Task-Adversarial Co-Generative Nets

Pei Yang
Arizona State University
South China University of Technology
cs.pyang@gmail.com

Qi Tan
South China Normal University
tanqi@scnu.edu.cn

Hanghang Tong
Arizona State University
hanghang.tong@gmail.com

Jingrui He
Arizona State University
jingrui.he@gmail.com

ABSTRACT

In this paper, we propose Task-Adversarial co-Generative Nets (TAGN) for learning from multiple tasks. It aims to address the two fundamental issues of multi-task learning, i.e., domain shift and limited labeled data, in a principled way. To this end, TAGN first learns the task-invariant representations of features to bridge the domain shift among tasks. Based on the task-invariant features, TAGN generates the plausible examples for each task to tackle the data scarcity issue. In TAGN, we leverage multiple game players to gradually improve the quality of the co-generation of features and examples by using an adversarial strategy. It simultaneously learns the marginal distribution of task-invariant features across different tasks and the joint distributions of examples with labels for each task. The theoretical study shows the desired results: at the equilibrium point of the multi-player game, the feature extractor exactly produces the task-invariant features for different tasks, while both the generator and the classifier perfectly replicate the joint distribution for each task. The experimental results on the benchmark data sets demonstrate the effectiveness of the proposed approach.

CCS CONCEPTS

• **Computing methodologies** → **Machine learning approaches**; **Multi-task learning**.

KEYWORDS

multi-task learning; generative adversarial nets

ACM Reference Format:

Pei Yang, Qi Tan, Hanghang Tong, and Jingrui He. 2019. Task-Adversarial Co-Generative Nets. In *The 25th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '19), August 4–8, 2019, Anchorage, AK, USA*. ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3292500.3330843

KDD '19, August 4–8, 2019, Anchorage, AK, USA

ACM ISBN 978-1-4503-6201-6/19/08...$15.00
https://doi.org/10.1145/3292500.3330843

1 INTRODUCTION

Domain shift [2] and limited labeled data are the two fundamental issues for deep multi-task learning. On one hand, although deep learning has achieved impressive success in various areas, the deep features learned on millions of examples are susceptible to domain shift [7], which usually refers to the difference of distributions between the data collected from the related tasks or domains. For example, the typical causes of visual domain shift include changes in the camera, image resolution, lighting, background, viewpoint, and post-processing [25]. On the other hand, it is expensive or unrealistic to collect a large amount of labeled data for each task. For instance, the tagged cancer images are very limited in each hospital, which severely hampers the generalization performance of the image analysis system [6].

In this paper, we propose Task-Adversarial co-Generative Nets (TAGN) for deep multi-task learning. The goal is to tackle the both issues, i.e., the domain shift and sparse labeled data, in a principled way. TAGN uses an adversarial strategy to generate both 'good' features and 'good' examples. Here, we regard the 'good' features as the task-invariant features shared across different tasks, which help bridge the domain gap. Also, we regard the 'good' examples as those generated examples which could act as the real training data to build the classifier. TAGN accommodates multiple game players, i.e., feature extractor, domain discriminator, classifier, generator, and label discriminator. They play in an adversarial way to co-generate the task-invariant features and the plausible examples. Specifically, TAGN first learns the task-invariant features by using a task-adversarial strategy. It encourages the feature extractor to generate the features which is indistinguishable by the domain discriminator. Based on the task-invariant features, TAGN not only builds the classifier to predict the label, but also builds the generator to generate the fake examples. Then, TAGN adopts the label discriminator to discern whether the pair (example, label) generated by either the generator or the classifier is fake or not. Since the task-invariant features encode the knowledge shared among different tasks, it could help both the generator and the classifier to generate high-quality pairs. Therefore, the generations of either task-invariant features or plausible examples are coupled together in TAGN.

The effectiveness of the task-adversarial co-generative nets is verified theoretically and empirically. We present the theoretical study of the TAGN approach. It shows that at the equilibrium point of the multi-player game, the feature extractor exactly produces the domain-invariant features across different tasks, while both the generator and the classifier perfectly duplicate the joint data distribution for each task. In other words, TAGN perfectly generates both features and examples in a unified model. Also, we conduct the detailed experiments on the benchmark data sets. The experimental results demonstrate that TAGN outperforms state-of-the-art methods by smoothing the domain shift and alleviating the scarcity of the labeled data. The main contributions of this work are summarized as follows:

- Adversarial co-generation of domain-invariant features and plausible examples to bridge the domain gap and tackle sparse data issue in a principled way.
- Guarantee of game equilibrium regarding the marginal distribution of the task-invariant features and the joint distributions of the plausible examples with labels.
- Experiments on the benchmark data demonstrating the effectiveness of the proposed method.

The rest of the paper is organized as follows. Section 2 reviews the related work. The proposed TAGN approach is introduced in Section 3, followed by the theoretical study in Section 4. We show the experimental results in Section 5, and conclude the paper in Section 6.

2 RELATED WORK

We review the related work on shallow or deep multi-task learning, as well as the related generative adversarial nets.

Multi-task learning [4] aims to improve the performance of each task by borrowing knowledge learned from related tasks. Different assumptions on task relatedness lead to different multi-task learning models. Some typical work include: multi-task feature learning [1], clustered multi-task learning [34], low-dimensional subspace learning [14], multi-task relationship learning [33], robust multi-task learning [5], sparsity-regularized multi-task learning [12, 17, 31], etc.

Recently, deep multi-task learning or deep domain adaptation becomes to receive attentions since it harnesses the power of deep learning [16] and multi-task learning [4] (or domain adaptation [2]). Although multi-task learning and domain adaptation (or transfer learning) are distinctive with each other, they technically share much commonness. Therefore, below we introduce them indiscriminately. The related approaches of deep multi-task learning can be roughly divided into four categorizations: domain-adversarial networks [3, 9, 23, 27, 28], domain distance-based method [10, 19, 21, 29], deep models with tensor regularization [20, 32], and adaptation based on image translation [13, 24].

First, inspired by idea of generative adversarial nets (GAN) [11], the domain-adversarial neural network [9] used adversarial training to promote the emergence of domain-invariant features via the use of a gradient reversal layer. In [27], they simultaneously aligned domains via domain confusion and aligned source and target classes via soft labels. The adversarial discriminative domain adaptation model [28] combined discriminative modeling, untied weight sharing, and a domain-adversarial loss into a unified framework. The multi-adversarial domain adaptation approach [23] captured multi-mode structures to enable fine-grained alignment of domains based on multiple domain discriminators. The domain separation networks [3] explicitly modeled both private and shared components of domain representations. Second, the joint adaptation networks [21] adopted an adversarial training strategy to maximize a joint maximum mean discrepancy (MMD) criterion. The domain adaptive neural network [10] incorporated MMD measure as a regularization embedded in the supervised back-propagation training. The deep domain confusion (DDC) model [29] had the MMD loss at one layer, while deep adaptation network (DAN) [19] had the MMD losses at multiple layers. Third, the deep multi-task learning model with tensor factorization [32] learned the shared feature subspace from multilayer parameter tensors, while the multilinear relationship networks (MRN) [20] learned multilinear task relationships from multiplayer parameter tensors. There are no pseudo example generation for the above three types of methods. In contrast, the translation-based methods adapted source images to appear as if drawn from the target domain. The cycle-consistent adversarial adaptation model [13] enforced both structural and semantic consistency during adaptation using a cycle-consistency loss and semantics loss. The bi-directional adaptive GAN [24] used the symmetric adversarial strategy to encourage the network to produce both target-like and source-like images.

Our proposed method is distinctive from the existing works in the following aspects. The domain-adversarial networks [3, 9, 23, 27] based on gradient reversal layer [9] could not guarantee that the feature extractor and the domain discriminator will finally reach the equilibrium. We remedy this defect and propose a task-adversarial method for feature generation with equilibrium guarantee. For the generation of examples, we do not start from the scratch as done by the translation-based adaptation methods [13, 24]. In contrast, we use the task-invariant features to generate the plausible pairs. Similar to the Triple-GAN [18] and the Triangle-GAN [8], we have to discern the multiple joint distributions. However, Triple-GAN [18] uses an asymmetrical objective and has to provide the explicit density form of conditional probability, while Triangle-GAN [8] couples the two discriminators in the objective which may render difficulty in optimization. Instead we use two decoupled discriminators to distinguish among three joint distributions in our objective. To summing up, we propose a novel task-adversarial co-generative nets with equilibrium guarantee.

3 THE TAGN METHOD

In this section, we introduce the proposed task-adversarial co-generative nets and the optimization algorithm.

3.1 Task-adversarial co-generative nets

Suppose we have multiple related learning tasks. Here, we focus on classification issue. For example, the categorization of the images is a multi-class classification problem. Also, the images may come from different domains, and each domain corresponds to a task. For instance, the artistic images include the paintings and artistic depictions, while the real-world images are the regular pictures captured with cameras, and they may have the same classes. The goal is to improve the performance of all the learning tasks by sharing the strength of each task via the proposed task-adversarial co-generative nets.

Figure 1 shows the high-level architecture of the proposed TAGN model. Suppose we have T related tasks. x_t is the data example from the t^{th} task. Each example $x_t \in \mathcal{X}$ is associated with a class label $y_t \in \mathcal{Y}$ and a domain (or task) label $d (1 \leq d \leq T)$. Denote the number of classes by c. The feature extractor F accepts the example x_t and produces the latent representation z_t. The domain discriminator D tries to discern which task the latent representation z_t belongs to. The classifier C_t takes z_t as input and attempts to predict its label y_t. The generator G_t accepts the latent representation z_t and a random signal v (e.g., standard normal distribution), and endeavors to generate the plausible example for the t^{th} task. Then, we use a label discriminators Q_t to determine whether the pair (x_t, y_t) is real or not. Either the feature extractor F or the domain discriminator D is shared by multiple tasks. Denote $G = \{G_t\}, C = \{C_t\}$, and $Q = \{Q_t\}$ for simplicity, where $1 \leq t \leq T$. In TAGN, the aforementioned components are parameterized as neural networks.

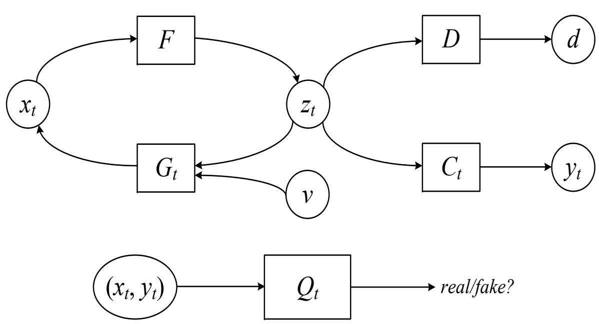

Figure 1: Task-adversarial co-generative nets.

We aim to address the two fundamental issues of deep multi-task learning, i.e., domain shift and limited training data, in a principled way. Our proposed strategy is to learn the 'good' features to bridge the domain gap, and generate the 'good' examples to tackle the sparse labeled data issue. As shown in Figure 1, TAGN has two pipelines. The first pipeline consists of the feature extractor F and the domain discriminator D, with the goal to learn the task-invariant features. The second pipeline is composed of the feature extractor F, the classifier C_t, the generator G_t, and the label discriminator Q_t, in order to generate the plausible examples for each task. The two pipelines are integrated into a unified way. Both pipelines share the feature extractor and the latent representation. The learned task-invariant features are used to build the high-quality classifiers and generators. Next, we elaborate the two pipelines in more details.

3.1.1 Adversarial feature generation. Although the domain-adversarial networks [3, 9, 23, 27] using gradient reversal layer [9] are popular for adversarial adaptation, they could not guarantee that the feature extractor and the domain discriminator will finally reach the game equilibrium. To remedy this drawback, we propose a novel task-adversarial method with equilibrium guarantee.

In the proposed TAGN model, the domain discriminator D and the feature extractor F compete with each other to learn the task-invariant features. The feature extractor tries to generate the features which are indistinguishable by the domain discriminator. The domain discriminator acts as a multi-class classifier to handle the multi-class discrimination. Therefore, D has T outputs. Denote the t^{th} output of D by D_t, which corresponds to the discrimination of the $t^{th} (1 \leq t \leq T)$ task from the other tasks. For the discrimination of the t^{th} task, z_t is viewed as the real example, while $z_k (k \neq t, 1 \leq k \leq T)$ from all the other tasks is viewed as the fake one. Let $p_t(z)$ be the probability distribution of the latent representation z_t, i.e., $z_t \sim p_t(z)$. Define the mean distribution $\bar{p}_t(z)$ of the latent representation $z_k (k \neq t)$ for all the other tasks $(1 \leq k \leq T)$ as:

$$\bar{p}_t(z) = \frac{1}{m-1} \sum_{k \neq t} p_k(z).$$

The domain discriminator D tries to distinguish the task-specific distribution $p_t(z)$ from the mean distribution $\bar{p}_t(z)$ of the other tasks. Therefore, the loss function of feature generation for the t^{th} task is as follows:

$$\begin{aligned} L_f(F, D_t) =& \mathbb{E}_{z \sim p_t(z)} \Big[log\big(D_t(z)\big) \Big] \\ &+ \mathbb{E}_{z \sim \bar{p}_t(z)} \Big[log\big(1 - D_t(z)\big) \Big] \end{aligned} \tag{1}$$

The feature extractor F and the domain discriminator D play an adversarial game in order to achieve the equilibrium such that

$$p_1(z) = p_2(z) = \cdots = p_T(z).$$

It means that the marginal distribution of the latent representation for each task is equal with each other. The formal proof will be presented in the next section. The equilibrium indicates that although different tasks have different data distributions in the original feature spaces, it is possible for them to have the same marginal distribution in the latent space. In this ideal situation, the domain distance among tasks is approaching to zero.

3.1.2 Adversarial example generation. In this pipeline, the multi-players including the feature extractor, generator, classifier, and label discriminator play an adversarial game to generate the plausible example, as well as accurate prediction.

Based on the task-invariant features, the classifier predicts its label for the unlabeled example. For the labeled example, the generator takes both its latent representation and the random signal as input, and produces the plausible example. Therefore, we have three types of pair fed to the label discriminator, i.e., true example with true label (x, y), true example with predicted label $(x, \hat{y})$, and pseudo example with true label $(\hat{x}, y)$. The label discriminator tries to discern the fake pair from the real one, while either the classifier or the generator attempts to generate the fake pair which is indistinguishable by the label discriminator.

For the t^{th} task, let $p_t(x, y)$ be the true joint distribution, $p_t^c(x, y)$ the joint distribution produced by the classifier, and $p_t^g(x, y)$ the joint distribution produced by the generator. The label discriminator Q_t uses two sub-networks to discriminate among the three types of joint distributions. The first sub-network is to distinguish $p_t(z, y)$ from the mean of the fake distributions $\bar{p}_t(x, y)$ which is defined as:

$$\bar{p}_t(x, y) = \frac{p_t^c(x, y) + p_t^g(x, y)}{2}.$$

The loss function for the first sub-network of Q_t is as follows:

$$\begin{aligned} L_e(F, G_t, C_t, Q_t^1) = &\mathbb{E}_{(x,y)\sim p_t(x,y)}\Big[log\big(Q_t^1(x, y)\big)\Big] \\ &+ \mathbb{E}_{(x,y)\sim \bar{p}_t(x,y)}\Big[log\big(1 - Q_t^1(x, y)\big)\Big] \end{aligned} \tag{2}$$

The second sub-network of Q_t is to distinguish between $p_t^c(z, y)$ and $p_t^g(z, y)$:

$$\begin{aligned} L_e(F, G_t, C_t, Q_t^2) = &\mathbb{E}_{(x,y)\sim p_t^c(x,y)}\Big[log\big(Q_t^2(x, y)\big)\Big] \\ &+ \mathbb{E}_{(x,y)\sim p_t^g(x,y)}\Big[log\big(1 - Q_t^2(x, y)\big)\Big] \end{aligned} \tag{3}$$

In total, the loss function of example generation for the t^{th} task is as follows:

$$L_e(F, G_t, C_t, Q_t) = \sum_{i=1}^{2} L_e(F, G_t, C_t, Q_t^i). \tag{4}$$

The label discriminator Q_t plays the adversarial game with the other components $\{F, G_t, C_t\}$ to guarantee that the generated fake joint distributions, $p_t^c(x, y)$ and $p_t^g(x, y)$, are indistinguishable from the real joint distribution $p_t(x, y)$ for each task $(1 \leq t \leq T)$, i.e.,

$$p_t(x, y) = p_t^c(x, y) = p_t^g(x, y).$$

The equilibrium suggests that it is possible for both the generator and the classifier to precisely duplicate the true joint distribution of each task. We will prove the game equilibrium in the next section.

3.1.3 Overall objective. We couple the two pipelines in a principled fashion. By learning the task-invariant features, we are able to bridge the domain gap, and better manipulate the knowledge sharing among different tasks. Also, since the task-invariant features encode the common knowledge shared between tasks, it allows both the generator and the classifier to make use of these knowledge to generate high-quality pairs. In summary, the co-generation of feature and example allows us to learn both the 'good' features which are transferable across domains, and generate 'reliable' examples which could act as the real labeled data.

The overall objective of the proposed TAGN method is as follows:

$$\begin{aligned} &\min_{F,G,C} \max_{D,Q} L(F, G, C, D, Q) \\ &= \min_{F,G,C} \max_{D,Q} \sum_{t=1}^{T} \Big[L_e(F, G_t, C_t, Q_t) + \alpha L_f(F, D_t)\Big] \end{aligned} \tag{5}$$

where α is the non-negative trade-off parameter.

Specifically, there are two min-max games in TAGN. The first min-max game is played between the domain discriminator D and the feature extractor F, in which F attempts to minimize the feature generation loss L_f, while D tries to maximize it. The second min-max game is played among Q_t and $\{F, G_t, C_t\}$, in which Q_t endeavors to maximize the example generation loss L_e, while $\{F, G_t, C_t\}$ try to minimize it. Therein the feature extractor F shared by both pipelines is responsible for generating both transferable features and reliable examples. In such a way, TAGN simultaneously learns both the marginal distribution of task-invariant features and the joint distributions of examples with labels for each task.

3.2 Algorithm

The proposed TAGN algorithm is summarized Algorithm 1. The outer loop iterates over the number of training epochs, while the inner loop iterates over the number of tasks. In Lines 3-4, we sample a batch data from the t^{th} task and pass it through the network. Lines 5-6 compute the loss of adversarial feature generation and update the domain discriminator. Lines 7-18 compute the loss of adversarial example generation and update label discriminator, the generator, and the classifier. Therein, Lines 7-12 handle the labeled data, while Lines 13-17 deal with the unlabeled data. The feature extractor involves in both pipelines, which is updated in Line 19. After training τ_{max} iterations, we append the generated pairs $(\hat{x}, y)$ into the training data, and rebuild the classifier to get the final prediction for the unlabeled data.

Since the sizes of labeled examples for each task are very small, the label discriminator learned from limited training data may reject the other types of examples from the true data distribution. Following the similar strategy used in [8, 18], we pick out some unlabeled examples with the high confidence (> 0.85), and use the pairs of unlabeled example and pseudo label as real examples to train the label discriminators.

4 THEORETICAL ANALYSIS

We analysis the multi-player game equilibrium of the proposed task-adversarial co-generative nets in this section.

As mentioned above, TAGN involves two min-max games played among three teams, i.e., $\{F, G, C\}$, $\{Q\}$, and $\{D\}$. It is important to investigate whether the game among the multiple players could achieve the equilibrium. Theorem 1 shows that the game has a global optimum.

THEOREM 1 (MULTI-PLAYER GAME EQUILIBRIUM). *In TAGN, the multi-player game among the three teams achieves the equilibrium if and only if*

$$p_t(x, y) = p_t^c(x, y) = p_t^g(x, y)(1 \leq t \leq T)$$

Algorithm 1 The **TAGN** Algorithm

Input: input data $\mathcal{X} \times \mathcal{Y}$ from T tasks, trade-off parameter α, batch size b, number of iterations τ_{max}.

Output: predictions for unlabeled data.

for number of iterations τ_{max} **do**
  for number of tasks T **do**
    Sample a batch of data $\{(x_t, y_t)\}$ of size b from the t^{th} task;
    Forward pass the batch through the network including $\{F, D, G_t, C_t\}$ and generate the task-invariant features z_t, pseudo label $\hat{y}_t$, and fake example $\hat{x}_t$;
    Compute the loss of adversarial feature generation L_f;
    Update the domain discriminator D by ascending along its stochastic gradient $\nabla_D \alpha L_f$;
    if labeled data **then**
      Forward pass the batch $\{(x_t, y_t)\}$ through Q_t;
      Forward pass the batch $\{(\hat{x}_t, y_t)\}$ through Q_t;
      Compute the loss of adversarial example generation L_e;
      Update the label discriminator Q_t by ascending along its stochastic gradient $\nabla_{Q_t} L_e$;
      Update the generator G_t by descending along its stochastic gradient $\nabla_{G_t} L_e$;
    else
      Forward pass the batch $\{(x_t, \hat{y}_t)\}$ through Q_t;
      Compute the loss of adversarial example generation L_e;
      Update the label discriminator Q_t by ascending along its stochastic gradient $\nabla_{Q_t} L_e$;
      Update the classifier C_t by descending along its stochastic gradient $\nabla_{C_t} L_e$;
    end if
    Update the feature extractor F by descending along its stochastic gradient $\nabla_F (L_e + \alpha L_f)$;
  end for
end for
Append the generated pairs $(\hat{x}, y)$ into training data;
Rebuild the classifier to get the final prediction.

and

$$p_1(z) = p_2(z) = \cdots = p_T(z).$$

At the equilibrium point, the multiple teams of game players reach their optimal values:

(i) For fixed $\{F, G, C, Q\}$, the domain discriminator D arrives at its maximum:

$$D_t^*(z) = \frac{p_t(z)}{p_t(z) + \bar{p}_t(z)} \tag{6}$$

(ii) For fixed $\{F, G, C, D\}$, the label discriminators Q_t $(1 \leq t \leq T)$ arrives at its maximums:

$$Q_t^{1*}(x, y) = \frac{p_t(x, y)}{p_t(x, y) + \bar{p}_t(x, y)} \tag{7}$$

$$Q_t^{2*}(x, y) = \frac{p_t^c(x, y)}{p_t^c(x, y) + p_t^g(x, y)} \tag{8}$$

(iii) Given the optimal $D^*(z)$ and $Q^*(x, y)$, the global minimum of objective Eq. 5 is:

$$L^*(F, G, C, D^*, Q^*) = -(2 + \alpha) T log 4 \tag{9}$$

Proof. In the multi-player game, the team $\{F, G, C\}$ tries to minimize the objective Eq. 5 while both teams, $\{D\}$ and $\{Q\}$, attempt to maximize this objective.

i) First, given $\{F, G, C, Q\}$, the training criterion for the domain discriminator D regarding its t^{th} output is to maximize:

$$\begin{aligned} & L_f(F, D_t) \\ =& \mathbb{E}_{z \sim p_t(z)} \Big[log\big(D_t(z)\big) \Big] + \mathbb{E}_{z \sim \bar{p}_t(z)} \Big[log\big(1 - D_t(z)\big) \Big] \\ =& \int p_t(z) log\big(D_t(z)\big) dz + \int \bar{p}_t(z) \Big[log\big(1 - D_t(z)\big) \Big] dz \end{aligned}$$

For any $(a, b) \in \mathbb{R}^2 \backslash \{0, 0\}$, the function $f \rightarrow alog(f) + blog(1 - f)$ achieves its maximum at $\frac{a}{a+b}$. Therefore, the domain discriminator reaches its maximum at

$$D_t^*(z) = \frac{p_t(z)}{p_t(z) + \bar{p}_t(z)}$$

for $1 \leq t \leq T$.

ii) Second, given $\{F, G, C, D\}$, the training criterion for the discriminator Q_t regarding its first output is to maximize:

$$\begin{aligned} & L_d(F, G, C, Q_t^1) \\ =& \mathbb{E}_{(x,y) \sim p_t(x,y)} \Big[log\big(Q_t^1(x, y)\big) \Big] \\ & + \mathbb{E}_{(x,y) \sim \bar{p}_t(x,y)} \Big[log\big(1 - Q_t^1(x, y)\big) \Big] \\ =& \int \int p_t(x, y) log\big(Q_t^1(x, y)\big) dx dy \\ & + \int \int \bar{p}_t(x, y) \Big[log\big(1 - Q_t^1(x, y)\big) \Big] dx dy. \end{aligned}$$

The label discriminator Q_t achieves its maximum for its first output at

$$Q_t^{1*}(x, y) = \frac{p_t(x, y)}{p_t(x, y) + \bar{p}_t(x, y)}.$$

The training criterion for the discriminator Q_t regarding its second output is to maximize:

$$\begin{aligned} & L_d(F, G, C, Q_t^2) \\ =& \mathbb{E}_{(x,y) \sim p_t^c(x,y)} \Big[log\big(Q_t^2(x, y)\big) \Big] \\ & + \mathbb{E}_{(x,y) \sim p_t^g(x,y)} \Big[log\big(1 - Q_t^2(x, y)\big) \Big] \\ =& \int \int p_t^c(x, y) log\big(Q_t^2(x, y)\big) dx dy \\ & + \int \int p_t^g(x, y) \Big[log\big(1 - Q_t^2(x, y)\big) \Big] dx dy. \end{aligned}$$

The label discriminator Q_t achieves its maximum for its second output at

$$Q_t^{2*}(x, y) = \frac{p_t^c(x, y)}{p_t^c(x, y) + p_t^g(x, y)}.$$

iii) Third, given the optimal $D^*(z)$ and $Q^*(x,y)$, the global minimum of objective Eq. 5 is:

$$
\begin{aligned}
&L^*(F,G,C,D^*,Q^*)\\
=&\sum_{t=1}^{T}\Big[L_e(F,G_t,C_t,Q_t^*)+\alpha L_f(F,D_t^*)\Big]\\
=&\sum_{t=1}^{T}\Big[\mathbb{E}_{(x,y)\sim p_t(x,y)}\Big[log\frac{p_t(x,y)}{p_t(x,y)+\bar{p}_t(x,y)}\Big]\\
&\quad+\mathbb{E}_{(x,y)\sim \bar{p}_t(x,y)}\Big[log\frac{\bar{p}_t(x,y)}{p_t(x,y)+\bar{p}_t(x,y)}\Big]\Big]\\
&+\sum_{t=1}^{T}\Big[\mathbb{E}_{(x,y)\sim p_t^c(x,y)}\Big[log\frac{p_t^c(x,y)}{p_t^c(x,y)+p_t^g(x,y)}\Big]\\
&\quad+\mathbb{E}_{(x,y)\sim p_t^g(x,y)}\Big[log\frac{p_t^g(x,y)}{p_t^c(x,y)+p_t^g(x,y)}\Big]\Big]\\
&+\alpha\sum_{t=1}^{T}\Big[\mathbb{E}_{z\sim p_t(z)}\Big[log\frac{p_t(z)}{p_t(z)+\bar{p}_t(z)}\Big]\\
&\quad+\mathbb{E}_{z\sim \bar{p}_t(z)}\Big[log\frac{\bar{p}(z)}{p_t(z)+\bar{p}_t(z)}\Big]\Big]\\
=&2\sum_{t=1}^{T}\Big[JSD\big(p_t(x,y),\bar{p}_t(x,y)\big)+JSD\big(p_t^c(x,y),p_t^g(x,y)\big)\Big]\\
&-(2+\alpha)Tlog4+2\alpha\sum_{t=1}^{T}JSD\Big(p_t(z),\bar{p}_t(z)\Big)\\
\geq&-(2+\alpha)Tlog4.
\end{aligned}
$$

where $JSD(\cdot)$ is the Jensen-Shannon divergence. The objective achieves its global minimum $-(2+\alpha)Tlog4$ if and only if the following conditions are satisfied:

(1) $p_t(x,y)=\bar{p}_t(x,y)$
(2) $p_t^c(x,y)=p_t^g(x,y)$
(3) $p_t(z)=\bar{p}_t(z)$

where $1\leq t\leq T$.

Based on the first two equilibrium conditions, we have $p_t(x,y)=p_t^c(x,y)=p_t^g(x,y)$, where $1\leq t\leq T$.

Now, consider the third equilibrium condition. Denote the matrix

$$
\mathbf{A}=\begin{bmatrix} T-1 & -1 & \cdots & -1\\ -1 & T-1 & \cdots & -1\\ \vdots & \vdots & \ddots & \vdots\\ -1 & -1 & \cdots & T-1 \end{bmatrix}
$$

and the column vector

$$
\mathbf{p}=\big[p_1(z),p_2(z),\cdots,p_T(z)\big]^T.
$$

The equilibrium conditions $p_t(z)=\bar{p}_t(z)(1\leq t\leq T)$ corresponds to the linear system

$$
\mathbf{A}\mathbf{p}=\mathbf{0}. \tag{10}
$$

Denote the rank of $\mathbf{A}$ by $r(\mathbf{A})$. Since $r(\mathbf{A})=T-1$, the linear system $\mathbf{A}\mathbf{p}=\mathbf{0}$ has one fundamental solution, i.e.,

$$
p_1(z)=p_2(z)=\cdots=p_T(z).
$$

In summary, the objective Eq. 5 achieves its global minimum if and only if $p_t(x,y)=p_t^c(x,y)=p_t^g(x,y)$ and $p_1(z)=p_2(z)=\cdots=p_T(z)$. □

Theorem 1 provides the insights into the proposed TAGN model regarding the game equilibrium among the multiple players. At the equilibrium point, the feature extractor exactly produces the task-invariant features across different tasks, while both the generator and the classifier precisely replicate the joint data distribution for each task. In other words, the task-adversarial co-generative nets perfectly generates both transferable features and reliable examples at the same time.

5 EXPERIMENTAL RESULTS

In order to verify the effectiveness of the proposed method, we compare TAGN with a variety of state-of-the-art algorithms on the image datasets, which are the standard benchmarks for the evaluation of multi-task learning algorithms.

5.1 Data sets

The Office-Home[1] [30] dataset consists of 15500 images from 4 different domains: 1) Artistic images (paintings, sketches and artistic depictions); 2) Clip art (clipart images); 3) Product images (images without background); and 4) Real-world images (regular images captured with a camera). For each domain, the dataset contains images of 65 object categories found typically in office and home settings. The images in the dataset were crawled through several search engines and online image directories. Therefore, we have four learning tasks corresponding four domains including Artistic (A), Clipart (C), Product (P), and Real-world images (R). Each learning task is a multi-class (c=65) classification problem.

The Office-Caltech dataset consists of 2533 images selected from the 10 common categories shared by the Office-31[2] [25] dataset and the Caltech-256[3] dataset. The Office-31 dataset is a collection of 4652 images in 31 categories collected from three distinct domains, i.e., Amazon, DSLR, and Webcam. The Amazon domain consists of images at medium resolution typically taken in an environment with studio lighting conditions. The DSLR domain consists of images that are captured with a digital camera in realistic environments. The Webcam domain is composed of images recorded with a simple webcam at low resolution. For the Caltech-256 dataset, it is a collection of 30607 images in 256 categories downloaded from Google Images. Hence, it yields four learning tasks corresponding to four domains: Amazon (A), Webcam (W), DSLR (D), and Caltech (C). Likewise, each learning task is a multi-class (c=10) classification problem.

5.2 Comparison methods

We compare TAGN with a variety of methods including the classic convolutional neural network (CNN) such as AlexNet [15] or VGG [26] , multi-task feature learning (MTFL) [1],

[1] http://hemanthdv.org/OfficeHome-Dataset/
[2] https://people.eecs.berkeley.edu/~jhoffman/domainadapt/
[3] http://www.vision.caltech.edu/Image_Datasets/Caltech256/

Table 1: The network architecture of TAGN.

Feature extractor	Domain discriminator	Classifier	Generator	Label discriminator
CNN Dropout Linear(512*7*7,4096) BatchNorm1d, ReLU Dropout Linear(4096,4096) BatchNorm1d, ReLU Dropout Linear(4096,256) BatchNorm1d, ReLU Dropout	Linear(256,1024) ReLU Dropout Linear(1024,1024) ReLU Dropout Linear(1024,T) Softmax	Linear(256,1024) ReLU Dropout Linear(1024,1024) ReLU Dropout Linear(1024,c) Softmax	ConvTranspose2d(512,2048) BatchNorm2d, ReLU ConvTranspose2d(2048,1024) BatchNorm2d, ReLU ConvTranspose2d(1024,512) BatchNorm2d, ReLU ConvTranspose2d(512,256) BatchNorm2d, ReLU ConvTranspose2d(256,128) BatchNorm2d, ReLU ConvTranspose2d(128,64) BatchNorm2d, ReLU ConvTranspose2d(64,3) Tanh	Conv2d(3,64) LeakyReLU Conv2d(64,128) BatchNorm2d, LeakyReLU Conv2d(128,256) BatchNorm2d, LeakyReLU Conv2d(256,512) BatchNorm2d, LeakyReLU Conv2d(512,1024) BatchNorm2d, LeakyReLU Conv2d(1024,2048) BatchNorm2d, LeakyReLU Conv2d(2048,512) BatchNorm2d Linear(512,c) Softmax

Table 2: Classification accuracy on Office-Caltech.

Method	5%					10%					20%				
	A	W	D	C	Avg	A	W	D	C	Avg	A	W	D	C	Avg
AlexNet ([15])	88.9	73.0	80.4	88.7	82.8	92.2	80.9	88.2	88.9	87.6	91.3	83.3	93.7	**94.9**	90.8
MTFL ([1])	90.0	78.9	90.2	86.9	86.5	92.4	85.3	89.5	89.2	89.1	93.5	89.0	95.2	92.6	92.6
RMTL ([5])	91.3	82.3	88.8	89.1	87.9	92.6	85.2	93.3	87.2	89.6	94.4	87.0	96.7	93.4	92.4
MTRL ([33])	86.4	83.0	95.1	89.1	88.4	91.1	87.1	97.0	87.6	90.7	90.0	88.8	99.2	94.3	93.1
DMTRL ([32])	91.2	88.3	92.5	85.6	89.4	92.2	91.9	97.4	86.8	92.0	92.6	97.6	94.5	88.4	93.3
MRN ([20])	92.5	**97.5**	97.9	87.5	93.8	93.6	**98.6**	98.6	87.3	94.5	94.4	98.3	**99.9**	89.1	95.5
TAGN	**92.8**	93.9	**98.2**	**91.1**	**94.0**	**93.9**	95.9	**98.8**	**91.0**	**94.9**	**94.8**	**98.7**	98.9	93.8	**96.6**

multi-task relationship learning (MTRL) [33], robust multi-task learning (RMTL) [5], deep multi-task learning with tensor factorization (DMTRL) [32], and multilinear relationship networks (MRN) [20]. MTFL [1], MTRL [33], and RMTL [5] are shallow multi-task learning algorithm, while DMTRL [32] and MRN [20] are deep multi-task learning methods proposed recently. MTFL [1] extracts the low-rank shared feature representations by learning feature covariance, while RMTL [5] extends MTFL [1] to further capture the task relationships using a low-rank structure and identify outlier tasks using a group-sparse structure. MTRL [33] captures the task relationships using task covariance of a matrix normal distribution. DMTRL [32] tackles multi-task deep learning by tensor factorization, which learns shared feature subspace instead of multilinear task relationship in multilayer parameter tensors. The multilinear relationship networks (MRN) [20] simultaneously learns transferable features and multilinear relationships of tasks and deep features.

5.3 Network architecture

The TAGN algorithm is implemented using the open source PyTorch [22] package[4]. It can be trained using the standard backpropagation algorithms based on min-batch stochastic gradient descent or its modifications. We adopt learning rate decaying strategy. The initial learning rate is set to 0.001, and the momentum is 0.9. The number of training iterations is set as $\tau_{max} = 2000$, and the sampling batch size b is 20. For the trade-off parameter, we empirically set $\alpha = 1$.

[4]https://github.com/pytorch

The network structure of TAGN is showed in Table 1. It consists of five components, i.e., feature extractor, domain discriminator, classifier, generator, and label discriminator. Specifically, the feature extractor is based on the classic CNNs, followed by multiple linear layers, ReLU, batch normalization, and Dropout layers. We adopt AlexNet [15] and VGG [26] for the Office-Caltech and Office-Home datasets, respectively. The Dropout layer randomly zeroes some of the elements of the input tensor using samples from a Bernoulli distribution. The non-linear activation ReLU applies the rectified linear unit function element-wise, $ReLU(x) = max(0, x)$, to the input data. Both of the domain discriminator and the classifier are composed of multiple linear layers, ReLU, Dropout, and Softmax layers. For the generator, it is made of multiple ConvTranspose2d layers and the batch normalization. The ConvTranspose2d layer applies a 2D transposed convolution operator over an input image composed of several input planes. The last layer Tanh applies the element-wise function, $Tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$, to its input. The label discriminator has two sub-networks, which have the same structure as shown in Table 1. Their differences lie in types of pairs fed to the sub-networks. The label discriminator mainly consists of multiple Conv2d, LeakyReLU, and batch normalization layers. The Conv2d layer applies a 2D convolution over an input signal composed of several input planes. The LeakyReLU layer applies the element-wise function, $LeakyReLU(x) = max(0, x) + \lambda * min(0, x)$, to its input, where λ controls the angle of the negative slope.

Table 3: Classification accuracy on Office-Home.

Method	5%					10%					20%				
	A	C	P	R	Avg	A	C	P	R	Avg	A	C	P	R	Avg
VGG ([26])	35.8	31.2	67.8	62.5	49.3	51.0	40.7	75.0	68.8	58.9	56.1	54.6	80.4	71.8	65.7
MTFL ([1])	40.1	30.4	61.5	59.5	47.9	50.3	35.0	66.3	65.0	54.2	55.2	38.8	69.1	70.0	58.3
RMTL ([5])	42.3	32.8	62.3	60.6	49.5	49.7	34.6	65.9	64.6	53.7	55.2	39.2	69.9	70.5	58.6
MTRL([33])	42.7	33.3	62.9	61.3	50.1	51.6	36.3	67.7	66.3	55.5	55.8	39.9	70.2	71.2	59.3
DMTRL ([32])	49.2	34.5	67.1	62.9	53.4	57.2	42.3	73.6	69.9	60.8	58.3	56.1	79.3	72.1	66.5
MRN ([20])	53.3	36.4	70.5	**67.7**	57.0	59.9	42.7	76.3	73.0	63.0	58.5	55.6	80.7	72.8	66.9
TAGN	**57.8**	**44.7**	**73.8**	66.1	**60.6**	**64.5**	**61.3**	**80.6**	**78.9**	**71.3**	**66.4**	**65.9**	**84.7**	**76.9**	**73.5**

5.4 Performance comparison

We follow the standard protocol [20, 33] for multi-task learning and randomly select 5%, 10%, and 20% (training ratio) examples from each task as trainset and use the rest as testset, respectively. We repeat five random experiments and report the average classification accuracy on the testset.

Tables 2-3 show the classification accuracy on Office-Caltech and Office-Home datasets, respectively. The results of the comparison methods are quoted from the related papers [1, 5, 20, 32, 33]. We have the following observations from the experiment results.

- CNN performs better than the shallow multi-task learning approaches such as MTFL [1], MTRL [33], and RMTL [5] on the Office-Home dataset when the training ratio is relatively large (e.g., 10% or 20%), verifying the superiority of CNN for feature learning. However, when the labeled data turn sparser (e.g., 5% in Office-Home) or the domains are more similar (as in Office-Caltech), the shallow multi-task learning methods outperform CNN, demonstrating the advantages of sharing the strength among the related tasks.
- All the deep multi-task learning methods including TAGN, DMTRL [32], and MRN [20] outperform both classic CNNs and the shallow multi-task learning approaches. It demonstrates that deep multi-task learning can further promote the performance by simultaneously learning the hierarchical features from data and sharing knowledge across tasks.
- Our proposed TAGN method outperforms all the comparison algorithms on both datasets in most case. The performance superiority of TAGN is more significant on the relatively difficult problem (i.e., Office-Home). It verifies the effectiveness of the proposed approach. TAGN co-learns the transferable features and plausible examples to bridge the domain gap and alleviate the scarcity of labeled data, leading to better performance.

5.5 Ablation study

We further conduct an ablation study to investigate how the individual component of TAGN impacts the classification performance. We setup the experiment with three settings:

- $TAGN$: the proposed TAGN algorithm.
- $TAGN_f$: TAGN with the generation of feature only.
- $TAGN_n$: the naïve implementation of TAGN, which has no the generation of features and examples.

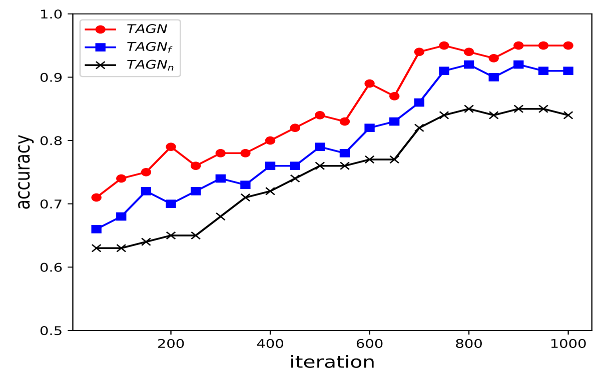

Figure 2: Ablation study on Office-Caltech dataset.

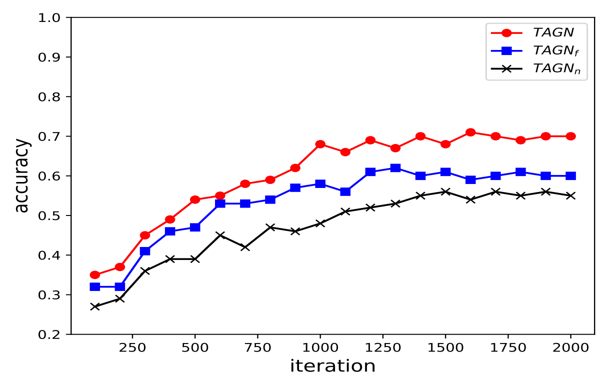

Figure 3: Ablation study on Office-Home dataset.

Specifically, both $TAGN_f$ and $TAGN_n$ are the reduced version of TAGN. For $TAGN_f$, we disable the generator. For $TAGN_n$, we disable both the generator and the domain discriminator, and use the classification loss to back-propagate the gradient. The ablation study is conducted on both the Office-Home and Office-Caltech datasets. The training ratio is fixed to 10%. We set the numbers of training iterations as 1000 and 2000 for Office-Caltech and Office-Home, respectively. Figures 2-3 plot the performance curves varying with the training epoch for the two datasets, respectively.

From the figures, we can see that $TAGN_f$ outperforms $TAGN_n$ by learning the task-invariant features, which facilitate the knowledge sharing among tasks. Furthermore, the inadequacy of labeled data could be alleviated by leveraging the generated examples, and as a result $TAGN$ performs better than $TAGN_f$. The results suggest that either the extraction of transferable features or the generation of the plausible examples is indispensable, and both of them help improve the performance.

Also, the classification accuracy of all the three algorithms improves along the training epochs, and finally converge to the stable values.

6 CONCLUSION

We propose TAGN, task-adversarial co-generative nets for deep multi-task learning. TAGN employs multiple game-players to simultaneously generate the task-invariant features and plausible examples, in order to smooth the domain shift and tackle the limited training data issue. Theoretically we prove the equilibrium of the multi-player game. The effectiveness of the proposed approach is also empirically verified on the benchmark data sets. Although we focus on multi-task learning in this paper, the proposed approach is flexible and widely applicable to other areas. As on-going work, we will adapt the proposed method to multi-modal scenarios.

ACKNOWLEDGMENTS

This work is supported by National Natural Science Foundation of China (Grant No. 61473123), Natural Science Foundation of Guangdong, China (Grant No. 2017A030313370 and 2018A030313356), National Science Foundation (Grant No. IIS-1552654, IIS-1813464, IIS-1651203, and CNS-1629888), the U.S. Department of Homeland Security (Grant Award No. 17STQAC00001-02-00), and an IBM Faculty Award. The views and conclusions are those of the authors and should not be interpreted as representing the official policies of the funding agencies or the governments.

REFERENCES

[1] Andreas Argyriou, Theodoros Evgeniou, and Massimiliano Pontil. 2008. Convex multi-task feature learning. *Machine Learning* 73, 3 (2008), 243–272.

[2] John Blitzer, Mark Dredze, and Fernando Pereira. 2007. Biographies, Bollywood, Boom-boxes and Blenders: Domain Adaptation for Sentiment Classification. In *ACL*.

[3] Konstantinos Bousmalis, George Trigeorgis, Nathan Silberman, Dilip Krishnan, and Dumitru Erhan. 2016. Domain Separation Networks. In *NIPS*. 343–351.

[4] Rich Caruana. 1993. Multitask Learning: A Knowledge-Based Source of Inductive Bias. In *ICML*. 41–48.

[5] Jianhui Chen, Lei Tang, Jun Liu, and Jieping Ye. 2013. A Convex Formulation for Learning a Shared Predictive Structure from Multiple Tasks. *IEEE Trans. Pattern Anal. Mach. Intell.* 35, 5 (2013), 1025–1038.

[6] Nicolas Coudray, Paolo Santiago Ocampo, Theodore Sakellaropoulos, Navneet Narula, Matija Snuderl, David Feny, Andre L. Moreira, Narges Razavian, and Aristotelis Tsirigos. 2018. Classification and mutation prediction from nonsmall cell lung cancer histopathology images using deep learning. *Nature Medicine* 24 (2018), 1559–1567.

[7] Jeff Donahue, Yangqing Jia, Oriol Vinyals, Judy Hoffman, Ning Zhang, Eric Tzeng, and Trevor Darrell. 2014. DeCAF: A Deep Convolutional Activation Feature for Generic Visual Recognition. In *ICML*. 647–655.

[8] Zhe Gan, Liqun Chen, Weiyao Wang, Yunchen Pu, Yizhe Zhang, Hao Liu, Chunyuan Li, and Lawrence Carin. 2017. Triangle Generative Adversarial Networks. In *NIPS*. 5253–5262.

[9] Yaroslav Ganin and Victor S. Lempitsky. 2015. Unsupervised Domain Adaptation by Backpropagation. In *ICML*. 1180–1189.

[10] Muhammad Ghifary, W. Bastiaan Kleijn, and Mengjie Zhang. 2014. Domain Adaptive Neural Networks for Object Recognition. In *PRICAI*. 898–904.

[11] 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 *NIPS*. 2672–2680.

[12] Lei Han and Yu Zhang. 2015. Learning Tree Structure in Multi-Task Learning. In *KDD*. 397–406.

[13] 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.

[14] Shuiwang Ji and Jieping Ye. 2009. An Accelerated Gradient Method for Trace Norm Minimization. In *ICML*. 457–464.

[15] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. 2012. ImageNet Classification with Deep Convolutional Neural Networks. In *NIPS*. 1106–1114.

[16] Yann LeCun, Yoshua Bengio, and Geoffrey E. Hinton. 2015. Deep learning. *Nature* 521, 7553 (2015), 436–444.

[17] Giwoong Lee, Eunho Yang, and Sung Ju Hwang. 2016. Asymmetric Multi-task Learning Based on Task Relatedness and Loss. In *ICML*. 230–238.

[18] Chongxuan Li, Taufik Xu, Jun Zhu, and Bo Zhang. 2017. Triple Generative Adversarial Nets. In *NIPS*. 4091–4101.

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

[20] Mingsheng Long, Zhangjie Cao, Jianmin Wang, and Philip S. Yu. 2017. Learning Multiple Tasks with Multilinear Relationship Networks. In *NIPS*. 1593–1602.

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

[22] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. 2017. Automatic differentiation in PyTorch. In *NIPS-W*.

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

[24] Paolo Russo, Fabio Maria Carlucci, Tatiana Tommasi, and Barbara Caputo. 2018. From Source to Target and Back: Symmetric Bi-Directional Adaptive GAN. In *CVPR*. 8099–8108.

[25] Kate Saenko, Brian Kulis, Mario Fritz, and Trevor Darrell. 2010. Adapting Visual Category Models to New Domains. In *ECCV*. 213–226.

[26] Karen Simonyan and Andrew Zisserman. 2015. Very Deep Convolutional Networks for Large-Scale Image Recognition. In *ICLR*. 1106–1114.

[27] Eric Tzeng, Judy Hoffman, Trevor Darrell, and Kate Saenko. 2015. Simultaneous Deep Transfer Across Domains and Tasks. In *ICCV*. 4068–4076.

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

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

[30] Hemanth Venkateswara, Jose Eusebio, Shayok Chakraborty, and Sethuraman Panchanathan. 2017. Deep Hashing Network for Unsupervised Domain Adaptation. In *CVPR*. 5385–5394.

[31] Pei Yang, Qi Tan, and Jingrui He. 2017. Multi-task Function-on-function Regression with Co-grouping Structured Sparsity. In *KDD*. 1255–1264.

[32] Yongxin Yang and Timothy M. Hospedales. 2017. Deep Multi-task Representation Learning: A Tensor Factorisation Approach. In *ICLR*.

[33] Yu Zhang and Dit-Yan Yeung. 2010. A Convex Formulation for Learning Task Relationships in Multi-Task Learning. In *UAI*. 733–442.

[34] Jiayu Zhou, Jianhui Chen, and Jieping Ye. 2011. Clustered Multi-Task Learning Via Alternating Structure Optimization. In *NIPS*. 702–710.