Transductive Reward Inference on Graph

Bohao Qu, Xiaofeng Cao, *Member, IEEE*, Qing Guo, *Member, IEEE*, Yi Chang, *Senior Member, IEEE*, Ivor W.Tsang, *Fellow, IEEE* and Chengqi Zhang, *Senior Member, IEEE*

***Abstract*—In this study, we present a transductive inference approach on that reward information propagation graph, which enables the effective estimation of rewards for unlabelled data in offline reinforcement learning. Reward inference is the key to learning effective policies in practical scenarios, while direct environmental interactions are either too costly or unethical and the reward functions are rarely accessible, such as in healthcare and robotics. Our research focuses on developing a reward inference method based on the contextual properties of information propagation on graphs that capitalizes on a constrained number of human reward annotations to infer rewards for unlabelled data. We leverage both the available data and limited reward annotations to construct a reward propagation graph, wherein the edge weights incorporate various influential factors pertaining to the rewards. Subsequently, we employ the constructed graph for transductive reward inference, thereby estimating rewards for unlabelled data. Furthermore, we establish the existence of a fixed point during several iterations of the transductive inference process and demonstrate its at least convergence to a local optimum. Empirical evaluations on locomotion and robotic manipulation tasks validate the effectiveness of our approach. The application of our inferred rewards improves the performance in offline reinforcement learning tasks.**

***Index Terms*—Reward propagation graph, reward inference, offline reinforcement learning.**

I. Introduction

OFFLINE reinforcement learning (RL) problems can be defined as a data-driven formulation of the reinforcement learning problem, that is, learning a policy from a fixed dataset without further environmental input [1], [2], [3]. Reliable and effective offline RL methods would significantly affect various fields, including robots [4], [5], autonomous driving [6], recommendation systems [7], [8], and healthcare [9]. Rewards are typically necessary for learning policies in offline RL, but they are rarely accessible in practice, and the rewards for state-action pairs need to be manually annotated, which is difficult and time-consuming. Meanwhile, real-world offline RL datasets always have a small amount with reward and a large amount always without reward. Thus, learning a model from limited data with rewards to label unrewarded data is critical for learning effective policies to apply offline RL to various applications.

Typical methods have attempted various types of supervision for reward learning. The method proposed by [10] and ORIL [11] learns reward functions and uses them in offline RL. [10] employs a reward sketching interface to elicit human preferences and use them as a signal for learning. In reward sketching, the annotator draws a curve where higher values correspond to higher rewards. ORIL [11] relies on demonstrated trajectories to obtain reward functions both from labelled and unlabelled data at the same time as training an agent. [12] propose the timestep annotations are binary and treat the reward prediction as a classification problem to focus on sample efficiency with limited human supervision.

Reward learning for offline RL is roughly divided into two categories: timestep-level (e.g., state-action pair reward annotations for the entire episode produced by humans [10]) and episode-level supervision (e.g., annotations of success for the whole episode [12], [11]). For episode-level supervision, [12] assumes rewards are binary, and it indicates if the task is solved. Episode annotations provide only limited information about the reward. They indicate that some of the state-action pairs from the episodes show successful behavior but do not indicate when the success occurs. So the episode-level supervision method is not adaptation to any value reward learning question. For timestep-level annotations, [12] and [13] need first to annotate demonstrated trajectories that are the successful trajectories (e.g., expert demonstrations). [10] employs a reward sketching interface to elicit human preferences and use them as a signal for learning, but the method is hard to solve tasks where variable speed is important or with cycles as in walking. Accordingly, existing methods are unsuitable for timestep-level reward learning with arbitrary values in offline reinforcement learning without any expert trajectories. The manual annotation of rewards for state-action pairs is costly, making it challenging to learn effective policies with limited reward labelled data. This makes it challenging to apply offline reinforcement learning to a variety of scenarios.

In a general sense, offline RL addresses the problem of learning to control a dynamic system, which is fully defined by a Markov decision process (MDP). An MDP is a sequential decision process for state-to-state transitions, which could be formed as a chain, and multiple chains of multiple MDPs can be combined to form a graph. In the graph, each node represents a state-action pair, and each edge is labelled with the probability of transitioning from one state to another state, given a particular action. The graph possesses the contextual properties

Bohao Qu, Xiaofeng Cao, and Yi Chang are with the School of Artificial Intelligence, Jilin University, Changchun, Jilin 130012, China, and also with the Engineering Research Center of Knowledge-Driven Human-Machine Intelligence, Ministry of Education, China. Yi Chang is also with the International Center of Future Science, Jilin University, Changchun, Jilin, 130012, China.
E-mail: qubohao@126.com, {xiaofengcao, yichang}@jlu.edu.cn.

Qing Guo and Ivor W. Tsang are with the Institute of High Performance Computing (IHPC) and Centre for Frontier AI Research (CFAR), Agency for Science, Technology and Research (A*STAR), Singapore. Ivor W. Tsang is also with the School of Computer Science and Engineering, Nanyang Technological University, Singapore.
E-mail: {Ivor_Tsang, guo_qing}@cfar.a-star.edu.sg.

Chengqi Zhang is with the Australian Artificial Intelligence Institute, University of Technology Sydney, Ultimo, NSW 2007, Australia. E-mail: chengqi.zhang@uts.edu.au.

Xiaofeng Cao and Yi Chang are the corresponding authors.

Manuscript created January, 2024.

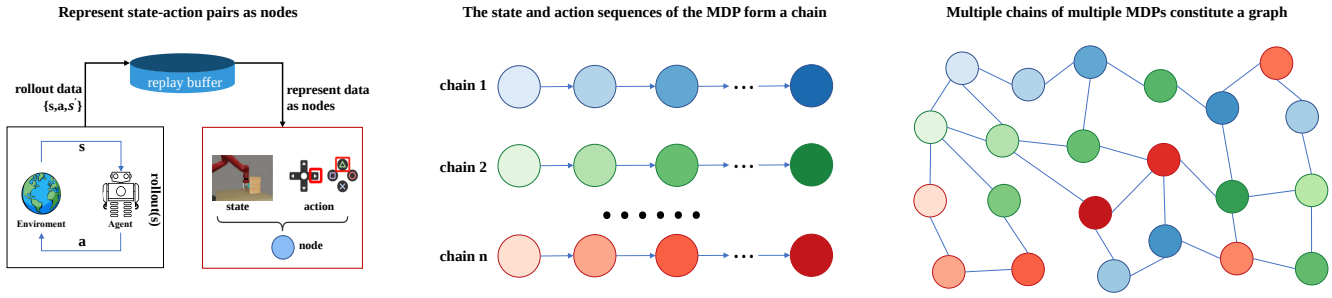


Fig. 1. We first represent each state-action pair within a Markov decision process (MDP) as an individual graph node. Then, we establish a foundation for modeling the state-action sequences across multiple MDPs as interconnected chains. Finally, these chains collectively form a comprehensive graph that encapsulates the dynamics of multiple MDPs. The graph structure is characterized by connectivity, where each node is connected to multiple other nodes. We leverage this feature to propagate reward-related information within the graph.

of information propagation, the structure is characterized by connectivity, where each node is connected to multiple other nodes. It is convenient to perform inference through a message passing mechanism, and the inference procedure essentially propagates information from the state-action pairs with rewards to the state-action pairs without rewards and results in a reward function over state-action pairs, as shown in Fig. 1. The model tries to find any information about the pattern in all state-action pairs and later uses this information for predicting the rewards of the unlabelled state-action pairs. Formally, such a situation requires the learning of a reward model to accurately predict the rewards for state-action pairs in a specific task, based on the data that is invariant to the distribution collected from agent-environment interactions within that task. In this manner, this situation involves a paradigm of transductive reward inference[1]. Meanwhile, the label-efficiency problem has been successfully addressed by label propagation (LPA) approaches [14], [15], [16], [17], [18], [19], [20] which is based on graph models and plays a significant role in leveraging unlabelled dataset to improve the model performance with low cost. Label propagation approaches formulate labelled and unlabelled data as a graph, where nodes represent sample data and edges represent relationships between nodes, and the node labels are propagated and aggregated along the edges.

Inspired by the label propagation approaches, we present TRAIN: **T**ransductive **R**ew**A**rds **IN**ference with Propagation Graph for Offline Reinforcement Learning. Each task is characterized by a unique composition of states with varying features. We need to infer rewards for a large number of unlabeled state-action pairs based on a smaller, reward-laden subset specific to each task. Utilizing transductive inference, we leverage the graph structure established by MDPs to incorporate information from all state-action pairs, regardless of whether they have associated rewards. Although the rewards for most state-action pairs remain unknown, this graph structure enables us to effectively propagate the limited available reward information. Specifically, TRAIN consists of two key ingredients:

- Reward Propagation Graph: We represent each state-action pair as a graph node and leverage the similarities and relationships between them to learn the edge weights of the nodes. It is worth noting that rewards are influenced by many factors, and all of the factors should be considered when learning the reward propagation graph.
- Transductive Reward Inference: We employ the reward propagation graph to infer rewards for state-action pairs that are without rewards and then utilize them for doing offline RL. The reward inference technique propagates rewards on the graph from state-action pairs with rewards to state-action pairs without rewards and will converge to a unique fixed point after a few iterations.

We remark that TRAIN is not a naïve application of the LPA technique but a novel scalable method of learning propagation graphs that integrates multiple influence reward factors to edge weights. The graph sufficiently leverages various relationship information between nodes, which can make reward inference more accurate. This has not been considered or evaluated in the context of offline RL reward learning. We also prove that the transductive inferred reward has a fixed point and at least can converge to a local optimum.

Our experiments demonstrate that the state-action pairs labeled by TRAIN significantly improve the offline reinforcement learning method when learning policy with limited reward annotations on complex locomotion and robotic manipulation tasks from DeepMind Control Suite [21] and Meta-World [22]. In particular, our method inherits the smooth characteristics of the LPA method [23], which can make the state-action pairs with smooth rewards and further make the process of offline RL algorithm learning policy more stable.

II. Related Work

Offline RL The offline reinforcement learning problem, which enables learning policies from the logged data instead of collecting it online, can be defined as a data-driven formulation of the reinforcement learning problem [1], [2], [3]. It is a promising approach for many real-world applications. Offline RL is an active area of research and many algorithms have been proposed recently, e.g., BCQ [24], MARWIL [25], BAIL

[1]Note that inductive inference usually focuses on building a relationship between the state feature and reward target by examining the hidden patterns in the state-action pairs with rewards and then generalizes effectively to unseen state-action pairs without rewards.

[26], ABM [27], AWR [28], CRR [29], F-BRC [30]. In this paper, we adopt CRR as our backbone algorithm due to its efficiency and simplicity.

Reward learning It is possible to learn the reward signal even when it is not constantly available in the environment. The reward can be learned if demonstrations are provided either directly with inverse RL [31], [32] or indirectly with generative adversarial imitation learning (GAIL) [33]. The end goal [34], [35] or reward values [10] for a subset of state-action pairs can be known, in which case reward functions can be learned by supervised learning. A significant instance of learning via limited reward supervision [10] is studied in some works. Rewards are commonly learned for online RL [36]. While learning from built or pre-trained state representations [37], [34], [38], [39], [40], [41], [42], [43] has achieved a lot of success, learning directly from pixel input is known to be difficult [44] and the quantity of supervision needed may become a bottleneck [10]. Unlike many other reward learning approaches for offline RL, we focus on learning rewards with multi-factors that influenced rewards from limited annotations.

Transduction The setting of transductive inference was first introduced by Vapnik [45]. Transductive Support Vector Machines (TSVMs) [46] is a margin-based categorization technique that reduces test set mistakes. Particularly for short training sets, it demonstrates considerable advantages over inductive techniques. Another classification of transduction methods involves graph-based methods [14], [16], [18], [47], [48]. Labels are transferred from labelled to unlabelled data instances through a process called label propagation, which is driven by the weighted graph. In prior works, the graph construction is done on a pre-defined feature space using only a single influence factor between nodes so that it is not possible to learn multi-factors influenced graph edge weights.

III. Problem formulation

The key to the TRAIN method is the prior assumption of consistency, which means: (1) nearby states and actions are likely to have similar or the same reward, and (2) state-action pairs on the same structure (typically referred to as a cluster or a manifold) are likely to have the similar or the same reward. This argument is akin to semi-supervised learning problems that in [49], [50], [51], [16], [52], [23], [53] and often called the *cluster assumption* [16], [51]. Orthodox supervised learning algorithms, such as k-NN, in general, depend only on the first assumption of local consistency [16], that is, k-NN makes every data point be similar to data points in its local neighborhood. Our method leverages the relation information between states and actions to formalize the intrinsic structure revealed by state-action pairs with reward and state-action pairs without reward and construct a reward inference function.

We assume that the training samples (both with reward and without reward) are given as $D = [(s_1, a_1), ..., (s_Z, a_Z)]$, where (s_i, a_i) denotes the state-action pair, and D has Z pairs. Given this, let D_L denote the labelled state-action pair set of D with v pairs, and D_U denote the unlabelled state-action pair set of D with g pairs, that is, $D_L = [(s_1^L, a_1^L), ..., (s_{Z_L}^L, a_{Z_L}^L)]$, $D_U = [(s_1^U, a_1^U), ..., (s_{Z_U}^U, a_{Z_U}^U)]$, s.t., $Z_L + Z_U = Z$.

The Z rewards are denoted by $R = [r_1, ..., r_Z]$, we split the reward set R into 2-sub-block, $R_L = [r_1^L, ..., r_{Z_L}^L]$ denotes the subset of known rewards and $R_U = [r_1^U, ..., r_{Z_U}^U]$ denotes the subset of unknown rewards. Suppose that we are given a small set D_L of the state-action pairs with reward. The rest of the state-action pairs $D_U = D \setminus D_L$ are without reward. TRAIN utilizes all samples and known rewards to learn a reward propagation graph and infer rewards for state-action pairs that are without reward.

IV. Methodology

A. Overview

We take advantage of the property of MDPs, where the reward depends only on the current state and action, as well as the relationship between states and actions, to construct a reward propagation graph. We then train this graph by the state-action pairs with rewards, facilitating transductive reward inference for unlabeled state-action pairs. In offline reinforcement learning, such state-action pairs are logged in dataset D. Practically, dataset D encompasses a wide array of state-action pairs generated for specific tasks via scripted, learned policies, and human demonstrations [10].

B. Construct Reward Propagation Graph

The graph possesses the contextual properties of information propagation and the structure is characterized by connectivity, which allows it to model the interrelationships between entities effectively. In a graph, nodes are connected by edges, facilitating the transmission of information across nodes through these connections. Given the limited number of state-action pairs with rewards, we could learn to infer rewards for those pairs without rewards. To achieve this, we model state-action pairs as nodes and construct a reward propagation graph. This graph leverages the relationships between nodes to transfer reward-related information from labelled (rewarded) nodes to unlabelled (unrewarded) nodes.

a) Graph construction: For most reinforcement learning tasks, rewards are influenced by many factors. For instance, in task *Humanoid*, which is part of the DeepMind Control Suite [21], [54], the state consists of six parts: joint angles, the height of the torso, extremity positions, torso vertical orientation, the velocity of the center of mass, and the generalized velocity, and action also consists of several parts that represent the torques applied at the hinge joints. The reward is related to the upright state of the robot, the control operation of the actuator, and the moving speed, etc., which are closely related to each part of the above state and action, so we regard each part of the state and action as a factor that influences the reward.

Specifically, we denote $s_i = [s_{i_1}, s_{i_2}, s_{i_3}, ..., s_{i_M}]$, where s_{i_m} is a sub-state with any given dimension, and s_i consists of M sub-states. Correspondingly, we denote $a_i = [a_{i_1}, a_{i_2}, a_{i_3}, ..., a_{i_N}]$, where a_i is all of the actions performed given state s_i, and composed of N specific sub-actions of a_{i_n}.

We design a reward propagation graph construction method integrating multi-factors influencing the reward to tune the edge weights. To be specific, we employ a distance function $\rho_s(s_{i_m}, s_{j_m}), \forall\, i \neq j$ to measure the similarity of the sub-states,

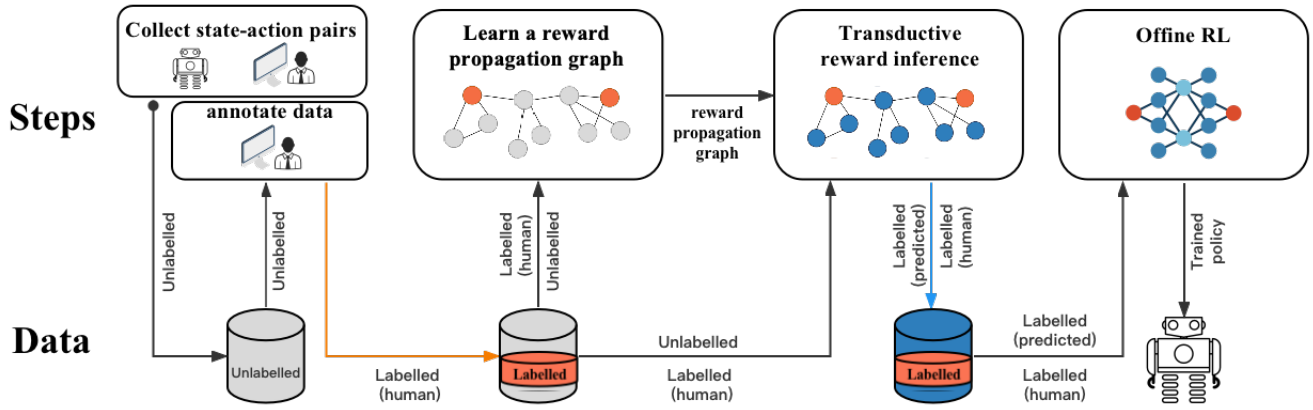


Fig. 2. TRAIN workflow: The process begins with the construction of a reward propagation graph using a pre-recorded dataset. Subsequently, this graph, in conjunction with state-action pairs that have rewards, is utilized to infer rewards for state-action pairs that lack rewards. In the final step, all state-action pairs, both with and without inferred rewards, are integrated into the offline reinforcement learning process.

and also employ a distance function $\rho_a(a_{i_n}, a_{j_n}), \forall\ i \neq j$ to measure the similarity of the sub-actions, the two distance function could be Euclidean distance or others. Then, we define the multi-factor measure for each state-action pair as:

$$\begin{aligned}\ell(\mathcal{S}_i, \mathcal{S}_j) = [\rho_s(s_{i_1}, s_{j_1}), ..., \rho_s(s_{i_M}, s_{j_M}),\\ \rho_a(a_{i_1}, a_{j_1}), ..., \rho_a(a_{i_N}, a_{j_N})] \in \mathbb{R}^{M+N},\end{aligned} \tag{1}$$

where $\mathcal{S}_i = (s_i, a_i)$ denotes the i-th state-action pair.

Further, we employ a reward shaping function f_Θ with the parameters Θ to tune multi-factors' contribution to the rewards and integrate them into the edge weight:

$$W_{ij} = \frac{\exp(-f_\Theta(\ell(\mathcal{S}_i, \mathcal{S}_j)))}{\sum_{j \neq i} \exp(-f_\Theta(\ell(\mathcal{S}_i, \mathcal{S}_j)))}, \forall\ i \neq j. \tag{2}$$

where W_{ij} is an element in matrix W, which is a $Z \times Z$ weight matrix. We let $W_{ii} = 0$ and we also have $\sum_j W_{ij} = 1, \forall i$. We define each state-action pair as a node in the graph and define the weight between every two nodes in the graph, thus completing the construction of the reward propagation graph G with weight matrix W.

b) Graph training: In different tasks, the number of factors influencing reward is different, and the degree of influence of different factors on reward is also different. Therefore, we need to tune the weight of graph edges so that optimizing the function f_Θ to make the multi-factors efficiently integrate to rewards. Intuitively, for the state-action pair (s_i, a_i), a larger edge weight W_{ij} means that state-action pair (s_j, a_j) will transfer more information to the reward for state-action pair (s_i, a_i). Specifically, we use the relationship between labelled data to train f_Θ to make it suitable for the current task. We design a predicted reward ξ_l:

$$\xi_l = \sum_{k \neq l} W_{lk} r_k, \quad l, k \in [1, ..., v], \tag{3}$$

where r_k is a label (reward) for a state-action pair (node). We use other labelled state-action pairs to predict the label of the current state-action pair (have a ground truth label) and then minimize the difference between the predicted label and the ground truth label.

Then, the goal of training the graph G is to optimize the parameters Θ for the function f_Θ, that is, minimize the difference between the predicted labels and their corresponding ground truth labels, the objective function $H(G)$ is given as:

$$\arg\min_{\Theta} \left\{ H(G) = \frac{1}{2Z_L} \sum_{l=1}^{Z_L} ||\xi_l - r_l||^2 \right\}. \tag{4}$$

There does not exist a closed-form solution, and we use the gradient descent method to seek the solution, details are in the appendix A.

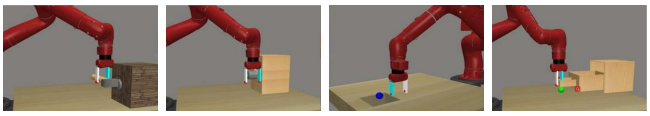
It should be noted that: the unlabelled data are not included in Equation (4), since the number of the unlabelled data is often much larger than that of the labelled data, the term on the unlabelled data may dominate the objective function, which in turn may degrade the algorithmic performance.

C. Transductive Reward Inference

In Section IV-B, we constructed the graph and trained the weights of the graph edges. In this section, we propagate reward-related information on the graph to infer the rewards for unlabelled state-action pairs based on the rewards of other state-action pairs.

We separate the weights associated with nodes without rewards from the weight matrix W of graph G formed by Equation (2), and represent them as two submatrices W_{UL} and W_{UU}. W_{UL} represents the weights between nodes with rewards and nodes without rewards, and W_{UU} represents the weights between nodes without rewards. We split the reward set R into two sub-blocks, R_L denotes the subset of known rewards and R_U denotes the subset of unknown rewards.

The inference of rewards for unlabelled nodes requires considering the information transfer between labelled and unlabelled nodes, we use $W_{UL}R_L$ for this calculation, while also taking into account the relationships between unlabeled nodes themselves, computed using $W_{UU}R_U$. Since unknown



(a) Hammer (b) Button Press (c) Sweep Into (d) Open Drawer

Fig. 3. Meta-World is a set of robotic manipulation tasks.

rewards R_U are a variable to be learned, we provide an iterative computation formula by:

$$R_U \longleftarrow W_{UU}R_U + W_{UL}R_L. \tag{5}$$

After t-th iterations, we obtain the following formula:

$$R_U^t \longleftarrow W_{UU}^t R_U^0 + (W_{UU}^{t-1} + ... + W_{UU} + 1)W_{UL}R_L. \tag{6}$$

The detailed derivation process is in Appendix B. Based on the weights calculated by Equation (2), the values in W_{UU} are all less than 1. Therefore, as t approaches infinity, W_{UU}^t tends to infinitesimal values, leading $W_{UU}^t R^0$ converges to 0. Meanwhile, $(W_{UU}^{t-1} + ... + W_{UU} + 1)$ forms a geometric series, and after applying the formula for the sum of a geometric series, we obtain the solution:

$$R_U = (I - W_{UU})^{(-1)}W_{UL}R_L, \tag{7}$$

which is a *fixed point*, and I is the identity matrix [14], [16]. TRAIN converges to a *fixed point* means that the reward inference error is within a certain range.

D. Policy Learning

We remark that TRAIN can be combined with any offline RL algorithm by learning rewards for state-action pairs without reward. For learning a policy, we use a pre-recorded dataset D. Dataset D contains some state-action pairs with reward D_L, and most of the rest are state-action pairs without reward D_U. We use TRAIN to predict rewards for D_U as $\tilde{D_U}$, and then replace the part of D_U with the predicted rewards $\tilde{D_U}$ to form $\tilde{D}$. We can then use $\tilde{D}$ to train the policy. In this study, we employ Critic-Regularized Regression (CRR) [29], a simple and efficient offline reinforcement learning algorithm, to train offline reinforcement learning policies on the dataset with the predicted rewards.

V. Experiments

In this section, we present and analyze our experimental results. Initially, in Section V-A, we detail the experimental setup, including descriptions of the environment and tasks, datasets, and baselines. Subsequently, in Section V-B, we display the experimental results and provide an analysis. Following this, in Section V-C, we present the findings from our ablation study. In Section V-D, we discuss the accuracy of inferred rewards across different proportions of labeled state-action pairs. Lastly, in Section V-E, we demonstrate the accuracy of inferred rewards under various norms.

(a) Cheetah (b) Walker (c) Fish (d) Humanoid (e) Cartpole

Fig. 4. DeepMind Control Suite is a set of popular continuous control environments with tasks of varying difficulty, including locomotion and simple object manipulation.

A. Experiments setup

a) Environment and tasks: We conduct the experiments with a variety of complex robotic manipulation and locomotion tasks from Meta-World [22] and DeepMind Control Suite [21], [54], respectively. Many factors influence the reward function of these two series of tasks. Meta-World consists of a variety of manipulation tasks designed for learning diverse manipulation skills. The second environment is the DeepMind Control Suite, which contains many continuous control tasks involving locomotion and simple manipulation. To investigate the performance of TRAIN with a small amount of annotated state-action pairs. We conduct experiments on more than 40 tasks in the Meta-World environment, and we select four tasks (*Hammer, Button Press, Sweep Into, Open Drawer*) (see Fig. 3) of them to show their learning curves. We also choose five complex environments from DeepMind Control Suite: *Cheetah Run, Walker Walk, Fish Swim, Humanoid Run, and Cartpole Swingup* (see Fig. 4) to evaluate the performance of TRAIN in another environment different from the Meta-World, to verify whether the algorithm works only in one environment. On each task, we leverage different algorithms to predict the rewards for the same dataset and employ the CRR algorithm (except Behavior cloning) to perform policy learning on the dataset after predicting the rewards. The learned policies are used to evaluate the performance of TRAIN and other baselines. We report the results on all tasks of 5 random seeds, and results are shown in Section V-B.

b) Datasets: To rigorously evaluate the efficacy of our method in learning rewards for diverse data and to showcase the performance of the offline RL algorithm across various seeds and diverse data, we organize datasets across two distinct domains. The total number of state-action pairs for each task is outlined in Table III.

Meta-World domain has been used to evaluate online RL agents, we create an *ad hoc* dataset suitable for offline learning. To do so, we train Soft Actor-Critic [55] from full states on each of the tasks, and record the entire training data, which forms the dataset D for each Meta-World task. We define each training session, from initialization to task resolution, as an independent run. Multiple independent runs are executed to ensure a rich diversity of data until a sufficient number of state-action pairs are collected.

For the DeepMind Control Suite, while the RL Unplugged [56] data provided are typically reduced through subsampling, we needed a larger volume and greater diversity. Hence, we adopted the RL Unplugged methodology, employing D4PG [57] as implemented in [58], to amass substantial state-action pairs for each task. Each collection session, from start to

task completion, is treated as an independent run. This process continues until the desired quantity of state-action pairs is amassed for each task.

From each task dataset, we random extract a small subset of state-action pairs and leverage them as labelled data D_L, while rewards are stripped from the remainder to create unlabeled data D_U. The labels in D_L constitute the ground truth rewards. Table III also displays the ratio of state-action pairs that include ground truth rewards for each task.

c) Baselines: Behavior cloning (**BC**) is a popular algorithm in the field of imitation learning and also an alternative way to learn a policy when the reward values are not available. BC agent does not require reward values as it attempts to directly imitate the demonstrated state-action pairs.

Time-guided reward (**UDS**) workflow, as outlined in [12], involves several key steps. Initially, it infers a reward function based on limited supervision, utilizing timestep-level annotations expressed as reward values on a subset of trajectories. Subsequently, it retroactively annotates all trajectories using the obtained reward function. Finally, the trajectories, now equipped with predicted rewards, are employed for offline reinforcement learning. Additionally, we integrated TGR with CRR (Critic Regularized Regression) [29] to learn policies as a baseline for our efforts. Specifically, TGR employs a two-step process for reward function inference. It starts by annotating demonstrated trajectories, assigning a flat zero synthetic reward to the unlabelled subset. The reward function is then trained using a loss function that jointly optimizes timestep-level annotations and synthetic flat labels. This comprehensive approach contributes to the effectiveness of TGR in the context of offline reinforcement learning. We also combined TGR with CRR [29] to learn policies.

Unlabeled data sharing (**UDS**) [13] addresses this scenario by treating the unlabeled dataset as if it has zero rewards, followed by the incorporation of reweighting techniques. This reweighting process is designed to adjust the distribution of interspersed zero-reward data. The primary objective is to synchronize the distribution of this external data with that of reward-containing data pertinent to the original task, thereby alleviating the bias introduced by inaccurate reward data. In particular, UDS initially assigns the minimum feasible reward (typically assumed to be 0) to all transitions within the unlabeled data. Subsequently, these unlabeled transitions undergo reweighting, altering the distribution of unlabeled data to mitigate reward bias. This strategy contributes to the overall goal of enhancing the alignment between labeled and unlabeled data distributions in UDS. Finally, train offline reinforcement learning policies using the reweighted reward distribution dataset.

B. Experiments results

a) Meta-world: We show the evaluation results on more than 40 tasks in the Meta-World environment in Table I. Apart from the five tasks (*button-press, drawer-close, handle-press, reach, reach-wall*), the performance of the algorithm TRAIN exhibits greater superiority compared to others. This indicates that TRAIN achieves more accurate reward learning on these tasks. Additionally, due to its inherent smoothness, the learned rewards in TRAIN are smoother, leading to more stable policy performance. UDS emphasizes the need for high-quality labeled data. Since the data in our environment consists of SAC training data, and the labeled data is randomly selected, UDS performs well in relatively easy-to-learn tasks, specifically those with a higher proportion of high-quality data in the dataset, with particularly outstanding performance in the tasks of *button-press, drawer-close, and handle-press*. But, its performance is subpar in many other tasks.

Regarding the two tasks where the TGR algorithm outperforms others, we conducted a detailed analysis to identify the reasons. These tasks involve relatively simple action trajectories that are easily explored. The TGR algorithm annotates the demonstrated trajectories and assigns a flat zero synthetic reward to the unlabelled subset, which amplifies the rewards annotated as one along the action trajectories, encouraging the policy to learn these trajectories more actively. Therefore, TGR achieves better performance on such tasks. However, in other relatively complex tasks where the action trajectories for task completion are diverse, the method fails to provide accurate rewards for some procedural actions, resulting in mediocre policy performance. On the other hand, the poor performance of the Behavioral Cloning (BC) algorithm in many tasks can be attributed to its reliance on training with the entire dataset. The dataset comprises both successful and unsuccessful episodes, causing BC to be significantly affected by data quality. In tasks with relatively simple actions, algorithms that collect data can quickly learn the action trajectories for task completion, leading to a higher proportion of high-quality data in the dataset and thus better performance of BC. Conversely, in relatively complex tasks, data collection algorithms require a longer exploration process to learn the action trajectories for task completion, resulting in a lower proportion of high-quality data and consequently poor performance of BC.

We select four tasks of the Meta-World (*Hammer, Button Press, Sweep Into, Open Drawer*) to show their learning curves as measured on the success rate, as shown in Fig. 5. TRAIN outperforms the baselines on all four tasks, showing that TRAIN is well suited to make effective use of the unlabelled, mixed quality, state-action pairs. In the two tasks of Hammer and Sweep Into, TRAIN has always shown a greater performance advantage compared to other baselines. In the two tasks of Button Press and Open Drawer, in the early stage of training, the performance of the three algorithms is equivalent, and the follow-up TRAIN gradually stands out. Especially in the Open Drawer task, a greater performance advantage has been achieved, while in the Button Press task, the training process of other baselines fluctuates greatly, and TRAIN has less fluctuation and achieves better performance. The conclusions drawn from the results in Fig. 5 are consistent with the conclusions drawn from the results in Table I. Italic numbers indicate the highest average return for each task, bold numbers indicate the statistically significant highest average return for each task, in which the average return is the highest, and there is not a clear overlap in the standard deviation among the baselines. TRAIN has an excellent performance from the aspect of the success rate demonstrating that TRAIN

TABLE I
EVALUATION RETURNS ON MORE THAN 40 META-WORLD TASKS. THE AVERAGE ± STANDARD DEVIATION IS SHOWN FOR FIVE RANDOM SEEDS.

Tasks	BC	TGR	UDS	TRAIN
basketball	3019 ± 439.6	978 ± 122.7	3661 ± 406.9	*4165* ± 315.9
box-close	1081 ± 106.4	1568 ± 119.8	1371 ± 134.5	$\mathbf{3897 \pm 412.3}$
button-press	1911 ± 356	3301 ± 291	*3508* ± 213.4	3435 ± 106.4
button-press-topdown	3190 ± 310.7	3401 ± 415.7	3496 ± 229.2	*3587* ± 75.4
button-press-topdown-wall	1892 ± 50.2	2085 ± 61.6	2069 ± 48.7	*2115* ± 70.4
coffee-button	3531 ± 710.6	3631 ± 610.8	3923 ± 352.4	*4128* ± 210.9
coffee-pull	517 ± 16.3	372 ± 11.5	314.2 ± 7.9	$\mathbf{637 \pm 35.2}$
dial-turn	1871 ± 262.7	4217 ± 395.4	4236 ± 162.1	*4428* ± 185.9
disassemble	215 ± 10.9	217 ± 18.1	239.7 ± 21.5	$\mathbf{889 \pm 10.5}$
door-close	3862 ± 167.1	4328 ± 212.0	4451 ± 184.2	*4512* ± 217.3
door-lock	3156 ± 306.9	3536 ± 285.1	3312 ± 182.8	*3753* ± 195.2
door-open	1082 ± 46.7	3985 ± 306.9	1949 ± 66.1	*4451* ± 197.1
door-unlock	1947 ± 216.1	4011 ± 75.3	2052 ± 101.6	*4189* ± 118.9
drawer-close	4697 ± 64.3	4839 ± 102.1	*4881* ± 79.8	4797 ± 20.3
drawer-open	1769 ± 247	2890 ± 86	1895 ± 40.1	$\mathbf{4466 \pm 41.3}$
faucet-close	4143 ± 170.6	4687 ± 821.6	4338 ± 102.1	*4712* ± 147.1
faucet-open	3660 ± 316.4	4702 ± 1503.5	4671 ± 361.8	*4715* ± 361.3
hammer	2205 ± 268	3898 ± 163	2692 ± 101.5	$\mathbf{4532 \pm 95.1}$
hand-insert	56 ± 16.1	443 ± 8.4	409 ± 10.8	$\mathbf{4016 \pm 598.6}$
handle-press	4598 ± 137.9	4522 ± 136.4	*4651* ± 82.4	4618 ± 102.8
handle-press-side	4241 ± 1353.4	4764 ± 243.7	4734 ± 185.2	*4785* ± 458.1
handle-pull	3892 ± 986.8	4348 ± 894.1	4138 ± 147.6	*4592* ± 125.7
handle-pull-side	3678 ± 1006.2	4095 ± 572.6	3958 ± 114.9	$\mathbf{4551 \pm 92.8}$
lever-pull	3864 ± 190.8	4184 ± 175.1	4008 ± 81.7	$\mathbf{4307 \pm 135.7}$
pick-out-of-hole	11 ± 0.4	209 ± 3.5	117 ± 12.2	$\mathbf{1035 \pm 234.9}$
pick-place	1879 ± 411.6	2975 ± 495.6	3228 ± 283.2	$\mathbf{4106 \pm 589.4}$
plate-slide	3984 ± 101.7	3674 ± 748.3	4064 ± 151.8	*4459* ± 171.4
plate-slide-back	3017 ± 331.6	3158 ± 958.4	3089 ± 163.1	$\mathbf{4658 \pm 165.3}$
plate-slide-back-side	4087 ± 887.5	4678 ± 172.6	4703 ± 136.8	*4734* ± 198.4
plate-slide-side	2698 ± 538.8	3002 ± 365.3	2928 ± 289.1	*3010* ± 429.5
push	1983 ± 381.9	2097 ± 261.4	2248 ± 175.2	$\mathbf{4268 \pm 210.7}$
push-back	9 ± 0.4	137 ± 1.5	79 ± 8.7	$\mathbf{201.3 \pm 21.7}$
push-wall	3642 ± 597.8	4347 ± 187.4	4182 ± 155.8	*4501* ± 204.6
reach	3209 ± 397.2	*4761* ± 476.6	4581 ± 181.2	4668 ± 215.6
reach-wall	4626 ± 91.8	*4816* ± 51.1	4672 ± 30.4	4810 ± 36.3
stick-pull	592 ± 10.8	442 ± 7.2	408 ± 7.8	$\mathbf{4128 \pm 121.5}$
stick-push	362 ± 16.2	887 ± 5.3	1065 ± 21.7	$\mathbf{2745 \pm 514.3}$
sweep	879 ± 145.6	3214 ± 412.7	3708 ± 237.1	*4106* ± 312.7
sweep-into	962 ± 137	1838 ± 149	2115 ± 176.5	*2257* ± 323.9
window-close	3846 ± 98.3	4104 ± 106.1	4354 ± 86.4	*4458* ± 88.4
window-open	3217 ± 193.2	2897 ± 954.5	3141 ± 105.5	$\mathbf{3829 \pm 208.4}$

can effectively label rewards for state-action pairs without rewards, and the smoothness of the algorithm can make the labelled data also have smooth characteristics. The policies trained using these data have more stable performance.

b) DeepMind Control Suite: We show the learning curves of the five DeepMind Control Suite tasks (*Cheetah Run, Walker Walk, Fish Swim, Humanoid, Cartpole Swingup*) in Fig. 6. Our method TRAIN achieves better performance than baselines in five tasks. It has achieved a great performance advantage in the Walker Walk task, and also performed well in the Fish Swim task. The Cheetah Run and Cartpole Swingup tasks, also showed a certain performance advantage compared to TGR, although the advantage is not very large, it can still reflect the ability of the TRAIN algorithm. In the Humanoid task, TRAIN, UDS, and TGR are evenly matched during the training process, but the final performance of TRAIN is still better than other baselines, reflecting the stability of the state-action pairs with rewards provided by the TRAIN algorithm.

The action spaces of these five tasks are continuous, and the rewards are also continuous. It is difficult to give a clear boundary to distinguish good actions and bad actions. Therefore, the performance of TRAIN and baselines are both very steady. TRAIN has smooth characteristics, which can make the labelled data also have smooth characteristics. Using these state-action pairs with smooth rewards makes the process of policy learning in offline RL more stable. Since the TGR algorithm annotates the demonstrated trajectories and assigns a flat zero synthetic reward to the unlabelled subset, it shows a large shock in the process of learning some tasks. UDS emphasizes the need for high-quality labeled data, resulting in slightly poorer performance. BC performed the poorest across the five tasks, particularly struggling with the Fish Swim and Cheetah Run tasks. Since BC trains using the entire dataset, it fails to differentiate between data qualities, preventing it from achieving a high-performing policy.

C. Ablation study

We conducted a comprehensive set of ablation studies aimed at thoroughly evaluating the effectiveness of our reward shaping function, denoted as f_{Θ}. These experiments were meticulously designed and carried out across a diverse set of environments,

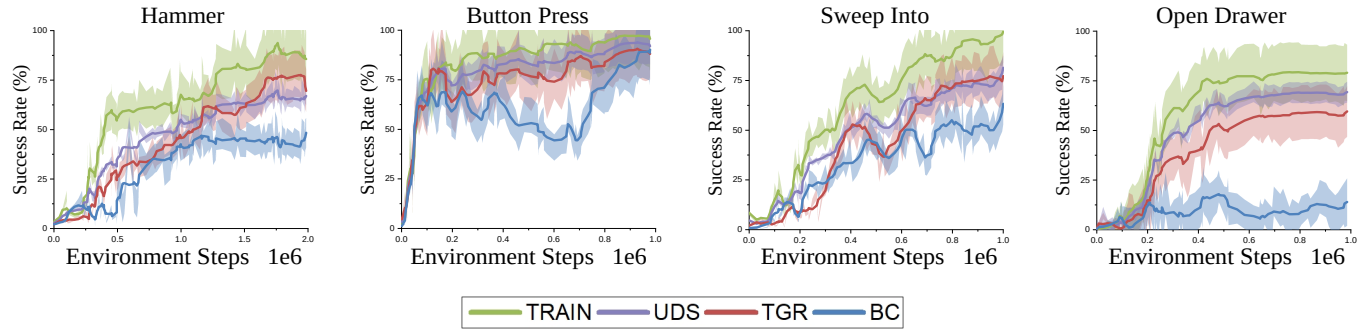


Fig. 5. Learning curves on the four Meta-World tasks as measured on the success rate. The solid line and shaded regions represent the mean and standard deviation, respectively, across five seeds.

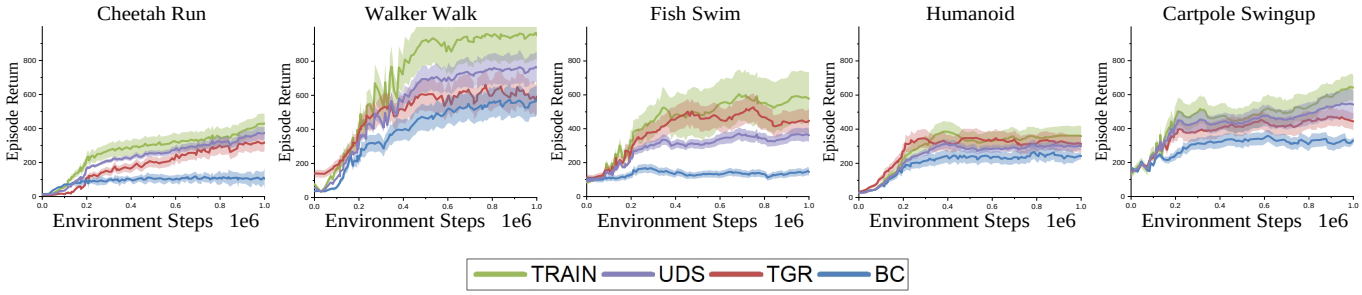


Fig. 6. Learning curves on the five DM Control tasks as measured on the episode return. The solid line and shaded regions represent the mean and standard deviation, respectively, across five seeds.

including four Meta-World environments and five DeepMind Control Suite environments.

In our investigation, we delved into the intricate interplay of various factors that influence the rewards associated with nine distinct tasks. To provide a detailed assessment, we employed four distinct composition methods, each shedding light on the role of factors related to both states and actions:

(1) Method 1: Both states and actions underwent decomposition into multiple factors, allowing us to scrutinize the combined impact of these nuanced elements.
(2) Method 2: We selectively decomposed states into multiple factors while treating actions as a unified entity. This method offered insights into how states, in isolation, contribute to the shaping of rewards.
(3) Method 3: We kept states as a single, undivided factor but decomposed actions into multiple components. This experiment assessed the significance of dissecting actions in the reward-shaping process.
(4) Method 4: We simplified the scenario by considering both states and actions as single, undifferentiated factors. This method served as a baseline for evaluating the performance of more complex factorization approaches.

The results are shown in Table II. Italic numbers indicate the highest average return for each task. Bold numbers indicate the statistically significant highest average return for each task, in which the average return is the highest, and there is not a clear overlap in the standard deviation among the baselines.

The compelling results that emerged from our extensive experimentation affirmed the superiority of Method 1, where both states and actions were decomposed into multiple factors. This approach consistently demonstrated the most favorable outcomes across the range of tasks we examined. On the other hand, Method 2, which decomposed states while treating actions as single entities, produced results that, while respectable, fell short of the peak performance achieved by Method 1.

TABLE II
EVALUATION RETURNS ON FOUR DIFFERENT COMPOSITION METHODS FOR THE MULTIPLE FACTORS THAT INFLUENCE THE REWARDS. THE AVERAGE ± STANDARD DEVIATION IS SHOWN FOR FIVE RANDOM SEEDS.

Tasks	Method 1	Method 2	Method 3	Method 4
Hammer	$\mathbf{4532 \pm 95}$	3158 ± 242	2185 ± 262	1034 ± 147
Sweep Into	$\mathit{2257} \pm 324$	1971 ± 243	1734 ± 276	665 ± 185
Button Press	$\mathit{3435} \pm 106$	3145 ± 227	2576 ± 85	1089 ± 290
Open Drawer	$\mathbf{4466 \pm 41}$	2998 ± 64	2514 ± 54	1727 ± 75
Cheetah Run	$\mathit{430} \pm 55$	361 ± 138	255 ± 126	183 ± 74
Walker Walk	$\mathit{950} \pm 155$	714 ± 83	463 ± 76	262 ± 38
Fish Swim	$\mathit{576} \pm 155$	453 ± 26	296 ± 89	198 ± 29
Humanoid Run	$\mathbf{359 \pm 57}$	201 ± 31	154 ± 38	26 ± 4
Cartpole Swingup	$\mathbf{642 \pm 68}$	441 ± 29	349 ± 18	208 ± 17

A noteworthy revelation emerged when we explored Method 4, where both states and actions were considered as single factors. This approach exhibited a stark drop in performance, and in some slightly complex environments, it led to outright failures. This finding underscores the critical importance of factorization and highlights the perils of oversimplifying the reward-shaping process.

In light of these insights, we conclude that decomposing both states and actions into multiple factors and seamlessly integrating them using f_Θ stands as the most effective strategy. This sophisticated approach enables a finer level of granularity in reward learning and significantly enhances the overall efficacy of policies trained on datasets subjected to such comprehensive factorization.

Image-based experiments TRAIN showcases a remarkable

degree of adaptability that extends beyond conventional full-state tasks, seamlessly accommodating image-based tasks with equal prowess. In our quest to assess TRAIN's performance in the realm of image-based tasks, we ingeniously conditioned the environment's output to generate images, thus opening up exciting possibilities for visual-based learning scenarios.

Furthermore, we integrated a widely endorsed strategy employed in diverse task domains, wherein we treated multiple frames as a single time-step state. These frames underwent a meticulous process of decomposition into interdependent frames, and each frame was subsequently subjected to further dissection into RGB channels, effectively treating each single image as an individual factor. The experimentation with TRAIN spanned a diverse set of environments, encompassing four distinct Meta-World environments and an additional five environments sourced from the esteemed DeepMind Control Suite. The breadth of this evaluation allowed us to gain comprehensive insights into TRAIN's capabilities in a variety of settings. The results are shown in Fig. 7.

The experimental results that emerged from these rigorous trials underscored TRAIN's robust performance in image-based experiments. This underscores the effectiveness of our approach in seamlessly integrating multiple images within a single time step, each image serving as a distinct factor. This multi-faceted approach capitalizes on the richness of information inherent in each image, ultimately enhancing the depth and quality of the learning process.

D. Accurate of inferred rewards

To rigorously assess the predictive accuracy of our approach, TRAIN, across varying ratios of labeled data, we embarked on a comprehensive evaluation campaign. Our experiments spanned a diverse range of environments, encompassing four challenging Meta-World scenarios and an additional five environments sourced from the prestigious DeepMind Control Suite.

In Fig. 8, we present a vivid representation of our findings, utilizing the mean squared error (MSE) as our primary evaluation metric. This heatmap graphically depicts the relationship between multiple tasks (on the horizontal axis) and the ratio of reward-labeled data in the dataset (on the vertical axis). Each value in the figure reflects the MSE associated with a specific task under the corresponding labeled data ratio. To ensure the robustness of our findings, we partitioned the dataset into multiple batches, with the calculated result representing the average MSE value across these batches.

The compelling insights derived from our experiments reveal a clear trend: as the ratio of reward-labeled data in the dataset increases, the corresponding MSE values decrease, indicating a higher degree of predictive accuracy. Conversely, a lower ratio of labeled reward data in the dataset is associated with higher MSE values, signifying a relatively lower predictive accuracy. Furthermore, our observations indicate that tasks characterized by higher-dimensional states and actions tend to exhibit elevated MSE values, suggesting that predictive challenges are more pronounced in these complex settings.

These findings offer valuable insights into the performance of TRAIN across a spectrum of labeled data ratios and task complexities, shedding light on its capabilities and areas for potential refinement. Such detailed evaluations are instrumental in understanding the nuances of our approach's predictive accuracy and provide a roadmap for its application in real-world scenarios across various domains.

E. Accurate of inferred rewards on different norms

To assess the accuracy of our proposed method, TRAIN, in relation to different norms, we conducted experiments using four Meta-World and five DeepMind Control Suite environments. Fig. 8 illustrates the results, utilizing the mean squared error (MSE) as the evaluation metric.

It's important to note that the 1.5 norm and the 2.5 norm are introduced purely for experimentation purposes and lack specific physical interpretations. These two norms are included to investigate the impact of norm selection on our method.

In the heatmap of the experimental outcomes, the horizontal axis represents various tasks, the vertical axis corresponds to different norms, and the values within the figure indicate the MSE for each task under a specific norm. The dataset was partitioned into multiple batches, and the calculated result represents the average MSE across these batches.

From the experimental results, it is evident that different norms have minimal influence on the MSE, indicating that our method TRAIN is not sensitive to the choice of norm, underscoring its robustness in various scenarios. Further experimental analysis can be found in Appendix D.

VI. Discussion

This paper adopts a transductive inference manner to learn the reward function for offline RL. The proposed TRAIN algorithm infers rewards for unlabeled state-action pairs and the experiments demonstrate the effectiveness of inferred rewards in agent training. From the above positive insight and effective performance, we would like to explore the potential mystery of this manner and clarify some interesting observations.

A. Reward inference mystery

Inductive inference builds models by identifying hidden patterns in state-action pairs that have rewards, aiming to generalize these findings to unseen pairs without rewards [59], [60]. Conversely, transductive learning exposes the model to both rewarded and unrewarded state-action pairs during the training phase [61], [62]. Our approach focuses on discerning patterns across all pairs and uses this knowledge to predict rewards for those without labels. Each task is distinguished by a unique array of state features, necessitating predictions for many unlabeled state-action pairs based on a smaller subset that contains rewards. Unlike inductive inference, which requires a large number of rewarded pairs to develop a generalizable model, transductive inference is more suitable for our purposes.

To augment the proposed method of transductive inference to enable it to support inductive reward inference, we consider integrating more advanced feature extraction techniques that identify fundamental patterns common across various tasks. Additionally, we aim to shift our focus from merely propagating reward information from labeled pairs to inferring the

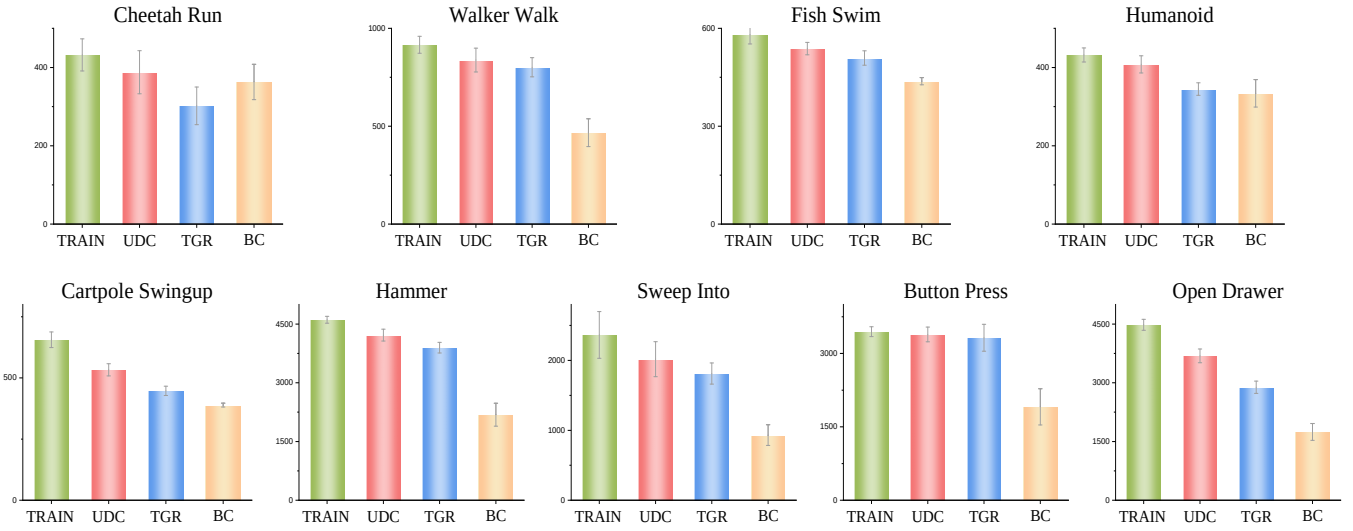


Fig. 7. Evaluation returns on the nine image-based tasks. The vertical lines depict the standard deviation across five random seeds of each experiment.

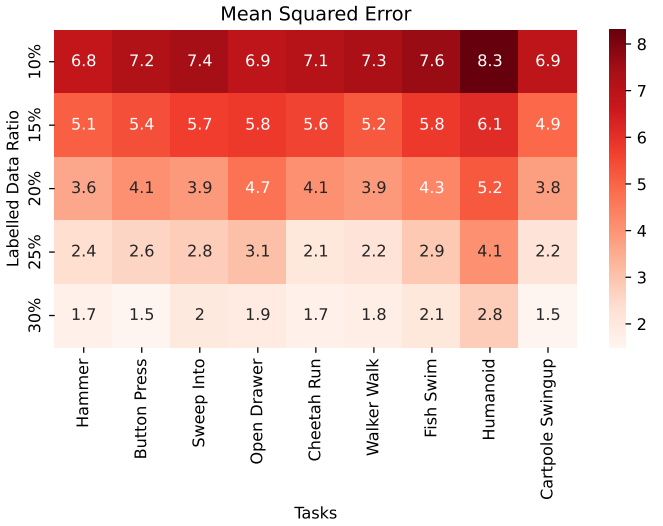


Fig. 8. The accuracy between predicted labels and ground truth labels of TRAIN under different labelled data ratios on four Meta-World and five DeepMind Control Suite environments, respectively. The evaluation metric is the mean squared error (MSE).

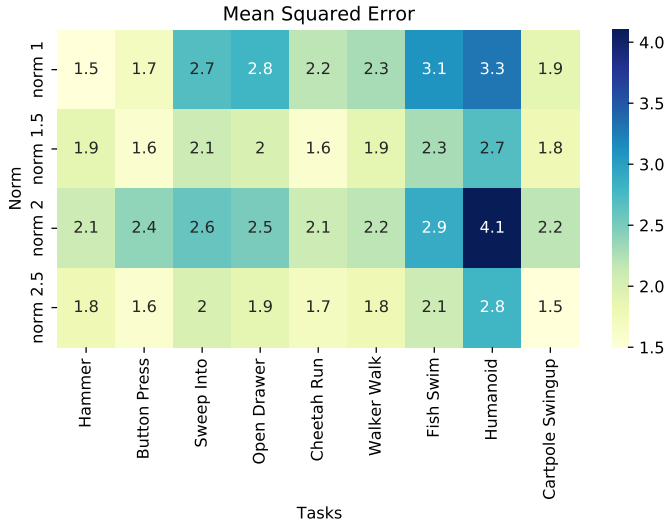


Fig. 9. The accuracy between predicted labels and ground truth labels of TRAIN under different norms on four Meta-World and five DeepMind Control Suite environments, respectively. The evaluation metric is the mean squared error (MSE).

underlying reward function from a wider range of state-action characteristics. This enhancement will help the model learn features that are broadly applicable, and enhance the model's applicability across different scenarios.

B. Significant deviation instability

From the experimental results, our method demonstrated stable performance across most tasks. We also observed some instances of instability. In the Meta-World domain, seven tasks exhibited significant variance due to their complexity and the random initialization of crucial elements, which led to substantial diversity in states. For instance, the task "pick-out-of-hole" involves picking up a puck from a hole with puck and goal positions randomized at each initialization. Similarly, the "box-close" task requires grasping a cover and closing the box, with the positions of the cover and box also randomized each time. In the DM Control domain, two tasks displayed considerable variance due to their complex reward structures and the multitude of factors influencing rewards. For example, in "fish swim," there are two distinct tasks: the upright task, where the fish is rewarded solely for righting itself vertically, and the swim task, where it is also rewarded for swimming towards a target. This setup results in a diverse array of reward-earning states.

The high diversity of collected states and the numerous state elements linked to rewards present challenges for reward learning. When integrating various factors that influence rewards into the graph, some oscillations occur. Moreover, because our scenarios feature a low proportion of reward-bearing states, problems such as insufficient or excessive smoothing and oscillations in the reward inference process may arise. This can lead to considerable performance variability when training agents using the annotated dataset. To address this issue, we could consider implementing feature extraction and integration techniques that capture the specific underlying

patterns or rules of the states for each task.

VII. Conclusion

In conclusion, our research proposes the TRAIN method to address a critical challenge in offline reinforcement learning by developing a reward inference method that leverages a constrained number of human reward annotations to estimate rewards for unlabelled data. TRAIN models MDPs as a graph and leverage the contextual properties of information propagation of the graph structure to construct a reward propagation graph that incorporates various influential factors, facilitating transductive reward inference. We have shown the existence of a fixed point during the iterative inference process, and our method converges at least to a local optimum. Empirical evaluations on locomotion and robotic manipulation tasks demonstrate the effectiveness of TRAIN, especially when dealing with limited reward annotations. This work has significant implications for practical scenarios where reward functions are challenging to access.

References

[1] S. Lange, T. Gabel, and M. Riedmiller, "Batch reinforcement learning," in *Reinforcement learning*. Springer, 2012, pp. 45–73.

[2] S. Levine, A. Kumar, G. Tucker, and J. Fu, "Offline reinforcement learning: Tutorial, review, and perspectives on open problems," *arXiv preprint arXiv:2005.01643*, 2020.

[3] R. F. Prudencio, M. R. Maximo, and E. L. Colombini, "A survey on offline reinforcement learning: Taxonomy, review, and open problems," *IEEE Transactions on Neural Networks and Learning Systems*, 2023.

[4] S. Cabi, S. G. Colmenarejo, A. Novikov, K. Konyushkova, S. Reed, R. Jeong, K. Zolna, Y. Aytar, D. Budden, M. Vecerik *et al.*, "Scaling data-driven robotics with reward sketching and batch reinforcement learning," *arXiv preprint arXiv:1909.12200*, 2019.

[5] S. Dasari, F. Ebert, S. Tian, S. Nair, B. Bucher, K. Schmeckpeper, S. Singh, S. Levine, and C. Finn, "Robonet: Large-scale multi-robot learning," in *Conference on Robot Learning*. PMLR, 2020, pp. 885–897.

[6] F. Yu, W. Xian, Y. Chen, F. Liu, M. Liao, V. Madhavan, and T. Darrell, "Bdd100k: A diverse driving video database with scalable annotation tooling," *arXiv preprint arXiv:1805.04687*, vol. 2, no. 5, p. 6, 2018.

[7] A. Strehl, J. Langford, L. Li, and S. M. Kakade, "Learning from logged implicit exploration data," *Advances in neural information processing systems*, vol. 23, 2010.

[8] L. Bottou, J. Peters, J. Quiñonero-Candela, D. X. Charles, D. M. Chickering, E. Portugaly, D. Ray, P. Simard, and E. Snelson, "Counterfactual reasoning and learning systems: The example of computational advertising." *Journal of Machine Learning Research*, vol. 14, no. 11, 2013.

[9] S. M. Shortreed, E. Laber, D. J. Lizotte, T. S. Stroup, J. Pineau, and S. A. Murphy, "Informing sequential clinical decision-making through reinforcement learning: an empirical study," *Machine learning*, vol. 84, no. 1, pp. 109–136, 2011.

[10] S. Cabi, S. G. Colmenarejo, A. Novikov, K. Konyushkova, S. Reed, R. Jeong, K. Zolna, Y. Aytar, D. Budden, M. Vecerik *et al.*, "Scaling data-driven robotics with reward sketching and batch reinforcement learning," *In Robotics: Science and Systems Conference*, 2020.

[11] K. Zolna, A. Novikov, K. Konyushkova, C. Gulcehre, Z. Wang, Y. Aytar, M. Denil, N. de Freitas, and S. Reed, "Offline learning from demonstrations and unlabeled experience," *arXiv preprint arXiv:2011.13885*, 2020.

[12] K. Konyushkova, K. Zolna, Y. Aytar, A. Novikov, S. Reed, S. Cabi, and N. de Freitas, "Semi-supervised reward learning for offline reinforcement learning," *arXiv preprint arXiv:2012.06899*, 2020.

[13] T. Yu, A. Kumar, Y. Chebotar, K. Hausman, C. Finn, and S. Levine, "How to leverage unlabeled data in offline reinforcement learning," in *International Conference on Machine Learning*. PMLR, 2022, pp. 25 611–25 635.

[14] X. Zhu and Z. Ghahramani, "Learning from labeled and unlabeled data with label propagation," *ProQuest number: information to all users*, 2002.

[15] X. Zhu, *Semi-supervised learning with graphs*. Carnegie Mellon University, 2005.

[16] D. Zhou, O. Bousquet, T. Lal, J. Weston, and B. Schölkopf, "Learning with local and global consistency," *Advances in neural information processing systems*, vol. 16, 2003.

[17] X. Zhang and W. Lee, "Hyperparameter learning for graph based semi-supervised learning algorithms," *Advances in neural information processing systems*, vol. 19, 2006.

[18] F. Wang and C. Zhang, "Label propagation through linear neighborhoods," in *Proceedings of the 23rd international conference on Machine learning*, 2006, pp. 985–992.

[19] M. Karasuyama and H. Mamitsuka, "Manifold-based similarity adaptation for label propagation," *Advances in neural information processing systems*, vol. 26, 2013.

[20] Y. Liu, J. Lee, M. Park, S. Kim, E. Yang, S. J. Hwang, and Y. Yang, "Learning to propagate labels: Transductive propagation network for few-shot learning," in *International Conference on Learning Representations*, 2019.

[21] Y. Tassa, Y. Doron, A. Muldal, T. Erez, Y. Li, D. d. L. Casas, D. Budden, A. Abdolmaleki, J. Merel, A. Lefrancq *et al.*, "Deepmind control suite," *arXiv preprint arXiv:1801.00690*, 2018.

[22] T. Yu, D. Quillen, Z. He, R. Julian, K. Hausman, C. Finn, and S. Levine, "Meta-world: A benchmark and evaluation for multi-task and meta reinforcement learning," in *Conference on robot learning*. PMLR, 2020, pp. 1094–1100.

[23] H. Wang and J. Leskovec, "Unifying graph convolutional neural networks and label propagation," *arXiv preprint arXiv:2002.06755*, 2020.

[24] S. Fujimoto, D. Meger, and D. Precup, "Off-policy deep reinforcement learning without exploration," in *International conference on machine learning*. PMLR, 2019, pp. 2052–2062.

[25] Q. Wang, J. Xiong, L. Han, H. Liu, T. Zhang *et al.*, "Exponentially weighted imitation learning for batched historical data," *Advances in Neural Information Processing Systems*, vol. 31, 2018.

[26] X. Chen, Z. Zhou, Z. Wang, C. Wang, Y. Wu, and K. Ross, "Bail: Best-action imitation learning for batch deep reinforcement learning," *Advances in Neural Information Processing Systems*, vol. 33, pp. 18 353–18 363, 2020.

[27] N. Y. Siegel, J. T. Springenberg, F. Berkenkamp, A. Abdolmaleki, M. Neunert, T. Lampe, R. Hafner, N. Heess, and M. Riedmiller, "Keep doing what worked: Behavioral modelling priors for offline reinforcement learning," in *International Conference on Learning Representations*, 2020.

[28] X. B. Peng, A. Kumar, G. Zhang, and S. Levine, "Advantage-weighted regression: Simple and scalable off-policy reinforcement learning," *arXiv preprint arXiv:1910.00177*, 2019.

[29] Z. Wang, A. Novikov, K. Zolna, J. S. Merel, J. T. Springenberg, S. E. Reed, B. Shahriari, N. Siegel, C. Gulcehre, N. Heess *et al.*, "Critic regularized regression," *Advances in Neural Information Processing Systems*, vol. 33, pp. 7768–7778, 2020.

[30] I. Kostrikov, R. Fergus, J. Tompson, and O. Nachum, "Offline reinforcement learning with fisher divergence critic regularization," in *International Conference on Machine Learning*. PMLR, 2021, pp. 5774–5783.

[31] P. Abbeel and A. Y. Ng, "Apprenticeship learning via inverse reinforcement learning," in *Proceedings of the twenty-first international conference on Machine learning*, 2004, p. 1.

[32] A. Y. Ng, S. Russell *et al.*, "Algorithms for inverse reinforcement learning." in *Icml*, vol. 1, 2000, p. 2.

[33] J. Ho and S. Ermon, "Generative adversarial imitation learning," *Advances in neural information processing systems*, vol. 29, 2016.

[34] A. Edwards, C. Isbell, and A. Takanishi, "Perceptual reward functions," *arXiv preprint arXiv:1608.03824*, 2016.

[35] A. Singh, L. Yang, K. Hartikainen, C. Finn, and S. Levine, "End-to-end robotic reinforcement learning without reward engineering," *arXiv preprint arXiv:1904.07854*, 2019.

[36] M. Klissarov and D. Precup, "Reward propagation using graph convolutional networks," *Advances in Neural Information Processing Systems*, vol. 33, pp. 12 895–12 908, 2020.

[37] N. Baram, O. Anschel, I. Caspi, and S. Mannor, "End-to-end differentiable adversarial imitation learning," in *International Conference on Machine Learning*. PMLR, 2017, pp. 390–399.

[38] C. Finn, S. Levine, and P. Abbeel, "Guided cost learning: Deep inverse optimal control via policy optimization," in *International conference on machine learning*. PMLR, 2016, pp. 49–58.

[39] J. Fu, K. Luo, and S. Levine, "Learning robust rewards with adversarial inverse reinforcement learning," *arXiv preprint arXiv:1710.11248*, 2017.

[40] Y. Li, J. Song, and S. Ermon, "Infogail: Interpretable imitation learning from visual demonstrations," *Advances in Neural Information Processing Systems*, vol. 30, 2017.

[41] J. Merel, Y. Tassa, D. TB, S. Srinivasan, J. Lemmon, Z. Wang, G. Wayne, and N. Heess, "Learning human behaviors from motion capture by adversarial imitation," *arXiv preprint arXiv:1707.02201*, 2017.
[42] P. Sermanet, K. Xu, and S. Levine, "Unsupervised perceptual rewards for imitation learning," *arXiv preprint arXiv:1612.06699*, 2016.
[43] Y. Zhu, Z. Wang, J. Merel, A. Rusu, T. Erez, S. Cabi, S. Tunyasuvunakool, J. Kramár, R. Hadsell, N. de Freitas *et al.*, "Reinforcement and imitation learning for diverse visuomotor skills," *arXiv preprint arXiv:1802.09564*, 2018.
[44] K. Zolna, S. Reed, A. Novikov, S. G. Colmenarejo, D. Budden, S. Cabi, M. Denil, N. de Freitas, and Z. Wang, "Task-relevant adversarial imitation learning," in *Conference on Robot Learning*. PMLR, 2021, pp. 247–263.
[45] V. N. Vapnik, "An overview of statistical learning theory," *IEEE transactions on neural networks*, vol. 10, no. 5, pp. 988–999, 1999.
[46] T. Joachims *et al.*, "Transductive inference for text classification using support vector machines," in *Icml*, vol. 99, 1999, pp. 200–209.
[47] M. Rohrbach, S. Ebert, and B. Schiele, "Transfer learning in a transductive setting," *Advances in neural information processing systems*, vol. 26, 2013.
[48] Y. Fu, T. M. Hospedales, T. Xiang, and S. Gong, "Transductive multi-view zero-shot learning," *IEEE transactions on pattern analysis and machine intelligence*, vol. 37, no. 11, pp. 2332–2345, 2015.
[49] M. Belkin and P. Niyogi, "Semi-supervised learning on manifolds," *Machine Learning Journal*, vol. 1, 2002.
[50] A. Blum and S. Chawla, "Learning from labeled and unlabeled data using graph mincuts," in *Proceedings of the Eighteenth International Conference on Machine Learning*, 2001, pp. 19–26.
[51] O. Chapelle, J. Weston, and B. Schölkopf, "Cluster kernels for semi-supervised learning," *Advances in neural information processing systems*, vol. 15, 2002.
[52] X. Zhu, Z. Ghahramani, and J. D. Lafferty, "Semi-supervised learning using gaussian fields and harmonic functions," in *Proceedings of the 20th International conference on Machine learning (ICML-03)*, 2003, pp. 912–919.
[53] A. Iscen, G. Tolias, Y. Avrithis, and O. Chum, "Label propagation for deep semi-supervised learning," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 5070–5079.
[54] S. Tunyasuvunakool, A. Muldal, Y. Doron, S. Liu, S. Bohez, J. Merel, T. Erez, T. Lillicrap, N. Heess, and Y. Tassa, "dm_control: Software and tasks for continuous control," *Software Impacts*, vol. 6, p. 100022, 2020.
[55] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," in *International conference on machine learning*. PMLR, 2018, pp. 1861–1870.
[56] C. Gulcehre, Z. Wang, A. Novikov, T. Le Paine, S. G. Colmenarejo, K. Zolna, R. Agarwal, J. Merel, D. Mankowitz, C. Paduraru *et al.*, "Rl unplugged: Benchmarks for offline reinforcement learning," *Advances in Neural Information Processing Systems*, vol. 33, pp. 7248–7259, 2020.
[57] G. Barth-Maron, M. W. Hoffman, D. Budden, W. Dabney, D. Horgan, D. Tb, A. Muldal, N. Heess, and T. Lillicrap, "Distributed distributional deterministic policy gradients," *arXiv preprint arXiv:1804.08617*, 2018.
[58] M. W. Hoffman, B. Shahriari, J. Aslanides, G. Barth-Maron, N. Momchev, D. Sinopalnikov, P. Stańczyk, S. Ramos, A. Raichuk, D. Vincent *et al.*, "Acme: A research framework for distributed reinforcement learning," *arXiv preprint arXiv:2006.00979*, 2020.
[59] S. Yi, F. Nie, Y. Liang, W. Liu, Z. He, and Q. Liao, "Fast extended inductive robust principal component analysis with optimal mean," *IEEE Transactions on Knowledge and Data Engineering*, vol. 34, no. 10, pp. 4812–4825, 2022.
[60] R. A. Rossi, R. Zhou, and N. K. Ahmed, "Deep inductive graph representation learning," *IEEE Transactions on Knowledge and Data Engineering*, vol. 32, no. 3, pp. 438–452, 2020.
[61] X. Kong, M. K. Ng, and Z.-H. Zhou, "Transductive multilabel learning via label set propagation," *IEEE Transactions on Knowledge and Data Engineering*, vol. 25, no. 3, pp. 704–719, 2013.
[62] V. N. Vapnik, V. Vapnik *et al.*, "Statistical learning theory," 1998.

Bohao Qu is currently pursuing the Ph.D. degree with the School of Artificial Intelligence, Jilin University, Changchun, China. He is also a visiting student with the A*STAR Centre for Frontier AI Research, Singapore. Before that, he was an engineer of Jingdong Mall Research, Beijing, China, from 2017 to 2019. His current research interests include deep reinforcement learning, machine learning, and hyperbolic (non-Euclidean) geometry.

Xiaofeng Cao received his Ph.D. degree at Australian Artificial Intelligence Institute, University of Technology Sydney, Australia, and was a visiting scholar at the Hong Kong University of Science and Technology. He is currently an Associate Professor at the School of Artificial Intelligence, Jilin University, China, and leading the Machine Perceptron Research Group with more than 20 members. He has more than 30 academic works published in IEEE T-PAMI, TNNLS, ICML, NeurIPS, ICLR, ECML, etc, and served as their PC members/reviewers. His research interests include the PAC learning theory, agnostic learning algorithm, generalization analysis, and hyperbolic (non-Euclidean) geometry.

Qing Guo received his Ph.D. degree in computer application technology from the School of Computer Science and Technology, Tianjin University, China. He is currently a senior research scientist and principal investigator (PI) at the Center for Frontier AI Research (CFAR), A*STAR in Singapore. He is also an adjunct assistant professor at the National University of Singapore (NUS), and senior PC member of AAAI. Before that, he was a Wallenberg-NTU Presidential Postdoctoral Fellow with the Nanyang Technological University, Singapore. His research interests include computer vision, AI security, and image processing. He is a member of IEEE.

Yi Chang is the dean of the School of Artificial Intelligence, Jilin University, Changchun, China. He was elected as a Chinese National Distinguished professor, in 2017 and an ACM distinguished scientist, in 2018. Before joining academia, he was the technical vice president with Huawei Research America, and the research director with Yahoo Labs. He is the author of two books and more than 100 papers in top conferences or journals. His research interests include information retrieval, data mining, machine learning, natural language processing, and artificial intelligence. He won the Best Paper Award on KDD'2016 and WSDM'2016. He has served as a Conference general chair for WSDM'2018 and SIGIR'2020.

Ivor W.Tsang (Fellow IEEE) is Director of A*STAR Centre for Frontier AI Research (CFAR) since Jan 2022. He is also an Adjunct Professor at School of Computer Science and Engineering (SCSE), Nanyang Technological University, Singapore. Previously, he was a Professor of Artificial Intelligence, at University of Technology Sydney (UTS), and Research Director of the Australian Artificial Intelligence Institute (AAII), the largest AI institute in Australia, which is the key player to drive the University of Technology Sydney to rank 10th globally and 1st in Australia for AI research, in the latest AI Research Index. Prof Tsang is working at the forefront of big data analytics and Artificial Intelligence. His research focuses on transfer learning, deep generative models, learning with weakly supervision, big data analytics for data with extremely high dimensions in features, samples and labels. His work is recognised internationally for its outstanding contributions to those fields.

Chengqi Zhang Chengqi Zhang received a Ph.D. from the University of Queensland, Brisbane, Australia, in 1991 and a DSc (higher doctorate) from Deakin University, Geelong, Australia, in 2002. Since December 2001, he has been a Professor of Information Technology with the University of Technology Sydney (UTS), Australia, where he was Director of the UTS Priority Investment Research Centre for Quantum Computation and Intelligent Systems from 2008-2016. His research interests mainly focus on data mining and its applications. He was General Co-Chair of KDD 2015 in Sydney, the Local Arrangements Chair of IJCAI-2017 in Melbourne, and a General Chair of IJCAI-2024 in Jeju Island, and is a Fellow of the Australian Computer Society and a Senior Member of the IEEE.