arXiv:2103.10583v1 [cs.CV] 19 Mar 2021

Dynamic Transfer for Multi-Source Domain Adaptation

Yunsheng Li[1], Lu Yuan[2], Yinpeng Chen[2], Pei Wang[1], Nuno Vasconcelos[1]
[1] UC San Diego, [2] Microsoft
{yul554,pew062,nvasconcelos}@ucsd.edu, {luyuan,yiche}@microsoft.com

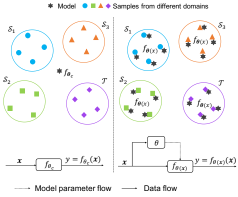
Abstract

Recent works of multi-source domain adaptation focus on learning a domain-agnostic model, of which the parameters are ***static****. However, such a static model is difficult to handle conflicts across multiple domains, and suffers from a performance degradation in both source domains and target domain. In this paper, we present* ***dynamic transfer*** *to address domain conflicts, where the model parameters are adapted to samples. The key insight is that adapting model across domains is achieved via adapting model across samples. Thus, it breaks down source domain barriers and turns multi-source domains into a single-source domain. This also simplifies the alignment between source and target domains, as it only requires the target domain to be aligned with any part of the union of source domains. Furthermore, we find dynamic transfer can be simply modeled by aggregating residual matrices and a static convolution matrix. Experimental results show that, without using domain labels, our dynamic transfer outperforms the state-of-the-art method by more than* 3% *on the large multi-source domain adaptation datasets – DomainNet. Source code is at https://github.com/liyunsheng13/DRT.*

1. Introduction

Multi-source domain adaptation addresses the adaptation from multiple source domains to a target domain. It is challenging because a clear domain discrepancy exists not only between source and target domains, but also among multiple source domains (see exemplar images in Figure 2). This suggests that successful adaptation requires significant *elasticity* of the model to adapt. A nature way to achieve this elasticity is to make model dynamic i.e. the mapping implemented by the model should vary with the input sample.

This hypothesis has not been explored by existing work, e.g. [20, 26], which instead aims to learn a domain agnostic model $f_{\boldsymbol{\theta}_c}$, of *static* parameters $\boldsymbol{\theta}_c$, that works well for all source $\{\mathcal{S}_1, \mathcal{S}_2, ..., \mathcal{S}_N\}$ and target $\mathcal{T}$ domains. We refer to this approach as *static transfer.* As illustrated in Figure 1 (a), the model implements a fixed mapping across all domains. However, learning a domain agnostic model is difficult, since different domains can give rise to very different image distributions. When forcing a model to be domain agnostic, it essentially averages the domain conflict. Thus the performance drops on each source domain. This is validated by our preliminary study. As shown in Figure 2, compared to the optimal model per domain, the static transfer model consistently degrades in each source domain.


Model
Samples from different domains
$\mathcal{S}_1$
$\mathcal{S}_3$
$\mathcal{S}_2$
$\mathcal{T}$
$f_{\boldsymbol{\theta}_c}$
$f_{\boldsymbol{\theta}(x)}$
θ
$\boldsymbol{x}$
$y = f_{\theta_c}(\boldsymbol{x})$
$y = f_{\boldsymbol{\theta}(x)}(\boldsymbol{x})$
Model parameter flow
Data flow

(a) Static Transfer (b) Dynamic Transfer

Figure 1: **Static Transfer vs. Dynamic Transfer**. (a) 'Static Transfer' implements domain adaptation with a static model $f_{\boldsymbol{\theta}_c}$, which has fixed parameters $\boldsymbol{\theta}_c$ to average domain conflict. (b) 'Dynamic Transfer' ($f_{\boldsymbol{\theta}(\boldsymbol{x})}$) adapts the model parameters $\boldsymbol{\theta}(\boldsymbol{x})$ according to samples, which generates a different model per sample and turns multi-source domain adaptation into single-source domain adaptation.

In this paper, we propose *dynamic transfer* to address this issue. As shown in Figure 1(b), it contains a parameter predictor that changes the model parameters on a per-sample basis, i.e. implements mapping $f_{\boldsymbol{\theta}(\boldsymbol{x})}$. It has the advantage of not requiring the definition of domains or the collection of domain labels. In fact, it unifies the problems of single-source and multi-source domain adaptation. By breaking down source domain barriers, it turns multiple source domain adaptation into a single-source domain problem. The only difference is the complexity of this domain.

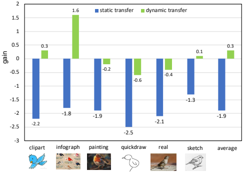


Figure 2: **Static Transfer vs. Dynamic Transfer on the performance degradation of source domains** compared to the oracle results. Both transfer models are tested across source domains. A clear performance degradation (1.9% in average) exists when using static transfer, indicating the conflicts across domains. The degradation is significantly reduced when using dynamic transfer, as it handles the domain variations well. (Best view in color)

The key insight is that adapting model according to domains is achieved statistically by adapting model per sample, since each domain is viewed as a distribution of image samples. The dynamic transfer learns how to adapt the model's parameters and fit to the union of source domains. Thus the alignment between source domains and target domain is significantly simplified, as it is no longer necessary to pull all source domains together with the target domain. In this case, as long as the target domain is aligned to any part of the source domains, the model can be easily adapted to the target samples.

When compared to the domain adaption literature, dynamic transfer introduces a significant paradigm shift. In the literature, most works assume the static network of Figure 1(a) and focus on loss functions. The goal is to define losses that somehow "pull all the domains" together into a shared latent representation. The problem is that the domains are usually very different at the network input. Hence, the force introduced by the loss at the output, to bring them together, is counter-balanced by an input force to keep them apart. This usually leads to a difficult optimization and compromises adaptation performance. The introduction of a dynamic network, as in Figure 1(b), enables a more elastic mapping. In this case, it is not necessary to pull all domains together. The model adaptation given by the dynamic transfer can be generalized to target domain easily when the target domain is shifted to the space formed by entire source domain. In this way, dynamic transfer shifts

the focus of the domain adaptation problem from the design of good *loss functions* to the design of good *network architectures* for dynamic transfer.

An immediate difficulty is that the architecture of Figure 1(b) can be very hard to train, since the parameter predictor cannot generate all parameters for a large model. The question is whether it is possible to perform the model adaptation by only modifying a small subset of parameters on a sample basis. In this work, we show that this is indeed possible by the addition of dynamic residuals to the convolution kernels of a static network. Since the residual blocks can be much smaller than the static ones, this has both very low additional computational cost (less than 0.1%) to aggregate dynamic residuals with static kernel and little tendency to overfit. However, it is shown to significantly enhance the domain adaptation performance. Experimental results show that the proposed *dynamic residual transfer* (DRT) can model domain variation in source domains (see Figure 2) and outperform its static counterpart (MCD [23] method) by a large margin (11.2% on DomainNet). Compared to state-of-the-art multi-source domain adaptation methods [26], it achieves a sizeable gain (3.9%) with a much simpler loss function and training algorithm.

2. Related Work

Single-Source Domain Adaptation approaches adapt a model from a source to a target domain. A common method is to minimize the distance between the two domains. While some methods [3, 24] minimize distance functions defined in terms of first and second order data statistics, others learn a latent space shared across domains by adversarial learning [25, 23, 9, 14]. Although these methods are effective for single-source domain distributions and relatively simple datasets (such as VisDA [21] or Office-31 [22]), they are not competitive for the multi-source domain adaptation problem, due to a more complex data distribution.

Multi-Source Domain Adaptation considers the domain adaptation problem when the source contains domains with a variety of styles. [31] pioneered this problem by adaptively picking the best among a set of hypothesis learned for different source domains. [1] derived an upper bound on the classification error achievable in the target domain, based on the $\mathcal{H}\Delta\mathcal{H}$ divergence. Several methods have been proposed after the introduction of deep learning. Some of these align domains pair-wise. [29] uses a discriminator to align each source domain with the target domain, while [20] matches moments across all pairs of source and target domains. These methods learn one classifier per domain and use their weighted combination to predict the class of target samples. [15] uses mutual learning techniques to align feature distributions among pairs of source and target domains. Other methods focus on the joint alignment of the feature distributions of all domains. [26] models interactions between domains with a knowledge graph. Target sample predictions are based on both their features and relationship to different domains. [13] proposes a meta-learning technique to search the best initial conditions for multi-source domain adaptation. [32] uses an auxiliary network to predict the transferability of each source sample and use it as a weight to learn a domain discriminator. All these works use a static transfer model. In this paper, we propose that the model should instead be dynamic, i.e. a function that changes with samples, and show that this can significantly enhance multi-source domain adaptation.

Dynamic Networks have architectures based on blocks [16, 28, 30, 4] or channels [10, 2, 27] that change depending on the input sample. [16, 28] proposed an input dependent block path that decides whether a network block should be kept or dropped. [30, 4] widen the network by adding new parallel blocks and train an attention module to choose the best combination of features dynamically. [10, 2, 27] rely on feature based attention modules that reweigh features depending on the input example. [27] shows that, for object detection, objects from different domains are best detected with domain dependent features. In this paper, we propose a dynamic convolution residual branch, which adds an input-dependent residual matrix to a static kernel, to implement dynamic multi-source domain adaptation.

3. Method

In this section, we introduce dynamic transfer for multi-source domain adaptation, in which the model is adaptive to the domain implicitly, but adaptive to the input explicitly. It not only has better performance, but also turns multi-source domains into a single-source domain.

3.1. Multi-Source Domain Adaptation

Multi-source domain adaptation (MSDA) aims to transfer a model learned on a source data distribution drawn from several domains $\mathcal{S} = \{\mathcal{S}_1, ..., \mathcal{S}_N\}$ to a target domain $\mathcal{T}$. While the following ideas can be applied to various tasks, we consider a classification model $f_{\boldsymbol{\theta}}$, of parameters $\boldsymbol{\theta}$, which maps images $\boldsymbol{x} \in \mathcal{X}$ to class predictions $y \in \mathcal{Y} = \{1, \dots, C\}$, where C is the number of classes and $\mathcal{X}$ is some image space. The goal is to adapt the parameters $\boldsymbol{\theta}$ of a model learned from a dataset $\mathcal{D}^{\mathcal{S}} = \{(\boldsymbol{x}_i^{\mathcal{S}}, \mathbf{y}_i)\}_{i=1}^{N_{\mathcal{S}}}$ of examples from the source distribution $\mathcal{S}$ ($\mathbf{y}_i$ is the one-hot encoding of the label of example $\boldsymbol{x}_i^{\mathcal{S}}$) to a dataset $\mathcal{D}^{T} = \{\boldsymbol{x}_i^{\mathcal{T}}\}_{i=1}^{N_{\mathcal{T}}}$ of unlabeled examples from the target distribution. Note that, in the most general formulation of the problem, the domain of origin of each source example, $(\boldsymbol{x}_i^{\mathcal{S}}, \mathbf{y}_i)$ is *unknown*. This is ignored by many approaches e.g. [20, 15], that assume a source dataset $\mathcal{D}^{\mathcal{S}} = \{(\boldsymbol{x}_i^{\mathcal{S}}, \mathbf{y}_i, z_i)\}_{i=1}^{N_{\mathcal{S}}}$ contains domain labels $z_i \in \{1, \dots, N\}$ and aligning pairs of domains. We refer to this a domain supervised multi-source domain adaptation.

3.2. Static vs. Dynamic Transfer

The model $f_{\boldsymbol{\theta}}$ is denoted static or dynamic depending on whether the model parameters $\boldsymbol{\theta}$ vary with samples $\boldsymbol{x}$. Static models have constant parameters $\boldsymbol{\theta} = \boldsymbol{\theta}_c$, while dynamic models have parameters $\boldsymbol{\theta} = \boldsymbol{\theta}(\boldsymbol{x})$ that depend on $\boldsymbol{x}$. In the case of deep networks, this implies that layer transfer functions depend on the input $\boldsymbol{x}$. Figure 1 illustrates the static transfer and dynamic transfer model built for multi-source domain adaptation.

Static Transfer. Static transfer, shown on Figure 1(a), consists of learning of a single model $f_{\boldsymbol{\theta}_c}$ that is applied to *all* examples from source and target domains. The model might, for instance, map images into a latent space where all the distributions are aligned. Since the big variation among the input samples, this is a difficult problem and the model $f_{\boldsymbol{\theta}_c}$ usually has sub-optimal performance on all domains.

Dynamic Transfer. In this case the model parameters are a function of the input example $\boldsymbol{x}$ directly, i.e. the model has the form $f_{\boldsymbol{\theta}(\boldsymbol{x})}$ where $\boldsymbol{x} \in \mathcal{S}_1 \bigcup \cdots \bigcup \mathcal{S}_N \bigcup \mathcal{T}$. This is illustrated in Figure 1(b), where there exists a model per sample. Compared to the static transfer, dynamic transfer varies the model according to sample explicitly and chooses domains *implicitly,* relying on the distribution of samples $\boldsymbol{x}$. Dynamic transfer learns to adapt the parameters to fit the model to the distribution formed by the union of source domains. The target domain is not required to be aligned with any specific domains $\mathcal{S}_i$ and there are no rigid domain boundaries. The model parameters $\boldsymbol{\theta}(\boldsymbol{x})$ can be similar for examples from different domains and different for examples from the same domain.

The key insight is that adapting model per domain is achieved statistically by adapting model per sample, as each domain can be considered as a distribution of image samples. The dynamic transfer learns to adapt model parameters over samples in the union of all source domains. This simplifies the alignment between source and target domains, as it is not necessary to pull all source domains and target domain together. As long as the target domain is aligned with any part of the union of source domains, the model can be easily adapted to the target samples.

Dynamic transfer has two advantages over static transfer. First, it turns multi-source domains into a single-source domain, voiding the need for domain labels. Second, it simplifies learning, since domain labels can be arbitrary. In practice, any "domain" can contain a mixture of unlabeled sub-domains and some of these can be shared by multiple "domains". Due to this, explicit assignment of data to domains can be difficult, and models learned over single domain can loose access to shared sub-domain data.

3.3. Dynamic Residual Transfer

The main difficulty of dynamic transfer is the model $f_{\boldsymbol{\theta}(\boldsymbol{x})}$ can be difficult to learn. Given the large number of

parameters of modern networks, it is impossible to simply predict all parameter values at inference time. The key is to restrict the model's dependence on input $\boldsymbol{x}$ to a *small number of parameters*. To guarantee this, we propose a model composed by a *static network* and *dynamic residual blocks*

$$f_{\boldsymbol{\theta}(\boldsymbol{x})} = f_0 + \Delta f_{\boldsymbol{\theta}(\boldsymbol{x})}, \tag{1}$$

where f_0 represents the static component and $\Delta f_{\boldsymbol{\theta}(\boldsymbol{x})}$ the dynamic residual that depends on the input sample $\boldsymbol{x}$. As usual, the residual is implemented by adding residual blocks to the various network layers. Since the static component f_0 is shared by all samples, static transfer is a special case of the proposed approach, where $\Delta f_{\boldsymbol{\theta}(\boldsymbol{x})} = 0$. This approach is denoted as *dynamic residual transfer* (DRT).

To implement DRT in convolution neural networks (CNNs), we represent a $k \times k$ convolution kernel as a $C_{out} \times C_{in}k^2$ weight matrix, where C_{in} and C_{out} are the number of input and output channels. We ignore bias terms in this discussion for the sake of brevity. DRT is implemented by applying Equation 1 to each convolution kernel in a CNN, i.e. defining the network convolutions as

$$\boldsymbol{W}(\boldsymbol{x}) = \boldsymbol{W}_0 + \Delta\boldsymbol{W}(\boldsymbol{x}), \tag{2}$$

where $\boldsymbol{W}_0$ is a static convolution kernel matrix, and $\Delta\boldsymbol{W}(\boldsymbol{x})$ a dynamic residual matrix. We next discuss several possibilities for the latter.

Channel Attention: in this case, the residual only rescales the output channels of $\boldsymbol{W}_0$. This is implemented as

$$\Delta\boldsymbol{W}(\boldsymbol{x}) = \boldsymbol{\Lambda}(\boldsymbol{x})\boldsymbol{W}_0, \tag{3}$$

where $\boldsymbol{\Lambda}(\boldsymbol{x})$ is a diagonal $C_{out} \times C_{out}$ matrix, whose entries are functions of $\boldsymbol{x}$. This can be seen as a dynamic feature-based attention mechanism.

Subspace Routing: the dynamic residual is a linear combination of K static matrices $\boldsymbol{\Phi}_i$

$$\Delta\boldsymbol{W}(\boldsymbol{x}) = \sum_{i=1}^{K} \pi_i(\boldsymbol{x})\boldsymbol{\Phi}_i, \tag{4}$$

whose weights depend on $\boldsymbol{x}$. The matrices $\boldsymbol{\Phi}_i$ can be seen as a basis for CNN weight space, although they are not necessarily linearly independent. And the dynamic coefficients $\pi_i(\boldsymbol{x})$ can be seen as the projections of the residual matrix in the corresponding weight subspaces. By choosing these projections in an input dependent manner, the network chooses different feature subspaces to route different $\boldsymbol{x}$.

To reduce the number of parameters and computation, the matrices can be further simplified into 1×1 convolution kernels and applied to the narrowest layer of the bottleneck architecture in ResNet [8]. In this case, only C_{in} rows of $\boldsymbol{\Phi}_i$ are non-zero.

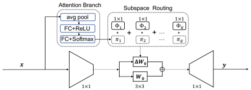


Figure 3: Subspace routing of DRT: dynamic coefficients are generated by a dynamic branch given the input $\boldsymbol{x}$. Each dynamic coefficient $\pi_i(\boldsymbol{x})$ is then multiplied by a matrix $\boldsymbol{\Phi}_i$, and the K matrices are aggregated as the residual kernel-$\boldsymbol{\Delta W}_0(\boldsymbol{x})$. For channel attention, softmax is replaced by sigmoid and the resulting coefficients in $\boldsymbol{\Lambda}(\boldsymbol{x})$ are multiplied to corresponding channels of $\boldsymbol{W}_0$.

Combination: the two mechanisms are combined into

$$\Delta\boldsymbol{W}(\boldsymbol{x}) = \boldsymbol{\Lambda}(\boldsymbol{x})\boldsymbol{W}_0 + \sum_{i=1}^{K} \pi_i(\boldsymbol{x})\boldsymbol{\Phi}_i. \tag{5}$$

Similar to squeeze-and-excitation block [11], the dynamic coefficients $\boldsymbol{\Lambda}(\boldsymbol{x})$ and $\{\pi_i(\boldsymbol{x})\}$ are implemented by a lightweight attention branch that includes average pooling and two fully connected layers (See Figure 3). A sigmoid is used to normalize $\boldsymbol{\Lambda}(\boldsymbol{x})$ and a softmax to normalize $\{\pi_i(\boldsymbol{x})\}$. As explained by [4, 30], the extra FLOPs caused by dynamic coefficient generation and residual aggregation of dynamic transfer is negligible (less than 0.1% in our implementation) compared to the static model.

3.4. Learning

As usual for domain adaptation problems, the DRT network is learned with a combination of two losses,

$$\mathcal{L} = \mathcal{L}_{ce} + \lambda\mathcal{L}_d, \tag{6}$$

where λ is a hyperparameter that controls the trade-off between the loss components. The first loss

$$\mathcal{L}_{ce} = \frac{1}{N_{\mathcal{S}}} \sum_{i=1}^{N_{\mathcal{S}}} \mathbf{y}_i^T \log f_{\boldsymbol{\theta}(\boldsymbol{x}_i^{\mathcal{S}})}(\boldsymbol{x}_i^{\mathcal{S}}), \tag{7}$$

is the cross entropy loss over the source data $\mathcal{D}^{\mathcal{S}}$. The second is a domain alignment loss that encourages the minimization of the distance between source and target domains

$$\mathcal{L}_d = \mathcal{H}\left(f_{\boldsymbol{\theta}(\mathcal{D}^{\mathcal{S}})}(\mathcal{D}^{\mathcal{S}}), f_{\boldsymbol{\theta}(\mathcal{D}^{\mathcal{T}})}(\mathcal{D}^{\mathcal{T}})\right), \tag{8}$$

where $\mathcal{D}^{\mathcal{T}}$ is the target data and $\mathcal{H}$ a measure of discrepancy between feature distributions of the source and target domains. $\mathcal{H}$ can be any distance function previously proposed for domain adaptation, e.g. the MMD [17] or adversarial learning [25]. Note that the two losses above operate on the

Models	inf,pnt,qdr rel,skt → clp	clp,pnt,qdr rel,skt → inf	clp,inf,qdr rel,skt → pnt	clp,inf,pnt rel,skt → qdr	clp,inf,pnt qdr,skt → rel	clp,inf,pnt qdr,rel → skt	Avg
static	54.3±0.64	22.1±0.70	45.7±0.63	7.6±0.49	58.4±0.65	43.5±0.57	38.5±0.61
Channel Attention	67.8±0.46	30.9±0.85	57.1±0.36	6.9±1.12	66.7±0.42	57.4±0.33	47.8±0.59
Subspace Routing	**69.7**±0.24	31.0±0.56	**59.5**±0.43	9.9±1.03	**68.4**±0.28	**59.4**±0.21	**49.7**±0.46
Combination	69.1±0.35	**31.6**±0.61	58.2±0.25	**11.9**±0.96	67.8±0.36	58.8±0.44	49.6±0.50

Table 1: Comparison of **different implementations for dynamic residual transfer**: Channel Attention (Equation 3), S
space Routing (Equation 4) and Combination (Equation 5).

entire source dataset $\mathcal{D}^{\mathcal{S}}$, i.e. there is no need for domain labels and not even a difference between the single domain and multiple domains adaptation problems. For the domain alignment losses commonly used in multi-source domain adaptation, Equation 8 also does require the evaluation of pairwise distances between all source domains and target domain, which is not necessary in dynamic transfer.

4. Experiments

In this section, adaptation performance of DRT is evaluated.

4.1. Datasets and Experimental Settings:

Following [20], we consider two datasets, Digit-five and DomainNet [20], which contain images from several domains but shared classes. Each domain is alternatively used as the target domain and the remaining ones as the source domain. All experiments are repeated with 5 times and mean and variance are reported.

Digit-five: Digit-five contains digit images from 5 domains: MNIST [12] (mt), Synthetic [6] (sy), MNIST-M [6] (mm), SVHN [19] (sv) and USPS [6] (up). These domains contribute 25,000 images for training and 9000 for validation, with the exception of USPS which uses 29752 and 1860, respectively. Since these datasets are relatively small, LeNet [12] is used as the backbone model. A dynamic residual is added on each convolutional layer. The model is trained from scratch with initial learning rate 0.002 and SGD optimizer. The learning rate is decayed by 0.1 every 100 epochs and decreased to $2e-5$ in 300 epochs.

DomainNet: DomainNet [20] is a dataset with 0.6 million images of 345 classes from 6 domains of different image styles: clipart (clp), infograph (inf), painting (pnt), quickdraw (qdr), real (rel) and sketch (skt). Results are obtained with ImageNet [5] pretrained ResNet-101 [8]. The dynamic residual is only added on the 3×3 kernel of each bottleneck block. The networks are trained with SGD for 15 epochs with initial learning rate of 0.001 and batch size as 64. The learning rate is decayed by 0.1 every 5 epochs.

4.2. Ablation Study

An ablation study was performed on DomainNet to evaluate the three key components of DRT: (a) the three im-
plementations of the dynamic transfer, (b) the numbe
basis used for subspace routing (Equation 4), and (c) dif
ent alignment losses $\mathcal{L}_d$. The default model uses subsp
routing with $K = 4$ and is trained with the MCD [23] lo

DRT Implementations: Table 1 shows that all implem
tations of DRT have significantly better adaptation per
mance than the static model. The average gains are of 9.
for channel attention, 10.8% for combined and 11.2%
subspace routing. The weaker performance of channel
tention suggests that it is not enough to re-scale the featu
of the static model. Routing the input $\boldsymbol{x}$ through diffe
subspaces appears to be more effective, although the
ferences are not staggering. While combining the two
proaches has no additional overall benefit, the combina
was beneficial for specific transfer problems. When 'i
graph' and 'quickdraw' were used as target domains,
combination model outperformed subspace routing. Si
these are the hardest transfer problems, this suggests that
enhanced dynamics of the combined implementation ca
beneficial as the domain gap increases. It is because the
hanced dynamics make the model more elastic. Theref
it is more likely to adapt models to target domain with la
gap. On the other hand, for the problems of smaller dom
gap, stronger dynamics can cause the model to overfit to
source domain, as is the case for the remaining target
mains. More experiments on datasets with more doma
will likely be needed to resolve this question. In any c
subspace routing and the combination model have sim
performance.

Number of Residual Basis. The impact of the numbe
basis K used in Equation 4 for subspace routing is abla
For different values of $K \in \{2, 4, 6, 8\}$, DRT achie
$\{48.8, 49.7, 49.5, 49.3\}$, all of which improve the ada
tion performance of static transfer (38.5%) by a large m
gin (more than 10%). Best performance is achieved
$K = 4$, although the results are not highly sensitive to
parameter.

Alignment Loss Function. Three different domain ali
ment losses with different forms of $\mathcal{H}$ (see Equation
were compared: ADDA [25], MCD [23] and M^3SDA [
They are representative of previously proposed losses fo
ducing single-source domain shift at the domain level
class level, and multi-source domain shift, respectively.

$\mathcal{L}_d$	inf,pnt,qdr rel,skt → clp	clp,pnt,qdr rel,skt → inf	clp,inf,qdr rel,skt → pnt	clp,inf,pnt rel,skt → qdr	clp,inf,pnt qdr,skt → rel	clp,inf,pnt qdr,rel → skt	Avg
Source Only	52.1±0.51	23.4±0.28	47.7±0.96	**13.0**±0.72	60.7±0.23	46.5±0.56	40.6±0.56
Source Only + **DRT**	**63.1**±0.62	**25.9**±0.84	**48.4**±1.02	6.4±0.98	**66.4**±0.54	**46.8**±0.44	**42.8**±0.74
ADDA [25]	47.5±0.76	11.4±0.67	36.7±0.53	**14.7**±0.50	49.1±0.82	33.5±0.49	32.2±0.63
ADDA+**DRT**	**63.6**±0.52	**27.6**±0.43	**52.3**±0.68	8.2±1.44	**67.9**±0.42	**49.6**±0.33	**44.9**±0.64
MCD [23]	54.3±0.64	22.1±0.70	45.7±0.63	7.6±0.49	58.4±0.65	43.5±0.57	38.5±0.61
MCD+**DRT**	**69.7**±0.24	**31.0**±0.56	**59.5**±0.43	**9.9**±1.03	**68.4**±0.28	**59.4**±0.21	**49.7**±0.46
M^{3}SDA-β [20]	58.6±0.53	26.0±0.89	52.3±0.55	6.3±0.58	62.7±0.51	49.5±0.76	42.6±0.64
M^{3}SDA-β+**DRT**	**67.4**±0.52	**31.3**±0.83	**56.5**±0.67	**13.6**±0.34	**66.9**±0.42	**56.8**±0.49	**48.8**±0.55

Table 2: **Static transfer vs. Dynamic transfer** evaluated on DomainNet with different domain alignment loss functions.

Table 2 shows that dynamic residual transfer (DRT) outperforms static transfer for all loss functions, by a large margin (12.7%, 11.2%, 6.2% respectively). Its improved performance is in part, due to the fact that DRT takes a much larger advantage of the domain alignment losses. It confirms our claim that DRT simplifies the domain alignment by unifying all source domains into a single domain. Thus the target samples are more likely to be aligned with the union of source domains and the same alignment loss will give more benefits to the dynamic model than the static one. However, the gains over the 'source only' case, where no alignment loss $\mathcal{L}_d$ is used in Equation 6, is only 2%. It means alignment loss is very critical for dynamic transfer. Without alignment loss, even though the model can adapt to the entire source domain very well, it can hardly adapt to target samples due to a large domain gap.

These conclusions also apply to the individual transfer problems, except when 'quickdraw' is the target domain. In this case, DRT is only effective with the M^{3}SDA-β [20] loss. It is because when 'quickdraw' is the target domain, the domain discrepancy is much larger and makes it harder for DRT to adapt model to this domain. Thus, M^{3}SDA-β [20] which proposed a more powerful alignment loss, can shift 'quickdraw' closer to source domains and works better with DRT. However, the strong alignment loss will cause 'over-alignment' for the domains e.g. 'clipart' that have much smaller gap. The 'over-alignment' reduces the adaptability of the dynamic model, which causes performance degradation compared to that given by simpler alignment losses e.g. MCD.

4.3. Comparisons to the state-of-the-art

DRT was compared to the results in the literature for Digit Five and DomainNet dataset. In these experiments, DRT is implemented with subspace routing (4 basis), using the MCD loss [23], and $\lambda = 50$ in Equation 6.

Evaluation on Digit Five Dataset: Table 3 shows a comparison to 6 baselines on Digit Five. DRT outperforms all other methods, beating the state of the art (CMSS) by more than 1%, despite a much simpler implementation. Comparing performance in individual adaptation problems, DRT has the best performance on four of the five problems considered. The only exception occurs when SVHN is the target domain, where DRT achieves the second best performance of all methods. Beyond this, the smallest gains occur when MNIST is the target domain. This was expected, since MNIST is easier to transfer to and somewhat saturated. In general, the gains of DRT increase with domain discrepancy, reaching 5.7% for the hardest transfer problem (MNIST-M as target domain).

Evaluation on DomainNet Dataset. For DomainNet [20], a ResNet-101 [8] was used as backbone and DRT was compared to 11 baselines. Among these, ADDA [25], DANN [7] and MCD [23] were developed for traditional unsupervised domain adaptation (UDA), where a single-source domain is assumed. The remaining are multi-source domain adaptation methods that require domain labels.

Table 4 shows that DRT improves on the state-of-the-art method- CMSS by more than 3% (49.7% vs 46.5%). When DRT is combined with a naive self-training method (DRT+ST), it achieves gains of 3.9% over LtC-MSDA [26], a methods that generates pseudo-labels for the target samples (during self-training, pseudo-labels for target samples with confidence greater than 0.8 are used to train DRT again with source samples). Compared to the adaptation methods that use no domain labels (single-source), DRT improves the best average adaptation results (DANN) by 6.7% (49.7% vs. 43%).

Regarding individual adaptation problems, DRT achieves the best performance for all target domains other than 'quickdraw'. This can be explained by the large domain gap between 'quickdraw' and the other domains, and the fact that the MCD loss does not fare well in this problem. Better results would likely be possible by using the M^{3}SDA loss, as was the case in Table 2.

4.4. Single-Source to Single-Target Adaptation

The performance of DRT on the traditional domain adaptation problem (single-source domain) was also evaluated on DomainNet [20]. In this case, for each target domain,

Models	mm,up,sv sy → mt	mt,up,sv sy → mm	mt,m sy →
Source Only	63.37±0.74	90.50±0.83	88.71
DANN [7]	71.30±0.56	97.60±0.75	92.33
ADDA [25]	71.57±0.52	97.89±0.84	92.83
MCD [23]	72.50±0.67	96.21±0.81	95.33
DCTN [29]	70.53±1.24	96.23±0.82	92.81
M^{3}SDA-β [20]	72.82±1.13	98.43±0.68	96.14
CMSS [32]	75.3±0.57	99.0±0.08	97.7
DRT	**81.03**±0.34	**99.31**±0.05	**98.40**

Table 3: Comparison between **dynamic residual transfer (DR**
source domains and target domain are shown at the top of each

Models	inf,pnt,qdr rel,skt → clp	clp,pnt,qdr rel,skt → inf	clp,inf,qdr rel,skt → pnt
Source Only	52.1±0.51	23.4±0.28	47.7±0.96
ADDA [25]	47.5±0.76	11.4±0.67	36.7±0.53
MCD [23]	54.3±0.64	22.1±0.70	45.7±0.63
DANN [7]	60.6±0.42	25.8±0.43	50.4±0.51
DCTN [29]	48.6±0.73	23.5±0.59	48.8±0.63
M^{3}SDA-β [20]	58.6±0.53	26.0±0.89	52.3±0.55
ML-MSDA [15]	61.4±0.79	26.2±0.41	51.9±0.20
Meta-MCD [13]	62.8±0.22	21.4±0.07	50.5±0.08
LtC-MSDA [26]	63.1±0.5	28.7±0.7	56.1±0.5
CMSS [32]	64.2±0.18	28.0±0.20	53.6±0.39
DRT	69.7±0.24	31.0±0.56	59.5±0.43
DRT+ST	**71.0**±0.21	**31.6**±0.44	**61.0**±0.32

Table 4: Comparison between **dynamic residual transfer**
('DRT+ST' represents the combination between dynamic resid

adaptation was performed from each of the other five domains (sources). The average and best performance among these adaptations is shown in Table 5, for each target domain. DRT again significantly outperforms the previous domain adaptation methods. For example, when 'clipart' is the target domain, its average adaptation performance is 10.5% better than that of the MCD method. On average, over all pairs of source and target domains, it outperforms MCD by more than 8%. These results show that, for problems with hundreds of classes, dynamic residual transfer can lead to very large adaptation gains even in the traditional domain adaptation setting. This confirms the claim that even these problems tend to have many sub-domains. When this is the case, the ability of dynamic residual transfer to adapt the model on a per-example basis can be a significant asset.

Finally, comparing the results of Tables 4 and 5 shows that DRT trained on multi-source domains performs 8.2% better (49.7% vs. 41.5%) than the average of the best single-source domain transfers. This improvement is about 2% better than that given by MCD (8.2% vs. 6.3%). This shows that considering a variety of source domains improves domain adaptation performance, especially when dynamic residual transfer is used. A main advantage of

m,sv → up	mt,mm,up sy → sv	mt,mm,up sv → sy	Avg
±0.89	63.54±0.93	82.44±0.65	77.71±0.81
±0.85	63.48±0.79	85.34±0.84	82.01±0.76
±0.74	75.48±0.48	86.45±0.62	84.84±0.64
±0.74	78.89±0.78	87.47±0.65	86.10±0.73
±0.27	77.61±0.41	86.77±0.78	84.79±0.72
±0.81	81.32±0.86	89.58±0.56	87.65±0.75
±0.13	**88.4**±0.54	93.7±0.21	90.8±0.31
±0.12	86.67±0.38	**93.89**±0.34	**91.86**±0.25

T) with the state-of-the-art models on Digit-five dataset. The column.

clp,inf,pnt rel,skt → qdr	clp,inf,pnt qdr,skt → rel	clp,inf,pnt qdr,rel → skt	Avg
13.0±0.72	60.7±0.23	46.5±0.56	40.6±0.56
14.7±0.50	49.1±0.82	33.5±0.49	32.2±0.63
7.6±0.49	58.4±0.65	43.5±0.57	38.5±0.61
7.7±0.68	62.0±0.66	51.7±0.19	43.0±0.46
7.2±0.46	53.5±0.56	47.3±0.47	38.2±0.57
6.3±0.58	62.7±0.51	49.5±0.76	42.6±0.64
19.1±0.31	57.0±1.04	50.3±0.67	44.3±0.24
15.5±0.22	64.6±0.16	50.4±0.12	44.2±0.07
16.3±0.5	66.1±0.6	53.8±0.6	47.4±0.6
16.0±0.12	63.4±0.21	53.8±0.35	46.5±0.24
9.9±1.03	68.4±0.28	59.4±0.21	49.7±0.46
12.3±0.38	**71.4**±0.23	**60.7**±0.31	**51.3**±0.32

(DRT) with the state-of-the-art models on DomainNet. ual transfer and self-training for domain adaptation)

DRT is that it can be applied to all settings, since it does not require domain labels. Its universal nature makes it irrelevant if the problem is single-source or multi-source. It suffices to collect training data and DRT will automatically figure out how to adapt the network to all settings. There is no need to even define "source domains."

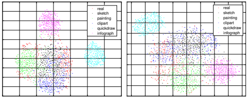
4.5. Visualization

To obtain further insight about dynamic residual transfer (DRT), we visualized the dynamic coefficients of Equation 4 with t-SNE [18]. For each sample, we created a vector $\Pi = \{\pi_i^l(\boldsymbol{x})\}, i \in \{1, 2, ...K\}, l \in \{1, 2, ...L\}$ by concatenating the dynamic coefficients from the L network layers. The vectors Π from different target domains are visualized in Figure 4. A more detailed visualization is given in Figure 5, by splitting Π into Π_{low} and Π_{high}, which include the coefficients from lower and higher network layers. For brevity, Figure 5, only visualizes the model trained with 'clipart' and 'real' as target domain and the other domains show similar trend.

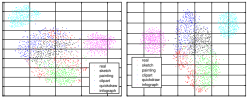
Figure 4 first shows that domain information is embedded into the dynamic coefficients $\{\pi_i^l(\boldsymbol{x})\}$. This can be observed by samples from same domains tend to group in identifiable clusters, which confirms our claim that adapting model across domains can be achieved through adapting model to samples. Secondly, the distance among clusters reflects domain shifts that explain how adaptation performance varies with target domain. For example, the fact that the dynamic coefficients of 'quickdraw' are always quite different from others, explains why adaptation performance is weaker when this is the target domain. Thus for 'quickdraw', either a more powerful alignment loss is needed to shift the samples close enough to the source domains to enable the dynamic model adapted to this domain or a more complex dynamic model e.g. combination of 'channel attention' and 'subspace routing' is required. Figure 5 further shows that the dynamic coefficients from lower layers (Figure 5(a)) form much more clear domain clusters than the coefficients of the higher layers (Figure 5(b)). This shows that the network features become more domain agnostic in the higher layers, confirming the effectiveness of DRT to reduce domain discrepancies.

Models	inf,pnt,qdr rel,skt → clp	clp,pnt,qdr rel,skt → inf	clp,inf,qdr rel,skt → pnt	clp,inf,pnt rel,skt → qdr	clp,inf,pnt qdr,skt → rel	clp,inf,pnt qdr,rel → skt	Avg
ADDA [25]	28.2/39.5	9.3/14.5	20.1/29.1	**8.4/14.9**	31.1/41.9	21.7/30.7	19.8/28.4
MCD [23]	31.4/42.6	13.1/19.6	24.9/42.6	2.2/3.8	35.7/50.5	23.9/33.8	21.9/32.2
DRT	**41.9/56.2**	**19.6/26.6**	**35.3/53.4**	8.0/12.2	**44.5/55.5**	**35.0/44.8**	**30.7/41.5**

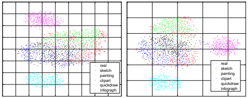
Table 5: **Single source domain adaptation performance on DomainNet**. Each column, shows the average/best classification accuracy for transfer from all source to the specified target domain.



(a) target: clipart (b) target: infograph

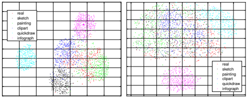


(c) target: painting (d) target: quickdraw

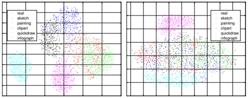


(e) target: real (f) target: sketch

Figure 4: The t-SNE **visualization of dynamic coefficients** $\Pi = \{\pi_i^l(\boldsymbol{x})\}$ when DRT is trained with target domain- 'clipart', 'infograph', 'painting', 'quickdraw', 'real' and 'sketch'. (Best view in color)



(a) target: clipart (Π_{low}) (b) target: clipart (Π_{high})



(c) target: real (Π_{low}) (d) target: real (Π_{high})

Figure 5: The t-SNE visualization of **first half** (Π_{low}) and **second half** (Π_{high}) dynamic coefficients when DRT is trained with target domain- 'clpart' and 'real'. (Best view in color)

5. Conclusion

In this paper, we introduce dynamic transfer for multi-source domain adaptation, in which the model parameters are not static but adaptive to input samples. Dynamic transfer mitigates conflicts across multiple domains and unifies multiple source domains into a single source domain, which simplifies the alignment between source and target domains. Experimental results show that dynamic transfer achieves a better adaptation performance compared to the state-of-the-art method for multi-source domain adaptation. We hope this paper can give a new understanding about multi-source domain adaptation.