Sequential Instance Refinement for Cross-Domain Object Detection in Images
Jin Chen, Xinxiao Wu, Member, IEEE, Lixin Duan, and Lin Chen

Abstract—Cross-domain object detection in images has attracted increasing attention in the past few years, which aims at adapting the detection model learned from existing labeled images (source domain) to newly collected unlabeled ones (target domain). Existing methods usually deal with the cross-domain object detection problem through direct feature alignment between the source and target domains at the image level, the instance level (i.e., region proposals) or both. However, we have observed that directly aligning features of all object instances from the two domains often results in the problem of negative transfer, due to the existence of (1) outlier target instances that contain confusing objects not belonging to any category of the source domain and thus are hard to be captured by detectors and (2) low-relevance source instances that are considerably statistically different from target instances although their contained objects are from the same category. With this in mind, we propose a reinforcement learning based method, coined as sequential instance refinement, where two agents are learned to progressively refine both source and target instances by taking sequential actions to remove both outlier target instances and low-relevance source instances step by step. Extensive experiments on several benchmark datasets demonstrate the superior performance of our method over existing state-of-the-art baselines for cross-domain object detection.

Index Terms—Cross-domain object detection, negative transfer, reinforcement learning.

I. INTRODUCTION

OBJECT detection is one of the most fundamental and challenging tasks in computer vision and has been an active research area for several decades [1]. The goal of object detection is to simultaneously localize and recognize all object instances belonging to the pre-defined categories in an image. It supports many applications, such as autonomous driving [2], [3], intelligent video surveillance [4], [5] and so on. Recently, with the development of deep learning [6], object detection has made remarkable breakthroughs [7]–[17], and achieved superior performances on large benchmark datasets [18], [19].

Despite its recent success, object detection still faces some problems in real-world applications. The deep learning based methods depend heavily on abundant manually labeled data. But for a new task, images are often unlabeled and it is time-consuming and labor-intensive to annotate them. To address this problem, cross-domain object detection is proposed, which aims at improving the detection performance on the unlabeled images (target domain) by leveraging an adaptive detector learned on a fully annotated, different but related domain (source domain). Since there exists a considerable domain shift between the source domain and the target domain due to the statistical differences caused by backgrounds, viewpoints, illumination and object appearances, directly applying object detection model trained on the source domain to the target domain would generally lead to performance degradation [20]. Thus several methods of cross-domain object detection are proposed to reduce the domain shift by learning domain-invariant features [21]–[24]. Among them, [21] is the first work to address the cross-domain object detection task and performs both image-level and instance-level feature alignment to reduce the domain mismatch. Several later works [22]–[24] focus on aligning instance features between the source and target domains to learn an adaptive object detector.

Nevertheless, directly aligning instance features between different domains is often prone to negative transfer, due to the existence of outlier target instances and low-relevance source instances. The outlier target instances refer to target instances containing objects that do not belong to any category of the source domain. The low-relevance source instances refer to source instances that are quite dissimilar to the target ones although they contain objects from the same categories as the target domain. For better understanding, we present several examples of outlier target instances and low-relevance source instances as illustrated in Fig. 1, where the PASCAL VOC dataset is considered as the source domain and the Clipart dataset as the target domain. The outlier target instances and low-relevance source instances are marked with red boxes in Fig. 1(a) and Fig. 1(b), respectively. For example, the bees with red boxes in the upper right corner of Fig. 1(a) are outlier target instances because the “bee” category is not in the category space of the source domain. In the left part of Fig. 1(b), horses with red boxes represent low-relevance source instances that are less similar to the target domain (shown in the right part) due to different appearances, viewpoints, etc.
To tackle these outlier target instances and low-relevance source instances in cross-domain object detection, we propose a reinforcement learning based method, called sequential instance refinement (SIR), under the framework of Faster R-CNN [8]. Our SIR trains two agents (defined as T-agent and S-agent) to progressively refine the source and target instances by taking sequential actions to remove the outlier target instances from the target domain and the low-relevance source instances from the source domain.

Specifically, T-agent is responsible for selecting out outlier target instances from the target domain to enhance the positive transfer. At each selection, T-agent takes an action to remove one target instance according to its Q-value from the target domain and the reward of taking this action is fed back to T-agent to update the selection policy. If the selected target instance is an outlier target instance, the mismatch of data distributions between the source and target domains will be reduced and thus the performance can be improved. It is worth noting that the relevance of outlier target instances to the source domain is low as the label space of source domain is different to that of outlier target instances. So we utilize the relevance of the target instances to the source domain as the reward of T-agent. Similarly, S-agent aims at removing the low-relevance source instances from the source domain to reduce the negative transfer caused by irrelevant source instances. Hence, the reward function of S-agent (resp., T-agent) is defined by a domain classifier that measures the relevance of the source instances to the target domain (resp., the target instances to the source domain).

By means of sequential actions taken by S-agent and T-agent, both target and source instances are refined and then used to train the domain classifier in an adversarial manner. In return, the rewards of the two agents computed by the domain classifier are further calibrated to further improve selection policies. Owing to such a progressive procedure, our method can simultaneously refine the source and target instances by removing unrelated ones, which helps alleviate negative transfer in cross-domain object detection.

We summarize our contributions as follows:
- To the best of our knowledge, we make the first attempt to explicitly address the negative transfer problem in cross-domain object detection through refining both source and target instances.
- We propose sequential instance refinement (SIR) based on reinforcement learning to progressively remove outlier target instances and low-relevance source instances by taking sequential actions of two agents.
- Evaluations on several benchmark datasets demonstrate that our SIR achieves superior performance over the existing state-of-the-art methods.

II. RELATED WORK

A. Cross-Domain Object Detection

Many existing cross-domain object detection methods resort to matching distributions of the image or instance features between the source and target domains [21]–[24], [26]. Chen et al. [21] are the first to tackle the cross-domain object detection and construct two domain classifiers on both image and instance levels to reduce the domain mismatch. Zhu et al. [24] first mine the discriminative regions that are directly pertinent to object detection and then align those regions in the source and target domains to reduce the domain shift. Saito et al. [22] propose weak alignment to focus on the similar part of images and strong alignment to focus on the local fields of the feature map. In [23], the object relationship is integrated into the mean teacher paradigm and the relation graphs between the source and target domains are matched to reduce the domain shift. Hsu et al. [26] first utilize CycleGAN [27] to generate the intermediate domain via translating source images to the target domain and then align the data distributions of the intermediate domain and the target domain via adversarial training at the feature level. Several methods focus on generating pseudo-labeled target data to fine-tune the source detector [28]–[30]. [28] utilizes the tracking information to label the target data and refines
the label of target data with the pseudo label predicted by the source detector. In [29], an object classifier is trained with bounding boxes of the source domain and then is used to label target instances detected by the detector trained on the source domain. Kim et al. [30] introduce a weak self-training method to diminish the effects of inaccurate pseudo-labels and propose an adversarial background score regularization to extract discriminative features for target backgrounds.

Rather than directly matching the data distributions of the source and target domains, we train two agents to select out the outlier target instances and low-relevance source instances, respectively, thus alleviating the negative transfer and improving the detection performance.

B. Domain Adaptation

Domain adaptation leverages the knowledge of the existing labeled domain to enhance the classification performance on the unlabeled but interested domain. Traditional domain adaptation methods can be roughly divided into three categories: instance-based [31], [32], feature-based [33]–[37] and parameter-based domain adaptation methods [38]–[41]. Recently, deep neural networks have made great process due to its strong power in feature learning. Several deep domain adaptation methods are proposed to learn domain-invariant features by minimizing the Maximum Mean Discrepancy (MMD) [42]–[44] or regularizing features by Batch Normalization (BN) layer [45]–[48]. Other methods [25], [49]–[55] introduce the generative adversarial learning [56] for domain adaptation, which aim at making the feature representations effective for handling the negative transfer in cross-domain object detection.

III. PROPOSED METHOD

In unsupervised cross-domain object detection, we are given a labeled source domain \( D_s = \{(x_i^s, y_i^s)\}_{i=1}^{N_s} \) with \( N_s \) images and an unlabeled target domain \( D_t = \{(x_i^t, y_i^t)\}_{i=1}^{N_t} \) with \( N_t \) images, where \( x_i^s \) and \( x_i^t \) represent the \( i \)-th images in \( D_s \) and \( D_t \), respectively. \( y_i^s \) is the set of annotations of objects in \( x_i^s \), i.e., \( y_i^s = \{y_{i1}^s, y_{i2}^s, \ldots, y_{IL}^s\} \), where \( L \) is the number of contained objects and \( y_{ij}^s \) is the corresponding annotation of the \( j \)-th object in \( x_i^s \). Each annotation \( y_{ij}^s \) is formulated as a 5-tuple, i.e., \( y_{ij}^s = (x, y, w, h, r) \in \mathbb{R}^5 \times 1 \), consisting of the category label of the \( j \)-th object, and the coordinates of the upper left corner, height and width of its corresponding bounding box.

In this work, we employ Faster R-CNN [8] as our base detector due to its robustness and flexibility. The region proposals (i.e., object instances) are generated via a region proposal network. Let \( \mathcal{P}_i^s = \{p_{ij}^s\}_{j=1}^{N_p^s} \) represent a set of region proposals in \( x_i^s \), where \( p_{ij}^s \) is the feature map of the \( j \)-th region proposal, and \( N_p^s \) is the number of source proposals. Let \( \mathcal{P}_i^t = \{p_{ij}^t\}_{j=1}^{N_p^t} \) represent a set of region proposals in \( x_i^t \), where \( N_p^t \) is the number of target proposals.

In order to improve the detection performance on the target domain, we aim to select out both outlier target region proposals that contain objects not belonging to any category of the source domain and low-relevance source region proposals that have low relevance to the target domain. We define two agents to make selections via a sequence of actions under a deep reinforcement learning framework. Specifically, T-agent learns to select out target region proposals from \( \mathcal{P}_i^t \) for generating an updated set \( \hat{\mathcal{P}}_i^t \). S-agent learns to select out low-relevance source region proposals from \( \mathcal{P}_i^s \) for generating an updated set \( \hat{\mathcal{P}}_i^s \). Then, we align \( \hat{\mathcal{P}}_i^t \) and \( \hat{\mathcal{P}}_i^s \) to learn domain-invariant features for reducing the domain shift.

The architecture of our SIR is shown in Fig. 2, which consists of a Faster R-CNN as the base detector and a sequential instance refinement module to refine target and source instances by T-agent and S-agent, respectively.

A. Faster R-CNN

Faster R-CNN [8] is a two-stage detector and consists of three major components: shared convolutional layers, a region proposal network (RPN) and a region of interest (RoI) based classifier. The shared convolutional layers firstly extract a feature map of an input image. Then RPN generates a set of region proposals with pre-defined anchor boxes. Finally, the RoI-based classifier predicts categories of those region proposals. The loss of Faster R-CNN is summarized as

\[
\mathcal{L}_{det} = \mathcal{L}_{rpn} + \mathcal{L}_{roi},
\]

where \( \mathcal{L}_{rpn} \) and \( \mathcal{L}_{roi} \) are the training losses of RPN and the RoI-based classifier, respectively. \( \mathcal{L}_{rpn} \) and \( \mathcal{L}_{roi} \) both have two loss terms: a cross-entropy loss about mis-classification error and a regression loss about localization error. The detection and localization in RPN take no account of object categories
and the RoI-based classifier is trained to predict the object categories.

\[ \text{B. Sequential Instance Refinement} \]

Due to the large size of proposal set, we randomly split the proposal set \( \mathcal{P}_i^t \) of each target image \( x_i \) into \( N_c^i \) candidate sets, denoted as \( \mathcal{P}_i^t = \bigcup_{n=1}^{N_c^i} \mathcal{C}_{i,n}^t \) with \( \mathcal{C}_{i,n}^t = \{ p_{i,n,k}^t \}_{k=1}^{N_c^i} \), where \( p_{i,n,k}^t \) represents the \( k \)-th region proposal in the \( n \)-th candidate set of \( x_i \) and is extracted by a RoI pooling layer of Faster R-CNN, and \( N_c \) is the size of candidate set. Similarly, the proposal set \( \mathcal{P}_i^s \) of each source image \( x_i \) is also randomly split into \( N_c^i \) candidate sets, denoted as \( \mathcal{P}_i^s = \bigcup_{n=1}^{N_c^i} \mathcal{C}_{i,n}^s \) with \( \mathcal{C}_{i,n}^s = \{ p_{i,n,k}^s \}_{k=1}^{N_c^i} \), where \( p_{i,n,k}^s \) represents the \( k \)-th proposal in the \( n \)-th candidate set of \( x_i \).

Each episode observed by the agent consists of a sequence of selections for one candidate set. In the selection process for each candidate set \( \mathcal{C}_{i,n}^t \), at time \( e \), T-agent first observes the current state \( s_{e}^t \) and takes an action \( a_{e}^t \) to remove a region proposal from \( \mathcal{C}_{i,n}^t \). Then, T-agent receives the next state \( s_{e+1}^t \) and a reward \( r_{e}^t \) of taking \( a_{e}^t \). When this episode terminates at time \( E \), T-agent completes the selections of \( \mathcal{C}_{i,n}^t \) and the current candidate set can be treated as an optimal candidate set, denoted as \( \hat{\mathcal{C}}_{i,n}^t \). If T-agent finishes selections for all the candidate sets of \( \mathcal{P}_i^t \), we can obtain the optimal target proposal set \( \hat{\mathcal{P}}_i^t = \bigcup_{n=1}^{N_c^i} \hat{\mathcal{C}}_{i,n}^t \) for the target image \( x_i^t \). In a similar way, we can obtain the optimal source proposal set \( \hat{\mathcal{P}}_i^s = \bigcup_{n=1}^{N_c^s} \hat{\mathcal{C}}_{i,n}^s \) for the source image \( x_i^s \).

1) State: Since each agent makes selections on a single candidate set, the state of each agent is defined by the region proposals in the corresponding candidate set. Specifically, the state of T-Agent is represented by the feature vectors of region proposals in a candidate set \( \mathcal{C}_{i,n}^t \), denoted as \( s^t = \{ f_{i,n,1}^t, \cdots, f_{i,n,N_c^t}^t \} \in \mathbb{R}^{d \times N_c^t} \), where \( f_{i,n,k}^t \) is a \( d \)-dimensional feature vector of the region proposal \( p_{i,n,k}^t \). When T-Agent removes the \( k \)-th region proposal \( p_{i,n,k}^t \) from the candidate set \( \mathcal{C}_{i,n}^t \), the corresponding feature vector \( f_{i,n,k}^t \) in \( s^t \) is replaced with a zero-valued feature vector to keep a constant size of \( s^t \). Similarly, the state of S-Agent is represented by \( s^s = \{ f_{i,n,1}^s, \cdots, f_{i,n,N_c^s}^s \} \in \mathbb{R}^{d \times N_c^s} \), where \( f_{i,n,k}^s \) is a \( d \)-dimensional feature vector of the region proposal \( p_{i,n,k}^s \).

2) Action: The actions of T-Agent are denoted by \( A^t = \{ 1, \cdots, N_c^t \} \) where the action \( k \) means selecting out the \( k \)-th proposal \( p_{i,n,k}^t \) from \( \mathcal{C}_{i,n}^t \). The actions of S-Agent are similarly denoted by \( A^s = \{ 1, \cdots, N_c^s \} \), where the action \( k \) means selecting out the \( k \)-th proposal \( p_{i,n,k}^s \) from \( \mathcal{C}_{i,n}^s \).

Since S-Agent and T-Agent make decisions in a similar way, we use \( s_e \in \{ s^t_e, s^s_e \} \) to denote the state of T-Agent or S-Agent, \( a_e \in \{ a^t_e, a^s_e \} \) to denote the action of T-Agent or S-Agent at time \( e \), and \( A \in \{ A^t, A^s \} \) to denote the action set of T-Agent or S-Agent. The agent takes an optimal action \( a_e \) to maximize the accumulated rewards \( R_e = \sum_{e=0}^{E} \gamma^{e-t} r_e \), where \( r_e \) is the immediate reward of taking action \( a_e \) under state \( s_e \), \( E \) is the terminal time of an episode, and \( \gamma \) is a discount factor. We apply a deep Q-learning network (DQN) to estimate the accumulated rewards by learning the action-value function \( Q(s_e, a_e) \). The agent takes an action \( a_e \) from \( A \) to get the maximum accumulated rewards via a policy defined by

\[ a_e^* = \max_{a_e} Q(s_e, a_e). \]  

3) Reward: For T-Agent, the reward of taking action \( a_e^t \) is determined by the relevance of the selected target proposal \( p_{i,n,a_e^t}^t \) to the source domain since the outlier target instances are less relevant to the source domain than the other target ones. Similarly, for S-Agent, the reward of taking action \( a_e^s \) is determined by the relevance of the selected source proposal \( p_{i,n,a_e^s}^s \) to the target domain.
So we propose a domain classifier whose classification score can be used to measure the relevance of the target proposal \( p_{i,n,a^t} \) to the source domain or the relevance of the source proposal \( p_{i,n,a^t} \) to the target domain. Specifically, we adopt a patch-based domain classifier \( D \) that predicts multiple domain labels for each pixel of a region proposal. Let \( W \) and \( H \) denote the width and height of a region proposal, respectively. The output of \( D \) is a domain prediction map with the size of \( W \times H \), where \( D(p_{i,n,k}^t(w,h)) \) and \( D(p_{i,n,k}^s(w,h)) \) denote domain predictions of the pixel \((w, h)\) of region proposals \( p_{i,n,k}^t \) and \( p_{i,n,k}^s \), respectively. Given the optimal region proposal sets of \( \hat{P}_n^t \) and \( \hat{P}_n^s \), we utilize a least-squares loss to train the domain classifier by following [65], which is formulated as

\[
\mathcal{L}_{grad} = \sum_{i,n,k,w,h} D^2(p_{i,n,k}^t(w,h)) + \sum_{i,n,k,w,h} (1 - D(p_{i,n,k}^s(w,h)))^2
\]  

(3)

When \( D(p_{i,n,k}^t(w,h)) = 1 \), the pixel \((w, h)\) of \( p_{i,n,k}^t \) is predicted as being from the target domain. When \( D(p_{i,n,k}^s(w,h)) = 0 \), the pixel \((w, h)\) of \( p_{i,n,k}^s \) is predicted as being from the source domain. We employ a gradient reverse layer (GRL) [66] to conduct the adversarial training between \( D \) and the backbone network of Faster R-CNN. Specifically, \( D \) is trained by the ordinary gradient descent to minimize \( \mathcal{L}_{adv} \), and the backbone network of Faster R-CNN is updated with the gradient whose sign is reversed through the GRL layer to maximize \( \mathcal{L}_{adv} \).

With the output of \( D \), the relevance measure function \( \varphi(p) \) is formulated as

\[
\varphi(p) = \begin{cases} 
1 & \frac{1}{W \times H} \sum_{(w,h)} D(p(w,h)), \quad p \in \hat{P}_n^t \\
1 - \frac{1}{W \times H} \sum_{(w,h)} D(p(w,h)), \quad p \in \hat{P}_n^s 
\end{cases}
\]  

(4)

where \( \frac{1}{W \times H} \sum_{(w,h)} D(p(w,h)) \) is the average domain predictions of all pixels of region proposal \( p \). The larger value of \( \varphi(p) \) means that the region proposal \( p \) is more similar to the opposite domain.

With the relevance measure function \( \varphi(p) \), the reward of action \( a^e_\epsilon \) is

\[
r^e_\epsilon = \begin{cases} 
1, & \varphi(p_{i,n,a^t}) < \tau \\
-1, & \text{otherwise}
\end{cases}
\]  

(5)

where the action \( a^t_\epsilon \) corresponds to removing the target region proposal \( p_{i,n,a^t} \) from \( C^t_{i,n} \), and \( \tau \) is a threshold. As \( \varphi(p_{i,n,a^t}) < \tau \) means that the selected target region proposal \( p_{i,n,a^t} \) is less relevant to the source domain, a positive reward is given to T-agent. The reward of action \( a^s_\epsilon \) is

\[
r^s_\epsilon = \begin{cases} 
1, & \varphi(p_{i,n,a^s}) < \tau \\
-1, & \text{otherwise}
\end{cases}
\]  

(6)

where action \( a^s_\epsilon \) corresponds to removing source region proposal \( p_{i,n,a^s} \) from \( C^s_{i,n} \). When S-agent takes an action of selecting out a region proposal with lower relevance to the target domain, i.e., \( \varphi(p_{i,n,a^s}) < \tau \), a positive reward is given to S-agent. For both S-agent and T-agent, we quantify the reward to 1 and -1 to help the agent clearly distinguish good or bad actions.

4) Loss Function Based on Q-Values: The DQN is trained with the temporal difference error, formulated as

\[
\mathcal{L}_q = \mathbb{E}_{s_e, a_e} \left[ (V(s_e) - Q(s_e, a_e))^2 \right],
\]  

(7)

where \( Q(s_e, a_e) \) is the output Q-value of action \( a_e \) under the current state \( s_e \). \( V(s_e) \) is the target value of \( Q(s_e, a_e) \), given by

\[
V(s_e) = \mathbb{E}_{s_{e+1}} \left[ r_e + \gamma \max_{a_{e+1}} Q(s_{e+1}, a_{e+1}) \right],
\]  

(8)

where the first term \( r_e \) is the immediate reward of taking the action \( a_e \) at time \( e \), the second term is the future reward estimated by the current deep Q-learning network with the next state \( s_{e+1} \) as input at time \( e + 1 \), and \( \gamma \) is a discount factor of reward.

C. Training

With the detection loss \( \mathcal{L}_{det} \) in Eq. (1), adversarial loss \( \mathcal{L}_{adv} \) in Eq. (3) and deep Q-learning loss \( \mathcal{L}_q^s \) and \( \mathcal{L}_q^t \) for the two agents from both domains as in Eq. (7), the overall objective function is given by

\[
\mathcal{L} = \mathcal{L}_{det} + \mathcal{L}_{adv} + \mathcal{L}_q^s + \mathcal{L}_q^t.
\]  

(9)

We use the \( \epsilon \)-greedy strategy [67] and the experience replay strategy [68] to train S-agent and T-agent. Specifically, the \( \epsilon \)-greedy strategy is used to balance the exploration and exploitation of an agent, which refers to that the agent has a certain probability to perform random actions. The selection policy of the agent (defined in Eq. (2)) is then rewritten by

\[
a^e_\epsilon = \begin{cases} 
\max_a Q(s_e, a_e), & \text{if } \lambda \geq \epsilon \\
a^t_\epsilon, & \text{otherwise}
\end{cases}
\]  

(10)

where \( \epsilon \) represents the probability of the agent to perform exploration and \( \lambda \) is a random variable drawn from \([0, 1]\). When \( \lambda \geq \epsilon \), the agent takes actions by Eq. (2). Otherwise, the agent performs exploration by taking a randomly action \( a^t \) from the action set \( A \), which can expand the solution space and avoid falling into a local optimal solution. The training of DQN requires data to be independent and identically distributed (i.i.d.) while the data obtained in the training process is strongly correlated sequentially. Hence, the experience replay strategy [68] is exploited to break the correlation between samples, which stores experiences in the experience pool and samples from the experience pool when updating the model. The whole training process of our SIR is summarized in Algorithm 1.

D. Discussion

In this paper, we investigate alleviating the negative transfer in cross-domain object detection via selecting out the outlier target instances and the low-relevance source instances. A reinforcement learning paradigm is applied to automatically learn policies for selecting out the two types instances. We remark the advantages of reinforcement learning as follows. First, the policies learned by reinforcement learning is optimized in a sequential decision process with the guidance of the
Algorithm 1: Sequential Instance Refinement

Input: Labeled source domain \( D_s = \{(x^s_i, y^s_i)\}_{i=1}^{N_s} \);
Unlabeled target domain \( D_t = \{(x^t_i)\}_{i=1}^{N_t} \);
Tradeoff parameters: \( \gamma, \tau, \epsilon \);
The size of candidate set: \( N_c \);
Terminal time of one episode: \( E \).

Output: Adaptive Faster R-CNN.

1. Initialize the experience pools of T-agent and S-agent \( M^t = \emptyset, M^s = \emptyset \);
2. while not converge do
   3. Generate the proposal set \( P^t_i \);
   4. Split \( P^t_i \) to \( N_c \) candidate sets \( C^t_{i,n} \);
   5. for \( n \leftarrow 1 \) to \( N_c^t \) do
      6. Generate the initial state \( s^t_{i,n} \) with \( C^t_{i,n} \);
      7. repeat
         8. Take an action \( a^t_{i,n} \) by Eq.(10);
         9. Remove \( p_{i,n,a^t_{i,n}} \) from \( C^t_{i,n} \);
        10. Generate the next state \( s^t_{i,n+1} \);
        11. Compute the reward \( r^t_{i,n} \) by Eq.(5);
        12. Insert \( (s^t_{i,n}, a^t_{i,n}, s^t_{i,n+1}, r^t_{i,n}) \) into \( M^t \);
        13. until Terminal time \( E \);
        14. Obtain the updated candidate set \( \tilde{C}^t_{i,n} \);
   15. Obtain the updated proposal set \( \tilde{P}^t_i \);
   16. Generate the proposal set \( P^t_i \);
   17. Split \( P^t_i \) to \( N_c^t \) candidate sets \( C^t_{i,n} \);
   18. for \( n \leftarrow 1 \) to \( N_c^t \) do
      19. Generate the initial state \( s^s_{i,n} \) with \( C^t_{i,n} \);
      20. repeat
         21. Take an action \( a^s_{i,n} \) by Eq.(10);
         22. Remove \( p_{i,n,a^s_{i,n}} \) from \( C^t_{i,n} \);
         23. Generate the next state \( s^s_{i,n+1} \);
         24. Compute the reward \( r^s_{i,n} \) by Eq.(5);
         25. Insert \( (s^s_{i,n}, a^s_{i,n}, s^s_{i,n+1}, r^s_{i,n}) \) into \( M^s \);
         26. until Terminal time \( E \);
         27. Obtain the updated candidate set \( \tilde{C}^s_{i,n} \);
   28. Obtain the updated proposal set \( \tilde{P}^s_i \);
   29. end
   30. Compute \( \mathcal{L}_{det} \) by Eq.(1) with \( P^t_i \) and \( P^s_i \);
   31. Compute \( \mathcal{L}_{adv} \) by Eq.(3) with \( P^t_i \) and \( P^s_i \);
   32. Compute \( \mathcal{L}^t_i \) and \( \mathcal{L}^s_i \) by Eq.(7) with sampling experiences from \( M^t \) and \( M^s \), respectively;
   33. Update model by Eq.(9);

accumulated rewards since the output of DQN represents both the immediate and future rewards. In this way, the agent learns to make selection at the set level, which is more correct compared with making selection based on the relevance measure function at the instance level. Second, reinforcement learning does not only make use of the learned relevance information but also explores in a wider space to find better solutions since the agent has \( \epsilon \) probability to take actions of small Q-values for conducting the exploration. Therefore, the agent can accumulate more rich experience and has ability to jump out the local optimum. For example, although some source instances have high relevance to the target domain, the high relevance is caused by similar scenes not by similar objects, and these instances should be removed to avoid the negative transfer. In this case, SIR can search such instances in a wide space and select out them to avoid the negative transfer while the method of selecting based on the learned relevance information cannot select out them due to the high relevance to the target domain.

IV. EXPERIMENTS

A. Datasets

To evaluate the effectiveness of our method, we conduct experiments using five image datasets as follows:

- The PASCAL VOC dataset [19] is a standardised image dataset for object detection, which has been created and maintained for many years (from 2005-2012). This dataset contains 20 object categories including “aeroplane”, “bicycle”, “bird”, “boat”, “bus”, “car”, “cat”, “chair”, “cow”, “table”, “dog”, “horse”, “motorbike”, “person”, “plant”, “sleep”, “sofa”, “train”, and “tv”. PASCAL VOC 2007 has 24,640 annotated instances in 9,963 images, and PASCAL VOC 2012 has 27,450 annotated instances in 11,530.

- The Clipart dataset [69] is a comical image dataset, which is collected from the CMPlaces dataset [70] and two image search engines (Openclipart1 and Pixabay2). The Clipart dataset consists of 1,000 images in total with the same 20 object categories as the PASCAL VOC dataset.

- The Watercolor dataset [69] is an artistic dataset, which is from the BAM! [71] dataset and includes six object categories: “bicycle”, “bird”, “cat”, “car”, “dog”, and “person”. The watercolor dataset contains 2,000 images with 3,315 annotated instances.

- The SIM 10K dataset [72] is a simulated dataset, which is rendered from the computer game Grand Theft Auto V. Images in this dataset are captured by a dash-cam under car driving scenes. There are 10,000 synthetic images with 58,701 annotated car instances.

- The Cityscapes dataset [73] is a benchmark dataset for instance segmentation with pixel-level annotations, which is captured by a dash-cam in urban street scenes. It has 2,975 images in the training set and 500 images in the validation set, covering eight object categories. Following [21], we use the tightest rectangles of each instance segmentation mask to generate the bounding box annotations.

- The Foggy Cityscapes dataset [74] is a collection of synthetic foggy images, which simulates fog on real scenes and is generated from Cityscapes by adding fog noise.

- The KITTI dataset [75] is a real dataset, which contains 7,481 images. In this dataset, images have original resolution of 1250 × 375.

1. https://openclipart.org/
2. https://pixabay.com/
Five settings for cross-domain object detection are constructed in our experiments as follows:

- **PASCAL VOC→Clipart (P→C):** The training and validation splits in the PASCAL VOC 2007 and PASCAL VOC 2012 datasets have totally 15,000 images that are used as the source domain. The Clipart dataset is used as the target domain. Since the number of source images is much larger than that of target images, all the images in the Clipart dataset are used for training (without labels) and evaluation, following [22], [30].

- **PASCAL VOC→Watercolor (P→W):** We use the training and validation splits of the PASCAL VOC 2007 and PASCAL VOC 2012 datasets as the source domain and the Watercolor dataset as the target domain, where the six common categories between the source and target domains are used. The training images of the Watercolor dataset (1,000 images) are used during training (without labels), and we evaluate our model on the test split of the Watercolor dataset, following [22], [30].

- **SIM 10K→Cityscapes (S→Ci):** We use the SIM 10K dataset as the source domain and the Cityscapes dataset as the target domain. Both the training images in the source and target domains are used for training and the validation split of the Cityscapes dataset is used for evaluating our model. Since the SIM 10K dataset only has annotations for cars, we evaluate the detection performance on the “car” category following [21], [22].

- **Cityscapes→Foggy Cityscapes (Ci→F):** We use the Cityscapes dataset as the source domain and the Foggy Cityscapes dataset as the target domain. The training sets of the two datasets are used for training and the validation set of the Foggy Cityscapes dataset is used for evaluation following [76].

- **KITTI→Cityscapes (K→Ci):** We use the KITTI dataset as the source domain and the Cityscapes dataset as the target domain. The training sets of the two datasets are used for training and the validation set of the Cityscapes dataset is used for evaluation. We report the results on the common “car” category following [21], [76].


Note that our work focuses on the unsupervised cross-domain object detection [21], where only the annotations of source images are provided when training, and the annotations of target images are only used for evaluation. The detailed settings are summarized in TABLE I.

### TABLE I

<table>
<thead>
<tr>
<th>Setting</th>
<th>Source domain</th>
<th>Target domain</th>
<th>Evaluation data</th>
</tr>
</thead>
<tbody>
<tr>
<td>PASCAL VOC→Clipart (P→C)</td>
<td>Train and validation split (15,000 images)</td>
<td>All 1,000 images</td>
<td>Test split (1,000 images)</td>
</tr>
<tr>
<td>PASCAL VOC→Watercolor (P→W)</td>
<td>Train and validation split (15,000 images)</td>
<td>Train split (1,000 images)</td>
<td>Validation split (500 images)</td>
</tr>
<tr>
<td>SIM 10K→Cityscapes (S→Ci)</td>
<td>Train split (10,000 images)</td>
<td>Train split (2,975 images)</td>
<td>Validation split (500 images)</td>
</tr>
<tr>
<td>Cityscapes→Foggy Cityscapes (Ci→F)</td>
<td>Train split (2,975 images)</td>
<td>Train split (2,975 images)</td>
<td>Validation split (500 images)</td>
</tr>
<tr>
<td>KITTI→Cityscapes (K→Ci)</td>
<td>Train split (7,481 images)</td>
<td>Train split (2,975 images)</td>
<td>Validation split (500 images)</td>
</tr>
</tbody>
</table>

B. Implementation Details

The domain classifier $D$ is constructed by three convolution layers ($512 \rightarrow 128 \rightarrow 1$) with the kernel size of 1. The first two layers are activated by the LeakyReLU function, and the last layer is activated by the Sigmoid function. Both architectures of S-agent and T-agent are built with three full-connected layers ($1024 \rightarrow 512 \rightarrow 16$) by using ReLU as the activation function. For both T-agent and S-agent, the size of each candidate is set to $N_c = 16$, and the number of proposals to be selected out from each candidate set (i.e., the terminal time $E$ of each episode) is set to 3. The discount factor $\gamma$ in Eq.(8) is set to 0.9 followed by [63], [77], and the probability of exploration $\epsilon$ in Eq.(10) is decayed from 0.9 to 0.01 during training. Moreover, we set the threshold $r$ in the reward function as 0.5. Following Faster R-CNN [8], we resize the shorter side of each image to 600 by preserving its aspect ratio. We train the overall network with a learning rate of 0.001 for the first 50,000 iterations and reduce the learning rate to 0.0001 for the rest 50,000 iterations. Each batch consists of one source image and one target image. Following [66], the learning rate ratio of domain classifier to the backbone of Faster R-CNN is set as 10 : 1, i.e., setting the parameter of GRL layer as 0.1.

For the P→C and P→W settings, we adopt the ImageNet pre-trained ResNet101 [78] as the backbone of Faster R-CNN by following [22]. For the S→Ci setting, we adopt both VGG16 [79] and ResNet50 [78] as the backbones of Faster R-CNN. For the Ci→F and K→Ci settings, ResNet50 [78] is used as the backbone of Faster R-CNN by following [76]. For evaluation, both per-category and mean average precisions (mAP) with a threshold of 0.5 are reported for all the settings.

C. Results

We compare our SIR with three existing methods for cross-domain object detection on all the settings with the same backbones: (1) Faster R-CNN [8] is trained on the source domain and directly applied to the target domain without any adaptation. (2) [21] is the first work for cross-domain object detection and reduces the domain shift both on image-level and instance-level. (3) [22] makes strong alignment on the local features of the source and target domains and performs weak alignment on the global features of the two domains. [21] and [22] both utilize Faster R-CNN as the base detection network. We report the results of [21] and [22] from [22]. With ResNet50 as the backbone network, we directly copy the results of [21], [22], [76] from [76].

TABLE II, TABLE III, TABLE IV, TABLE V and TABLE VI show results on the P→C, P→W, S→Ci, Ci→F and K→Ci settings, respectively. From the results, we have the following observations:
• SIR outperforms all the compared methods on the five settings, clearly demonstrating the effectiveness of sequential instance refinement for cross-domain object detection. In particular, SIR substantially promotes the performance on difficult settings, e.g., improving Faster R-CNN with a gain of 14.8% on the P→C setting, where the domain shift is serious due to the large variances in object styles, backgrounds, and viewpoints.

• SIR performs much better than [21] and [22] on both P→C and P→W settings. This is probably because our SIR can successfully refine both source and target instances by removing outlier target instances and low-relevance source instances, relieving the negative transfer and thereby enhancing the detection performance.

• The improvement of SIR on the S→Ci setting is lower than that on other settings. The possible reason is that object categories in the SIM 10K dataset are similar to the Cityscapes dataset and there is a small variance in the appearance of cars in different domains. Hence, there are few outlier target instances and low-relevance source instances. In this situation, our SIR can still outperform the state-of-the-art method [76] defined in Eq. (4) (denoted as “SIR-relevance”). In “SIR w/o S&T-agents”, all the source and target instances are utilized for empirically evaluating the importance of each individual component. We compare SIR with five variants summarized in TABLE VII: without S-agent (denoted as “SIR w/o S-agent”), without T-agent (denoted as “SIR w/o T-agent”), without S-agent and T-agent (denoted as “SIR w/o S&T-agents”), replacing the patch-based domain classifier with a standard domain classifier (denoted as “SIR-stanD”), and selecting instances directly based on the relevance measure function as defined in Eq. (4) (denoted as “SIR-relevance”). In “SIR w/o S&T-agents”, all the source and target instances are utilized for the adversarial training between the backbone network of Faster R-CNN and the domain classifier by minimizing \( L_{adv} \) defined in Eq. (3). The results of ablation study on the P→C setting are shown in TABLE VIII.

D. Ablation Study

To analyze our method in depth, ablation study is conducted for empirically evaluating the importance of each individual component. We compare SIR with five variants summarized in TABLE VII: without S-agent (denoted as “SIR w/o S-agent”), without T-agent (denoted as “SIR w/o T-agent”), without S-agent and T-agent (denoted as “SIR w/o S&T-agents”), replacing the patch-based domain classifier with a standard domain classifier (denoted as “SIR-stanD”), and selecting instances directly based on the relevance measure function as defined in Eq. (4) (denoted as “SIR-relevance”). In “SIR w/o S&T-agents”, all the source and target instances are utilized for the adversarial training between the backbone network of Faster R-CNN and the domain classifier by minimizing \( L_{adv} \) defined in Eq. (3). The results of ablation study on the P→C setting are shown in TABLE VIII.

1) Effect of S-Agent and T-Agent: From the results shown in TABLE VIII, “SIR w/o S-agent” and “SIR w/o T-agent” work worse than SIR with a drop of 2% and 0.8% in terms of mAP, respectively, which validates that both S-agent and T-agent can contribute to instance refinement for improving the cross-domain object detection performance.
“SIR w/o S-agent” and “SIR w/o T-agent” work better than “SIR w/o S&T-agents” by gains of 1.9% and 3.1% in terms of mAP, respectively, which shows that both source and target instances need refinement to alleviate the negative transfer.

2) Effect of the Patch-Based Domain Classifier: To evaluate the patch-based domain classifier, we compare SIR with SIR-stanD in TABLE VIII. SIR outperforms SIR-stanD with a gain of 7.1%, possibly due to the fact that the patch-based domain classifier provides a robust relevance measure by averaging the classification scores of all the pixels in the region proposal. Moreover, from the results in TABLE VIII and TABLE II, “SIR w/o S&T-agents” achieves 10.9% improvement over the source only method (“Faster R-CNN” in TABLE II), clearly demonstrating the effectiveness of performing adaptation between different domains. Moreover, “SIR w/o S&T-agents” outperforms [21], [22], showing that it is beneficial to perform fine-grained alignment at the instance-level via the patch-based domain classifier.

3) Effect of Reinforcement Learning: To evaluate the effect of reinforcement learning, we compare SIR with “SIR w/o S&T-agents” and SIR-relevance. The difference between “SIR w/o S&T-agents” and SIR is whether handling the negative transfer by selecting out the outlier target instances and the low-relevance source instances. From the results in TABLE VIII, SIR outperforms “SIR w/o S&T-agents” with an improvement of 3.9%, which demonstrates the benefit of alleviating the negative transfer by selecting out the outlier target instances and the low-relevance source instances. As shown in TABLE VIII, SIR outperforms SIR-relevance by 2.8% since the instance selection in SIR is optimized based on a sequential decision procedure, where both the immediate and future rewards are considered. That is to say, the decisions of SIR are made according to the accumulated rewards, while the decisions of SIR-relevance only consider the immediate rewards. For example, in the leftmost column of Fig. 3, SIR-relevance wrongly selects out the source instance (in red box) as low-relevance source instance according to the low immediate reward, ignoring that this instance contains a horse similar to the target horses. In contrast, SIR does not select out this source instance by taking into account future rewards.

E. Statistical and Divergence Analysis

To illustrate the importance of sloving the negative transfer in cross-domain object detection task, we conduct statistics on outlier target instances and low-relevance source instances on all settings to quantify the severity of negative transfer. The statistical results are shown in TABLE IX. From the results, it is noteworthy that outlier target instances and low-relevance source instances exist in each experiment, clearly confirming the significance of addressing the negative transfer. For the P→C and P→W settings, there are many outlier target instances due to the large domain gap between the target domain and the source domain. For the Ci→F setting, when adding foggy noise on images, the appearances of objects are changed, leading to more low-relevance source instances.

To further evaluate the effectiveness of SIR in handling negative transfer, we make the statistical analysis of the number of selected outlier target instances and selected low-relevance source instances on the P→C setting. The statistic results are shown in TABLE X. From the results, it is noteworthy that SIR can select out more outlier target instances and more low-relevance source instances than SIR-relevance, which shows the effectiveness of reinforcement learning in instance refinement. Moreover, SIR achieves better selection results than SIR-relevance, especially in some categories, such as “bird”, “bottle”, “cow”, “horse” and so on. The reason is that objects of those categories with different viewpoints have large variances in appearance representation and the immediate rewards may not be sufficient to take actions right. In those cases, SIR works better by considering accumulated rewards than SIR-relevance that independently considers the relevance of instance. For some other categories, e.g., “pottedplant (plnt)” and “train”, the detection precisions are high in TABLE II, showing that the variance in appearances is relatively small. So the immediate reward may be sufficient to take actions right, and SIR-relevance works well.

Moreover, we also conduct the divergence analysis on the P→C setting. Specifically, we compute the Jensen-Shannon divergence (JS divergence) between the source and target domains, including the JS divergence between all source and target instances (denoted as $J_{S/all}$), the JS divergence between the refined source instances and all the target instances

<table>
<thead>
<tr>
<th>Method</th>
<th>person</th>
<th>rider</th>
<th>car</th>
<th>truck</th>
<th>bus</th>
<th>train</th>
<th>motorcycle</th>
<th>bicycle</th>
<th>mAP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Faster-R-CNN [8]</td>
<td>26.9</td>
<td>35.2</td>
<td>35.6</td>
<td>18.3</td>
<td>32.4</td>
<td>9.6</td>
<td>25.8</td>
<td>28.6</td>
<td>26.9</td>
</tr>
<tr>
<td>Chen et al. [21]</td>
<td>29.2</td>
<td>40.4</td>
<td>43.4</td>
<td>19.7</td>
<td>38.3</td>
<td>28.5</td>
<td>23.7</td>
<td>32.7</td>
<td>32.0</td>
</tr>
<tr>
<td>Saito et al. [22]</td>
<td>31.6</td>
<td>44.3</td>
<td>48.9</td>
<td>21.0</td>
<td>43.8</td>
<td>28.0</td>
<td>28.9</td>
<td>35.8</td>
<td>35.3</td>
</tr>
<tr>
<td>Xu et al. [76]</td>
<td>32.9</td>
<td>46.7</td>
<td>54.1</td>
<td>24.7</td>
<td>45.7</td>
<td>41.1</td>
<td>32.4</td>
<td>38.7</td>
<td>39.5</td>
</tr>
<tr>
<td>SIR w/o S&amp;T-agents</td>
<td>25.3</td>
<td>44.1</td>
<td>45.0</td>
<td>25.1</td>
<td>47.1</td>
<td>38.6</td>
<td>24.8</td>
<td>33.9</td>
<td>35.5</td>
</tr>
<tr>
<td>SIR</td>
<td>33.0</td>
<td>45.6</td>
<td>52.3</td>
<td>30.5</td>
<td>50.8</td>
<td>40.9</td>
<td>32.6</td>
<td>35.9</td>
<td>40.2</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Method</th>
<th>AP on car</th>
</tr>
</thead>
<tbody>
<tr>
<td>Faster R-CNN [8]</td>
<td>37.6</td>
</tr>
<tr>
<td>Chen et al. [21]</td>
<td>41.8</td>
</tr>
<tr>
<td>Saito et al. [22]</td>
<td>43.2</td>
</tr>
<tr>
<td>Xu et al. [76]</td>
<td>47.9</td>
</tr>
<tr>
<td>SIR w/o S&amp;T-agents</td>
<td>43.5</td>
</tr>
<tr>
<td>SIR</td>
<td>46.5</td>
</tr>
</tbody>
</table>
Fig. 3. Examples of selection results on the source domain of the P→C setting. Bounding boxes denote the instances that are selected to be removed from the source domain. Yellow boxes indicate correctly selected source instances, and red boxes indicate wrongly selected source instances. Low-relevance source instances selected by SIR-relevance and S-agent in SIR are shown in (a) and (b), respectively.

Fig. 4. Examples of selection results on the target domain of the P→C setting. Bounding boxes denote the instances that are selected to be removed from the target domain. Yellow boxes indicate correctly selected target instances, and red boxes indicate wrongly selected target instances. Outlier target instances selected by SIR-relevance and T-agent in SIR are shown in (a) and (b), respectively.

TABLE VII
FIVE VARIANTS OF SIR

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SIR w/o S-agent</td>
<td>Without selecting low-relevance source instances</td>
</tr>
<tr>
<td>SIR w/o T-agent</td>
<td>Without selecting outlier target instances</td>
</tr>
<tr>
<td>SIR w/o S&amp;T-agents</td>
<td>Without making selection and directly performing adaptation</td>
</tr>
<tr>
<td>SIR-stdD</td>
<td>Replacing the patch-based domain classifier with a standard domain classifier</td>
</tr>
<tr>
<td>SIR-relevance</td>
<td>Making selections according to the relevance measure function defined in Eq.(4)</td>
</tr>
</tbody>
</table>

TABLE VIII
ABLATION STUDY OF SIR FROM THE PASCAL VOC DATASET TO THE CITYSCAPES DATASET (P→C)

<table>
<thead>
<tr>
<th>Method</th>
<th>mAP (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>SIR w/o S-agent</td>
<td>40.6</td>
</tr>
<tr>
<td>SIR w/o T-agent</td>
<td>41.8</td>
</tr>
<tr>
<td>SIR w/o S&amp;T-agents</td>
<td>38.7</td>
</tr>
<tr>
<td>SIR-stdD</td>
<td>35.5</td>
</tr>
<tr>
<td>SIR-relevance</td>
<td>39.8</td>
</tr>
<tr>
<td>SIR</td>
<td>42.6</td>
</tr>
</tbody>
</table>

(2) (denoted as $JS_{refine}^s$), the JS divergence between all the source instances and the refined target instances (denoted as $JS_{refine}^t$), and the JS divergence between the refined source and target instances (denoted as $JS_{refine}^{st}$). The results are shown in TABLE XI. From the results, we can have the following observations. First, $JS_{refine}^s < JS_{all}$ indicates that the refined source instances are closer to the target domain, demonstrating that s-agent can select out low-relevance source instances to avoid the negative transfer. Second, $JS_{refine}^t < JS_{all}$ shows that after the selection conducted by t-agent, the outlier target instances are removed from the target domain, demonstrating the effectiveness of the t-agent. Third, $JS_{refine}^{st} < JS_{all}$, which shows that the refined source and target instances are closer to each other and the instance refinements in both the source and target domains facilitate the positive transfer.

F. Qualitative Evaluation

Fig. 3 and Fig. 4 show the selection comparisons between SIR and SIR-relevance on the P→C setting. As shown
in Fig. 3(b), we find that S-agent removes more low-relevance source instances than SIR-relevance. In addition, some instances containing similar objects to the target ones are wrongly selected by SIR-relevance such as the horse in the rightmost column of Fig. 3(a). The reason is probably that the person on the horse has a different pose to the target ones and the immediate reward is small. In contrast, SIR does not select out this horse by considering accumulated rewards, which validates that the agent can make more correct decisions than independently considering the relevance of instance.

The selection results of T-agent are shown in Fig. 4. We can find that T-agent correctly removes more outlier target instances from the target domain than SIR-relevance, such as fox in the rightmost column of Fig. 4(b). Since the appearance of fox is similar to that of cat in the source domain, resulting in the low immediate reward, SIR-relevance does not filter out this fox. In contrast, SIR successfully selects out the fox with the exploration ability of reinforcement learning.

To go deeper with the effectiveness of sequential instance refinement, we visualize the selection processes of S-agent and T-agent on the P→C setting. In Fig. 5, we show two examples of the selection processes of S-agent for the “horse” category. Since the goal of S-agent is selecting out low-relevance source instances, we present some object examples (horses) from the target domain (the Clipart dataset) in the leftmost column for comparison. The second column shows the input image, where the low-relevance source instances are marked with yellow boxes. The following three columns show 24 proposals that are selected from 128 proposals of the input image by S-agent at epoch 1, 3 and 5. We use the same color box to denote the same object. The rightmost column demonstrates the selection results at epoch 10. From the results, it is worth noting that S-agent selects out increasing low-relevance source instances with the increasing epoch. Specifically, in Fig. 5(a), at epoch 1, S-agent selects out two horses with different viewpoints to the target ones. At epoch 3 and 5, with the accumulated experience, S-agent selects out more horses that are heavily occluded.

In Fig. 6, we visualize two examples of the selection processes of T-agent at epoch 1, 3 and 5 in the target domain (the Clipart dataset). The leftmost column shows the object categories of the source domain (the PASCAL VOC dataset). The second column shows the input image, where instances with yellow boxes are outlier target instances. The following three columns show the selected proposals by T-agent at epoch 1, 3 and 5, where we denote all the proposals of the same object with the same color boxes. For example, all the proposals of the bigger dolphin in the input image of Fig. 6(b) are marked with red boxes. From the results, T-agent progressively selects out all the outlier target instances with the

**TABLE X**

The statistics of the selected outlier target instances and low-relevance source instances on the P→C setting.

<table>
<thead>
<tr>
<th>Method</th>
<th>#Outlier</th>
<th>#Low-relevance</th>
</tr>
</thead>
<tbody>
<tr>
<td>SIR-relevance</td>
<td>1542</td>
<td>3263</td>
</tr>
<tr>
<td>SIR</td>
<td>1724</td>
<td>2386</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Examples in the target domain</th>
<th>Input image</th>
<th>Epoch 1</th>
<th>Epoch 3</th>
<th>Epoch 5</th>
<th>Selection results</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Fig. 5. Two example of the selection processes of S-agent in SIR for the “horse” category on the P→C setting. The leftmost column represents several horses in the target domain (the Clipart dataset) for comparison. In the second column, the low-relevance source instances of the input image are marked with yellow boxes, which are less relevant to the target domain and should be removed. In the following three columns, we present 24 proposals selected out by S-agent at epoch 1, 3 and 5, and mark all proposals of the same object with the same color boxes. The rightmost column represents the final selection results of the input image at epoch 10.

<table>
<thead>
<tr>
<th>Examples in the target domain</th>
<th>Input image</th>
<th>Epoch 1</th>
<th>Epoch 3</th>
<th>Epoch 5</th>
<th>Selection results</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**TABLE XI**

The Jensen-Shannon divergence between domains on the P→C setting.

<table>
<thead>
<tr>
<th>JS\textsubscript{all}</th>
<th>JS\textsubscript{refine} \textsuperscript{r}</th>
<th>JS\textsubscript{refine}</th>
<th>JS\textsubscript{refine} \textsuperscript{r}</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.2635</td>
<td>0.2628</td>
<td>0.2619</td>
<td>0.2618</td>
</tr>
</tbody>
</table>
setting to evaluate the effect of the threshold in Table XII. From the results, it is noteworthy that the

(b) The selection process of T-agent on an image of a dolphin. The leftmost column shows the object categories of the source domain (the PASCAL VOC dataset). In the second column, the outlier target instances of the input image are marked with yellow boxes, which does not belong to any category of the source domain and should be removed. In the following three columns, we present 24 proposals selected out by T-agent at epoch 1, 3 and 5, and mark all proposals of the same object with the same color boxes. The rightmost column shows the final selection results of the input image at epoch 10.

![Selection process of T-agent on an image of a dolphin.](image)

increasing epoch. Concretely, as shown in Fig. 6(a), at epoch 1, two deers are selected out by T-agent. Since the fox is more similar to the dog in the source domain and the squirrel is more similar to the cat in the source domain, instances of fox and squirrel are not selected out by T-agent at first. It is interesting to observe that at epoch 5, instances containing the fox and the squirrel are selected out by T-agent owing to better Q-value estimated by DQN. In Fig. 6(b), T-agent only selects out the bigger dolphin at epoch 1. With the epoch increases, T-agent selects out the smaller dolphin at epoch 3 and selects out more proposals of the two dolphins at epoch 5, which clearly demonstrates the effectiveness of T-agent in refining target proposals.

G. Hyperparameter Analysis

In this section, we conduct experiments on the P→C setting to evaluate the effect of the threshold $\tau$, the terminal time $E$, the size $N_c$ of the candidate set, and the probability of exploration $\epsilon$. The results of different $\tau$, $E$ and $N_c$ are shown in Fig. 7, and the results of different $\epsilon$ are shown in Table XII. From the results, it is noteworthy that the performances of SIR with different hyperparameters are all over 39.0% and are better than “SIR w/o S&T-agents”, clearly demonstrating the effectiveness of sequential instance refinement for cross-domain object detection. The followings are the detailed hyperparameter analysis.

1) Discussion of the Threshold $\tau$: We select $\tau$ in the range of $\{0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9\}$ and show the mAP-threshold curves in Fig. 7 (a), where the horizontal axis represents the value of $\tau$ in Eq. (5) and Eq. (6), and the vertical axis represents the mAP. From the results, we can find that the mAP first increases and then decreases as the threshold $\tau$ increases. Specifically, when the threshold $\tau$ is small, few outlier target instances and low-relevance

![Hyperparameter analysis on the P→C setting.](image)

<table>
<thead>
<tr>
<th>Method</th>
<th>Initial value</th>
<th>0.1</th>
<th>0.2</th>
<th>0.3</th>
<th>0.4</th>
<th>0.5</th>
<th>0.6</th>
<th>0.7</th>
<th>0.8</th>
<th>0.9</th>
</tr>
</thead>
<tbody>
<tr>
<td>SIR</td>
<td>41.9</td>
<td>40.1</td>
<td>40.7</td>
<td>40.5</td>
<td>42.6</td>
<td>40.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Table XII**

Average Precisions (%) on the P → C Setting with Different Values of $\epsilon$
source instances are selected out, and some outlier target instances and low-relevance source instances still remain in the candidate set. In this case, the negative transfer is mildly relieved and the mAP is low. When \( \tau \) becomes large, some target instances in the shared classes are wrongly selected out, leading to the decreasing mAP. Based on the experimental results, we set \( \tau = 0.5 \) to achieve the best performance.

2) Discussion of the Terminal Time \( E \): We tune \( E \) in the range of \( \{1, 2, 3, 4, 5, 6, 7\} \) and show the mAP-\( E \) curves in Fig. 7 (b), where the horizontal axis represents the value of \( E \), and the vertical axis represents the mAP. From the results, the highest mAP on the target domain is achieved when \( E \) is set to 3, which means that selecting 3 instances from each candidate set most properly relieves the negative transfer. When \( E \) is large, the agents continue selection although all the outlier target instances and low-relevance source instances have been selected out, leading to wrong selections of instances. When \( E \) is small, the agents stop selection before all the outlier target instances and low-relevance source instances are selected out, resulting in mildly relieving the negative transfer.

3) Discussion of the Size \( N_c \) of the Candidate Set: Fig. 7 (c) shows the performance of our method trained with different values of \( N_c \). From the results, the mAP first increases, achieves the highest value when \( N_c = 16 \), and then decreases. The possible reason is that a large \( N_c \) leads to fewer candidate sets and fewer instances are selected out from the proposal set. A small \( N_c \) means more candidate sets and more instances are selected out from the proposal set, which are prone to wrong selections of instances. We set \( N_c = 16 \) for the best performance in our experiments.

4) Discussion of the Probability of Exploration \( \epsilon \): TABLE XII shows the performance of our method trained with different decay schemes of \( \epsilon \), where \( \epsilon \) is decayed from the initial value to the final value during training. The larger \( \epsilon \) is, the more the agent explores. The larger \( \epsilon \) is, the more the agent explores. For convenience, we denote the initial \( \epsilon \) as \( \epsilon_s \) and the final \( \epsilon \) as \( \epsilon_f \). When \( \epsilon_s = 1 \), the agent explores more in the initial stage and requires more exploitation in the final stage, so \( \epsilon_f = 0 \) achieves better result than \( \epsilon_f = 0.01 \) and \( \epsilon_f = 0.1 \). When \( \epsilon_s = 0.9 \), the agent explores less in the initial stage and requires more exploration in the final stage. So \( \epsilon_f = 0.01 \) achieves better result than \( \epsilon_f = 0 \). In other words, the best performance is achieved when \( \epsilon \) is decayed from 0.9 to 0.01, which shows that under this setting, the agent can better balance the exploration and the exploitation during learning the policies of instance refinement.

V. CONCLUSION

We have presented a reinforcement learning based method, namely sequential instance refinement (SIR), to address the negative transfer problem in cross-domain object detection. In our SIR, S-agent and T-agent learn to remove the low-relevance source instances and outlier target instances, respectively. Via the sequential actions in the reinforcement learning process, the two agents can progressively refine both source and target instances and thus successfully alleviate negative transfer. Extensive experiments conducted on several benchmark datasets clearly demonstrate that our SIR outperforms the existing state-of-the-art methods for the cross-domain object detection task. As we believe that SIR is a general solution for tackling the negative transfer problem in object detection and can be readily incorporated by existing cross-domain methods to improve the overall performance.

REFERENCES


---

**Jin Chen** received the B.S. degree in computer science from the Beijing Institute of Technology, Beijing, China, in 2017, where she is currently pursuing the Ph.D. degree in computer science with the Beijing Laboratory of Intelligent Information Technology. Her current research interests include domain adaptation, reinforcement learning, and machine learning.

**Xinxiao Wu** (Member, IEEE) received the B.S. degree in computer science from the Nanjing University of Information Science and Technology, Nanjing, China, in 2005, and the Ph.D. degree in computer science from the Beijing Institute of Technology (BIT), Beijing, China, in 2010. From 2010 to 2011, she was a Postdoctoral Research Fellow with Nanyang Technological University, Singapore. She is currently an Associate Professor with the School of Computer Science, BIT. Her research interests include machine learning, computer vision, and video analysis and understanding.

**Lixin Duan** received the B.Eng. degree from the University of Science and Technology of China in 2008 and the Ph.D. degree from the Nanyang Technological University in 2012. He is currently a Full Professor with the School of Computer Science and Engineering, University of Electronic Science and Technology of China (UESTC). His research interests include machine learning algorithms especially in transfer learning and domain adaptation, and their applications in computer vision. He was a recipient of the Microsoft Research Asia Fellowship in 2009 and the Best Student Paper Award at the IEEE Conference on Computer Vision and Pattern Recognition 2010.

**Lin Chen** received the B.E. degree from the University of Science and Technology of China in 2009 and the Ph.D. degree from Nanyang Technological University, Singapore, in 2014. He is currently a Senior Principal Scientist and the Head of AI with Wyze Labs, Kirkland, WA, USA. His current research interests include machine learning, such as transfer learning, domain adaptation, weakly-supervised learning, few-shot-learning, and their applications in computer vision, such as object detection, face recognition, and video understanding. He was a recipient of *IEEE TRANSACTIONS ON MULTIMEDIA* Prize Paper Award in 2014.