Resource Efficient Domain Adaptation

Junguang Jiang, Ximei Wang, Mingsheng Long (✉), and Jianmin Wang
School of Software, BNRist, Research Center for Big Data, Tsinghua University, China
{jjg20,wxm17}@mails.tsinghua.edu.cn,{mingsheng,jimwang}@tsinghua.edu.cn

ABSTRACT

Domain Adaptation (DA) aims at transferring knowledge from a labeled source domain to an unlabeled target domain. While remarkable advances have been witnessed recently, the power of DA methods still heavily depends on the network depth, especially when the domain discrepancy is large, posing an unprecedented challenge to DA in low-resource scenarios where fast and adaptive inference is required. How to bridge transferability and resource-efficient inference in DA becomes an important problem. In this paper, we propose Resource Efficient Domain Adaptation (REDA), a general framework that can adaptively adjust computation resources across "easier" and "harder" inputs. Based on existing multi-exit architectures, REDA has two novel designs: 1) Transferable distillation to distill the transferability of top classifier into the early exits; 2) Consistency weighting to control the distillation degree via prediction consistency. As a general method, REDA can be easily applied with a variety of DA methods. Empirical results and analyses justify that REDA can substantially improve the accuracy and accelerate the inference under domain shift and low resource.

CCS CONCEPTS

• **Computing methodologies** → *Transfer learning*; *Neural networks*;

KEYWORDS

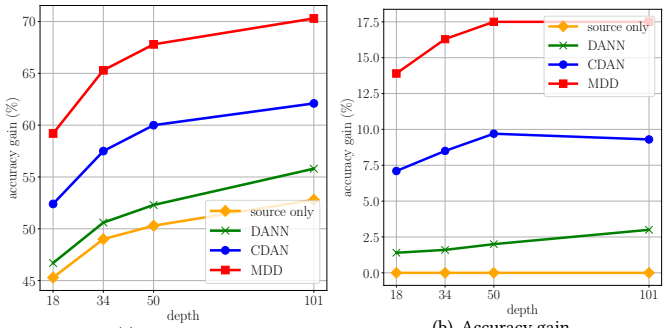
Domain adaptation, transfer learning, resource efficient learning

ACM Reference Format:
Junguang Jiang, Ximei Wang, Mingsheng Long (✉), and Jianmin Wang. 2020. Resource Efficient Domain Adaptation. In *Proceedings of the 28th ACM International Conference on Multimedia (MM '20), October 12–16, 2020, Seattle, WA, USA.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3394171.3413701

1 INTRODUCTION

Recent years have witnessed great progress in Deep Neural Network (DNN) in many machine learning problems and applications. Although most state-of-the-art DNN models have achieved increasingly higher accuracy, they are often *data-hungry* and *computation-consuming*. In the real world, however, not all problems have large-scale datasets and not all devices can endure the heavy computation burden of DNN. For problems lacking labeled data (e.g. real-world), it is reasonable to transfer knowledge from a related domain (e.g. simulation) with a large-scale labeled dataset, though performance degradations may happen due to the *dataset shift* [42, 65].

To tackle the dataset shift problem between the source and target domains, a variety of *domain adaptation* (DA) [10, 36, 37, 49, 69] methods have been proposed, aiming to push the accuracy of DA towards that of supervised learning. However, most DA methods focus on improving the accuracy while rare attention has been given to speeding up the inference on the unlabeled target domain, thereby a high computation cost is with most existing DA models. Further, in many practical applications, reducing the inference time of those models would be beneficial or even compulsory. For instance, labeled data is expensive in robotics, therefore, generating training data through simulation and then applying the models in real scenarios becomes a common technique [3, 30]. In such a simulation to real setup, transfer models with deep neural networks will face the problem of high computation cost, making the industry robots hard to respond quickly to fast manufacturing. Meanwhile, the power of DA methods still heavily depends on the network depth, especially when the domain discrepancy is large, posing an unprecedented challenge to DA in low-resource scenarios where both fast and adaptive inference is required.



(a) Accuracy (b) Accuracy gain

Figure 1: Accuracy and accuracy gain by domain adaptation as a function of the *depth of ResNets* on *Office-Home*.

Motivated by previous work on supervised learning of resource-efficient models [27], we aim to speed up the inference of DA models on the target domain by reducing the computation spent on "easier" inputs. Unfortunately, these models fail to generalize to the target domain due to the dataset shift problem. Even after introducing some state-of-the-art DA methods, they still cannot reach a satisfying balance between speed and accuracy in our experiments. A natural question arises: *why is it so difficult to be simultaneously fast and accurate in the DA setup?* Before answering this question,

MM '20, October 12–16, 2020, Seattle, WA, USA

ACM ISBN 978-1-4503-7988-5/20/10…$15.00
https://doi.org/10.1145/3394171.3413701

Table 1: Comparison of the proposed REDA framework with several existing classification models

Method	Adaptive Inference	Domain Adaptive	Adaptive Distillation
ResNet [22]	✘	✘	✘
DANN [10]	✘	✔	✘
CDAN [37]	✘	✔	✘
MSDNet [27]	✔	✘	✘
Skipnet [63]	✔	✘	✘
REDA (Proposed)	✔	✔	✔

we need to analyze some observations first. *Observation 1*: When the domain gap is within a certain range, a deeper network can improve both accuracy and *accuracy gain* yielded by the adaptation module (Figure 1(a), 1(b), see Section 4.3 for details). *Observation 2*: When the domain discrepancy is decreased, a shallower network is enough to reach similar accuracy on the *same* target domain. Observation 1 tells us that an effective way to enhance transferability is to increase the network depth during training. Observation 2 indicates that small networks could exhibit comparable test error when the domain gap decreases and using deep networks is wasting computational resources. Therefore, to achieve a balance between speed and accuracy, the architecture during training and inference should not be identical. In other words, we need a deeper neural network during training to attain high accuracy and a shallower one during testing to speed up the inference.

Based on the above observations and analyses, we propose a novel framework called REDA, enabling Resource Efficient Domain Adaptation. First, we tailor the popular multi-exit architectures [27] in IID setup into the domain adaptation scenario. Further, since deeper networks are more transferable as in Observation 1, REDA embeds adaptation modules to deeper networks during training. Due to the weak transferability of shallower networks, we propose a novel Transferability Distillation (TransDistill) technique, through which deeper networks can help shallower ones transfer better by distilling the transferability of top classifier into the early exits. Meanwhile, we also propose a novel design of consistency weighting to control the distillation degree via prediction consistency. Our contributions are summarized as follows:

- To our knowledge, no previous work has discussed how to save computation resources of DA models, and the proposed REDA is the first general framework that allows dynamic resource adaptation across domains.
- We explore the impact of depth on domain adaptation through novel experiments. Based on these empirical insights, we propose TransDistill, which is the key module for fast inference of DA methods.
- We conduct extensive experiments on three standard DA datasets and demonstrate that REDA can speed up inference on all datasets under guaranteed accuracies.

2 RELATED WORK

Domain Adaptation. Domain adaptation is proposed to alleviate the *dataset shift* problem between the source and target domains [49]. It finds wide applications in computer vision [17, 24, 52] and natural language processing [6, 13]. Early DA methods attempt to bridge the source and target domains through learning domain-invariant representations [15, 43] or instance importances [14, 28]. Since DNNs achieved success, adaptation modules have been embedded into deep neural networks. **1)** *Moment matching* is used to minimize the distribution discrepancy for aligning feature distributions across domains [35, 36, 38, 39, 58, 60]. **2)** *Adversarial training*, which was first proposed in DANN [10], becomes popular recently [11, 37, 56, 57]. It takes the spirit from Generative Adversarial Networks (GANs) [16] and introduces a domain discriminator which encourages the feature extractor to learn domain-invariant features. CDAN [37] generalizes DANN by aligning features in a class-conditional adversarial manner. MDD [69] establishes a new domain adaptation margin theory. ADDA [57], MADA [45], and MCD [53] extend DANN to architectures with multiple feature extractors or multiple discriminators to reduce domain discrepancy. CyCADA [25] performs adaptation at both pixel and feature levels. Many novel methods have emerged recently [46, 54, 62, 70]. For instance, Ciga et al. [5] introduced domain discriminators at multiple layers of deep networks. However, we found that simply applying DA loss to multiple layers in multi-exit architectures [27] is not satisfactory. Thus we propose REDA to improve the transferability of DNN for low-resource scenarios.

Adaptive Inference. The key idea of adaptive inference is to save the computation time of "easy" samples during inference so that the overall computation is reduced while the accuracy remains unchanged. Common methods fall into three categories. **1)** *Skip networks* selectively skip some layers during testing, thereby reducing the inference time. Many selection strategies have been proposed, including skipping convolutional blocks [8, 63, 64] or channel-wise pruning [34] through reinforcement learning, and adaptive network topology by introducing gate for each layer [29, 59]. Meanwhile, **2)** *Dynamic recursive network* has a small computation cost itself. To improve the accuracy of "hard" examples, the same block might be executed multiple times during inference [20, 33, 41]. Further, **3)** *Multi-exit architectures* are a family of architectures with multiple classifiers at different depths of the network. Early exits run faster, while later ones usually achieve higher accuracy. BranchyNet [55] first proposes multi-exit architecture and adds early exits to LeNet, AlexNet, and ResNet. MSDNet [27] solves the issue that early exits lack coarse-level features and interfere with later classifiers, by introducing multi-scale feature maps and dense connections. Graph HyperNetwork [67] has found better multi-exit networks through neural architecture search (NAS). Most of the multi-exit architectures including MSDNet and BranchyNet minimize the sum of cross-entropy loss of all exits. Among these methods, we focus

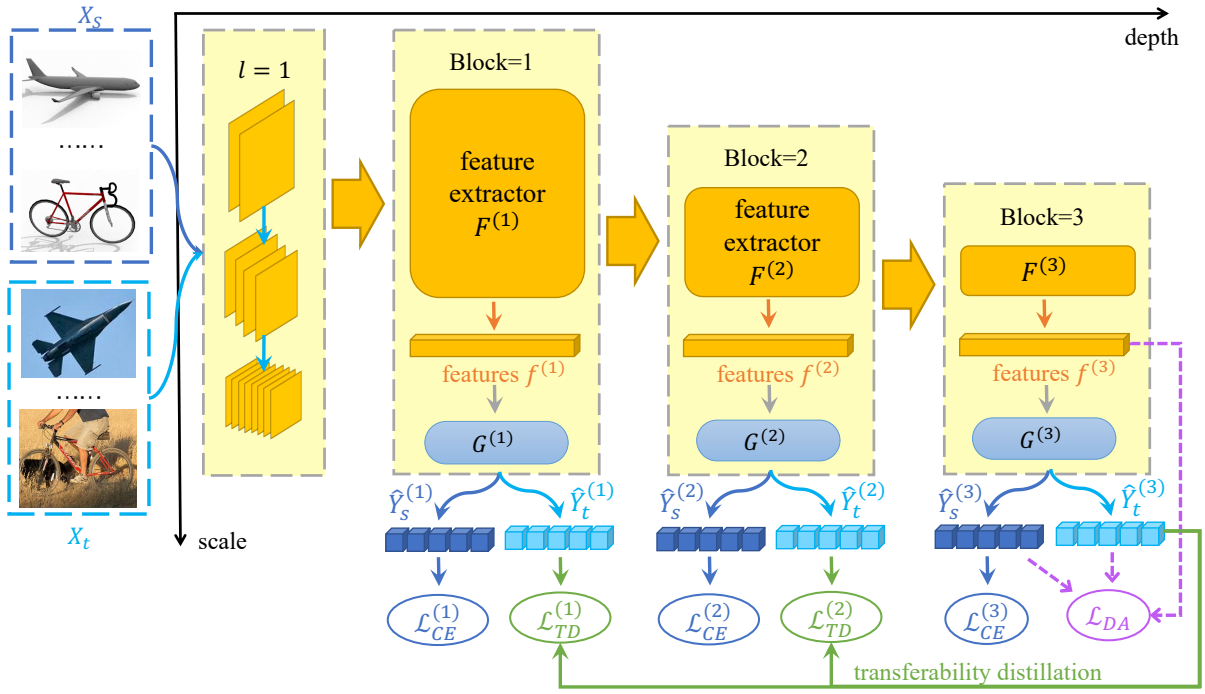


Figure 2: The Resource Efficient Domain Adaptation (REDA) framework includes $M = 3$ cascaded feature extractors (orange) and M label predictors (blue). Training loss combines the label prediction loss (dark blue) of all classifiers, the DA loss (purple) of the top classifier, and the TransDistill loss (green) between the bottom $M-1$ exits and the top classifier. (*Best viewed in color*)

our attention on multi-exit architectures due to their straightforward implementation and superior accuracy. We will discuss them in more details in Section 3.1.

Knowledge Distillation. *Knowledge distillation* is a method to transfer knowledge from complex models to simple models. It has found increasingly wide adoptions in many applications, including network compression [26, 48], defending against adversarial attack [44], heterogeneous architecture transfer [12], generating pseudo-labels in omni-supervised learning (a.k.a. *data distillation*) [50], avoiding catastrophic forgetting in life-long learning [32] and so on. Knowledge distillation is also used to transfer knowledge between different domains [1] or different tasks [31] (through distilling the feature maps) to increase the generalizability of the student models.

Table 1 compares REDA with several well-known or related models. As far as we know, REDA is the first general framework to speed up the inference dynamically in domain adaptation. The transferability distillation (TransDistill) in REDA is different from the *knowledge distillation* [47, 68] used in supervised learning. First, TransDistill enhances the transferability of multi-exits on fully unlabeled target domains, while the knowledge distillation attempts to improve the accuracy of shallower neural networks with labeled data. Further, TransDistill is the first one to adaptively adjust the distillation according to the consistency of the predictions on the unlabeled target domain, which proves effective in our experiments.

3 REDA: RESOURCE EFFICIENT DOMAIN ADAPTATION

In this paper, we present Resource Efficient Domain Adaptation (REDA), a general framework to dynamically reduce the inference time for different DA methods. We mainly focus our study on the standard DA setup, Unsupervised Domain Adaptation (UDA), with a labeled source domain $\mathcal{S} = \{(\mathbf{x}_s^i, y_s^i)\}_{i=1}^{n_s}$ of n_s labeled examples and an unlabeled target domain $\mathcal{T} = \{\mathbf{x}_t^j\}_{j=1}^{n_t}$ of n_t unlabeled examples.

3.1 Multi-Exit Architectures in IID Setup

To accelerate inference in the unlabeled target domain, we first explore how supervised learning dynamically reduces computation resources. Among various adaptive inference methods, we adopt multi-exit architectures as they are easier to implement and show higher performance in practice. Given an input $\mathbf{x}$, multi-exit architectures with M exits will output M class-probability distributions $\{\widehat{\mathbf{y}}^{(1)}, \widehat{\mathbf{y}}^{(2)}, ..., \widehat{\mathbf{y}}^{(M)}\}$. We will use MSDNet [27] as an example. Figure 2 briefly illustrates the key insights of MSDNet (depicted in yellow). To improve the accuracy of early classifiers in the lower layers, MSDNet keeps multi-scale feature maps throughout the network. The first layer produces the original S representations by down-sampling (along the vertical black line). Multiple scales ensure that even the early exits have access to coarse-level features, which are essential for the accurate inference of early exits. The subsequent

layers are divided into M blocks and each block corresponds to an equivalent feature extractor and a classifier. The dense connections between layers prevent early classifiers from interfering with later ones. However, those connections are simplified since our focus is on how to improve the transferability of multi-exit architectures.

In the independent and identically distributed (IID) setup, MSDNet substantially outperforms ResNets and DenseNets when the computational budget is limited. However, when the training and test datasets come from different domains, MSDNet often lags behind ResNets (Figure 4(c)). The weak transferability of MSDNet is closely related to its built-in design. The coarse-level features of early exits have large cross-domain distribution discrepancies since they are from shallow networks. Because MSDNet performs unsatisfactorily when the IID assumption fails, it is essential to introduce existing DA methods into the multi-exit architectures.

3.2 Multi-Exit Architectures in DA Setup

As multi-exit architectures have multiple exits $\mathbf{G} = \{G^{(1)}, ..., G^{(M)}\}$ at different depths, a natural idea is to perform domain adaptation on the features of each exit. Take DANN [10] as an instance. DANN introduces a minimax game into domain adaptation, where a domain discriminator attempts to distinguish the source from the target, while a feature extractor tries to fool the domain discriminator. Thus, we need to introduce M domain discriminators $\mathbf{D} = \{D^{(1)}, ..., D^{(M)}\}$ in multi-exit architectures. Feature extractors $\mathbf{F}$ are forced to learn domain-invariant representations at all depth to fool the domain discriminators. The extension of the two-player game into multiple domain discriminators can be formalized as

$$\min_{\mathbf{F},\mathbf{G}} \max_{\mathbf{D}} \sum_{m=1}^{M} \mathbb{E}_{(\mathbf{x}_s,\mathbf{y}_s)\in\mathcal{S}} \mathcal{L}_{\mathrm{CE}}(\widehat{\mathbf{y}}_s^{(m)}, \mathbf{y}_s) - \lambda \mathbb{E}_{\mathbf{x}\in\mathcal{S}\cup\mathcal{T}} \mathcal{L}_{\mathrm{CE}}(D^{(m)}(\widehat{\mathbf{f}}^{(m)}), \mathbf{d}), \tag{1}$$

where $\mathcal{L}_{\mathrm{CE}}$ is the cross-entropy loss, $\widehat{\mathbf{f}}^{(m)} = F^{(m)}(\mathbf{x})$ is the feature of the m-th layer learned to confuse the domain discriminator $D^{(m)}$, $\mathbf{d}$ is the domain label, and λ is the trade-off hyper-parameter between source cross-entropy loss and the DA loss.

Many domain adaptation methods, including DANN [11] and CDAN [37], only align high-layer features. Due to the vanishing gradient problem of back-propagation, lower-layer features might remain misaligned. It seems that Equation (1) can fill this gap by aligning features of different depths at the same time. Unfortunately, our experiments show that introducing domain discriminators at the low levels does not help the *top* classifier achieve better transferability and sometimes even deteriorate its performance. Why does this happen? Yosinski et al. [65] revealed that features of different depths have different transferability. Besides, adversarial DA methods tend to sacrifice discriminability for transferability [4] and early exits often fail to balance between those two aspects. Thus, *directly* aligning the low-layer features may even worsen transferability. Our experiments (Section 4.3) also indicate that the depth of neural networks has a huge impact on the transferability of features. It is obvious that the early exits are faster while the later ones are more transferable. Then a natural question is: *how to have them both?*

3.3 Transferability Distillation

Towards overcoming the challenges when applying domain adaptation in multi-exit architectures, we propose Transferability Distillation (**TransDistill**), through which the top classifier can help the lower ones in the early exists to improve their transferability. In this paper, we focus on the predictions of all classifiers following [19, 23]. First, we denote the input on the target domain as $\mathbf{x}$ (omit subscript t for clarity) and the corresponding logit output of the m-th exit as $\mathbf{z}^{(m)}$. Meanwhile, the temperature-scaled probability $\widehat{y}_k^{(m)}$ that the instance belongs to the k-th class is given by

$$\widehat{y}_k^{(m)} = \frac{\exp(z_k^{(m)}/T)}{\sum_{k'=1}^{C} \exp(z_{k'}^{(m)}/T)}, \tag{2}$$

where T is the temperature and C is the number of classes. Vanilla softmax output can be regarded as a special form of $\widehat{\mathbf{y}}^{(m)}$ when $T = 1$. Note that, $\widehat{\mathbf{y}}_k^{(m)}$ indicates the confidence that $\mathbf{x}$ belongs to the k-th class at the m-th exit. In this way, $\widehat{\mathbf{y}}^{(m)}$ measures the knowledge transferred from the source domain to the target domain by the m-th classifier, although $\widehat{\mathbf{y}}^{(m)}$ may not agree with the ground truth label $\mathbf{y}$ (we will come back to this problem in Section 3.4).

According to our experiments (see Figure 1(b)), deeper networks imply better transferability. Hence we use the top classifier (exit M) as the teacher and the remaining shallower classifiers (exit $m, m \in [1, M-1]$) as students. To encourage students to learn from the teacher, we try to minimize the *Jensen-Shannon Divergence* (JSD) [7, 9] between their temperature-scaled probability distributions,

$$\mathcal{L}_{\mathrm{distill}}^{(m)}(\mathbf{x}) = \mathrm{JSD}(\widehat{\mathbf{y}}^{(m)}, \widehat{\mathbf{y}}^{(M)}). \tag{3}$$

Note that the teacher's prediction $\widehat{\mathbf{y}}^{(M)}$ is treated as constant when computing the gradient for $\mathcal{L}_{\mathrm{distill}}^{(m)}(\mathbf{x})$, since what we want is that only the students learn from the teacher but not vice versa. JSD measures the discrepancy between two probability distributions $\mathbf{p}$ and $\mathbf{q}$ and its definition can be formalized as

$$\mathrm{JSD}(\mathbf{p}, \mathbf{q}) = \frac{1}{2}\mathrm{KL}(\mathbf{p}, \mathbf{m}) + \frac{1}{2}\mathrm{KL}(\mathbf{q}, \mathbf{m}) \tag{4}$$

where $\mathbf{m} = \frac{1}{2}(\mathbf{p} + \mathbf{q})$ and $\mathrm{KL}(\mathbf{p}, \mathbf{q}) = \sum_{k=1}^{C} \mathbf{p}_k \log\left(\frac{\mathbf{p}_k}{\mathbf{q}_k}\right)$. Compared with Kullback-Leibler divergence [21], JSD is *symmetric* and *finite*. By minimizing the JSD between the probability predictions of the top classifier and the remaining shallower classifiers, we can distill the transferability of the top classifier into the early exits. Since this kind of distillation is tailored into the domain adaptation setup and thus successfully distills the transferability from the top layer to shallower layers, we call it Transferable Distillation (**TransDistill**).

3.4 Consistency Weighting

As mentioned before, there is still a problem with distillation in the unsupervised domain adaptation scenario. The prediction of the deeper network may not be consistent with the ground-truth label, thus the guidance from the top exit might be wrong occasionally, which is harmful to the early exits. To enable safer transfer, we further introduce a novel consistency weighting mechanism to *adaptively* adjust the transferability distillation loss.

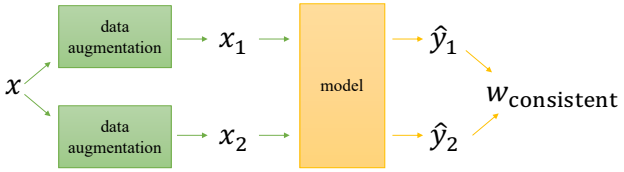


Figure 3: The pipeline to calculate the consistency weight.

Given the input data $\mathbf{x}$, we can perform data augmentation twice to obtain $\mathbf{x}_1$ and $\mathbf{x}_2$ (see Figure 3). Since the data augmentation is performed randomly, $\mathbf{x}_1$ and $\mathbf{x}_2$ are usually different. Suppose the predictions of the teacher model are $\widehat{\mathbf{y}}_1$ and $\widehat{\mathbf{y}}_2$. In an unsupervised scenario, we still cannot judge whether $\widehat{\mathbf{y}}_1$ and $\widehat{\mathbf{y}}_2$ are correct or not. Yet one thing is for sure. When $\widehat{\mathbf{y}}_1$ and $\widehat{\mathbf{y}}_2$ are correct, they must be consistent with each other. Otherwise, when their predicted labels are inconsistent, at least one of the predictions is wrong, and it is not wise to distill the knowledge from top exit to the early exits at this time. Hence, we define the consistency weighting scheme as

$$w_{\text{consistent}}(\mathbf{x}) = \mathbb{1}(h_{\widehat{\mathbf{y}}_1} = h_{\widehat{\mathbf{y}}_2}) + \mathbb{1}(h_{\widehat{\mathbf{y}}_1} \neq h_{\widehat{\mathbf{y}}_2}) \cdot \alpha, \tag{5}$$

where h is the labeling function, $\mathbb{1}$ is the indicator function and α is a trade-off hyperparameter. We should place less importance on an instance whose predictions of different data augmentations are inconsistent when distilling the transferability from the top exit. We set the value of α much smaller than 1 to achieve this goal.

Finally, integrating the distillation loss in Equation (3) and the consistency weighting in Equation (5), we achieve the **TransDistill (TD)** loss as

$$\mathcal{L}_{\text{TD}}^{(m)}(\mathbf{x}) = w_{\text{consistent}}^{(M)}(\mathbf{x}) \cdot \text{JSD}(\widehat{\mathbf{y}}^{(m)}, \widehat{\mathbf{y}}^{(M)}). \tag{6}$$

In this way, the proposed consistency weighting mechanism can control the distillation degree via prediction consistency of the top classifier. In summary, this new TransDistill loss has two important insights. First, since the features extracted by the shallow neural networks are hard to be aligned directly, we adopt transferable distillation to distill the transferability of the top classifier into the early exits. Secondly, since the transferability varies across different instances, we use the consistency of predictions corresponding to each instance to reweigh the distillation loss as in Equation (3).

3.5 REDA with Transferability Distillation

Resource Efficient Domain Adaptation (REDA) can be implemented by integrating the TransDistill loss in Equation (6) with existing DA methods. The two-player minimax game in Section 3.2 becomes

$$\begin{aligned}\min_{\mathbf{F},\mathbf{G}} \max_{\mathbf{D}} & \sum_{m=1}^{M} \mathbb{E}_{(\mathbf{x}_s,\mathbf{y}_s)\in\mathcal{S}} \mathcal{L}_{\text{CE}}(\widehat{\mathbf{y}}_s^{(m)}, \mathbf{y}_s) \\ & - \lambda\, \mathbb{E}_{\mathbf{x}\in\mathcal{S}\cup\mathcal{T}} \mathcal{L}_{\text{CE}}(D^{(m)}(\widehat{\mathbf{f}}^{(m)}), \mathbf{d}) \\ & + \mu \sum_{m=1}^{M-1} \mathbb{E}_{\mathbf{x}_t\in\mathcal{T}} \mathcal{L}_{\text{TD}}^{(m)}(\mathbf{x}_t),\end{aligned} \tag{7}$$

where μ is a hyper-parameter for the TransDistill loss. The entire framework is illustrated in Figure 2.

The TransDistill loss is general to readily work with a variety of existing DA methods, turning them into REDA approaches. Take Margin Disparity Discrepancy (MDD) [69], which is the current state-of-art DA method, for another instance. MDD uses an adversarial classifier $\mathbf{G}'$ to maximize the margin disparity discrepancy:

$$\begin{aligned}\max_{\mathbf{G}'} \mathcal{D}_\gamma(\mathcal{S},\mathcal{T}) = \gamma \mathbb{E}_{\mathbf{x}_s\in\mathcal{S}} \log\left(\frac{\exp(\widehat{\mathbf{y}}_s^{\text{adv}}[h_{\widehat{\mathbf{y}}_s^{(M)}}])}{\sum_j \exp(\widehat{\mathbf{y}}_s^{\text{adv}}[j])}\right) \\ + \mathbb{E}_{\mathbf{x}_t\in\mathcal{T}} \log\left(1 - \frac{\exp(\widehat{\mathbf{y}}_t^{\text{adv}}[h_{\widehat{\mathbf{y}}_t^{(M)}}])}{\sum_j \exp(\widehat{\mathbf{y}}_t^{\text{adv}}[j])}\right),\end{aligned} \tag{8}$$

where γ is the hyperparameter corresponding to the margin, $\widehat{\mathbf{y}}^{\text{adv}}$ is the output of the adversarial classifier and h is the labeling function. The objective of classifier $\mathbf{G}$ and feature extractor $\mathbf{F}$ is to minimize the cross-entropy loss on the source domain, the margin disparity discrepancy of the top exit and the transferability distillation loss:

$$\begin{aligned}\min_{\mathbf{F},\mathbf{G}} \max_{\mathbf{G}'} & \sum_{m=1}^{M} \mathbb{E}_{(\mathbf{x}_s,\mathbf{y}_s)\in\mathcal{S}} \mathcal{L}_{\text{CE}}(\widehat{\mathbf{y}}_s^{(m)}, \mathbf{y}_s) \\ & + \lambda\, \mathcal{D}_\gamma(\mathcal{S},\mathcal{T}) \\ & + \mu \sum_{m=1}^{M-1} \mathbb{E}_{\mathbf{x}_t\in\mathcal{T}} \mathcal{L}_{\text{TD}}^{(m)}(\mathbf{x}_t),\end{aligned} \tag{9}$$

where μ is a hyper-parameter for the TransDistill loss. The entire framework is shown in Figure 2, which is the same with Equation (7). As can be confirmed, the REDA framework is general to incorporate a variety of DA methods without modifying the architecture.

In summary, our contributions are highlighted as follows. We propose a unified framework called Resource Efficient Domain Adaptation (REDA), which is not only fast and adaptive at inference, but also friendly with various existing DA methods. We design a novel transferability distillation loss function that can overcome the domain shift problem in the lower layers of the neural network for early exits and speed up the inference on the unlabeled domain.

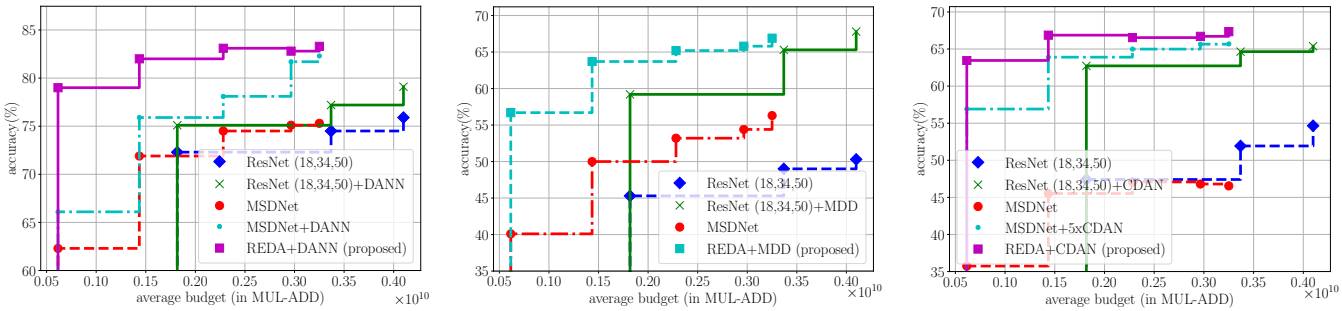
4 EXPERIMENTS

We evaluate the effectiveness of REDA with several state-of-the-art domain adaptation methods on three real-world domain adaptation datasets: *Office-31*, *Office-Home*, and *VisDA-2017*. Code is made available at https://github.com/thuml/Transfer-Learning-Library.

4.1 Setup

Datasets. *Office-31* contains 4,625 images and 31 categories from 3 domains: *Amazon* (**A**), *Webcam* (**W**), *DSLR* (**D**). *Office-Home* includes 15,500 images and 65 classes collected from 4 domains: *Artistic images* (**Ar**), *Clip Art* (**Cl**), *Product images* (**Pr**), and *Real-World images* (**Rw**). *VisDA-2017* is a simulation-to-real dataset with 12 categories and over 280K images from two domains: *Synthetic* and *Real*. On each dataset, we randomly split the dataset from both domains and use the **first** 80% for training and the **remaining** 20% as the test set. For a fair comparison, we keep the split of each dataset the same for different methods.

Training Details. We implement ResNet-based methods and MSDNet-based methods both pretrained from ImageNet [51] in



(a) *Office-31* (b) *Office-Home* (c) *VisDA-2017*

Figure 4: Accuracy (top-1) of *anytime prediction* as a function of compute budget for unsupervised DA (UDA) on three datasets.

PyTorch [61]. Unless extra specifications, the MSDNet has $M = 5$ exits in our experiments and one for every seven layers. We set the learning rate of the new layers and the classifier layers 10 times larger than that of the pre-trained layers. We adopt mini-batch SGD with a batch-size of 28, and Nesterov momentum with a momentum weight of 0.9 and a weight decay of 0.001. Deep Embedded Validation (**DEV**) [66] is conducted to select the hyper-parameters. We find that for REDA, $\lambda = 1$, $\mu = 1$, $T = 2.5$, $\alpha = 0.5$ works well on all datasets. All models are trained for $10k$, $15k$, and $30k$ iterations on *Office-31*, *Office-Home*, and *VisDA-2017* respectively. When calculating the consistency weight, we use the common data augmentation pipeline: first, resize to 256, then crop to 224 and flip randomly. We run each experiment three times and report the average accuracy on the test set of the target domain.

Tasks. We evaluate REDA mainly on two tasks: *anytime prediction* and *budgeted batch classification*. 1) In the **anytime prediction** setting [18], we don't know the computational budget for each instance in advance and the models are forced to output their most up-to-date prediction at an arbitrary time. On each dataset, we average the accuracy of all the tasks and plot the anytime prediction graph. 2) In the **budgeted batch classification** setting, the model is asked to classify a group of examples given a total amount of computational budget B. Therefore, the model can allocate the computation to each sample in advance. We use dynamic evaluation following Huang et al. [27]: "easy" instances will exit early while "hard" examples will propagate through the entire networks. The difficulty of each instance is measured by *confidence*, i.e. the max value of the softmax output. We find the confidence threshold for each exit on the validation set. During the test phase, the inference will stop early when the confidence exceeds the threshold limit of the current exit.

Baselines. We compare REDA with several baselines, including **MSDNet** [27], **MSDNet+M×DA** (MSDNet with DA loss on all M exits; See Section 3.2 for more details), **ResNets** [22] with depth varying from 18 to 50 layers, and **ResNets+DA** (Standard ResNets with different DA methods, including mainstream ones: **DANN** [11], **CDAN** [37], and the new state-of-the-art: **MDD** [69].

4.2 Empirical Results

Office-31. We evaluate the performance on Office-31 with **DANN** as the base DA method. There are 6 tasks in Office-31 and we report the average accuracy. As shown in Figure 4(a), **REDA+DANN** yields a larger boost than any other method, especially when the budget is $\leq 0.2\times10^{10}$ FLOPS. It's astonishing that while the *top* classifier of **MSDNet** has similar accuracy as **ResNet50**, the *bottom* exit of **REDA+DANN** achieves higher accuracy than **ResNet50+DANN**. There are two possible reasons. First, Chen et al. [4] reveal that DANN sacrifices discriminability for transferability. Yet we have implicitly improved the discriminability of features through minimizing the class disagreement between different exits. More importantly, most samples of the *Office-31* dataset are easy to classify, yet hard to transfer between domains. **REDA+DANN** improves the transferability of early exits through TransDistill. Thus, most samples can be correctly classified by the early exits of **REDA+DANN**.

Office-Home. We evaluate our method on Office-Home taking **MDD** as the base DA method. There are 12 tasks in Office-Home and we report the average accuracy. Note that the accuracy of **MSDNet+5×MDD** always drops to zero, probably because the low-level features are not suitable for adversarial training in MDD. Thus we didn't compare it with other methods. As shown in Figure 4(b), **REDA+MDD** substantially outperforms any other methods in the anytime prediction setting when the budget is $\leq 0.4\times10^{10}$ FLOPS. In particular, **REDA+MDD** achieves 15% $\sim$ 17% higher accuracy than **MSDNet** when the budget ranges from 0.05×10^{10} to 0.2×10^{10} FLOPS. **REDA+MDD** also achieves better performance than **ResNet34+MDD** while taking less computational budget.

VisDA-2017. The anytime prediction and dynamic evaluation results on VisDA-2017 are presented in Figure 4(c) and Figure 5 respectively. We adopt **CDAN** as our base DA method. The first exit of **REDA+CDAN** is ~7% better than **MSDNet+5×CDAN**, therefore taking ~2× times fewer FLOPS to achieve the same classification accuracy when doing the dynamic evaluation. It is inspiring that the top classifier of **REDA+CDAN** achieves ~3% higher accuracy than **ResNet34+CDAN** while taking less computational cost. In comparison, the top classifier of **MSDNet** lags behind its counterpart **ResNet34**, achieving ~6% lower accuracy.

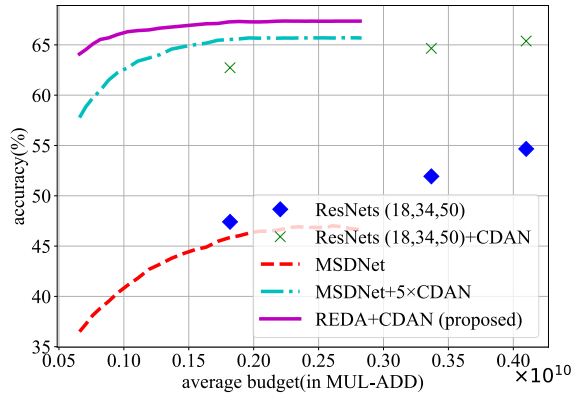


Figure 5: Accuracy (top-1) of budgeted batch classification on *VisDA-2017* dataset as a function of compute budget.

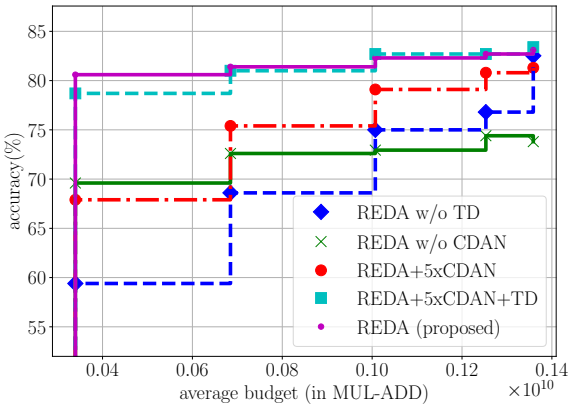


Figure 7: Ablation: anytime prediction on *Office-31* dataset.

4.3 Insight Analysis

Depth matters. We explore how the depth of ResNets influences the performance of DA methods, including DANN, CDAN, and MDD. Figure 1(a) shows the average performance of ResNets on *Office-home*. As the depth of ResNets increases, the accuracy of the baseline (**w/o DA**) increases and it's easy to think that the accuracy gain brought by DA should continue to decrease. Yet we're surprised to find that the accuracy gap between **MDD** and **w/o DA** enlarges as the depth increases from 18 to 50 (see Figure 1(b)). We find similar results on **CDAN**. Why does this trend happen? We argue that both **MDD** and **CDAN** rely on a prior that the features extracted by the ResNets backbone are clustered well by category. The deeper the network, the better the prior is satisfied, thus the larger the gap between the accuracy of **w/ DA** and **w/o DA**.

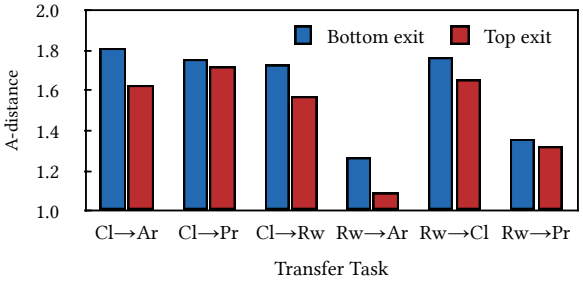


Figure 6: $d_{\mathcal{H}\Delta\mathcal{H}}(\mathcal{S},\mathcal{T})$ of bottom exit and top exit on *Office-Home* tasks. MSDNet was trained with CDAN loss added to each exit.

Where exactly does the larger accuracy gap come from? We analyze it based on the learning bound of domain adaptation [2, 40]. Denote the hypothesis space of classifier G as $\mathcal{H}$, then the probabilistic bound of the target risk is given by $\epsilon_{\mathcal{T}}(G) \leq \epsilon_{\mathcal{S}}(G) + \frac{1}{2}d_{\mathcal{H}\Delta\mathcal{H}}(\mathcal{S},\mathcal{T}) + \lambda$, where $\epsilon_{\mathcal{S}}(G)$ is the source risk, $d_{\mathcal{H}\Delta\mathcal{H}}(\mathcal{S},\mathcal{T})$ is a measure of domain discrepancy, and λ is the error of the ideal joint hypothesis $G^* \in \mathcal{H}$. As the network gets deeper, $\epsilon_{\mathcal{S}}(G)$ and λ would decrease, resulting in the accuracy increase in all methods. We further calculate $d_{\mathcal{H}\Delta\mathcal{H}}(\mathcal{S},\mathcal{T})$ on *Office-Home* with MSDNet. We observe that $d_{\mathcal{H}\Delta\mathcal{H}}(\mathcal{S},\mathcal{T})$ of the top exit is always smaller than that of the bottom exit (Figure 6), implying that increasing network depth can reduce domain discrepancy.

Ablation study. To justify that **TransDistill** and **DA loss** of the top classifier are the sources of performance gains, we do an ablation study using MSDNet (with $M = 5$ exits, one for every *fourth* layer) on *Office-31*. Note that we use TD to refer to TransDistill. Figure 7 reveals that TransDistill and CDAN are both vital since **REDA w/o TD** does a bad job when the computation budget is small while **REDA w/o CDAN** achieves low overall accuracy. Surprisingly, **MSDNet+5×CDAN+TD**, which adds CDAN loss to all exits and TransDistill loss to all early exits, is less accurate than **REDA**. The result indicates that the DA loss on the early exits is unnecessary. Thus, **REDA** provides a better way to improve shallow neural networks' transferability.

Effect of consistency weighting. We further examine the design of consistency weighting and adopt TransDistill and TransDistill (w/o consistency weighting) on Office-Home. Table 2 shows that consistency weighting can improve the accuracy of all exits effectively. To verify how consistency weighting works, we conducted the following experiment. On the Office-Home Ar→Cl task, we trained the ResNet50 with MDD loss. Every 1000 iterations, we calculated the accuracy of all instances and the accuracy of consistent instances on the target domain. Given data input $\mathbf{x}$ and ground truth $\mathbf{y}$ on the target domain, the model has two predictions $\widehat{\mathbf{y}}_1$ and $\widehat{\mathbf{y}}_2$ since we do data augmentation twice. The accuracy of all instances is defined by $\frac{\sum_{(\mathbf{x},\mathbf{y})\in\mathcal{T}} \mathbb{1}(h_{\widehat{\mathbf{y}}} = h_{\widehat{\mathbf{y}}_1})}{|\mathcal{T}|}$ and the accuracy of consistent instances is defined by $\frac{\sum_{(\mathbf{x},\mathbf{y})\in\mathcal{T}} \mathbb{1}(h_{\widehat{\mathbf{y}}_1} = h_{\widehat{\mathbf{y}}_2} = h_{\mathbf{y}})}{\sum_{(\mathbf{x},\mathbf{y})\in\mathcal{T}} \mathbb{1}(h_{\widehat{\mathbf{y}}_1} = h_{\widehat{\mathbf{y}}_2})}$. Figure 8 shows the results. The accuracy of consistent instances is about 20% higher than the accuracy of all instances. Therefore, the consistency weighting can filter out some instances with incorrect predictions. It has two benefits in the unsupervised scenarios. First, it improves the transferability of the early exits since it reduces the incorrect distillation from the teacher model. Second, it improves

Table 2: Ablation study on consistency weighting. Accuracy (top-1) of $M = 5$ exits on *Office-Home*. DA loss is MDD.

Method	Exit	Avg	Ar→Cl	Ar→Pr	Ar→Rw	Cl→Ar	Cl→Pr	Cl→Rw	Pr→Ar	Pr→Cl	Pr→Rw	Rw→Ar	Rw→Cl	Rw→Pr
REDA (w/o consistency weight)	0	54.8	41.9	63.5	64.8	44.0	58.7	59.9	41.8	44.9	66.1	52.9	48.9	70.7
	1	62.9	49.3	69.1	71.8	56.6	65.9	67.9	55.4	51.4	72.5	62.6	55.0	77.3
	2	64.5	50.6	70.9	72.5	59.1	67.1	69.2	54.7	52.3	74.2	69.8	56.2	76.9
	3	64.8	51.1	70.8	72.0	61.1	66.1	69.4	56.6	52.9	75.2	68.3	55.6	78.4
	4	66.2	52.3	71.6	75.7	61.9	68.0	70.2	58.2	54.3	76.8	70.2	55.8	79.8
REDA (TD)	0	**56.8**	43.8	63.9	67.1	46.1	63.7	63.1	44.7	45.2	67.0	55.4	49.8	72.2
	1	**63.9**	50.8	69.0	73.1	58.2	67.0	70.2	55.6	50.7	73.9	65.4	55.1	77.6
	2	**65.2**	52.0	70.9	73.2	60.9	67.3	69.8	57.2	53.6	74.5	69.1	56.2	78.2
	3	**65.9**	52.5	71.1	72.9	60.3	67.8	70.8	58.2	54.6	75.6	71.0	56.6	79.6
	4	**67.0**	53.7	71.1	75.6	63.0	67.9	72.1	58.6	55.1	76.8	71.0	57.6	81.2

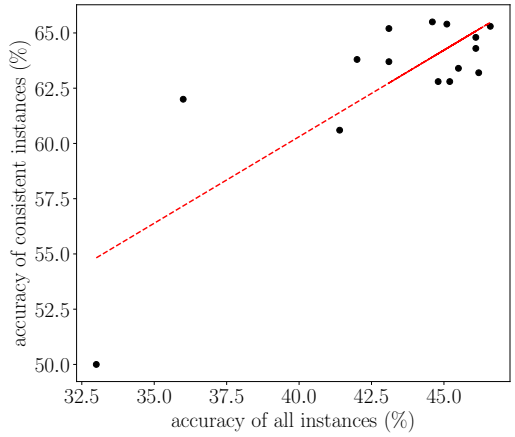


Figure 8: The accuracy of all instances and the consistent instances on *Office-Home* task Ar→Cl.

the transferability of the entire network, thereby improving the accuracy of the top exit.

5 CONCLUSION

In this paper, we uncovered that increasing network depth during training can improve the accuracy gain from adaptation modules, while the additional computation is unnecessary during inference. Based on our observations, we presented REDA, a novel and general framework for resource-efficient domain adaptation. REDA performs transferability distillation across different layers in the multi-exit architectures. Extensive empirical results validated the effectiveness and efficiency of REDA.

ACKNOWLEDGMENTS

The work was supported by the Natural Science Foundation of China (61772299, 71690231), and China University S&T Innovation Plan by Ministry of Education.

REFERENCES

[1] Taichi Asami, Ryo Masumura, Yoshikazu Yamaguchi, Hirokazu Masataki, and Yushi Aono. 2017. Domain adaptation of DNN acoustic models using knowledge distillation. In *IEEE*. 5185–5189.

[2] Shai Ben-David, John Blitzer, Koby Crammer, Alex Kulesza, Fernando Pereira, and Jennifer Wortman Vaughan. 2010. A theory of learning from different domains. *Machine Learning* 79, 1-2 (2010), 151–175.

[3] Konstantinos Bousmalis, Alex Irpan, Paul Wohlhart, Yunfei Bai, Matthew Kelcey, Mrinal Kalakrishnan, Laura Downs, Julian Ibarz, Peter Pastor, Kurt Konolige, Sergey Levine, and Vincent Vanhoucke. 2018. Using Simulation and Domain Adaptation to Improve Efficiency of Deep Robotic Grasping. In *ICRA*. 4243–4250.

[4] Xinyang Chen, Sinan Wang, Mingsheng Long, and Jianmin Wang. 2019. Transferability vs. Discriminability: Batch Spectral Penalization for Adversarial Domain Adaptation. In *ICML*. 1081–1090.

[5] Ozan Ciga, Jianan Chen, and Anne L. Martel. 2019. Multi-layer Domain Adaptation for Deep Convolutional Networks. In *MICCAI*. 20–27.

[6] Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel P. Kuksa. 2011. Natural Language Processing (Almost) from Scratch. *JMLR* (2011), 2493–2537.

[7] Dominik Maria Endres and Johannes E. Schindelin. 2003. A new metric for probability distributions. *Trans. Information Theory* 49, 7 (2003), 1858–1860.

[8] Michael Figurnov, Maxwell D. Collins, Yukun Zhu, Li Zhang, Jonathan Huang, Dmitry P. Vetrov, and Ruslan Salakhutdinov. 2017. Spatially Adaptive Computation Time for Residual Networks. In *CVPR*. 1790–1799.

[9] Bent Fuglede and Flemming Topsøe. 2004. Jensen-Shannon divergence and Hilbert space embedding. In *International Symposium on Information Theory*. 31.

[10] Yaroslav Ganin and Victor Lempitsky. 2015. Unsupervised domain adaptation by backpropagation. In *JMLR*.

[11] Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario Marchand, and Victor S. Lempitsky. 2016. Domain-Adversarial Training of Neural Networks. *JMLR* (2016), 59:1–59:35.

[12] Krzysztof J. Geras, Abdel rahman Mohamed, Rich Caruana, Gregor Urban, Shengjie Wang, Ozlem Aslan, Matthai Philipose, Matthew Richardson, and Charles Sutton. 2015. Blending LSTMs into CNNs. (2015). arXiv:cs.LG/1511.06433

[13] Xavier Glorot, Antoine Bordes, and Yoshua Bengio. 2011. Domain Adaptation for Large-Scale Sentiment Classification: A Deep Learning Approach. In *ICML*. 513–520.

[14] Boqing Gong, Kristen Grauman, and Fei Sha. 2013. Connecting the Dots with Landmarks: Discriminatively Learning Domain-Invariant Features for Unsupervised Domain Adaptation. In *ICML*.

[15] Boqing Gong, Yuan Shi, Fei Sha, and Kristen Grauman. 2012. Geodesic flow kernel for unsupervised domain adaptation. In *CVPR*.

[16] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron C. Courville, and Yoshua Bengio. 2014. Generative Adversarial Nets. In *NeurIPS*. 2672–2680.

[17] Raghuraman Gopalan, Ruonan Li, and Rama Chellappa. 2011. Domain adaptation for object recognition: An unsupervised approach. In *ICCV*. 999–1006.

[18] Alexander Grubb and Drew Bagnell. 2012. SpeedBoost: Anytime Prediction with Uniform Near-Optimality. In *AISTATS*. 458–466.

[19] Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q. Weinberger. 2017. On Calibration of Modern Neural Networks. In *ICML*. 1321–1330.

[20] Qiushan Guo, Zhipeng Yu, Yichao Wu, Ding Liang, Haoyu Qin, and Junjie Yan. 2019. Dynamic Recursive Neural Network. In *CVPR*. 5147–5156.

[21] Evgueni A. Haroutunian. 2011. Information Theory and Statistics. In *International Encyclopedia of Statistical Science*. 666–667.

[22] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep Residual Learning for Image Recognition. In *CVPR*. 770–778.

[23] Geoffrey E. Hinton, Oriol Vinyals, and Jeffrey Dean. 2015. Distilling the Knowledge in a Neural Network. *CoRR* abs/1503.02531 (2015).

[24] Judy Hoffman, Sergio Guadarrama, Eric Tzeng, Ronghang Hu, Jeff Donahue, Ross B. Girshick, Trevor Darrell, and Kate Saenko. 2014. LSDA: Large Scale Detection through Adaptation. In *NeurIPS*. 3536–3544.

[25] Judy Hoffman, Eric Tzeng, Taesung Park, Jun-Yan Zhu, Phillip Isola, Kate Saenko, Alexei A. Efros, and Trevor Darrell. 2018. CyCADA: Cycle-Consistent Adversarial Domain Adaptation. In *ICML*. 1994–2003.

[26] Andrew G. Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. 2017. MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications. *CoRR* abs/1704.04861 (2017).

[27] Gao Huang, Danlu Chen, Tianhong Li, Felix Wu, Laurens van der Maaten, and Kilian Q. Weinberger. 2018. Multi-Scale Dense Networks for Resource Efficient Image Classification. In *ICLR*.

[28] Jiayuan Huang, Alexander J. Smola, Arthur Gretton, Karsten M. Borgwardt, and Bernhard Schölkopf. 2006. Correcting Sample Selection Bias by Unlabeled Data. In *NeurIPS*.

[29] Shu Kong and Charless C. Fowlkes. 2019. Pixel-Wise Attentional Gating for Scene Parsing. In *WACV*. 1024–1033.

[30] Kevin Lai and Dieter Fox. 2010. Object Recognition in 3D Point Clouds Using Web Data and Domain Adaptation. *I. J. Robotics Res.* 29, 8 (2010), 1019–1037.

[31] Xingjian Li, Haoyi Xiong, Hanchao Wang, Yuxuan Rao, Liping Liu, and Jun Huan. 2019. Delta: Deep Learning Transfer using Feature Map with Attention for Convolutional Networks. In *ICLR*.

[32] Zhizhong Li and Derek Hoiem. 2018. Learning without Forgetting. *TPAMI* 40, 12 (2018), 2935–2947.

[33] Zhichao Li, Yi Yang, Xiao Liu, Feng Zhou, Shilei Wen, and Wei Xu. 2017. Dynamic Computational Time for Visual Attention. In *ICCV*. 1199–1209.

[34] Ji Lin, Yongming Rao, Jiwen Lu, and Jie Zhou. 2017. Runtime Neural Pruning. In *NeurIPS*. 2181–2191.

[35] M. Long, Y. Cao, Z. Cao, J. Wang, and M. I. Jordan. 2018. Transferable Representation Learning with Deep Adaptation Networks. *TPAMI* (2018).

[36] Mingsheng Long, Yue Cao, Jianmin Wang, and Michael I. Jordan. 2015. Learning Transferable Features with Deep Adaptation Networks. In *ICML*. 97–105.

[37] Mingsheng Long, Zhangjie Cao, Jianmin Wang, and Michael I. Jordan. 2018. Conditional Adversarial Domain Adaptation. In *NeurIPS*. 1647–1657.

[38] Mingsheng Long, Han Zhu, Jianmin Wang, and Michael I. Jordan. 2016. Unsupervised Domain Adaptation with Residual Transfer Networks. In *NeurIPS*.

[39] Mingsheng Long, Han Zhu, Jianmin Wang, and Michael I. Jordan. 2017. Deep Transfer Learning with Joint Adaptation Networks. In *ICML*. 2208–2217.

[40] Yishay Mansour, Mehryar Mohri, and Afshin Rostamizadeh. 2009. Domain Adaptation: Learning Bounds and Algorithms. In *COLT*.

[41] Lane McIntosh, Niru Maheswaranathan, David Sussillo, and Jonathon Shlens. 2018. Recurrent Segmentation for Variable Computational Budgets. In *CVPR*. 1648–1657.

[42] Maxime Oquab, Leon Bottou, Ivan Laptev, and Josef Sivic. 2014. Learning and Transferring Mid-level Image Representations Using Convolutional Neural Networks. In *CVPR*.

[43] Sinno Jialin Pan, Ivor W. Tsang, James T. Kwok, and Qiang Yang. 2011. Domain Adaptation via Transfer Component Analysis. *IEEE Trans. Neural Networks* 22, 2 (2011), 199–210.

[44] Nicolas Papernot, Patrick D. McDaniel, Xi Wu, Somesh Jha, and Ananthram Swami. 2016. Distillation as a Defense to Adversarial Perturbations Against Deep Neural Networks. In *SP*. 582–597.

[45] Zhongyi Pei, Zhangjie Cao, Mingsheng Long, and Jianmin Wang. 2018. Multi-Adversarial Domain Adaptation. In *AAAI*. 3934–3941.

[46] Xingchao Peng, Zijun Huang, Ximeng Sun, and Kate Saenko. 2019. Domain Agnostic Learning with Disentangled Representations. In *ICML*. 5102–5112.

[47] Mary Phuong and Christoph H. Lampert. 2019. Distillation-Based Training for Multi-Exit Architectures. In *ICCV*.

[48] Antonio Polino, Razvan Pascanu, and Dan Alistarh. 2018. Model compression via distillation and quantization. In *ICLR*.

[49] Joaquin Quionero-Candela, Masashi Sugiyama, Anton Schwaighofer, and Neil D. Lawrence. 2009. *Dataset Shift in Machine Learning*. The MIT Press. 274–274 pages.

[50] Ilija Radosavovic, Piotr Dollár, Ross B. Girshick, Georgia Gkioxari, and Kaiming He. 2018. Data Distillation: Towards Omni-Supervised Learning. In *CVPR*. 4119–4128.

[51] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael S. Bernstein, Alexander C. Berg, and Fei-Fei Li. 2014. ImageNet Large Scale Visual Recognition Challenge. *CoRR* abs/1409.0575 (2014).

[52] K. Saenko, B. Kulis, M. Fritz, and T. Darrell. 2010. Adapting Visual Category Models to New Domains. In *ECCV*.

[53] Kuniaki Saito, Kohei Watanabe, Yoshitaka Ushiku, and Tatsuya Harada. 2018. Maximum Classifier Discrepancy for Unsupervised Domain Adaptation. In *CVPR*.

[54] Yu Sun, Eric Tzeng, Trevor Darrell, and Alexei A. Efros. 2019. Unsupervised Domain Adaptation through Self-Supervision. *CoRR* abs/1909.11825 (2019).

[55] Surat Teerapittayanon, Bradley McDanel, and H. T. Kung. 2016. BranchyNet: Fast inference via early exiting from deep neural networks. In *ICPR*. 2464–2469.

[56] Eric Tzeng, Judy Hoffman, Trevor Darrell, and Kate Saenko. 2015. Simultaneous Deep Transfer Across Domains and Tasks. In *ICCV*.

[57] Eric Tzeng, Judy Hoffman, Kate Saenko, and Trevor Darrell. 2017. Adversarial Discriminative Domain Adaptation. In *CVPR*. 2962–2971.

[58] Eric Tzeng, Judy Hoffman, Ning Zhang, Kate Saenko, and Trevor Darrell. 2014. Deep Domain Confusion: Maximizing for Domain Invariance. *CoRR* abs/1412.3474 (2014).

[59] Andreas Veit and Serge J. Belongie. 2018. Convolutional Networks with Adaptive Inference Graphs. In *ECCV*. 3–18.

[60] Hemanth Venkateswara, Jose Eusebio, Shayok Chakraborty, and Sethuraman Panchanathan. 2017. Deep Hashing Network for Unsupervised Domain Adaptation. In *CVPR*.

[61] Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily B. Fox, and Roman Garnett (Eds.). 2019. *NeurIPS*.

[62] Ximei Wang, Ying Jin, Mingsheng Long, Jianmin Wang, and Michael I Jordan. 2019. Transferable Normalization: Towards Improving Transferability of Deep Neural Networks. In *NeurIPS*. 1951–1961.

[63] Xin Wang, Fisher Yu, Zi-Yi Dou, Trevor Darrell, and Joseph E. Gonzalez. 2018. SkipNet: Learning Dynamic Routing in Convolutional Networks. In *ECCV*. 420–436.

[64] Zuxuan Wu, Tushar Nagarajan, Abhishek Kumar, Steven Rennie, Larry S. Davis, Kristen Grauman, and Rogério Schmidt Feris. 2018. BlockDrop: Dynamic Inference Paths in Residual Networks. In *CVPR*. 8817–8826.

[65] Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. 2014. How transferable are features in deep neural networks?. In *NeurIPS*.

[66] Kaichao You, Ximei Wang, Mingsheng Long, and Michael I. Jordan. 2019. Towards Accurate Model Selection in Deep Unsupervised Domain Adaptation. In *ICML*. 7124–7133.

[67] Chris Zhang, Mengye Ren, and Raquel Urtasun. 2019. Graph HyperNetworks for Neural Architecture Search. In *ICLR*.

[68] Linfeng Zhang, Jiebo Song, Anni Gao, Jingwei Chen, Chenglong Bao, and Kaisheng Ma. [n. d.]. Be Your Own Teacher: Improve the Performance of Convolutional Neural Networks via Self Distillation. *CoRR* ([n. d.]).

[69] Yuchen Zhang, Tianle Liu, Mingsheng Long, and Michael I. Jordan. 2019. Bridging Theory and Algorithm for Domain Adaptation. In *ICML*. 7404–7413.

[70] Han Zhao, Remi Tachet des Combes, Kun Zhang, and Geoffrey J. Gordon. 2019. On Learning Invariant Representations for Domain Adaptation. In *ICML*. 7523–7532.