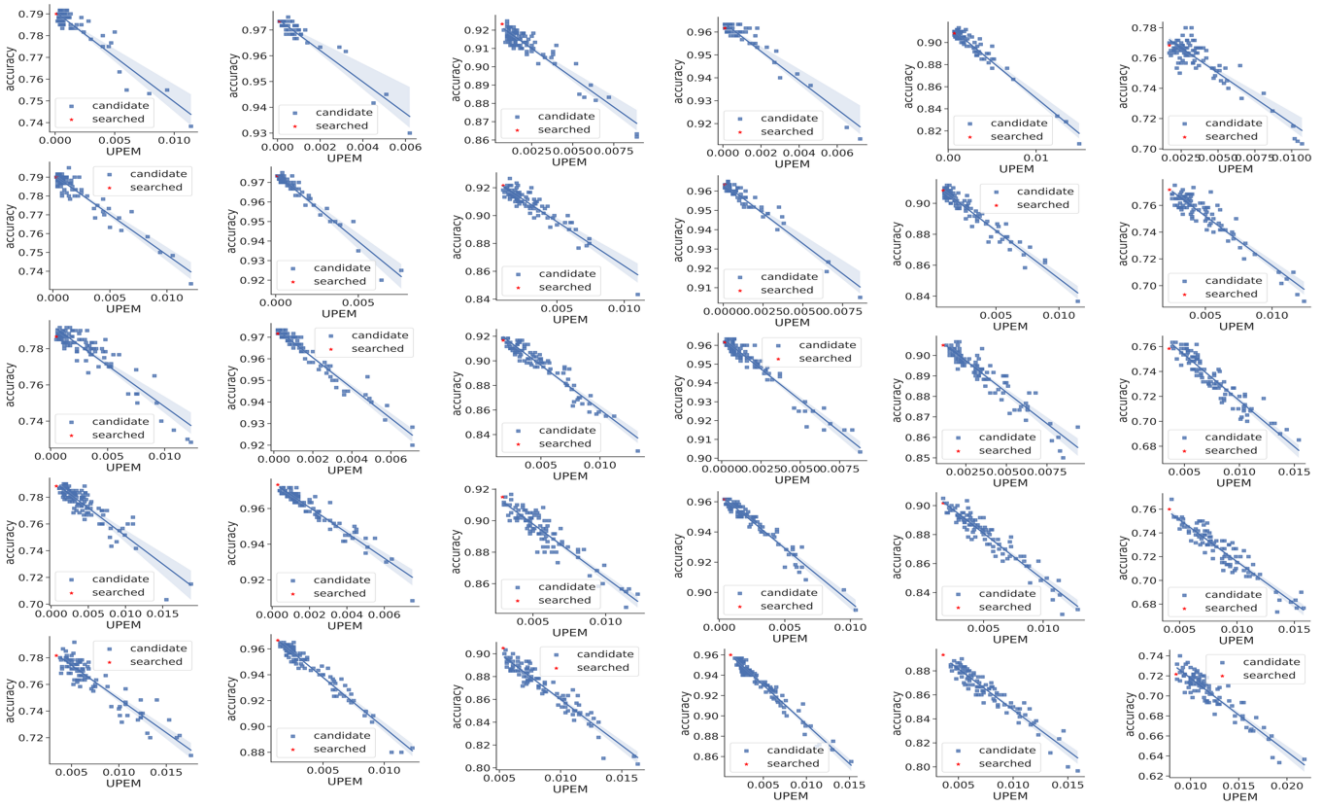

Figure 2. The visualization of the relation of numerous models with different computational complexities on different domain adaptation tasks. Here we present the results of six adaptation tasks (six columns) on the ImageCLEF-DA dataset and five computational budgets (five rows) from $1/2\times$ to $1/32\times$. Each column from left to right presents adaptation task I→P, I→C, P→I, P→C, C→I, and C→P, respectively, while each row presents one computational budget on six adaptation tasks. In each sub-figure, we present the accuracy relation of 100 randomly-sampled models, and the red dot denotes our searched one, which is superior to most of the candidate models.

methods	FLOPs	plane	bcycl	bus	car	horse	knife	mcycl	person	plant	sktbrd	train	truck	mean
Source only	1×	55.1	53.3	61.9	59.1	80.6	17.9	79.7	31.2	81.0	26.5	73.5	8.5	52.4
Stand-alone MCD	1×	86.6	42.9	77.6	76.7	69.0	30.6	73.8	70.2	68.6	60.5	68.6	9.7	61.2
SlimDA with MCD	1×	94.3	77.9	84.7	66.0	91.9	71.0	88.7	75.8	91.2	77.8	86.4	36.3	78.5
	1/2×	94.0	77.7	84.5	65.5	91.7	70.9	88.6	75.6	90.9	77.7	86.4	36.2	78.3
	1/4×	93.7	77.6	84.3	65.4	91.5	70.6	88.4	75.3	90.8	77.7	86.4	36.1	78.1
	1/8×	93.6	77.7	84.2	65.5	91.6	70.6	88.5	75.5	90.5	77.4	86.4	36.1	78.1
	1/16×	93.5	77.5	84.2	65.5	91.7	70.6	88.3	75.4	90.6	77.5	86.3	36.0	78.1
	1/32×	93.3	77.3	84.0	65.1	91.7	70.9	88.1	75.1	90.6	77.7	86.3	35.6	78.0
	1/64×	93.5	77.0	83.9	64.9	91.6	70.5	88.1	74.9	90.6	77.1	85.7	34.9	77.7

Table 1. Comparison among models with different FLOPs in SlimDA on VisDA datasets. We report the accuracy for each class. Moreover, "mean" indicates the average value among 12 per-class accuracy.

Specifically, we begin with the slimmest model (the $1/64\times$ FLOPs model) and set it as our initial model for searching. Then we divide the FLOPs gap (F) between the slimmest and the widest models into k parts equally. In this way, we can search the optimal models under different computational budgets from the slimmest and the widest models in k-1 steps with steady FLOPs growing. In the first step, we sample q models by randomly adding the channels to each block of the initial model to fit the incremental FLOPs (F/k), and then we leverage UPEM to search the optimal model among q models in this step. In the next step, we use the previous searched model as the initial model to repeat the same process in the first step, and we call this sampling method "Inherited Sampling". Benefited from the Inherited Greedy Search, we can significantly reduce the complexity of the search process.

Methods	FLOPs	I → P	P → I	I → C	C → I	C → P	P → C	Avg.	Δ
SlimDA	1×	79.2	92.3	97.5	91.2	76.7	96.5	88.9	–
	1/2×	79.0	92.3	97.3	90.8	76.8	96.2	88.7	0.2↓
	1/4×	79.0	92.2	97.3	90.8	77.2	96.3	88.8	0.1↓
	1/8×	78.7	91.7	97.2	90.5	75.8	96.2	88.4	0.5↓
	1/16×	78.8	91.5	97.3	90.2	76.0	96.2	88.3	0.6↓
	1/32×	78.2	90.5	96.7	89.3	72.2	96.0	87.2	1.7↓
	1/64×	78.3	90.7	95.8	88.3	71.8	94.8	86.6	2.3↓
Inplaced Distillation	1×	78.0	87.8	94.2	90.3	75.7	91.8	86.3	-
	1/2×	77.1	87.0	94.0	89.0	74.0	90.9	85.3	1.0 ↓
	1/4×	76.3	86.3	93.5	87.7	72.6	89.7	84.3	2.0↓
	1/8×	75.5	84.8	92.9	85.5	70.9	89.3	82.7	3.6↓
	1/16×	73.3	82.6	91.5	83.9	68.0	87.4	81.2	5.1↓
	1/32×	71.1	81.0	90.5	82.2	66.5	86.5	79.6	6.7↓
	1/64×	70.0	80.3	90.2	81.5	65.8	86.2	79.0	7.3↓

Table 2. Comparison with Inplaced Distillation on ImageCLEF-DA dataset. The model batch size is set to 10 for both methods. "Δ" means the averaged performance gap between 1× FLOPs of ResNet-50 and other models, respectively. Note that "Δ" is calculated for SlimDA and Inplaced Distillation, respectively.

Method	mean acc.	Method	mean acc.
ResNet-50 [6]	52.1	CDAN [9]	70.0
DANN [3]	57.4	MDD [21]	74.6
DAN [10]	61.6	GVB-GD [2]	75.3
MCD [15]	69.2	SHOT [6]	76.7
GTA [16]	69.5	SHOT++ [7]	77.2
1× FLOPs in SlimDA	78.5	1/64× FLOPS in SlimDA	77.7

Table 3. Comparison with different UDA methods on the VisDA dataset. Note that the network backbone is set ResNet-50 in this table. "1× FLOPs in SlimDA" and "1/64× FLOPs in SlimDA" mean the models with 1× and 1/64× FLOPs of ResNet-50 in SlimDA, respectively. "mean acc." means the averaged accuracy among 12 per-class accuracies.

Methods	I → P	P → I	I → C	C → I	C → P	P → C	Overall
stand alone 1×	0.11H	0.11H	0.11H	0.11H	0.11H	0.11H	0.66H
SlimDA	0.63H	0.67H	0.60H	0.65H	0.63H	0.67H	3.85H

Table 4. Analysis for training time on ImageCLEF-DA. We report the number of hours on one V100 GPU for training the standalone ResNet-50 model and our SlimDA framework with SymNet. "Overall " means the total training time for 6 adaptation tasks.

s	1×	1/4×	1/16×	1/64×
1.0	88.3	88.2	87.5	85.7
0.5	88.3	88.2	87.9	86.1
0.1	**89.0**	**88.9**	**88.4**	86.5
0.0	88.9	88.8	88.3	**86.6**

Table 5. Ablation study of s in Eq.1.

B. Additional Experiments and Analysis

B.1. Additional Experiments on VisDA

We also report the results of our SlimDA on the large-scale domain adaptation benchmark, VisDA [13], to evaluate the effectiveness of our SlimDA. VisDA is a challenging large-scale image classification benchmark for UDA. It consists of 152k synthetic images rendered by the 3D model with annotations as source data and 55k real images without annotations from MS-COCO [8] as target data. There are 12 categories in VisDA. There is a large domain discrepancy between the synthetic style and real style in VisDA. We follow the network configuration and implementation details from the experiments on ImageCLEF-DA [11], Office-31 [14], and Office-Home [17] as described in the main text of the paper.

As shown in Table 1, our SlimDA can simultaneously boost the adaptation performance of models with different FLOPs. It can be observed that the model with 1/64 FLOPs of ResNet-50 [5] merely has a drop of 0.8% on the mean accuracy. In addition, from Table 3, we can observe that our SlimDA with only 1/64× FLOPs can surpass other SOTA UDA methods by a large margin. Overall, our SlimDA can achieve impressive performance for the large-scale dataset with significant domain shifts such as VisDA.

B.2. Additional Analysis

Comparison with Inplaced Distillation. Inplaced Distillation [18–20] is an improving technique provided in slimmable neural network [18]. With Inplaced Distillation, the largest model generates soft labels to guide the learning of the remaining models in the model batch. However, it cannot remedy the issue of uncertainty brought by domain shift and unlabeled data. We combine Inplaced Distillation with SymNet [22] straightforwardly as another baseline. As shown in Table 2, we can observe that Inplaced Distillation leads to negative transfer for the model with 1× FLOPS. Moreover, the performance of models with fewer FLOPs is limited with Inplaced Distillation. Compared with

Inplaced Distillation, our SlimDA can bring performance improvements consistently under different computational budgets. In addition, the smallest model with 1/64× FLOPs in SlimDA surpasses the corresponding counterpart in Inplace Distillation by 7.6% on the averaged accuracy. Furthermore, even comparing the smallest model in our SlimDA to the largest one in Inplaced Distillation, the model can still achieve improvement of 0.3% on the averaged accuracy.

Additional analysis for UPEM. We provide additional analysis for the effectiveness of our proposed UPEM. As shown in Fig.2, we visualize the relation between accuracy and UPEM with different adaptation tasks and computational budgets on the ImageCLEF-DA dataset. The accuracy of 100 models with different network configurations under the same computational budgets vary a lot, which indicates the necessity for architecture adaptation. Moverover, the UPEM will have strict negative correlation with the accuracy for different adaptation tasks and different computational budgets (the blue lines in 30 sub-figures). Specifically, the models with the smallest UPEM (the red dots in 30 sub-figures) always have the top accuracy among models with the same computational budgets.

Analysis for Time Complexity of SlimDA. As shown in Table 4, our SlimDA framework only takes about 6× the time to train a ResNet-50 model alone on imageCLEF-DA dataset. But SlimDA can improve the performance of numerous models impressively at the same time. We report the training time with one NVIDIA V100 GPU.

Analysis for Architecture Configurations. We provide details of channel configurations for models with different FLOPs in our SlimDA. As shown in Table 6-9 (in the last page), we report 30 models with the smallest UPEM for I→P adaptation task under each FLOPs reduction. Note that the reported models with different architecture configurations are trained simultaneously in SlimDA with 6× the time to train a ResNet-50 model alone. Also, we sample these models from the model bank without re-training. Otherwise, it is inconceivably expensive and time-consuming if these model are trained individually. The numerous models and their impressive performances reflect that SlimDA significantly remedies the issue of real-world UDA, which is ignored by previous work.

A More General Method to Determine Model Confidence: A more general formula of Eq.9 in the paper:

$$\begin{aligned} \mathbf{g}_j &= 0.5\text{sign}(a_j)|a_j|^s + 0.5 \\ a_j &= 2r_j - 1 \end{aligned} \tag{1}$$

where $\text{sign}(\cdot)$ is a sign function to produce +1 or -1, and $|\cdot|$ is to produce an absolute value. If $s \to 0$ or $s \to 1$, this formula will be specified as Eq.9 in the paper or $\mathbf{g}_j = r_j / \sum_{j'} r_{j'}$, respectively. The performances tend to degrade a bit when $s \to 1$ due to the increasing weights of the small models for intra-model adaptation and knowledge ensemble. We find we can roughly set the model confidence in a hard way as defined in Eq.9 in the paper to achieve considerable performance.

block1	block2	block3	block4	acc.	block1	block2	block3	block4	acc.	block1	block2	block3	block4	acc.
210	339	573	1370	79.0	100	341	559	2025	78.7	39	167	774	2048	78.8
96	305	913	1563	78.8	168	201	976	1254	78.8	138	149	725	2044	78.8
147	446	775	931	79.0	139	541	469	1017	78.7	95	480	538	1560	78.7
87	236	1177	966	78.8	95	231	1193	852	78.7	78	458	588	1649	78.7
46	380	860	1564	78.8	193	357	821	849	78.8	79	556	529	1176	79.0
125	260	507	2048	78.8	108	233	1115	1119	79.0	125	228	671	2048	79.2
81	544	707	830	78.8	262	319	527	846	79.2	196	392	632	1151	79.0
125	332	819	1550	79.0	176	336	869	1011	79.0	299	214	261	1063	78.5
82	414	685	1674	78.8	239	354	489	1111	79.2	230	228	572	1527	78.8
118	353	641	1822	78.7	75	159	1149	1346	78.8	64	442	680	1616	79.0

Table 6. Architecture configurations for the I→P adaptation task on ImageCLEF-DA. We report the architecture configurations for models with 1/2× FLOPs of ResNet-50 in SlimDA. We report 30 models with the smallest UPEM. ["block1"→ "block4"] represents the channel number for each block consisting of layers with the same spatial resolution of feature maps.

block1	block2	block3	block4	acc.	block1	block2	block3	block4	acc.	block1	block2	block3	block4	acc.
74	241	598	970	79.0	38	242	489	1293	79.0	101	98	779	654	78.5
92	113	415	1492	78.7	134	222	353	1099	78.8	113	250	252	1260	79.2
95	137	472	1373	79.0	114	129	307	1479	78.8	90	249	577	902	78.8
63	130	566	1361	79.2	92	152	254	1578	78.5	134	149	557	951	79.0
182	149	308	898	79.0	49	261	643	873	79.0	48	319	404	1112	79.0
142	207	514	790	78.8	104	354	297	775	78.8	154	192	208	1174	79.0
63	130	555	1380	79.2	86	200	666	876	79.0	67	334	440	899	78.8
180	141	359	869	78.8	95	195	705	692	79.0	63	136	777	834	78.8
87	195	736	628	79.0	139	272	459	633	78.8	115	110	662	926	79.0
45	367	450	754	78.8	87	323	427	878	78.8	149	109	660	547	79.0

Table 7. Architecture configurations for the I→P adaptation task on ImageCLEF-DA. We report the architecture configurations for models with 1/4× FLOPs of ResNet-50 in SlimDA. We report 30 models with the smallest UPEM.

block1	block2	block3	block4	acc.	block1	block2	block3	block4	acc.	block1	block2	block3	block4	acc.
52	142	387	685	78.7	42	189	325	673	78.5	66	86	286	915	78.7
67	140	385	605	79.0	93	122	316	615	78.8	90	126	350	549	78.5
52	160	367	667	78.8	73	171	302	628	78.8	112	100	244	602	78.5
48	173	399	554	78.7	81	154	381	425	79.2	45	90	497	564	78.8
41	150	418	631	78.7	63	124	412	624	78.7	48	85	486	600	78.7
52	117	504	401	78.8	108	79	286	624	78.5	92	115	280	703	78.5
58	161	177	899	79.0	39	81	485	650	78.8	68	102	179	987	78.3
55	83	492	541	78.7	57	198	333	533	79.0	69	202	341	375	78.7
41	70	413	835	78.3	69	169	410	367	78.0	39	228	199	690	78.8
43	86	225	1063	78.2	101	83	330	599	78.3	71	223	198	544	78.7

Table 8. Architecture configurations for the I→P adaptation task on ImageCLEF-DA. We report the architecture configurations for models with 1/8× FLOPs of ResNet-50 in SlimDA. We report 30 models with the smallest UPEM.

block1	block2	block3	block4	acc.	block1	block2	block3	block4	acc.	block1	block2	block3	block4	acc.
41	83	136	477	78.2	45	88	138	435	78.0	37	82	241	298	77.8
49	85	143	407	76.8	32	77	163	505	77.3	35	73	157	507	77.2
36	132	139	284	78.7	32	64	284	257	78.5	53	77	140	406	77.8
53	99	151	297	78.0	36	136	134	268	77.8	33	113	192	311	78.0
53	103	150	279	78.2	32	64	132	564	77.3	39	85	198	391	77.3
48	95	194	273	77.8	51	74	183	360	77.7	53	81	189	297	76.8
46	89	140	423	77.8	46	97	173	338	76.8	40	132	130	267	78.2
41	64	128	529	77.2	59	72	142	366	77.5	49	103	135	350	77.7
42	82	134	480	77.5	40	92	213	323	77.8	37	68	152	517	77.2
54	65	178	374	76.0	34	128	137	328	78.3	65	81	128	284	79.2

Table 9. Architecture configurations for the I→P adaptation task on ImageCLEF-DA. We report the architecture configurations for models with 1/32× FLOPs of ResNet-50 in SlimDA. We report 30 models with the smallest UPEM.

References

[1] Han Cai, Chuang Gan, Tianzhe Wang, Zhekai Zhang, and Song Han. Once-for-all: Train one network and specialize it for efficient deployment. *arXiv preprint arXiv:1908.09791*, 2019. 1

[2] Shuhao Cui, Shuhui Wang, Junbao Zhuo, Chi Su, Qingming Huang, and Qi Tian. Gradually vanishing bridge for adversarial domain adaptation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12455–12464, 2020. 3

[3] Yaroslav Ganin and Victor Lempitsky. Unsupervised domain adaptation by backpropagation. In *International conference on machine learning*, pages 1180–1189. PMLR, 2015. 3

[4] Zichao Guo, Xiangyu Zhang, Haoyuan Mu, Wen Heng, Zechun Liu, Yichen Wei, and Jian Sun. Single path one-shot neural architecture search with uniform sampling. In *European Conference on Computer Vision*, pages 544–560. Springer, 2020. 1

[5] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016. 3

[6] Jian Liang, Dapeng Hu, and Jiashi Feng. Do we really need to access the source data? source hypothesis transfer for unsupervised domain adaptation. In *International Conference on Machine Learning*, pages 6028–6039. PMLR, 2020. 3

[7] Jian Liang, Dapeng Hu, Yunbo Wang, Ran He, and Jiashi Feng. Source data-absent unsupervised domain adaptation through hypothesis transfer and labeling transfer. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2021. 3

[8] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *ECCV*, pages 740–755, 2014. 3

[9] Mingsheng Long, Zhangjie Cao, Jianmin Wang, and Michael I Jordan. Conditional adversarial domain adaptation. *arXiv preprint arXiv:1705.10667*, 2017. 3

[10] M. Long, C. Yue, Z. Cao, J. Wang, and M. I. Jordan. Transferable representation learning with deep adaptation networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 41:3071–3085, 2018. 3

[11] 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. 3

[12] Rang Meng, Weijie Chen, Di Xie, Yuan Zhang, and Sshiliang Pu. Neural inheritance relation guided one-shot layer assignment search. In *AAAI*, 2020. 1

[13] Xingchao Peng, Ben Usman, Neela Kaushik, Judy Hoffman, Dequan Wang, and Kate Saenko. Visda: The visual domain adaptation challenge. *arXiv preprint arXiv:1710.06924*, 2017. 3

[14] K. Saenko, B. Kulis, M. Fritz, and T. Darrell. Adapting visual category models to new domains. In *ECCV*, 2010. 3

[15] Kuniaki Saito, Kohei Watanabe, Yoshitaka Ushiku, and Tatsuya Harada. Maximum classifier discrepancy for unsupervised domain adaptation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3723–3732, 2018. 3

[16] Swami Sankaranarayanan, Yogesh Balaji, Carlos D Castillo, and Rama Chellappa. Generate to adapt: Aligning domains using generative adversarial networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8503–8512, 2018. 3

[17] H. Venkateswara, J. Eusebio, S. Chakraborty, and S. Panchanathan. Deep hashing network for unsupervised domain adaptation. In *CVPR*, 2017. 3

[18] J. Yu and T. Huang. Autoslim: Towards one-shot architecture search for channel numbers. *ArXiv*, abs/1903.11728, 2019. 3

[19] J. Yu and T. Huang. Universally slimmable networks and improved training techniques. In *ICCV*, 2019. 3

[20] J. Yu, L. Yang, N. Xu, J. Yang, and T. Huang. Slimmable neural networks. 2018. 3

[21] Yuchen Zhang, Tianle Liu, Mingsheng Long, and Michael Jordan. Bridging theory and algorithm for domain adaptation. In *International Conference on Machine Learning*, pages 7404–7413. PMLR, 2019. 3

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