Learning Feature Alignment Architecture for Domain Adaptation

Zhixiong Yue[1,2,3], Pengxin Guo[2], Yu Zhang[2,4], and Christy Liang[3]
[1]*Research Institute of Trustworthy Autonomous Systems, Southern University of Science and Technology*
[2]*Department of Computer Science and Engineering, Southern University of Science and Technology*
[3]*School of Computer Science, University of Technology Sydney*
[4]*Peng Cheng Laboratory*
{yuezx,12032913}@mail.sustech.edu.cn, yu.zhang.ust@gmail.com, jie.liang@uts.edu.au

***Abstract*—In domain adaptation, where the feature distributions of the source and target domains are different, various distance-based methods have been proposed to handle the domain shift by minimizing the discrepancy between the source and target domains. These methods use hand-crafted bottleneck networks, which might hinder the alignment of hidden feature representations extracted from both domains. In this paper, we propose a new method called Alignment Architecture Search with Population Correlation (AASPC) to automatically learn the architecture of the bottleneck network that can align the source and target domains. The proposed AASPC method introduces a new similarity function called Population Correlation (PC) to measure the domain discrepancy. The proposed AASPC method leverages PC to learn the alignment architecture and domain-invariant feature representation. Experiments on several benchmark datasets, including Office-31, Office-Home, and VisDA-2017, show the effectiveness of the proposed AASPC method.**

***Index Terms*—domain adaptation, neural architecture search, transfer learning**

I. Introduction

With access to large-scale labeled data, deep neural networks have achieved state-of-the-art performance among a variety of machine learning problems and applications [1]–[3]. However, with intolerably time-consuming and labor-expensive costs, it is hard for a target domain of interest to collect enough labeled data for model training. One solution is to transfer a deep neural network trained on a data-sufficient source domain to the target domain where only unlabeled data is available. However, this learning paradigm suffers from the shift in data distributions across different domains, which brings a major obstacle in adapting predictive models for the target task.

Domain Adaptation (DA) [4], [5] aims to learn a high-performance learner on a target domain via utilizing the knowledge transferred from a source domain, which has a different but related data distribution to the target domain. Many DA methods aim to bridge the gap between source and target domains to apply the classifier learned in the source domain to the target domain. To achieve this goal, recent DA works can be grouped into two main categories: *distance-based* methods [6]–[15] and *adversarial-based* DA methods [16]–[20]. For distance functions adopted by DA, the first attempt is the *Proxy $\mathcal{A}$-distance* [7], which aims to minimize the generalization error by discriminating between source and target samples. *Maximum Mean Discrepancy* is a popular distance measure between two domains and it has been used in Deep Domain Confusion [9] and Deep Adaptation Network [10].

Corresponding author: Yu Zhang.

Although numerous distance-based DA methods have been proposed, learning the domain-invariant feature representation is still challenging. Distance alone in a high-dimensional space may be difficult to reflect the domain discrepancy adequately [21], [22]. The alignment of hidden feature representations also relies heavily on the network architecture design. However, the network architecture of existing methods is manually designed by experts, hence it is hard to guarantee that hidden feature representations from different domains can be well aligned since the difficulty levels of different DA tasks vary. For instance, a complex task may require a more sophisticated network architecture than an easy task, making a hand-crafted network architecture fail to do the alignment for tasks with varying difficulty levels, limiting DA methods' capacity and versatility.

To alleviate those limitations, in this paper, we propose a new similarity function called Population Correlation (PC) to measure the similarity between the source and target domains. A learning model can learn a domain-invariant feature representation by maximizing the PC between the source and target domains. Specifically, maximizing PC can force the two domains to have similar distributions since the PC is the maximum pairwise correlations between source and target samples. To further align hidden feature representations between source and target domains, we design a reinforcement-based Neural Architecture Search (NAS) method called Alignment Architecture Search with Population Correlation (AASPC) to learn deep alignment architecture. In this way, AASPC can better learn domain-invariant feature representations for different DA tasks. To the best of our knowledge, the proposed AASPC method is the first NAS framework designed for distance-based DA methods. AASPC is also one of few works integrating NAS methods into deep DA methods. Our contributions are summarized as follows:

- We propose a new similarity measure, i.e., PC, to measure

the domain similarity. By maximizing the PC between the source and target domains, our method can learn domain-invariant feature representation.
- We design the AASPC framework to search an optimal network architecture to align hidden features between the source and target domains.
- Experimental results on three benchmark datasets demonstrate the effectiveness of the proposed methods.

II. Related Work

a) Domain Adaptation: DA aims to transfer the knowledge learned from a source domain with labeled data to a target domain without labeled data, where there is a domain shift between domains. As discussed in the introduction, recent works in DA can be mainly grouped into two categories: distance-based methods and adversarial DA methods. In this paper, we mainly focus on *distance-based* methods, which minimize the discrepancy between the source and target domains via some measures, including the maximum mean discrepancy used in Deep Domain Confusion [9], Deep Adaptation Network [10] and Joint Adaptation Networks (JAN) [23], the *second-order statistics* utilized in CORrelation ALignment (CORAL) [12], [13], and the Central Moment Discrepancy (CMD) [14].

b) Neural Architecture Search: NAS aims to design the architecture of a neural network in an automated way. Compared with the manually designed architecture of neural networks, NAS has demonstrated the capability to find architecture with state-of-the-art performance in various tasks [24]–[26]. For example, the NAS-FPN method [26] leverages NAS to learn an effective architecture of the feature pyramid network for object detection.

Although NAS can achieve satisfactory performance, the high computational cost of the searching procedure makes NAS less attractive. To accelerate the search procedure, one-shot NAS method leverages a supernet, which contains all the candidate architecture in the search space. In the supernet, weights of operations on edges are shared across different candidate architecture. ENAS [24] employs a reinforcement-based method to train a controller that samples architecture from a supernet with a weight sharing mechanism. DARTS [25] search architecture with a differentiable objective function based on a supernet that uses the softmax function to contain all candidate operations on each edge. The final architecture is determined based on the weights corresponding to the candidate operations on each edge.

c) Learning Architecture for Domain Adaptation: There are few works on NAS for DA. To improve the generalization ability of neural networks for DA, [27] analyze the generalization bound of neural architecture and propose the AdaptNAS method to adapt neural architecture between domains. [28] propose a DARTS-like method for DA, which combines DARTS and DA into one framework. [29] aim to learn an auxiliary branch network from data for an adversarial DA method. Different from those works, this paper aims to leverage NAS to learn alignment architecture and domain-invariant feature representation based on PC.

III. Methodology

In this section, we introduce the proposed PC similarity and the AASPC method.

A. Population Correlation

We first present the definition of PC. Here we study DA under the unsupervised setting. That is, the target domain has unlabeled data only. In DA, the source domain $\mathcal{D}_s = \{(\mathbf{x}_i^s, \mathbf{y}_i^s)\}_{i=1}^{n_s}$ has n_s labeled samples and the target domain $\mathcal{D}_t = \left\{\mathbf{x}_j^t\right\}_{j=1}^{n_t}$ has n_t unlabeled samples. To adapt the classifier trained on the source domain to the target domain, one solution is to minimize the domain discrepancy or equivalently maximize the domain similarity. To achieve this, we propose the PC to measure the similarity between the source and target domains. Specifically, suppose $F(\cdot)$ is the feature extraction network. Then the PC between the source and target domains can be computed based on each pair of source and target samples as

$$\begin{aligned}\mathrm{PC}(\mathcal{D}^s, \mathcal{D}^t) =& \frac{1}{n_s}\sum_{i=1}^{n_s}\max_{j\in[n_t]}\operatorname{corr}\left(F(\mathbf{x}_i^s), F(\mathbf{x}_j^t)\right) \\ &+ \frac{1}{n_t}\sum_{j=1}^{n_t}\max_{i\in[n_s]}\operatorname{corr}\left(F(\mathbf{x}_i^s), F(\mathbf{x}_j^t)\right),\end{aligned} \tag{1}$$

where $\|\cdot\|_2$ denotes the ℓ_2 norm of a vector, $\operatorname{corr}(\mathbf{x}_1, \mathbf{x}_2) = \frac{\mathbf{x}_1^T\mathbf{x}_2}{\|\mathbf{x}_1\|_2\|\mathbf{x}_2\|_2}$ denotes the correlation between two vectors, and $[n]$ denotes a set of integers $\{1, \ldots, n\}$ for an integer n. Here we use the cosine similarity to calculate the correlation between two vectors, thus the larger the PC value is, the more similar the two domains are.

For DA tasks, the hidden feature representations learned by the feature extraction network should be not only discriminative to train a strong classifier but also domain-invariant to both the source and target domains. Only maximizing the PC can help learn a domain-invariant feature representation and only minimizing the classification loss is to learn a discriminative feature representation. Therefore, we combine the classification loss and the PC to obtain the final objective function, which is formulated as

$$\mathcal{L}_{\mathrm{PC}} = \frac{1}{n_s}\sum_{i=1}^{n_s} l(C(F(\mathbf{x}_i^s)), y_i^s) - \lambda \mathrm{PC}(\mathcal{D}^s, \mathcal{D}^t), \tag{2}$$

where λ is a trade-off parameter, $C(\cdot)$ denotes the classification layer, and $l(\cdot, \cdot)$ denotes the classification loss such as the cross-entropy loss.

By minimizing Eq. (2), the final learned feature representations are not only discriminative for classification but also domain-invariant for the adaptation.

B. AASPC

In this section, we introduce the proposed AASPC framework that finds an optimal alignment architecture for source and target domains. An overview of the AASPC framework is shown in Figure 1.

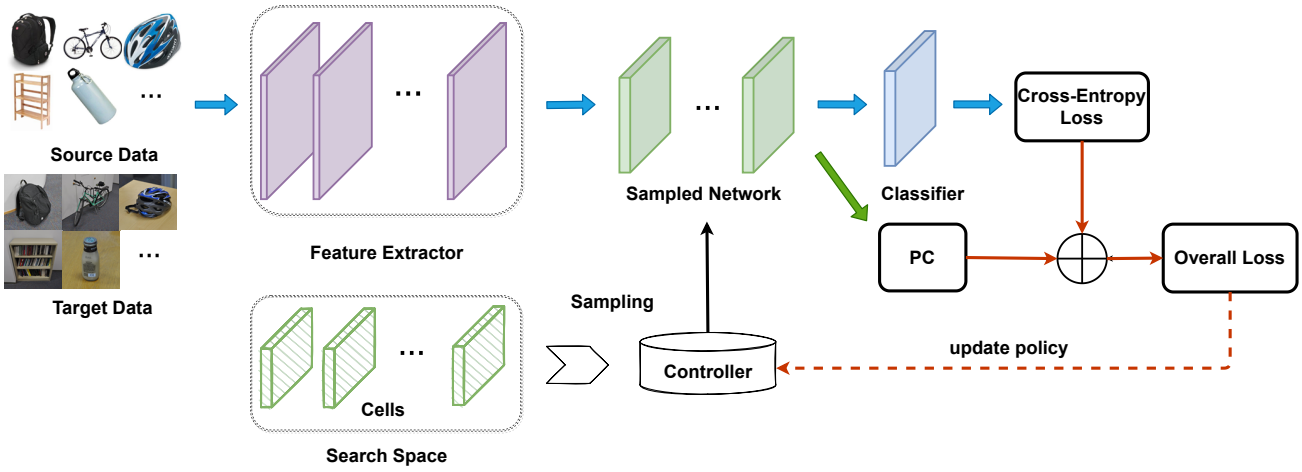


Fig. 1. Overview of the AASPC framework. Source and target data first go through the feature extractor to extract hidden features. The controller samples cell choices for each cell and connections between the cells from search space to generate the architecture of the sampled network. Source and target data with the extracted feature representation then go through the sampled network. Finally, the cross-entropy loss is minimized and the PC is maximized. The controller's policy is updated by the reward of the negative overall loss.

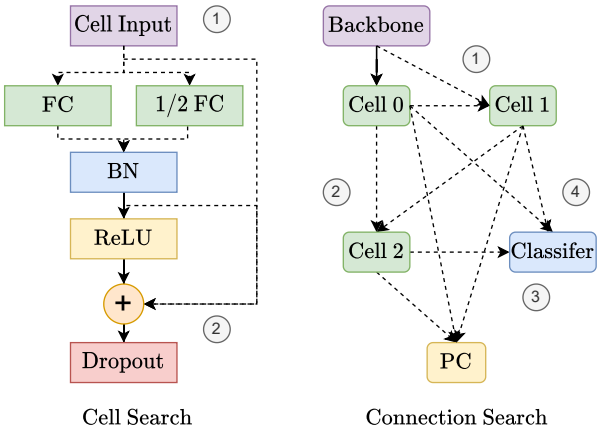


Fig. 2. The search space of the DAMPC-NAS method. Dashed lines represent possible search choices and numbered grey circles indicate the order of choices generated from the controller.

Algorithm 1: AASPC

Input : source data $\mathcal{D}_s$, target data $\mathcal{D}_t$, the number of training epochs n_{epochs}

Output: The searched architecture with learned weights

initialize controller;
for $i \leftarrow 0$ **to** n_{epochs} **do**
  sample $\mathcal{A}_m$ from $\mathcal{A}_{space}$ with policy $\pi(m;\theta)$;
  for *mini-batch in* $\mathcal{D}_s$ *and* $\mathcal{D}_t$ **do**
    compute $\mathcal{L}_{\text{PC}}$ in Eq. (2) with $\mathcal{A}_m$;
    update ω_m in $\mathcal{A}_{space}$ with $\mathcal{L}_{\text{PC}}$;
  end
  calculate reward of $\mathcal{A}_m$ as $R_m = -\mathcal{L}_{\text{PC}}$;
  update θ in $\pi(m;\theta)$ with reward R_m;
end
Return: $\mathcal{A}_m$ with trained weights ω_m

1) Cell-based Search Space: We design the search space on the top of the Resnet-50 backbone, whose architecture is kept fixed, and hence we only search the architecture after the backbone. The search space of the AASPC method consists of two parts: within cells and between cells. We design the cell as the composition of the fully connected layer, batch-norm layer, dropout layer, and the associated activation functions. Within the cell, we search for the size of the fully connected layer and the location of the skip connection. Specifically, the search choice of the fully connected layer in a cell can be 'the same as input size' or 'the half of input size'. The starting location of the skip connection can be chosen from the cell input, the fully connected layer, and the batch-norm layer. We search for input and output connections between the cells of the N cells. For example, if there are three cells in the search space, i.e., $N = 3$, the input of "Cell 1" can be chosen from the outputs of "Backbone" and "Cell 0", and the input of "Cell 2" can be chosen from the outputs of "Cell 0" and "Cell 1", hence the input of a cell can be chosen from the outputs of the previous two cells. The calculation of PC can be chosen from one of all cells' outputs. Moreover, One of the outputs from the N cells, i.e., "Cell 0", "Cell 1" and "Cell 2", can connect to the classifier trained on source domain data. Hence, the total search space has $(2\times3)^N 2^{N-1} N^2$ configurations. An illustration of the search space in the AASPC method is shown in Figure 2. In experiments, for efficiency, we use the search space with $N = 3$ cells for all experiments.

2) Searching Alignment Architecture: The searching algorithm for the AASPC method is described in Algorithm 1. AASPC is a reinforcement-based NAS framework which leverages a controller network to sample architecture from the search space. The controller network is a LSTM that samples search choice via a softmax classifier. We denote by θ the learnable parameters of the controller. The policy of the controller is denoted by $\pi(m;\theta)$.

In each epoch, the training procedure of AASPC consists of two phases. In the first phase, we fix the parameters of the

TABLE I
ACCURACY (%) ON THE OFFICE-31 DATASET WITH RESNET-50 AS THE BACKBONE.

Type	Method	A→D	A→W	D→A	D→W	W→A	W→D	Avg
Source Only	ResNet-50 [3]	68.9	68.4	62.5	96.7	60.7	99.3	76.1
Dist. Based	JDA [30]	80.7	73.6	64.7	96.5	63.1	98.6	79.5
	DDC [9]	76.5	75.6	62.2	96.0	61.5	98.2	78.3
	DAN [10]	78.6	80.5	63.6	97.1	62.8	99.6	80.4
	D-CORAL [13]	81.5	77.0	65.9	97.1	64.3	99.6	80.9
	JAN [23]	84.7	85.4	68.6	97.4	70.0	99.8	84.3
	MDDA [31]	86.3	86.0	**72.1**	97.1	**73.2**	99.2	85.7
Adv. Based	DANN [32]	79.7	82.0	68.2	96.9	67.4	99.1	82.2
	ADDA [19]	77.8	86.2	69.5	96.2	68.9	98.4	82.9
	CAN [33]	85.5	81.5	65.9	98.2	63.4	99.7	82.4
	DDAN [31]	84.9	88.8	65.3	96.7	65.0	**100.0**	83.5
With NAS	ABAS [29]	87.6	89.4	64.1	98.4	69.3	99.8	84.8
	AASPC (Ours)	**90.8**	**93.1**	70.4	**98.7**	69.1	**100.0**	**87.0**

controller θ and train the shared weights ω in the search space $\mathcal{A}_{space}$. Specifically, the controller samples an architecture $\mathcal{A}_m$ from the search space $\mathcal{A}_{space}$ with policy $\pi(m;\theta)$. For each mini-batch from $\mathcal{D}_s$ and $\mathcal{D}_t$, $\mathcal{L}_{\text{PC}}$ is computed according to Eq. (2) and the shared weights ω_m of the sampled architecture are updated by minimizing $\mathcal{L}_{\text{PC}}$. In the second phase, we fix all the shared weights ω in the search space $\mathcal{A}_{space}$ and update the parameter θ of the controller. Specifically, after one epoch of training, $-\mathcal{L}_{\text{PC}}$ is used as the reward to update the policy $\pi(m;\theta)$ in the controller. The gradient is computed via the REINFORCE algorithm [34] with a moving average baseline.

In summary, the AASPC method trains a supernet that contains all shared parameters in the search space during the searching process. The AASPC method samples a child network in each epoch to calculate the loss function defined in Eq. (2) and updates its shared parameters in the search space. Parameters in the controller are updated by the reward, which is the negative loss of the sampled child network. After searching, all weights of the final architecture are retained for testing. Different from two-stage one-shot NAS methods, there is no need for the AASPC method to retrain the final architecture from scratch for testing since AASPC can directly optimize the objective in Eq. (2), which is just the negative reward for the controller, in an end-to-end manner. In this way, the architecture is optimized alongside child networks' parameters. Therefore, the final architecture derived from the AASPC method can be deployed directly without parameter retraining, which improves the overall efficiency.

IV. EXPERIMENTS

This section empirically evaluates the proposed method.

A. Setup

We conduct experiments on three benchmark datasets, including Office-31 [35], Office-Home [36], and VisDA-2017 [37]. The Office-31 dataset has 4,652 images in 31 categories collected from three distinct domains: *Amazon* (**A**), *Webcam* (**W**) and *DSLR* (**D**). We can construct six transfer tasks: **A** $\rightarrow$ **W**, **D** $\rightarrow$ **W**, **W** $\rightarrow$ **D**, **A** $\rightarrow$ **D**, **D** $\rightarrow$ **A**, and **W** $\rightarrow$ **A**. The Office-Home dataset consists of 15,500 images in 65 object classes under the office and home settings, forming four extremely dissimilar domains: *Artistic* (**Ar**), *Clip Art* (**Cl**), *Product* (**Pr**), and *Real-World* (**Rw**) and 12 transfer tasks. The VisDA-2017 dataset has over 280K images across 12 classes. It contains two very distinct domains: **Synthetic**, which contains renderings of 3D models from different angles and with different lighting conditions, and **Real** that are natural images. We study a transfer task: Synthetic $\rightarrow$ Real on this dataset.

We compare the proposed **AASPC** method with state-of-the-art DA methods, including Joint Distribution Adaptation (**JDA**) [30], Deep Domain Confusion (**DDC**) [9], Deep Adaptation Network (**DAN**) [10], Domain Adversarial Neural Network (**DANN**) [32], Correlation Alignment for Deep Domain Adaptation (**D-CORAL**) [13], Residual Transfer Networks (**RTN**) [38], Joint Adaptation Networks (**JAN**) [23], Adversarial Discriminative Domain Adaptation (**ADDA**) [19], Conditional Domain Adversarial Networks (**CDAN**) [17], Collaborative and Adversarial Network (**CAN**) [33], Manifold Dynamic Distribution Adaptation (**MDDA**) [31], and Dynamic Distribution Adaptation Network (**DDAN**) [31]. The results of baseline methods are directly reported from DDAN [31] and CDAN [17].

We leverage the ResNet-50 network [3] pretrained on the ImageNet dataset as the backbone for the feature extraction. For optimization, we use the mini-batch SGD with the Nesterov momentum 0.9. The learning rate is adjusted by $\eta_p = \eta_0(1+\alpha p)^{-\beta}$, where p is the index of training steps, η_0 = 0.1, α = 0.001, and β = 0.75. The batch size is set to 128 for all the datasets.

B. Results

The classification results on the Office-31 dataset are shown in Table I. As illustrated in Table I, the proposed AASPC method achieves the best average accuracy.

In four out of six transfer tasks, AASPC performs the best, especially on transfer tasks A→D and A→W, which is transferring from a large source domain to a small target

TABLE II
ACCURACY (%) ON THE OFFICE-HOME DATASET WITH RESNET-50 AS THE BACKBONE.

Type	Method	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
Source Only	ResNet-50 [3]	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
Dist. Based	JDA [30]	38.9	54.8	58.2	36.2	53.1	50.2	42.1	38.2	63.1	50.2	44.0	68.2	49.8
	DAN [10]	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
	D-CORAL [13]	42.2	59.1	64.9	46.4	56.3	58.3	45.4	41.2	68.5	60.1	48.2	73.1	55.3
	JAN [23]	45.9	61.2	68.9	50.4	59.7	61.0	45.8	43.4	70.3	63.9	52.4	76.8	58.3
Adv. Based	DANN [32]	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 [17]	46.6	65.9	73.4	55.7	62.7	64.2	51.8	**49.1**	74.5	68.2	**56.9**	**80.7**	62.8
	DDAN [31]	**51.0**	66.0	73.9	57.0	63.1	65.1	52.0	48.4	72.7	65.1	56.6	78.9	62.5
With NAS	AASPC (Ours)	46.53	**68.42**	**75.24**	**58.3**	**66.3**	**67.48**	**56.94**	44.77	**75.33**	**69.26**	51.94	80.33	**63.4**

domain and in the other two tasks, the AASPC method performs slightly worse than the best baseline method, which implies that the proposed AASPC model works well when the source data is sufficient and it can learn transferable feature representations for effective domain adaptation.

Table II shows the classification results on the Office-Home dataset. According to the results, AASPC achieves the best average accuracy and performs the best in eight out of twelve transfer tasks, while transferring from a large source domain to a small target domain (i.e., Cl→Ar, Pr→Ar, and Rw→Ar), AASPC achieves the best performance and this phenomenon is similar to the Office-31 dataset, which again demonstrates that the proposed AASPC model works well when the source data is sufficient.

TABLE III
ACCURACY (%) ON THE VISDA-2017 DATASET WITH RESNET-50 AS THE BACKBONE.

Type	Method	Synthetic→Real
Source Only	ResNet-50 [3]	45.6
Dist. Based	DAN [10]	53.0
	RTN [38]	53.6
	JAN [23]	61.6
Adv. Based	DANN [32]	55.0
	CDAN [17]	66.8
With NAS	AASPC (Ours)	**68.75**

According to experimental results on the most challenging VisDA-2017 dataset shown in Table III, the proposed AASPC method outperforms all the baseline methods by improving by 1.9% over state-of-the-art baseline methods (i.e., CDAN) on this dataset, which again demonstrates the effectiveness of the proposed method.

C. Discussion

1) Ablation Study: To investigate the efficacy of key designs of the proposed AASPC method, we conduct ablation study on the Office-31 dataset by comparing with variants of AASPC, including Source Only (no distance calculation and architecture search), AAS (AASPC without population correlation), and PC (AASPC without alignment architecture search). According to the results shown in Table V, the AASPC method outperforms both AAS and PC methods.

TABLE IV
COMPARISON OF PC WITH OTHER DISTANCE FUNCTIONS ON THE VISDA-2017 DATASET WITH RESNET-50 AS THE BACKBONE.

Measurement	Synthetic→Real
None	57.68
Proxy $\mathcal{A}$-distance	56.36
KL-divergence	56.27
MMD	58.76
CORAL	56.66
CMD	56.65
PC (Ours)	65.25

AAS is inferior to AASPC with a drop of 3.26%, while it performs better than Source Only by 2.1% in terms of the average accuracy. PC performs better than Source Only by 5.36% on the average accuracy. AASPC further improves over PC by 2.41% and 1.76% on the A→D and A→W tasks in terms of the accuracy, respectively. This experiment verifies the effectiveness of both the AAS and PC components in the AASPC method.

2) Effectiveness of Population Correlation: To demonstrate the effectiveness of the proposed PC, we replace the measurement with other widely used distance functions on the Office-31, Office-Home, and VisDA-2017 datasets. We then compare the performance of PC with these distance functions, including Proxy $\mathcal{A}$-distance, Kullback-Leibler divergence (KL-divergence), Maximum Mean Discrepancies (MMD), CORrelation ALignmen (CORAL), and Central Moment Discrepancy (CMD). For a fair comparison, we only replace the minus of the PC with these distance functions in Eq. (2). Specifically, we adopt the ResNet-50 as the backbone, following with the bottleneck layer (consisting of a fully connected layer, a batch normalization layer, a ReLU activation function, and a dropout function) used for generating hidden features and a fully connected layer used for prediction. According to experimental results shown in Tables VI, IV and VII, we can see that none of the distance functions can obtain performance improvement compared with no distance function used (i.e., ResNet-50). One possible reason is that the normalization layer used in the bottleneck layer has improved the performance of the ResNet-50 and adapting these distance functions can not improve the performance further. However, the proposed PC can still obtain performance improvement over ResNet-50, which indicates the effectiveness of the proposed PC.

TABLE V
ABLATION STUDY ON THE OFFICE-31 DATASET WITH RESNET-50 AS THE BACKBONE.

Method	A→D	A→W	D→A	D→W	W→A	W→D	Avg
Source Only	83.53	80.50	64.61	98.49	62.69	100.0	81.64
AAS (AASPC w/o PC)	86.35	89.18	64.75	98.24	63.90	100.0	83.74
PC (AASPC w/o AAS)	88.35	91.32	70.36	98.49	69.05	100.0	86.26
AASPC	90.76	93.08	70.36	98.74	69.05	100.0	87.0

TABLE VI
COMPARISON OF PC WITH OTHER DISTANCE FUNCTIONS ON THE OFFICE-31 DATASET WITH RESNET-50 AS THE BACKBONE.

Measurement	A→D	A→W	D→A	D→W	W→A	W→D	Avg
None	83.53	80.50	64.61	98.49	62.69	100.0	81.64
Proxy $\mathcal{A}$-distance	82.73	81.01	64.04	98.11	61.77	100.0	81.28
KL-divergence	83.94	79.75	63.90	97.86	63.51	99.80	81.46
MMD	83.13	79.25	64.11	98.74	63.12	100.0	81.39
CORAL	84.34	80.25	64.61	98.24	62.80	99.80	81.67
CMD	82.93	79.50	64.29	98.62	63.10	100.0	81.41
PC	88.35	91.32	70.36	98.49	69.05	100.0	86.26

TABLE VII
COMPARISON OF PC WITH OTHER DISTANCE FUNCTIONS ON THE OFFICE-HOME DATASET WITH RESNET-50 AS THE BACKBONE.

Measurement	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
None	43.41	66.55	74.64	56.61	63.98	65.32	53.36	39.36	72.64	64.73	46.30	76.55	60.29
Proxy $\mathcal{A}$-distance	43.21	65.44	74.85	55.09	62.51	65.37	52.33	38.63	72.83	64.57	46.23	76.66	59.81
KL-divergence	44.01	66.75	74.50	55.75	63.42	66.51	52.74	38.14	73.43	65.84	44.79	77.13	60.25
MMD	43.78	66.28	74.48	55.62	64.07	66.19	53.40	38.30	73.15	64.89	45.52	77.43	60.26
CORAL	44.15	65.85	74.16	55.42	63.01	66.83	52.95	39.38	72.53	65.14	45.96	77.07	60.20
CMD	44.40	65.92	74.50	54.68	63.37	67.07	52.78	38.88	72.94	65.64	45.29	77.36	60.24
PC	46.19	66.03	73.7	57.89	63.48	65.80	56.94	44.19	75.58	69.02	51.11	78.89	62.24

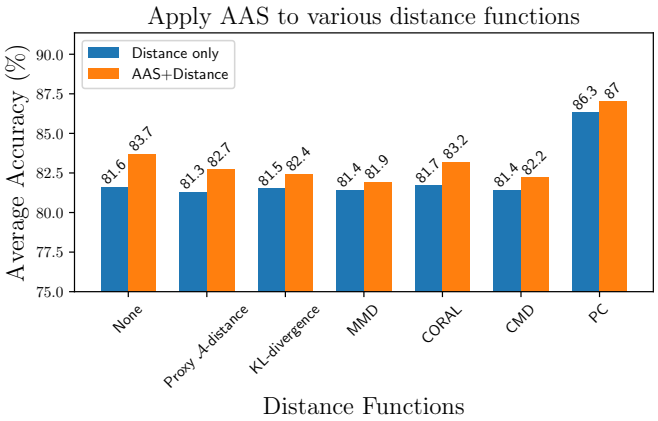


Fig. 3. Apply AAS to various distance functions.

3) Effectiveness of Alignment Architecture Search: To demonstrate the effectiveness of the alignment architecture search (AAS) in the AASPC method, we apply AAS to various distance functions on the Office-31 dataset. Specifically, we modify Algorithm 1 to search alignment architecture for other measurements by replacing the minus of the PC with other distance functions in $\mathcal{L}_{\text{PC}}$. According to experimental results shown in Figure 3, AAS can improve the performance of various distance functions on the Office-31 dataset, which demonstrates the effectiveness and generalization ability of the alignment architecture search.

TABLE VIII
COMPARISON OF TRAINING COMPLEXITY ON OFFICE-31 DATASET. TRAINING TIME PER EPOCH AND GPU MEMORY CONSUMPTION ARE RECORDED ON ONE SINGLE NVIDIA TESLA V100S GPU WITH BATCH SIZE 128.

Method	Time per Epoch (s)	GPU memory (M)	Avg Acc. (%)
Source Only	11.4	24865	65.3
PC	12.9	25085	86.3
AASPC	13.6	25185	87.0

4) Hyper-parameter Sensitivity: We investigate the sensitivity with respect to two hyper-parameters: the training batch size and trade-off parameter λ in Eq. (2). We set the value of batch size from 32 to 256 and $\lambda \in \{0.1, 1, 10\}$ to obtain the performance change of AASPC. According to Figure 6, when the batch size is small, the performance becomes worse. This is because minimizing the loss in Eq. (1) may make samples in different classes close to each other, especially when λ is relatively large. The performance of AASPC is stable when the batch size is larger than 128 for both $\lambda = 0.1$ and $\lambda = 1$, which indicates that AASPC is relatively insensitive to a large batch size when λ is not so large, i.e., 0.1 or 1.

5) Complexity Analysis: In Table VIII, we compare the training time and performance of AASPC and PC with the source only baseline, which does not compute any distance functions during the training process. Compared with the source only baseline, the training time per epoch and occupied

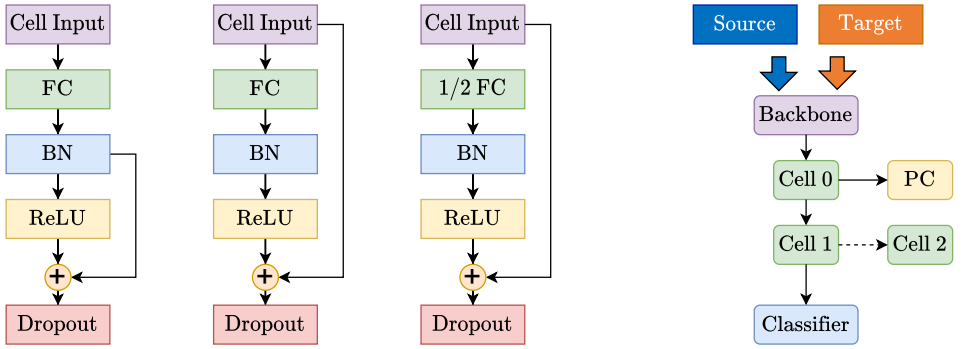
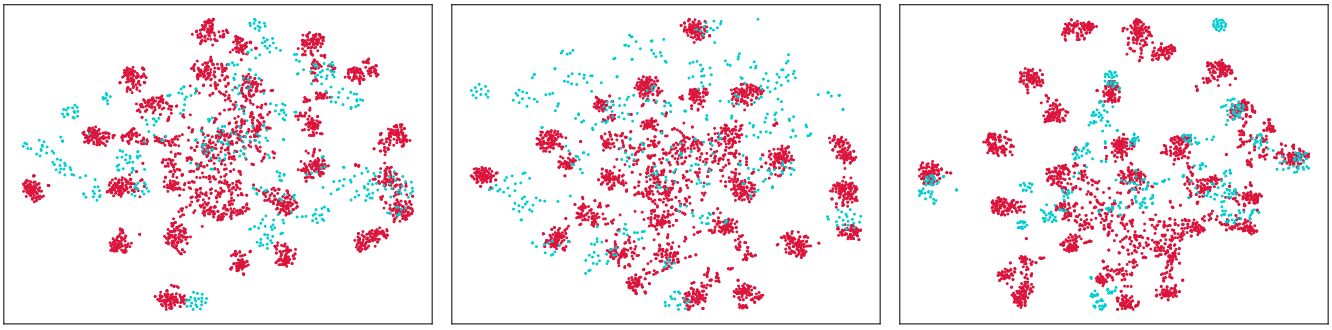


Fig. 4. Searched architecture for transfer task D→W of the Office-31 dataset. Left: architecture within the three cells. Right: connections between the three cells, PC, and classifier.



(a) Source Only (b) PC (c) AASPC (Ours)

Fig. 5. t-SNE visualization of different methods for the transfer task A→D in the Office-31 dataset.

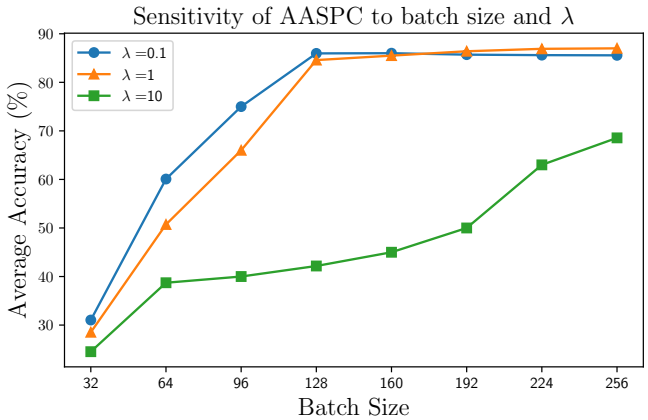


Fig. 6. Sensitivity of AASPC to batch size and λ on the Office-31 dataset.

GPU memory slightly increase for PC and AASPC while the average accuracy dramatically improves. Hence, both PC and AASPC methods introduce negligibly additional computation costs for considerable performance improvement.

6) Learned Architecture: Figure 4 shows the architecture found by AASPC for the transfer task D→W constructed on the Office-31 dataset. The left part of Figure 4 shows the search choice within the three cells found by the AASPC method and the right part of Figure 4 shows the connections among the three cells, PC and classifier. In Cell 0, the AASPC method chooses the FC layer with the same size as the input and the skip connection is connected to the batch-norm layer. In Cell 1, the choice of FC is the same as Cell 0 but the skip connection starts from the cell input. In Cell 2, the skip connection is the same as Cell 2 but the FC layer is of half size of the input. For connections between cells, the AASPC method chooses to use the output of Cell 0 to calculate the PC and the output of Cell 1 to calculate the classification loss. For a simple transfer task D→W, the searched architecture only has two cells, which indicates that the AASPC method can adaptively learn architecture depending on the complexity of the DA task. Moreover, the location of the skip connection moves forward in Cell 1 and Cell 2 when compared with Cell 0, which helps reduce the network depth and alleviates the vanishing gradient problem.

7) Feature Visualization: In Figure 5, we visualize the hidden feature representations of the transfer task A→D constructed on the Office-31 dataset learned by source samples only, source and target samples with PC, and source and target samples with AASPC, respectively. According to Figure 5, we can see that samples with the representations learned by PC are more distinguishable than those by source only. The representations learned by AASPC are more separable than those by PC, which implies that the proposed AASPC method can learn discriminative and transferable feature representations for DA.

V. Conclusion

In this paper, we propose a new PC function that can measure domain similarity. We further design the AASPC framework that searches deep alignment architecture for DA tasks. Experiments results on the Office-31, Office-Home, and VisDA-2017 datasets demonstrate the effectiveness of the proposed method. Moreover, the proposed AASPC framework has shown its potential to search alignment architecture for various DA methods. In our future studies, we try to extend the proposed AASPC framework to learn architecture for other DA methods and settings.

Acknowledgements. This work is supported by Shenzhen fundamental research program JCYJ20210324105000003 and NSFC general grant 62076118.

References

[1] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Advances in neural information processing systems*, vol. 25, pp. 1097–1105, 2012.

[2] M. Oquab, L. Bottou, I. Laptev, and J. Sivic, "Learning and transferring mid-level image representations using convolutional neural networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2014, pp. 1717–1724.

[3] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[4] S. J. Pan and Q. Yang, "A survey on transfer learning," *IEEE Transactions on knowledge and data engineering*, vol. 22, no. 10, pp. 1345–1359, 2009.

[5] Q. Yang, Y. Zhang, W. Dai, and S. J. Pan, *Transfer learning*. Cambridge University Press, 2020.

[6] S. Ben-David, J. Blitzer, K. Crammer, F. Pereira *et al.*, "Analysis of representations for domain adaptation," *Advances in neural information processing systems*, vol. 19, p. 137, 2007.

[7] S. Ben-David, J. Blitzer, K. Crammer, A. Kulesza, F. Pereira, and J. W. Vaughan, "A theory of learning from different domains," *Machine learning*, vol. 79, no. 1, pp. 151–175, 2010.

[8] F. Zhuang, X. Cheng, P. Luo, S. J. Pan, and Q. He, "Supervised representation learning: Transfer learning with deep autoencoders," in *Twenty-Fourth International Joint Conference on Artificial Intelligence*, 2015.

[9] E. Tzeng, J. Hoffman, N. Zhang, K. Saenko, and T. Darrell, "Deep domain confusion: Maximizing for domain invariance," *arXiv preprint arXiv:1412.3474*, 2014.

[10] M. Long, Y. Cao, J. Wang, and M. Jordan, "Learning transferable features with deep adaptation networks," in *International conference on machine learning*. PMLR, 2015, pp. 97–105.

[11] N. Courty, R. Flamary, D. Tuia, and A. Rakotomamonjy, "Optimal transport for domain adaptation," *IEEE transactions on pattern analysis and machine intelligence*, vol. 39, no. 9, pp. 1853–1865, 2016.

[12] B. Sun, J. Feng, and K. Saenko, "Return of frustratingly easy domain adaptation," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 30, no. 1, 2016.

[13] B. Sun and K. Saenko, "Deep coral: Correlation alignment for deep domain adaptation," in *European conference on computer vision*. Springer, 2016, pp. 443–450.

[14] W. Zellinger, T. Grubinger, E. Lughofer, T. Natschläger, and S. Saminger-Platz, "Central moment discrepancy (cmd) for domain-invariant representation learning," *arXiv preprint arXiv:1702.08811*, 2017.

[15] C. Chen, Z. Chen, B. Jiang, and X. Jin, "Joint domain alignment and discriminative feature learning for unsupervised deep domain adaptation," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 33, no. 01, 2019, pp. 3296–3303.

[16] 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.

[17] M. Long, Z. Cao, J. Wang, and M. I. Jordan, "Conditional adversarial domain adaptation," *arXiv preprint arXiv:1705.10667*, 2017.

[18] Z. Pei, Z. Cao, M. Long, and J. Wang, "Multi-adversarial domain adaptation," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 32, no. 1, 2018.

[19] E. Tzeng, J. Hoffman, K. Saenko, and T. Darrell, "Adversarial discriminative domain adaptation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 7167–7176.

[20] K. Saito, K. Watanabe, Y. Ushiku, and T. Harada, "Maximum classifier discrepancy for unsupervised domain adaptation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 3723–3732.

[21] S. Li, C. Liu, Q. Lin, B. Xie, Z. Ding, G. Huang, and J. Tang, "Domain conditioned adaptation network," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, no. 07, 2020, pp. 11 386–11 393.

[22] S. Li, M. Xie, K. Gong, C. H. Liu, Y. Wang, and W. Li, "Transferable semantic augmentation for domain adaptation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 11 516–11 525.

[23] M. Long, H. Zhu, J. Wang, and M. I. Jordan, "Deep transfer learning with joint adaptation networks," in *International conference on machine learning*. PMLR, 2017, pp. 2208–2217.

[24] H. Pham, M. Y. Guan, B. Zoph, Q. V. Le, and J. Dean, "Efficient neural architecture search via parameter sharing," *arXiv preprint arXiv:1802.03268*, 2018.

[25] H. Liu, K. Simonyan, and Y. Yang, "DARTS: differentiable architecture search," in *Proceedings of the 7th International Conference on Learning Representations*, 2019.

[26] G. Ghiasi, T.-Y. Lin, and Q. V. Le, "Nas-fpn: Learning scalable feature pyramid architecture for object detection," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 7036–7045.

[27] Y. Li, Z. Yang, Y. Wang, and C. Xu, "Adapting neural architectures between domains," *Advances in Neural Information Processing Systems*, vol. 33, 2020.

[28] Y. Li and X. Peng, "Network architecture search for domain adaptation," *arXiv preprint arXiv:2008.05706*, 2020.

[29] L. Robbiano, M. R. U. Rahman, F. Galasso, B. Caputo, and F. M. Carlucci, "Adversarial branch architecture search for unsupervised domain adaptation," *arXiv preprint arXiv:2102.06679*, 2021.

[30] M. Long, J. Wang, G. Ding, J. Sun, and P. S. Yu, "Transfer feature learning with joint distribution adaptation," in *Proceedings of the IEEE international conference on computer vision*, 2013, pp. 2200–2207.

[31] J. Wang, Y. Chen, W. Feng, H. Yu, M. Huang, and Q. Yang, "Transfer learning with dynamic distribution adaptation," *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 11, no. 1, pp. 1–25, 2020.

[32] Y. Ganin and V. Lempitsky, "Unsupervised domain adaptation by backpropagation," in *International conference on machine learning*. PMLR, 2015, pp. 1180–1189.

[33] W. Zhang, W. Ouyang, W. Li, and D. Xu, "Collaborative and adversarial network for unsupervised domain adaptation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 3801–3809.

[34] R. J. Williams, "Simple statistical gradient-following algorithms for connectionist reinforcement learning," *Machine learning*, vol. 8, no. 3, pp. 229–256, 1992.

[35] K. Saenko, B. Kulis, M. Fritz, and T. Darrell, "Adapting visual category models to new domains," in *European conference on computer vision*. Springer, 2010, pp. 213–226.

[36] H. Venkateswara, J. Eusebio, S. Chakraborty, and S. Panchanathan, "Deep hashing network for unsupervised domain adaptation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 5018–5027.

[37] X. Peng, B. Usman, N. Kaushik, J. Hoffman, D. Wang, and K. Saenko, "Visda: The visual domain adaptation challenge," *arXiv preprint arXiv:1710.06924*, 2017.

[38] M. Long, H. Zhu, J. Wang, and M. I. Jordan, "Unsupervised domain adaptation with residual transfer networks," *arXiv preprint arXiv:1602.04433*, 2016.