Task-Adversarial Co-Generative Nets

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

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

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

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

ABSTRACT

In this paper, we propose Task-Adversarial co-Generative Nets (TANG) 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, TANG first learns the task-invariant representations of features to bridge the domain shift among tasks. Based on the task-invariant features, TANG generates the plausible examples for each task to tackle the data scarcity issue. In TANG, 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

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 (TANG) 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. TANG 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. TANG 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, TANG 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, TANG not only builds the classifier to predict the label, but also builds the generator to generate the fake examples. Then, TANG 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 TANG.

The effectiveness of the task-adversarial co-generative nets is verified theoretically and empirically. We present the theoretical study of the TANG approach. It shows that at the
We review the related work on shallow or deep multi-task learning with tensor regularization [20, 32], and adaptation based on multi-linear relationship networks [20] learned multi-linear task relationships from multi-layer parameter tensors. Therefore, below we introduce them indiscriminately. The related approaches 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 includes: 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 distinct 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 multi-linear task relationships from multi-layer parameter tensors.

There are no pseudo example generation for the above three types of methods. In contrast, 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 TGN model. Suppose we have $T$ related tasks. $x_t$ is the data example from the $t^{th}$ task. Each example $x_t \in X$ is associated with a class label $y_t \in Y$ 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 $G = \{G_t\}, \ C = \{C_t\}, \text{and} \ Q = \{Q_t\}$ for simplicity, where $1 \leq t \leq T$. In TGN, the aforementioned components are parameterized as neural networks.

![Figure 1: Task-adversarial co-generative nets.](image)

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, TGN 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 TGN 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 $\bar{p}(z)$ of the latent representation $\bar{z}_t(k \neq t)$ for all the other tasks $(1 \leq k \leq T)$ as:

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

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

$$L_f(F, D_t) = \mathbb{E}_{z_t \sim p_t(z)} \left[ \log(D_t(z)) \right] + \mathbb{E}_{z \sim \bar{p}(z)} \left[ \log(1 - D_t(z)) \right]$$

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

$$p_t(z) = p_2(z) = \cdots = p_r(z).$$

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
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, $p'_t(x, y)$ the joint distribution produced by the classifier, and $p''_t(x, y)$ 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 $p_t(x, y)$ from the mean of the fake distributions $\bar{p}_t(x, y)$ which is defined as:

$$\bar{p}_t(x, y) = \frac{p'_t(x, y) + p''_t(x, y)}{2}.$$  

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

$$L_e(F, G_t, C_t, Q'_t) = E_{(x,y) \sim p_t(x,y)} \left[ \log(Q'_t(x, y)) \right] + E_{(x,y) \sim \bar{p}_t(x,y)} \left[ \log(1 - Q'_t(x, y)) \right]$$  \hspace{1cm} (2)

The second sub-network of $Q_t$ is to distinguish between $p'_t(x, y)$ and $p''_t(x, y)$:

$$L_e(F, G_t, C_t, Q''_t) = E_{(x,y) \sim p'_t(x,y)} \left[ \log(Q''_t(x, y)) \right] + E_{(x,y) \sim p''_t(x,y)} \left[ \log(1 - Q''_t(x, y)) \right]$$  \hspace{1cm} (3)

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

$$L_e(F, G_t, C_t, Q_t) = \sum_{i=1}^{2} L_e(F, G_t, C_t, Q'_i).$$  \hspace{1cm} (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, $p'_t(x, y)$ and $p''_t(x, y)$, are indistinguishable from the real joint distribution $p_t(x, y)$ for each task $(1 \leq t \leq T)$, i.e.,

$$p_t(x, y) = p'_t(x, y) = p''_t(x, y).$$

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 this 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:

$$\min_{F, G, C, D, Q} \max \sum_{t=1}^{T} \left[ L_e(F, G_t, C_t, Q_t) + \alpha L_f(F, D_t) \right]$$  \hspace{1cm} (5)

where $\alpha$ 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 $Q_t$ attempts 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 the $t^{th}$ task, and pass it through the network. Lines 7-12 compute the loss of the $t^{th}$ task for the labeled data, and pass it through the network. In Lines 13-17, the generator and the classifier are trained to minimize the joint distribution loss $L_e$, and pass it through the network. In Lines 18-20, the domain discriminator is trained to maximize the joint distribution loss $L_f$, and pass it through the network. Finally, in Lines 21-22, update the parameters of the models.

4 THEORETICAL ANALYSIS

We analyze 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

$$p_t(x, y) = p_t'(x, y) = p_t''(x, y) (1 \leq t \leq T)$$
At the equilibrium point, the multiple teams of game players reach their optimal values:

1. Forward pass the batch through the network including \( \{F, D, G_1, C_1\} \) and generate the task-invariant features \( z_t \), pseudo label \( \hat{y}_t \), and fake example \( \hat{x}_t \);
2. Compute the loss of adversarial feature generation \( L_f \);
3. Update the feature extractor \( F \) by descending along its stochastic gradient \( \nabla_F \alpha L_f \);
4. Update the generator \( G_1 \) by descending along its stochastic gradient \( \nabla_{G_1} L_c \);
5. Update the classifier \( C_1 \) by descending along its stochastic gradient \( \nabla_{C_1} L_c \);
6. Forward pass the batch \( \{(x_t, y_t)\} \) through \( Q_1 \);
7. Compute the loss of adversarial example generation \( L_{e_1} \);
8. Update the label discriminator \( Q_1 \) by descending along its stochastic gradient \( \nabla_{Q_1} L_{e_1} \);
9. Update the classifier \( C_1 \) by descending along its stochastic gradient \( \nabla_{C_1} L_{e_1} \);
10. Forward pass the batch \( \{(x_t, \hat{y}_t)\} \) through \( Q_1 \);
11. Update the label discriminator \( Q_1 \) by descending along its stochastic gradient \( \nabla_{Q_1} L_{e_1} \);
12. Update the domain discriminator \( D \) by descending along its stochastic gradient \( \nabla_D \alpha L_{f} \);
13. Update the generator \( G \) by descending along its stochastic gradient \( \nabla_G \theta \).

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

\[
L^*(F, G, C, D^*, Q^*) = -(2 + \alpha)T \log 4
\]

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 first output is to maximize:

\[
L_f(F, D_t) = \mathbb{E}_{x \sim p_t(x)} \left[ \log(D_t(z)) \right] + \mathbb{E}_{z \sim \bar{p}_t(z)} \left[ \log(1 - D_t(z)) \right]
\]

\[
= \int p_t(x) \log(D_t(z)) dz + \int \bar{p}_t(z) \left[ \log(1 - D_t(z)) \right] dz
\]

For any \( (a, b) \in \mathbb{R}^2 \setminus \{0, 0\} \), the function \( f \to a \log(f) + b \log(1 - f) \) 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:

\[
L_d(F, G, C, Q_t) = \mathbb{E}_{(x, y) \sim p_t(x, y)} \left[ \log(Q_t^1(x, y)) \right] + \mathbb{E}_{(z, y) \sim \bar{p}_t(x, y)} \left[ \log(1 - Q_t^1(x, y)) \right]
\]

\[
= \int \int p_t(x, y) \log(Q_t^1(x, y)) dxdy + \int \int \bar{p}_t(x, y) \left[ \log(1 - Q_t^1(x, y)) \right] dxdy.
\]

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:

\[
L_d(F, G, C, Q_t^2) = \mathbb{E}_{(x, y) \sim p_t^2(x, y)} \left[ \log(Q_t^2(x, y)) \right] + \mathbb{E}_{(z, y) \sim \bar{p}_t^2(x, y)} \left[ \log(1 - Q_t^2(x, y)) \right]
\]

\[
= \int \int p_t^2(x, y) \log(Q_t^2(x, y)) dxdy + \int \int \bar{p}_t^2(x, y) \left[ \log(1 - Q_t^2(x, y)) \right] dxdy.
\]

The label discriminator \( Q_t \) achieves its maximum for its second output at

\[
Q_t^2(x, y) = \frac{p_t^2(x, y)}{p_t^2(x, y) + \bar{p}_t^2(x, y)}
\]
The equilibrium conditions of the system are:

\[ \text{Denote the rank of } A \text{ by } r(A). \)

Since \( r(A) = T - 1 \), the linear system \( AP = 0 \) has one fundamental solution, i.e.,

\[ p_1(z) = p_2(z) = \cdots = p_T(z). \]

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

\[
L^*(F, G, C, D^*, Q^*) = \sum_{t=1}^{T} \left[ L_t(F, G_t, C_t, Q_t^*) + \alpha L_f(F, D_t^*) \right]
\]

\[
= \sum_{t=1}^{T} \left[ \mathbb{E}_{(x, y) \sim p_t(x, y)} \left[ \log \frac{p_t(x, y)}{\bar{p}_t(x, y) + \bar{p}(x, y)} \right] + \mathbb{E}_{(x, y) \sim \bar{p}_t(x, y)} \left[ \log \frac{\bar{p}_t(x, y)}{\bar{p}(x, y) + \bar{p}(x, y)} \right] \right]
\]

\[
+ \sum_{t=1}^{T} \left[ \mathbb{E}_{(x, y) \sim p_t^c(x, y)} \left[ \log \frac{p_t^c(x, y)}{p_t^c(x, y) + p_t^c(x, y)} \right]\right]
\]

\[
+ \sum_{t=1}^{T} \left[ \mathbb{E}_{(x, y) \sim \bar{p}_t^c(x, y)} \left[ \log \frac{\bar{p}_t^c(x, y)}{\bar{p}_t^c(x, y) + \bar{p}_t^c(x, y)} \right] \right] + \alpha \sum_{t=1}^{T} \left[ \mathbb{E}_{z \sim p_t(z)} \left[ \log \frac{p_t(z)}{\bar{p}(z) + \bar{p}(z)} \right] \right]
\]

\[
= 2 \sum_{t=1}^{T} \left[ JSD(p_t(x, y), \bar{p}_t(x, y)) + JSD(p_t^c(x, y), \bar{p}_t^c(x, y)) \right]
\]

\[
- (2 + \alpha)T \log 4 + 2\alpha \sum_{t=1}^{T} JSD(p_t(z), \bar{p}_t(z)) \]

\[
\geq - (2 + \alpha)T \log 4.
\]

where \( JSD(\cdot) \) is the Jensen-Shannon divergence. The objective achieves its global minimum \( - (2 + \alpha)T \log 4 \) if and only if the following conditions are satisfied:

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

where \( 1 \leq t \leq T \).

Based on the first two equilibrium conditions, we have

\[
p_t(x, y) = \bar{p}_t(x, y) = p_t^c(x, y), \text{ where } 1 \leq t \leq T.
\]

Now, consider the third equilibrium condition. Denote the matrix

\[
A = \begin{bmatrix}
T - 1 & -1 & \cdots & -1 \\
-1 & T - 1 & \cdots & -1 \\
\vdots & \vdots & \ddots & \vdots \\
-1 & -1 & \cdots & T - 1
\end{bmatrix}
\]

and the column vector

\[
p = \begin{bmatrix} p_t(z), p_2(z), \ldots, p_T(z) \end{bmatrix}^T.
\]

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

\[
Ap = 0.
\]

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

\[
\square
\]

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\textsuperscript{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 to 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\textsuperscript{2} [25] dataset and the Caltech-256\textsuperscript{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\textsuperscript{4} [15] or VGG\textsuperscript{5} [26] , multi-task feature learning (MTFL) [1],

\footnotetext[1]{http://hemanthdv.org/OfficeHome-Dataset/}
\footnotetext[2]{https://people.eecs.berkeley.edu/~jhoffman/domainadapt/}
\footnotetext[3]{http://wwwvision.caltech.edu/Image_Datasets/Caltech256/}

5.3 Network architecture

The TAGN algorithm is implemented using the open source PyTorch package\(^1\). It can be trained using the standard backpropagation algorithms based on mini-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 \(\tau_{\text{max}} = 2000\), and the sampling batch size \(b\) is 20. For the trade-off parameter, we empirically set \(\alpha = 1\).

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, which is defined linear unit function element-wise, to its input, \(\text{ReLU}(x) = \max(0, x)\), to its input, and \(\min(0, x)\), to its input. 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, \(\text{Tanh}(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}\), to its input. The label discriminator has two sub-networks, which have the same structure as shown in Table 1. Their differences lie in type 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, \(\text{LeakyReLU}(x) = \max(0, x) + \lambda \min(0, x)\), to its input, where \(\lambda\) controls the angle of the negative slope.

---

\(^1\)https://github.com/pytorch

---

**Table 1: The network architecture of TAGN.**

<table>
<thead>
<tr>
<th>Feature extractor</th>
<th>Domain discriminator</th>
<th>Classifier</th>
<th>Generator</th>
<th>Label discriminator</th>
</tr>
</thead>
<tbody>
<tr>
<td>CNN Dropout</td>
<td>Linear(256,1024) ReLU ConvTranspose2d(512,2048)</td>
<td>Linear(256,1024) ReLU ConvTranspose2d(512,2048)</td>
<td>Conv2d(3,64) LeakyReLU</td>
<td></td>
</tr>
<tr>
<td>Dropout</td>
<td>ReLU BatchNorm2d, LeakyReLU</td>
<td>ReLU BatchNorm2d, LeakyReLU</td>
<td>Conv2d(64,128)</td>
<td></td>
</tr>
<tr>
<td>Linear(512<em>7</em>7,4096)</td>
<td>Dropout BatchNorm2d, LeakyReLU</td>
<td>Dropout BatchNorm2d, LeakyReLU</td>
<td>Conv2d(128,256)</td>
<td></td>
</tr>
<tr>
<td>BatchNorm1d, ReLU</td>
<td>ConvTranspose2d(1024,1024) BatchNorm2d, LeakyReLU</td>
<td>ConvTranspose2d(1024,1024) BatchNorm2d, LeakyReLU</td>
<td>Conv2d(256,128)</td>
<td></td>
</tr>
<tr>
<td>Dropout</td>
<td>ReLU BatchNorm2d, LeakyReLU</td>
<td>Dropout BatchNorm2d, LeakyReLU</td>
<td>Conv2d(512,1024)</td>
<td></td>
</tr>
<tr>
<td>Linear(4096,4096)</td>
<td>Softmax ConvTranspose2d(512,256)</td>
<td>Softmax ConvTranspose2d(512,256)</td>
<td>BatchNorm2d, LeakyReLU</td>
<td></td>
</tr>
<tr>
<td>BatchNorm1d, ReLU</td>
<td>ConvTranspose2d(1024,T) ConvTranspose2d(64,3)</td>
<td>ConvTranspose2d(1024,T) ConvTranspose2d(64,3)</td>
<td>Conv2d(256,128)</td>
<td></td>
</tr>
<tr>
<td>Dropout</td>
<td>Tanh ConvTranspose2d(64,3)</td>
<td>Tanh ConvTranspose2d(64,3)</td>
<td>BatchNorm2d, LeakyReLU</td>
<td></td>
</tr>
</tbody>
</table>

**Table 2: Classification accuracy on Office-Caltech.**

<table>
<thead>
<tr>
<th>Method</th>
<th>5% A W D C Avg</th>
<th>10% A W D C Avg</th>
<th>20% A W D C Avg</th>
</tr>
</thead>
<tbody>
<tr>
<td>AlexNet ([15])</td>
<td>88.9 73.0 80.4 88.7 82.8</td>
<td>92.2 80.9 88.2 88.9 87.6</td>
<td>91.3 83.3 93.7 94.9 90.8</td>
</tr>
<tr>
<td>MTLF ([1])</td>
<td>90.0 78.9 90.2 86.9 86.5</td>
<td>92.4 85.3 89.5 89.2 89.1</td>
<td>93.5 89.0 95.2 92.6 92.6</td>
</tr>
<tr>
<td>MTFL ([51])</td>
<td>91.3 82.3 88.8 89.1 87.9</td>
<td>92.6 85.2 93.3 87.2 89.6</td>
<td>94.4 87.0 96.7 93.4 92.4</td>
</tr>
<tr>
<td>MTFL ([33])</td>
<td>91.2 88.3 92.5 85.6 89.4</td>
<td>92.2 91.9 97.4 86.8 92.0</td>
<td>92.6 97.6 94.5 88.4 93.3</td>
</tr>
<tr>
<td>RMTL ([32])</td>
<td>91.2 88.3 92.5 85.6 89.4</td>
<td>92.2 91.9 97.4 86.8 92.0</td>
<td>92.6 97.6 94.5 88.4 93.3</td>
</tr>
<tr>
<td>MRN ([20])</td>
<td>92.5 97.5 97.9 87.5 93.5</td>
<td>93.6 98.6 98.6 87.3 94.5</td>
<td>94.4 98.4 99.9 89.1 95.5</td>
</tr>
<tr>
<td>TAGN</td>
<td>92.8 93.9 98.2 91.1 94.0</td>
<td>93.9 95.9 98.8 91.0 94.9</td>
<td>94.8 98.7 98.9 93.8 96.6</td>
</tr>
</tbody>
</table>

\[\text{Leak}\text{yReLU}(x) = \max(0, x) + \lambda \min(0, x)\]
Table 3: Classification accuracy on Office-Home.

<table>
<thead>
<tr>
<th>Method</th>
<th>5%</th>
<th>10%</th>
<th>20%</th>
<th>Avg</th>
<th>5%</th>
<th>10%</th>
<th>20%</th>
<th>Avg</th>
<th>5%</th>
<th>10%</th>
<th>20%</th>
<th>Avg</th>
</tr>
</thead>
<tbody>
<tr>
<td>VGG ([26])</td>
<td>35.8</td>
<td>67.8</td>
<td>49.3</td>
<td>62.5</td>
<td>49.3</td>
<td>67.8</td>
<td>59.8</td>
<td>56.1</td>
<td>54.6</td>
<td>60.4</td>
<td>58.9</td>
<td>65.7</td>
</tr>
<tr>
<td>MTFL ([1])</td>
<td>40.1</td>
<td>70.5</td>
<td>47.9</td>
<td>50.3</td>
<td>35.0</td>
<td>66.3</td>
<td>54.2</td>
<td>55.2</td>
<td>38.8</td>
<td>69.1</td>
<td>50.0</td>
<td>58.8</td>
</tr>
<tr>
<td>RMTL ([5])</td>
<td>42.3</td>
<td>62.3</td>
<td>49.5</td>
<td>50.7</td>
<td>34.6</td>
<td>65.9</td>
<td>53.7</td>
<td>55.2</td>
<td>39.2</td>
<td>69.9</td>
<td>50.5</td>
<td>58.6</td>
</tr>
<tr>
<td>MTRL ([33])</td>
<td>42.7</td>
<td>61.3</td>
<td>50.1</td>
<td>51.6</td>
<td>36.3</td>
<td>67.7</td>
<td>55.5</td>
<td>55.8</td>
<td>39.9</td>
<td>70.2</td>
<td>51.2</td>
<td>59.3</td>
</tr>
<tr>
<td>MTFL ([1])</td>
<td>49.7</td>
<td>34.6</td>
<td>65.9</td>
<td>64.6</td>
<td>53.7</td>
<td>69.9</td>
<td>60.8</td>
<td>58.3</td>
<td>56.1</td>
<td>79.3</td>
<td>72.1</td>
<td>66.3</td>
</tr>
</tbody>
</table>

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:

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

5.5 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:

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

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
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.

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), National 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