PoRank: A Practical Framework for Learning to Rank Policies

Pengjie Gu[1], **Mengchen Zhao**[2,†], **Xu He**[3] and **Yi Cai**[2] and **Bo An**[1,4]
[1]School of Computer Science and Engineering, Nanyang Technological University, Singapore
[2]School of Software Engineering, South China University of Technology, China
[3]Huawei Noah' Ark Lab
[4]Skywork AI, Singapore
{pengjie.gu, boan}@ntu.edu.sg, {zzmc, ycai}@scut.edu.cn, hexu27@huawei.com

Abstract

In many real-world scenarios, we need to select from a set of candidate policies before online deployment. Although existing off-policy evaluation (OPE) methods can be used to estimate the online performance, they suffer from high variance. Fortunately, we care only about the ranking of the candidate policies, rather than their exact online rewards. Based on this, we propose a novel framework PoRank for learning to rank policies. In practice, learning to rank policies faces two main challenges: 1) generalization over the huge policy space and 2) lack of supervision signals. To overcome the first challenge, PoRank uses a Policy Comparison Transformer (PCT) for learning cross-policy representations, which capture the core discrepancies between policies and generalizes well across the whole policy space. The second challenge arises because learning to rank requires online comparisons of policies as ground-truth labels, whereas deploying policies online might be highly expensive. To overcome this, PoRank adopts a crowdsourcing based learning-to-rank (LTR) framework, where a set of OPE algorithms are employed to provide weak comparison labels. Experimental results show that PoRank not only outperforms baselines when the ground-truth labels are provided, but also achieves competitive performance when the ground-truth labels are unavailable.

1 Introduction

[1] In many real-world scenarios such as trading [Zhang *et al.*, 2020], advertising [Cai *et al.*, 2023], autonomous vehicles [Shi *et al.*, 2021], and drug trials [Yang *et al.*, 2023], the task often involves selecting the most promising policy from a set of candidates prior to online deployment. This selection is critical, as online evaluation of each policy can be costly and potentially risky. The conventional approach for policy selection is to estimate the online performance of candidate policies via Off-Policy Evaluation (OPE) [Paduraru, 2013]. OPE requires only off-policy data, which means that we can estimate the performance of a target policy using data generated by other policies.

Although OPE is a promising direction, existing OPE methods are still far from reliable in practice. For example, standard Inverse Propensity Scoring (IPS) based estimators such as importance sampling suffer from high variance due to the product of importance weights [Hanna *et al.*, 2019]. Direct Methods (DM) requires extra estimators of environmental dynamics or value functions, which are hard to learn when the observation data is high-dimensional or insufficient. Hybrid Methods (HM) such as doubly robust estimators combine IPS and DM [Jiang and Li, 2016], yet it often comes with additional hyperparameters that need to be carefully chosen.

Fortunately, in many real-world scenarios, we do not need to estimate the exact online performance of candidate policies. Instead, we only care about which policy would perform the best when deployed online. This inspires us to develop a policy ranker focusing on predicting the ranking of target policies regarding to their online performance. Learning such a policy ranker is similar to learning item rankers in recommender systems [Liu, 2009]. However, learning to rank policies faces two unique challenges. First, the policy space could be extremely large, even in simple environments. Therefore, a policy ranker with poor generalization ability would fail to rank unseen policies. Second, unless deployed online, we can hardly know the real performance of the policies, which means that we are lack of ground-truth labels for training the policy ranker. These challenges make it extremely hard to train useful policy rankers in real-world tasks.

In this paper, we propose a novel and practical framework called PoRank for learning to rank policies. PoRank composes of a Policy Comparison Transformer (PCT) and a learning-to-rank (LTR) module, where the PCT aims to learn compact representations of policy pairs and the LTR module focuses on predicting the order of input policies. Instead of directly encoding the raw trajectories to policy representations (as is done in existing works), PCT encodes different behaviors of two policies at the same sates to represent a policy pair. In such a way, PCT successfully extracts the most useful features for policy ranking and significantly improves the generalization over the policy space. The LTR module is built upon the PCT. Conventionally, given two input policies, LTR requires a label to tell which policy performs bet-

[1]† Corresponding author.

ter. If we can easily obtain such labels, for example in the game environments, we can directly train the policy ranker using supervised learning. However, in many industrial scenarios, obtaining such labels could be very expensive due to deployment cost. To resolve this, we propose a novel crowdsourcing based LTR module, where a set of OPE methods are employed to estimate the policy comparison labels. Specifically, these labels are constructed by comparing the estimated accumulated rewards of the target policies. In such a way, we can easily collect plenty of labels without deploying training policies online. In the experiments, we first demonstrate the advantage of the PCT architecture by comparing it with the state-of-the-art policy ranking network, where both networks are trained using ground-truth labels. Hence, we show that a simple crowdsourcing mechanism could compensate for the lack of ground-truth labels, which makes PoRank practical in many real-world scenarios.

2 Related Works

Off-Policy Evaluation/Selection/Ranking The goal of OPE is to precisely predict the online performance of target policies given trajectory data collected by some other behavior policies. Standard importance sampling approach suffers from exponential variance with respect to the time horizon [Li *et al.*, 2015; Jiang and Li, 2016]. Recent works such as Fitted-Q evaluation [Hoang *et al.*, 2019] and marginalized importance sampling [Liu *et al.*, 2018] achieve polynomial variance, yet they rely on additional function approximators. Direct methods avoid the large variance by learning the dynamic model or Q-function, which could be biased especially when the data is insufficient. Some works study the offline policy selection problem, yet their methods require running from scratch for each candidate policy [Zhang and Jiang, 2021; Mengjiao *et al.*, 2022]. By contrast, in offline policy ranking (OPR) problem we aim to develop a policy ranker that could directly rank policies. A recent work on OPR collects online performance of a set of policies and uses these labeled data to train the policy ranker [Jin *et al.*, 2022]. However, collecting such data might be extremely expensive in practice.

Learning from Crowds Crowdsourcing systems enable machine learners to collect labels of large datasets from crowds. One big issue with crowdsourcing is that the labels provided by crowds are often noisy [S. and Zhang, 2019]. To tackle this challenge, various probabilistic generative methods are proposed for statistical inference [Yuchen *et al.*, 2016; Tian and Zhu, 2015]. Another line of works use discriminative models that find the most likely label for each instance [Jing *et al.*, 2014; Jing *et al.*, 2015]. A recently work called Crowd Layer (CL) first describes an algorithm for jointly learning the target model and the reliability of workers [Filipe and Pereira, 2018]. CL proposes a simple yet efficient crowd layer that can train deep neural networks end-to-end directly from the noisy labels. In our work, we treat existing OPE methods as workers and adopt CL to process multiple labels due to its simplicity and effectiveness.

Policy Representations Compact but informative representations of policies not only benefit the policy learning process [Tang *et al.*, 2022], but also help with the policy transfer among different tasks [Isac *et al.*, 2019; G. *et al.*, 2017]. A straightforward idea is to represent a policy by its network parameters, yet this leads to a very sparse representation space. Network Fingerprint [Harb *et al.*, 2020] proposes a differentiable representation that uses the concatenation of the vectors of actions outputted by the policy network on a set of probing states. Some recent works try to encode policy parameters as well as state-action pair data into a low-dimensional embedding space [Tang *et al.*, 2022; Jin *et al.*, 2022]. However, existing works focus on single policy representations, which fail to capture the relative discrepancies between different policies.

3 Problem Statement

Markov Decision Process We consider the underlying environment as a Markov decision process (MDP) and define an MDP as a finite–horizon tuple $\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{P}, \mathcal{R}, \gamma)$. Here, $\mathcal{S}$ is the state space, and $\mathcal{A}$ is the action space. $\mathcal{T}$ is the length of time horizon. $\mathcal{P}$ and $\mathcal{R}$ are the transition function and the reward function, respectively. $\mathcal{P}(s_{t+1}|s_t, a_t)$ represents the probability of transitioning from state s_t to state $s_{t+1} \in \mathcal{S}$ when the agent takes action $a_t \in \mathcal{A}$ under state $s_t \in \mathcal{S}$ and $\mathcal{R}(s_t, a_t)$ represents the immediate reward the agent receives. The expected return of a policy π can be computed by $\mathbb{E}_{\mathcal{P}}\Big[\sum_{t=1}^{\mathcal{T}}\left[\gamma^t \mathcal{R}(s_t, \pi(s_t))\right]\Big]$, where $\gamma \in (0, 1]$ is the discount factor.

Ranking Policies without Online Deployment The goal of OPE is to estimate the expected return of a policy π without deploying it online, given an offline dataset $\mathcal{D} = \left\{\tau^k\right\}_{k=1}^N$, where $\tau^k = \left(s_0, a_0, r_0, \cdots, s_{\mathcal{T}}, a_{\mathcal{T}}, r_{\mathcal{T}}\right)^k$ are trajectories generated by some behavior policies. OPE is usually used for model selection: We are required to select the most promising policy from a candidate set of available policies before actual deployment. Take recommender systems as example, we can easily obtain a set of candidate policies by adjusting the training data or the hyperparameters of the model. However, we often need to select very few policies from the candidates for online test, since a bad policy would harm the user experience. Therefore, we care more about the ranking of the candidate policies, instead of their exact expected reward. We formally define the off-policy ranking problem as follows.

Definition 1 (Offline Policy Ranking, OPR)**.** Given a set of trajectory data $\mathcal{D} = \left\{\tau^k\right\}_{k=1}^N$ generated by some behavior policies and a set of target policies $\Pi = \left\{\pi^i\right\}_{i=1}^M$, the OPR problem is to seek a ranking of the target policies that aligns with their online expected accumulated rewards.

Intuitively, OPR is a simpler problem than OPE because we do not care about the exact online performace of the candidate polcies. While OPE as well as off-policy selection (OPS) can also be used to rank candidate policies, they often need to inputrun an estimation procedure from scratch for each candidate policy, thus leading to poor efficiency. In this work, we aim to learn an universal policy ranker that could directly return the ranking results for any input candidate policies.

4 Approach

In this section, we will elaborate our PoRank framework. PoRank borrows the idea from learning-to-rank literature and explicitly solves two unique challenges in ranking policies. Firstly, given the vastness of the policy space, we need an effective policy representation scheme so that the ranking module can accurately comprehend and compare different policies. Secondly, although we can easily generate many policies for training, we can hardly know their actual online performance, therefore we are lack of ground-truth labels for training the ranker. Overall, PoRank consists of a Policy Comparison Transformer (PCT) and a Learning-to-Rank (LTR) module, as shown in Figure 2(a). We will introduce each of them in the following subsections.

4.1 Learning Cross-policy Representations

Cross-policy Representation A policy can be considered as a conditional distribution over actions given the state. Therefore, a policy can be naturally represented by a set of state-action pairs where the actions are sampled from the policy. However, such a naive policy representation could be inefficient since the number of state-action pairs can be extremely large. Previous works address this issue by extracting high-level features from the state-action pairs using deep neural networks [Jin *et al.*, 2022]. Although these representations reflect the features of single policies, they fail to capture the discrepancies of different policies at some crucial states.

To this end, we aim to learn cross-policy representations by comparing two policies' decisions at the same set of states. Formally, given a set of states $\{s_1, ..., s_K\}$ and two policies π^i, π^j, we can construct the following sequence of state-action pairs by taking actions at these states:

$$\xi^{i \succ j} = \left\{(s_1, a_1^i), (s_1, a_1^j), \cdots, (s_K, a_K^i), (s_K, a_K^j)\right\}, \quad (1)$$

where $a^i \sim \pi^i(\cdot|s)$, and $a_j \sim \pi^j(\cdot|s)$. We denote by $\chi^{i \succ j} = g(\xi^{i \succ j}) \in \mathbb{R}^n$ the cross-policy representation where g is a function that maps $\xi^{i \succ j}$ to an n-dimensional representation space. Figure 2(a) shows the computation of cross-policy representations. By contrast, Figure 2(b) shows the computation of single-policy representations, which is adopted in [Jin *et al.*, 2022]. Intuitively, cross-policy representations focus on encoding the discrepancies between policies, while single-policy representations only encode features of single policies. Thus, cross-policy representations are more effective for downstream learning-to-rank tasks.

Policy Comparison Transformer (PCT) Transformers are proved to be effective for learning dependencies between different positions in sequences. Prior works has employed transformers to extract features from trajectory sequences [Lili *et al.*, 2021; Michael *et al.*, 2021]. However, existing transformer architectures fail to capture the differences of two policies' decisions. In our work, we propose the PCT architecture to learn cross-policy representations. Unlike previous works where the positional encodings indicate the positions of state-action pairs in a trajectory, PCT uses positional encoding to distinguish the relative order of two policies. In this way, the learned cross-policy representation $\chi^{i \succ j}$ can be directly used to predict whether π^i performs better than π^j.

Figure 1 shows the construction of input tokens. We first sample K states from $\mathcal{D}$ and then use a linear encoder f to map the K state-action pairs into $2K$ tokens:

$$x_k^i = f(s_k, a_k^i), \quad x_k^j = f(s_k, a_k^j), \quad k = 1, ..., K \quad (2)$$

where i and j represent the indexes of two policies. In order to represent the relative order of π^i and π^j, we introduce two one-hot positional encodings $e_+ = [1, 0]$ and $e_- = [0, 1]$, where e_+ indicates the policy ranked higher and e_- indicates the policy ranked lower. We also use an aggregation token e_0, which is a learnable vector for aggregating the information from the other $2K$ tokens [Zhu *et al.*, 2021]. The final inputs that indicate π^i ranked higher than π^j can be represented as:

$$z^{i \succ j} = \left[e_0, x_1^i + e_+, x_1^j + e_-, \cdots x_K^i + e_+, x_K^j + e_-\right] \quad (3)$$

This construction of inputs has two advantages. First, the two policies share the same set of states, thus their discrepancies are naturally represented by the different actions taken at these states. Second, we can easily get a mirrored representation $z^{j \succ i}$ by simply exchange the positional encoding $\mathbf{e}_\alpha$ and $\mathbf{e}_\beta$ used in $z^{i \succ j}$.

We adopt a widely used transformer architecture as our encoder [Dosovitskiy *et al.*, 2021]. It contains L alternating layers of multi-head self-attention (MSA) and multi-layer perception (MLP) blocks. Layernorm (LN) and residual connections are applied to the outputs of each block. For brevity, we rewrite the inputs in Equation 3 as $\mathbf{z}^{(0)}$. And the computations at each block can be represented as:

$$\begin{aligned} \hat{\mathbf{z}}^l &= \mathtt{MSA}(\mathtt{LN}(\mathbf{z}^{(l-1)})) + \mathbf{z}^{(l-1)} \qquad l = 1, \cdots, L \\ \mathbf{z}^l &= \mathtt{MLP}(\mathtt{LN}(\hat{\mathbf{z}}^{(l-1)})) + \hat{\mathbf{z}}^{l-1} \qquad l = 1, \cdots, L \\ \chi^{i \succ j} &= \mathtt{LN}(\mathbf{z}^L). \end{aligned} \quad (4)$$

The final cross-policy representation $\chi^{i \succ j}$ is the corresponding outputs of the aggregation token e_0 taken from $\mathbf{z}^L$. Note that $\chi^{i \succ j}$ changes to $\chi^{j \succ i}$ when we exchange the positional encodings e_+ and e_-, but they are permutation invariant to the order of inputted state-action pairs.

4.2 Learning to Rank Policies

In this section, we will introduce how to train the PCT in two cases regarding to the existence of ground-truth ranking labels of policies. First, we show that the policy ranking problem can be reduced to a binary classification problem since our cross-policy representations can be directly used to predict the ranking of two policies. Second, we will introduce a learning paradigm where multiple OPE methods are modeled as label providers. We will also show how to train the PCT leveraging the inaccurate labels provided by the workers.

Reducing OPR to Binary Classification We first consider the case when there is a training set $\Pi = \{(\pi^i, R^i)\}_{i=1}^T$ consisting of T deployed policies π^i as well as their real expected accumulated rewards R^i. In this case, we can directly construct binary labels by comparing the performance of the two

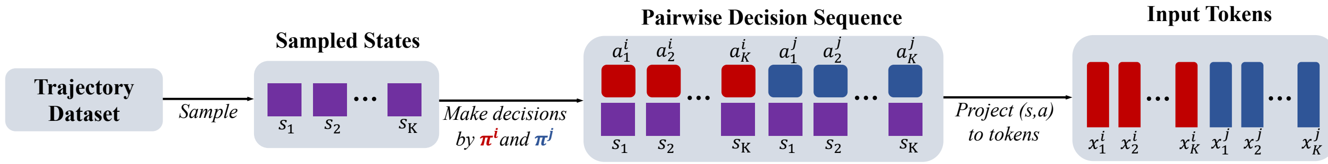

Figure 1: **Generation of input tokens.**

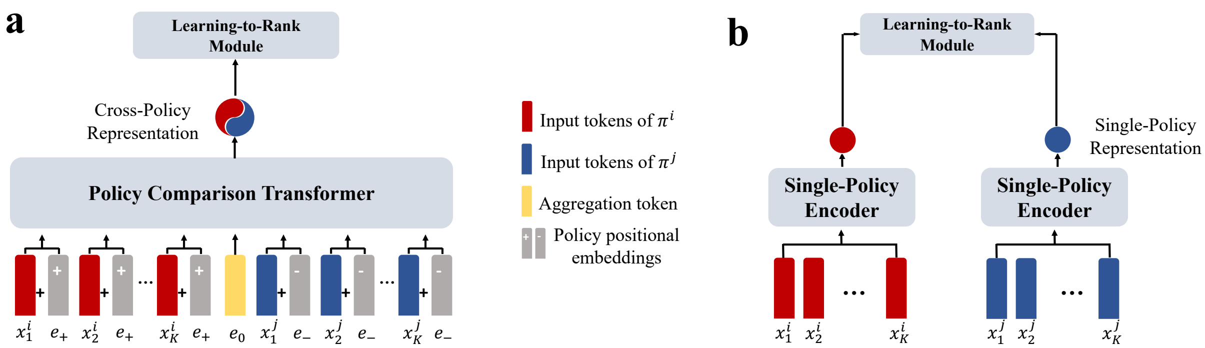

Figure 2: **Comparison between Cross-Policy Representation (left) and Single-Policy Representation (right).**

policies. We use an indicator $\mathbb{1}_{R^i>R^j}$ to represent the label of a pair of policies (π^i, π^j). The PCT can be trained by minimizing the following binary cross entropy loss:

$$\mathcal{L}_{sup} = -\mathop{\mathbb{E}}_{\pi^i,\pi^j\sim\Pi}\Bigg[\left(\mathbb{1}_{R^i>R^j}\right)\cdot\log\left(\hat{y}_{i\succ j}\right) + \left(\mathbb{1}_{R^i\leq R^j}\right)\cdot\log\left(1-\hat{y}_{i\succ j}\right)\Bigg], \tag{5}$$

where $\hat{y}_{i\succ j} = \texttt{sigmoid}(\phi(\chi^{i\succ j}))$ represents the predicted probability that π^i performs better than π^j. ϕ is a function that projects $\chi^{i\succ j}$ to a real number. The final ranking of test policies is based on their scores computed by:

$$score_i = \frac{1}{N}\sum_{j\neq i}\hat{y}_{i\succ j}, i = 1, ..., N, \tag{6}$$

which can be interpreted as the expected probability that π^i performs better than other test policies [Rodrigo *et al.*, 2019].

Figure 3 shows the framework of the Learning-to-rank module. In the presence of ground-truth label $y_{i>j}$, the LTR module is simplified to Situation 1. Otherwise, we will rely on OPE workers to provide supervision signals. Situation 2 will be illustrated bellow.

Learning from OPE Workers Supervised training is efficient when the dataset $\Pi = \{(\pi^i, R^i)\}_{i=1}^T$ contains enough policies. Unfortunately, collecting such training data can be extremely expensive in many real applications. Meanwhile, we note that although existing OPE methods are not robust enough, they actually provide candidate solutions to the OPR problem. To this end, we borrow ideas from crowdsourcing domain as an alternative way to approach the OPR problem. Specifically, suppose that there exists a set of OPE algorithms estimating the policy performance, we can employ them as crowd workers to generate possibly inaccurate labels and make use of these labels to train our models. The intuition is that the inaccurate labels generated by OPE workers are implicitly conditioned on the ground-truth performance of policies. If we can take advantage of these labels and learn their relationships with the ground-truth labels, our prediction $\hat{y}_{i\succ j}$ would be more close to the ground-truth labels.

In the framework of PoRank, we adopt Crowd Layer (CL, [Filipe and Pereira, 2018]) as our backend for learning from crowd labels. CL is able to automatically distinguish the good from the unreliable workers and capture their individual biases in many other domains, such as image annotation [Guan *et al.*, 2018; Li *et al.*, 2022] and music genre classification [Rodrigues *et al.*, 2013]. In addition, CL is naturally compatible with deep learning approaches since it simply adds a crowd layer to the deep neural networks and can be trained in an end-to-end way. As shown in Figure 2, we add CL to the top of our predicted probability $\hat{y}_{i\succ j}$. During training, CL adjusts the gradients coming from these noisy labels according to its reliability by scaling them and adjusting their bias. The adjusted gradients are then backpropagated to PCT according to the chain rule.

Formally, assume that there are W workers of OPE methods. For each worker w_m, its estimation about the expected return of π^i is denoted as R_m^i. The goal of CL is to train a mapping function $\hat{y}_{i\succ j}^m = \zeta^m(\hat{y}_{i\succ j})$ to predict the noisy binary label generated by worker w_m: $y_{i,j}^m = \mathbb{1}_{R_m^i>R_m^j}$. The overall objective can be written as:

$$\mathcal{L}_{CL} = -\mathop{\mathbb{E}}_{\substack{m=1,\cdots,W\\ \pi^i,\pi^j\sim\Pi}}\left[y_{i,j}^m\cdot\log\left(\hat{y}_{i\succ j}^m\right)+(1-y_{i,j}^m)\cdot\log\left(1-\hat{y}_{i\succ j}^m\right)\right]. \tag{7}$$

The complete training procedures of PoRank is summa-

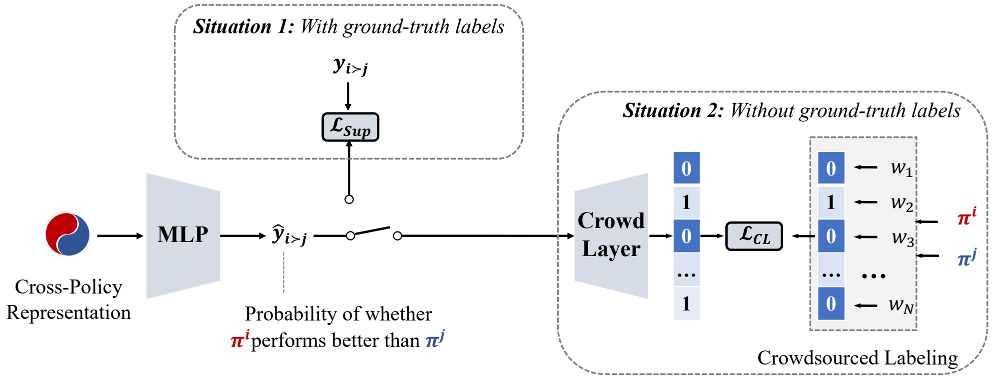

Figure 3: **Framework of the Learning-to-Rank Module.**

rized in Appendix. In practice, to reduce the computational cost brought by CL, we set ζ^m as a linear projection followed by a `sigmoid` function. Therefore, the number of additional parameters only grows linearly with the number of workers. Note that the CL is only available during training since we still use $\hat{y}_{i \succ j}$ to generate the predicted ranking of policies.

5 Experiments

In this section, we compare PoRank with widely-used OPE methods on various tasks. We present ablation studies with respect to PCT and CL, which are the main components of PoRank. **Note that** hyper-parameter selection and more experimental results can be found in Appendix.

5.1 Experimental settings

Trajectory Set We evaluate PoRank and all baseline OPE methods on D4RL dataset consisting of various trajectory sets [Fu *et al.*, 2020]. Overall, we adopt trajectory sets collected from 2 environments of Mujoco games: `HalfCheetah-v2` and `Walker2d-v2`. Besides, there are 3 different types of trajectory sets for each environment: `expert`, `full-replay` and `medium`. The difference between them is that the behavioral policies collecting these 3 types of trajectories show different performance. And these behavioral policies are trained by the Soft Actor-Critic (SAC) algorithm online [Haarnoja *et al.*, 2018].

Policy Set To evaluate the abilities of all methods to correctly rank policies. We use the policy set released by [Jin *et al.*, 2022] . For each trajectory set mentioned above, there are 2 types of policy sets (referred as `Set I` and `Set II`) in which the expected return of policies are evenly spaced in the performance range of them. As mentioned in [Jin *et al.*, 2022], `Set I` and `Set II` aim to simulate two kinds of OPE cases. The policies contained in `Set I` are trained by offline RL algorithms (CQL [Kumar *et al.*, 2020], BEAR [Kumar *et al.*, 2019], CRR [Wang *et al.*, 2020]) and show diverse behavioral performance. This is aligned with practical cases where the sources of policies are diverse and unknown. `Set II` contains policies trained by SAC, which is the same as the algorithm of training the behavioral policies. Therefore, `Set II` corresponds to the practical OPE cases in which the target policies share many common properties with the policies generating the trajectory data.

Baselines [2] We compare PoRank with six state-of-the-art baselines. **i**) Fitted Q-Evaluation (`FQE` [Hoang *et al.*, 2019]). It is a value-based OPE method, which learns a neural network to approximate the Q-function of the evaluated policy by temporal difference learning on the trajectory set. **ii**) Model-based estimation (`MB` [Paduraru, 2013]). It learns the dynamics model of environment, and estimates the expected return of evaluated policies by computing their average returns of Monte-Carlo rollouts in the model environment. **iii**) Weighted importance sampling (`IW` [Mahmood *et al.*, 2014]). It leverages weighted importance sampling to correct the weight of the reward, regarding the collected trajectory data distribution to the data distribution of the evaluated policy. **iv**) `DualDICE` [Nachum *et al.*, 2019]. It also aims to achieve distribution correction yet without directly using importance sampling. It learns an estimation of the state-action stationary distribution for achieving distribution correction. **v**) Doubly Robust (`DR` [Jiang and Li, 2016]). It utilizes an unbiased estimator that leverages an estimated environment model to decrease the variance of the unbiased estimates produced by importance sampling techniques.

Worker Set In all experiments, we use 15 models trained by 5 OPE methods mentioned above (IW, MB, DR, DualDICE, and FQE, each trained 3 models with different seeds) as our OPE workers. We present the architecture details for them in Appendix.

Evaluation Metrics We evaluate all models according two widely-used metrics. **i**) `Spearman's Rank Correlation`. It is the Pearson correlation between the ground truth rank sequence and the evaluated rank sequence of the evaluated policies. So, a higher rank correlation indicates a better policy ranker. **ii**) `Normalized Regret@k`. It is the normalized difference between the actual value of the best policy in the policy set, and the actual value of the best policy in the estimated top-k set. Mathematically, it can

[2] We leverage a popular implementation of OPE algorithms: https://github.com/google-research/google-research/tree/master/policy_eval. It contains the first 5 baselines used in our paper

Table 1: Comparing PoRank with other OPE baselines.

		HalfCheetah-v2				Walker2d-v2			
		Rank Correlation ↑		Regret @1 ↓		Rank Correlation ↑		Regret @1 ↓	
		Set I	Set II	Set I	Set II	Set I	Set II	Set I	Set II
Expert	PoRank (Ours)	0.65 ± 0.10	0.35 ± 0.03	0.00 ± 0.00	0.32 ± 0.02	0.85 ± 0.12	0.83 ± 0.08	0.01 ± 0.01	0.02 ± 0.03
	FQE	-0.53 ± 0.14	0.31 ± 0.10	0.23 ± 0.06	0.60 ± 0.07	0.62 ± 0.20	-0.08 ± 0.03	0.04 ± 0.07	0.02 ± 0.07
	DualDICE	0.47 ± 0.18	0.27 ± 0.06	0.38 ± 0.02	0.22 ± 0.04	0.52 ± 0.16	0.31 ± 0.14	0.43 ± 0.07	0.28 ± 0.03
	MB	0.39 ± 0.04	0.24 ± 0.08	0.34 ± 0.05	0.18 ± 0.02	0.46 ± 0.11	0.29 ± 0.12	0.39 ± 0.03	0.23 ± 0.08
	IW	0.18 ± 0.03	0.09 ± 0.02	0.23 ± 0.04	0.02 ± 0.01	0.24 ± 0.07	0.06 ± 0.03	0.28 ± 0.02	0.14 ± 0.07
	DR	-0.12 ± 0.03	-0.18 ± 0.04	0.13 ± 0.06	0.08 ± 0.01	0.14 ± 0.09	0.12 ± 0.06	0.19 ± 0.03	0.04 ± 0.02
Full-replay	PoRank (Ours)	0.72 ± 0.19	0.34 ± 0.08	0.01 ± 0.02	0.31 ± 0.01	0.82 ± 0.04	0.81 ± 0.17	0.09 ± 0.06	0.21 ± 0.09
	FQE	0.24 ± 0.18	0.52 ± 0.01	0.36 ± 0.09	0.03 ± 0.02	0.71 ± 0.13	-0.19 ± 0.19	0.06 ± 0.04	0.48 ± 0.02
	DualDICE	-0.57 ± 0.18	0.06 ± 0.09	0.53 ± 0.08	0.27 ± 0.03	-0.26 ± 0.14	-0.24 ± 0.18	0.42 ± 0.01	0.28 ± 0.06
	MB	0.23 ± 0.04	-0.19 ± 0.02	0.17 ± 0.06	0.34 ± 0.07	0.63 ± 0.13	0.71 ± 0.03	0.08 ± 0.01	0.14 ± 0.09
	IW	-0.24 ± 0.01	-0.31 ± 0.04	0.42 ± 0.07	0.46 ± 0.02	0.11 ± 0.09	-0.65 ± 0.08	0.09 ± 0.01	0.43 ± 0.08
	DR	0.03 ± 0.04	0.18 ± 0.03	0.34 ± 0.08	0.09 ± 0.02	0.31 ± 0.06	0.02 ± 0.07	0.06 ± 0.04	0.18 ± 0.05
Medium	PoRank (Ours)	0.82 ± 0.13	0.81 ± 0.04	0.02 ± 0.01	0.03 ± 0.04	0.82 ± 0.11	0.72 ± 0.13	0.13 ± 0.02	0.08 ± 0.03
	FQE	0.48 ± 0.12	-0.12 ± 0.09	0.05 ± 0.03	0.12 ± 0.08	0.71 ± 0.14	0.72 ± 0.19	0.08 ± 0.04	0.13 ± 0.01
	DualDICE	-0.42 ± 0.19	-0.28 ± 0.11	0.32 ± 0.07	0.13 ± 0.02	0.43 ± 0.17	0.23 ± 0.14	0.03 ± 0.08	0.28 ± 0.06
	MB	0.12 ± 0.03	-0.19 ± 0.14	0.23 ± 0.02	0.12 ± 0.01	0.47 ± 0.11	0.02 ± 0.10	0.18 ± 0.03	0.13 ± 0.02
	IW	-0.53 ± 0.02	-0.78 ± 0.04	0.63 ± 0.01	0.57 ± 0.06	0.27 ± 0.09	0.67 ± 0.03	0.38 ± 0.01	0.27 ± 0.09
	DR	0.63 ± 0.01	0.17 ± 0.04	0.03 ± 0.08	0.28 ± 0.09	0.07 ± 0.05	0.37 ± 0.02	0.28 ± 0.04	0.38 ± 0.08

be computed by $regret@k = \frac{V_{max}-V_{topk}}{V_{max}-V_{min}}$, where V_{max} and V_{min} is the expected return of the best and the worse policies, respectively, in the entire set, while V_{topk} is the estimated top k policies. So, a lower regret value indicates a better policy ranker.

5.2 Experimental Results

Comparison with Other OPE Baselines We evaluated PoRank against five baseline methods in two environments. Figure 1 presents the rank correlation and regret@1 values of the estimated rank sequences generated by each model.

Our results demonstrate that PoRank consistently outperforms the baseline methods. Specifically, PoRank achieved the highest rank correlations in 11 of the 12 policy sets and the lowest regret@1 values in 7 sets. This indicates PoRank's capability to deliver robust and effective performance across diverse policy scenarios.

Notably, PoRank excels in learning from noisy labels generated by other OPE workers. In our experiments, the OPE workers, which generate these noisy labels for PoRank, were directly sampled from the trained models of the baselines. For instance, in the `Set I` of the expert trajectory set in HalfCheetah-v2, while all OPE baselines exhibited poor performance , PoRank achieved a high rank correlation of about 0.65—despite relying on these low-quality labels. We attribute this resilience to two key factors: **i)** The Policy Comparison Transformer (PCT) in PoRank effectively mitigates the biases of inaccurate workers. **ii)** The crowd layer adeptly distinguishes reliable OPE workers from unreliable ones and adjusts for their individual biases. In conclusion, PoRank demonstrates highly effective and robust OPR results across a variety of policy sets. It effectively reduces the biases induced by OPE workers, thereby outperforming these workers significantly.

Comparing with Other Label Aggregation Methods We propose to use PCT and crowd layer to aggregate the information of multiple OPE workers. There are also other simple mechanism to aggregate worker information. For example, ranking policies by the average score of workers (denoted by **Average**) or ranking policies by the number of worker votes (denoted by **Major Voting**).

Table 2: Performance with different label aggregation methods.

Environment	Avg. Score	Major Voting	Ours	Ours with RA
HalfCheetah-expert	-0.34	-0.27	0.71	**0.72**
HalfCheetah-full-replay	0.24	0.31	**0.74**	0.73
HalfCheetah-medium	0.32	0.57	**0.81**	0.80
Walker2d-expert	0.53	0.23	**0.85**	0.83
Walker2d-full-replay	0.41	0.31	**0.82**	0.75
Walker2d-medium	0.21	0.29	0.80	**0.87**

Table 3: Performance with different batch size of state-action pairs.

Batch size	Avg. Rank Correlation
8	0.21
16	0.27
32	0.37
64	0.39
128	0.56
256	0.65
512	0.64
1024	0.66
2048	0.65

Actually, the superiority of CL over these baslines has been demonstrated in [Filipe and Pereira, 2018]. However, it is still valuable to reproduce this superiority in the context of OPR. Specifically, we report the rank correlations in the following Table 2. We can see from the first four columns that our method dominates these two baselines in all of the six environements. Note that the framework of PoRank can also combine with more advanced crowdsourcing methods other than CL. On the other hand, the line of works in rank aggregation, focusing on aggregating a set of pairwise comparisons

Table 4: **Ablations.** Comparison of policy ranking performance among PoRank (`PCT with CL`), and its ablation models `PCT with GL`, `SOPR-T with GL`, and `SOPR-T with CL` across different policy sets. Models with PCT architectures consistently achieves higher scores than models with SOPR-T architectures, underscoring the advantage of PCT in generating meaningful cross-policy representations. Notably, `PCT with CL` and `SOPR-T with CL` show competitive performance against their GL counterparts, despite lacking additional supervised information from deployed policies, affirming the effectiveness of the CL in learning from OPE-generated noisy labels.

		HalfCheetah-v2				Walker2d-v2			
		Rank Correlation ↑		Regret @1 ↓		Rank Correlation ↑		Regret @1 ↓	
		Set I	Set II	Set I	Set II	Set I	Set II	Set I	Set II
Expert	PoRank (PCT w/ CL)	0.65 ± 0.10	0.35 ± 0.03	0.00 ± 0.00	0.32 ± 0.02	0.85 ± 0.12	0.83 ± 0.08	0.01 ± 0.01	0.02 ± 0.03
	PCT w/ GL	0.43 ± 0.14	0.63 ± 0.10	0.13 ± 0.06	0.05 ± 0.07	0.82 ± 0.20	0.78 ± 0.03	0.04 ± 0.07	0.05 ± 0.07
	SOPR-T w/ CL	-0.47 ± 0.18	0.47 ± 0.06	0.58 ± 0.02	0.52 ± 0.04	0.72 ± 0.16	0.71 ± 0.14	0.13 ± 0.07	0.18 ± 0.03
	SOPR-T w/ GL	-0.29 ± 0.04	0.70 ± 0.08	0.34 ± 0.05	0.18 ± 0.02	0.56 ± 0.11	0.80 ± 0.12	0.19 ± 0.03	0.03 ± 0.08
Full-replay	PoRank (PCT w/ CL)	0.72 ± 0.19	0.34 ± 0.08	0.01 ± 0.02	0.31 ± 0.01	0.82 ± 0.04	0.81 ± 0.17	0.09 ± 0.06	0.21 ± 0.09
	PCT w/ GL	0.57 ± 0.18	0.37 ± 0.06	0.08 ± 0.02	0.32 ± 0.04	0.72 ± 0.16	0.81 ± 0.14	0.03 ± 0.07	0.18 ± 0.05
	SOPR-T w/ CL	-0.37 ± 0.18	0.36 ± 0.06	0.58 ± 0.02	0.52 ± 0.04	0.74 ± 0.16	0.71 ± 0.11	0.13 ± 0.07	0.19 ± 0.03
	SOPR-T w/ GL	-0.29 ± 0.04	0.24 ± 0.28	0.34 ± 0.05	0.08 ± 0.02	0.66 ± 0.11	0.79 ± 0.12	0.19 ± 0.03	0.23 ± 0.08
Medium	PoRank (PCT w/ CL)	0.82 ± 0.13	0.81 ± 0.04	0.02 ± 0.01	0.03 ± 0.04	0.82 ± 0.11	0.72 ± 0.13	0.13 ± 0.02	0.08 ± 0.03
	PCT w/ GL	0.73 ± 0.14	0.88 ± 0.10	0.03 ± 0.01	0.00 ± 0.00	0.72 ± 0.20	0.88 ± 0.03	0.14 ± 0.07	0.02 ± 0.07
	SOPR-T w/ CL	-0.27 ± 0.18	0.47 ± 0.06	0.58 ± 0.02	0.02 ± 0.04	0.72 ± 0.16	0.81 ± 0.14	0.14 ± 0.07	0.08 ± 0.03
	SOPR-T w/ GL	0.59 ± 0.04	0.84 ± 0.08	0.14 ± 0.05	0.08 ± 0.02	0.76 ± 0.11	0.89 ± 0.12	0.19 ± 0.03	0.03 ± 0.08

into a ranking list. Therefore, we use a more recent and simple RA method [Maystre and Grossglauser, 2017] (denoted by **RA**) to replace Equation 6 in our work. From the last two columns we can see that this method indeed further improves PoRank in some evironments. However, this does not contradicts to our main contribution: modeling the OPR problem from the perspective of crowdsourcing.

Selection of the Batch Size of State-action Pairs In the training phase, the batch size of state-action pairs feeded into the Transformer is an important hyper-parameter in our model. It play the role in balancing the computational cost and the performance. We chose the number 256 as the batch size. This choice is supported by the experimental results reported in Table 3, which show the averaged rank correlations of our model with the batch size growing. We can find that when the batch size is larger than 256, the performance of our model tends to be stable.

Ablations We conducted ablation studies to evaluate the importance of each component in our framework, using the same policy sets as the primary experiments. The results are depicted in Table 4. In our framework, termed PoRank, we integrate two key components: the Policy Comparison Transformer (PCT) for generating cross-policy representations, and the Crowd Layer (CL) that aggregates information from OPE workers. This integration allows PoRank to rank policies without requiring additional ground truth labels. We compared `PCT with CL` (PoRank) against three different ablations: **i)** `PCT with GL`: This model uses our PCT architecture but discards the CL, relying on extra ground truth labels (GL) for training. It is trained using additional sets of deployed policies released by [Jin *et al.*, 2022]. Without CL, it lacks the capability to learn from OPE workers. **ii)** `SOPR-T with GL`: [Jin *et al.*, 2022], a transformer-based model designed to learn individual policy representations. Unlike PCT, SOPR-T does not focus on capturing the differences between policies at the decision level. Like `PCT with GL`, it also requires extra ground truth labels for training. **iii)** `SOPR-T with CL`: This variation of SOPR-T attempts to learn from OPE workers by incorporating the CL, similar to PoRank, but does not require additional supervised information from deployed policies.

As shown in Table 4, `PCT with CL` outperforms `SOPR-T with CL` on 8 policy sets, while `PCT with GL` surpasses `SOPR-T with GL` on 8 policy sets. This suggests that our cross-policy representation, which aims to discern the nuanced decision differences between policies, exhibits stronger representational power compared to single-policy representations used by `SOPR-T`. Moreover, `PCT with CL` (PoRank) and `SOPR-T with CL` both demonstrate competitive performance against their counterparts `PCT with GL` and `SOPR-T with GL`, despite the absence of additional supervised information from deployed policies. This underscores the effectiveness of using the CL to learn from OPE-generated noisy labels when extra policies providing supervised labels are not readily available.

6 Conclusions

In this study, we introduced PoRank, a novel framework designed to learn robust off-policy rankers from a set of unreliable off-policy estimators. Unlike existing approaches that require the deployment of policies online for gathering supervision signals, PoRank innovatively leverages labels generated by existing Off-Policy Evaluation (OPE) methods. This feature significantly reduces the training cost of the ranker. Our theoretical analysis elucidates the relationships between a worker's bias, variance, and overall quality. We further contribute a unique Policy Comparison Transformer (PCT) architecture, developed to discern the relative discrepancies between policies through effective cross-policy representations. Empirical results confirm PoRank's superior performance over baseline models across diverse tasks and policy sets. Importantly, PoRank demonstrates excellent generalizability across various policy sets. Our ablation studies further validate the effectiveness of each individual component within the PoRank framework.

Acknowledgments

This research is supported by the National Research Foundation, Singapore under its Industry Alignment Fund – Pre-positioning (IAF-PP) Funding Initiative. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not reflect the views of National Research Foundation, Singapore.

References

[Cai *et al.*, 2023] Tianchi Cai, Jiyan Jiang, Wenpeng Zhang, Shiji Zhou, Xierui Song, Li Yu, Lihong Gu, Xiaodong Zeng, Jinjie Gu, and Guannan Zhang. Marketing budget allocation with offline constrained deep reinforcement learning. In *Proceedings of the Sixteenth ACM International Conference on Web Search and Data Mining*, pages 186–194, 2023.

[Dosovitskiy *et al.*, 2021] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*, 2021.

[Filipe and Pereira, 2018] Rodrigues Filipe and Francisco Pereira. Deep learning from crowds. In *Proceedings of the AAAI Conference on Artificial Intelligence*, page 1611–1618, 2018.

[Fu *et al.*, 2020] Justin Fu, Aviral Kumar, Ofir Nachum, George Tucker, and Sergey Levine. D4RL: Datasets for deep data-driven reinforcement learning. *arXiv*, 2020.

[G. *et al.*, 2017] Bellemare Marc G., Will Dabney, and Rémi Munos. A distributional perspective on reinforcement learning. In *International Conference on Machine Learning*, pages 449–458, 2017.

[Guan *et al.*, 2018] Melody Guan, Varun Gulshan, Andrew Dai, and Geoffrey Hinton. Who said what: Modeling individual labelers improves classification. In *Proceedings of the AAAI Conference on Artificial Intelligence*, page 3109–3118, 2018.

[Haarnoja *et al.*, 2018] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International Conference on Machine Learning*, pages 1861–1870, 2018.

[Hanna *et al.*, 2019] Josiah Hanna, Niekum Scott, and Peter Stone. Importance sampling policy evaluation with an estimated behavior policy. In *International Conference on Machine Learning*, pages 2605–2613, 2019.

[Harb *et al.*, 2020] Jean Harb, Tom Schaul, Doina Precup, and Pierre-Luc Bacon. Policy evaluation networks. *arXiv*, 2020.

[Hoang *et al.*, 2019] Le Hoang, Cameron Voloshin, and Yisong Yue. Batch policy learning under constraints. In *International Conference on Machine Learning*, pages 3703–3712, 2019.

[Isac *et al.*, 2019] Arnekvist Isac, Danica Kragic, and Johannes A. Stork. VPE: Variational policy embedding for transfer reinforcement learning. In *ICRA*, pages 36–42, 2019.

[Jiang and Li, 2016] Nan Jiang and Lihong Li. Doubly robust off-policy value evaluation for reinforcement learning. In *International Conference on Machine Learning*, pages 652–661, 2016.

[Jin *et al.*, 2022] Yue Jin, Yue Zhang, Tao Qin, Xudong Zhang, Jian Yuan, Houqiang Li, and Tie-Yan Liu. Supervised off-policy ranking. In *International Conference on Machine Learning*, pages 10323–10339, 2022.

[Jing *et al.*, 2014] Zhang Jing, Xindong Wu, and Victor S. Sheng. Imbalanced multiple noisy labeling. *Transactions on Knowledge and Data Engineering*, 27(2):489–503, 2014.

[Jing *et al.*, 2015] Zhang Jing, Victor S. Sheng, Jian Wu, and Xindong Wu. Multi-class ground truth inference in crowdsourcing with clustering. *Transactions on Knowledge and Data Engineering*, 28(4):1080–1085, 2015.

[Kumar *et al.*, 2019] Aviral Kumar, Justin Fu, George Tucker, and Sergey Levine. Stabilizing off-policy Q-Learning via bootstrapping error reduction. In *Advances in Neural Information Processing Systems*, pages 11761–11771, 2019.

[Kumar *et al.*, 2020] Aviral Kumar, Aurick Zhou, George Tucker, and Sergey Levine. Conservative Q-Learning for offline reinforcement learning. In *Advances in Neural Information Processing Systems*, pages 1179–1191, 2020.

[Li *et al.*, 2015] Lihong Li, Rémi Munos, and Csaba Szepesvári. Toward minimax off-policy value estimation. In *Artificial Intelligence and Statistics*, pages 608–616, 2015.

[Li *et al.*, 2022] Junbing Li, Changqing Zhang, Joey Tianyi Zhou, Huazhu Fu, Shuyin Xia, and Qinghua Hu. Deep-lift: Deep label-specific feature learning for image annotation. *IEEE Transactions on Cybernetics*, 52(8):7732–7741, 2022.

[Lili *et al.*, 2021] Chen Lili, Lu Kevin, Rajeswaran Aravind, Lee Kimin, Grover Aditya, Laskin Michael, Abbeel Pieter, Srinivas Aravind, and Mordatch Igor. Decision transformer: Reinforcement learning via sequence modeling. In *Advances in Neural Information Processing Systems*, pages 15084–15097, 2021.

[Liu *et al.*, 2018] Qiang Liu, Lihong Li, Ziyang Tang, and Dengyong Zhou. Breaking the curse of horizon: Infinite-horizon off-policy estimation. In *Advances in Neural Information Processing Systems*, page 5361–5371, 2018.

[Liu, 2009] Tie-Yan Liu. Learning to rank for information retrieval. *Information Retrieval*, 3(3):225–331, 2009.

[Mahmood *et al.*, 2014] A. Rupam Mahmood, Hado P van Hasselt, and Richard S Sutton. Weighted importance sampling for off-policy learning with linear function approximation. In *Advances in Neural Information Processing Systems*, 2014.

[Maystre and Grossglauser, 2017] Lucas Maystre and Matthias Grossglauser. Just sort it! a simple and effective approach to active preference learning. In *International Conference on Machine Learning*, pages 2344–2353. PMLR, 2017.

[Mengjiao *et al.*, 2022] Yang Mengjiao, Bo Dai, Ofir Nachum, George Tucker, and Dale Schuurmans. Offline policy selection under uncertainty. In *International Conference on Artificial Intelligence and Statistics*, pages 4376–4396, 2022.

[Michael *et al.*, 2021] Janner Michael, Qiyang Li, and Sergey Levine. Offline reinforcement learning as one big sequence modeling problem. In *Advances in Neural Information Processing Systems*, pages 1273–1286, 2021.

[Nachum *et al.*, 2019] Ofir Nachum, Yinlam Chow, Bo Dai, and Lihong Li. DualDICE: Behavior-agnostic estimation of discounted stationary distribution corrections. In *Advances in Neural Information Processing Systems*, 2019.

[Paduraru, 2013] Cosmin Paduraru. *Off-policy evaluation in Markov decision processes*. PhD thesis, McGill University Libraries, 2013.

[Rodrigo *et al.*, 2019] Nogueira Rodrigo, Wei Yang, Kyunghyun Cho, and Jimmy Lin. Multi-stage document ranking with bert. *arXiv preprint arXiv:1910.14424*, 2019.

[Rodrigues *et al.*, 2013] Filipe Rodrigues, Francisco Pereira, and Bernardete Ribeiro. Learning from multiple annotators: Distinguishing good from random labelers. *Pattern Recognition Letters*, 34(12):1428–1436, 2013.

[S. and Zhang, 2019] Sheng Victor S. and Jing Zhang. Machine learning with crowdsourcing: A brief summary of the past research and future directions. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 9837–9843, 2019.

[Shi *et al.*, 2021] Tianyu Shi, Dong Chen, Kaian Chen, and Zhaojian Li. Offline reinforcement learning for autonomous driving with safety and exploration enhancement. *arXiv preprint arXiv:2110.07067*, 2021.

[Tang *et al.*, 2022] Hongyao Tang, Zhaopeng Meng, Jianye Hao, Chen Chen, Daniel Graves, Dong Li, Changmin Yu, Hangyu Mao, Wulong Liu, Yaodong Yang, Wenyuan Tao, and Li Wang. What about inputting policy in value function: Policy representation and policy-extended value function approximator. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 8441–8449, 2022.

[Tian and Zhu, 2015] Tian Tian and Jun Zhu. Max-margin majority voting for learning from crowds. In *Advances in Neural Information Processing Systems*, page 1621–1629, 2015.

[Wang *et al.*, 2020] Ziyu Wang, Alexander Novikov, Konrad Zolna, Josh S Merel, Jost Tobias Springenberg, Scott E Reed, Bobak Shahriari, Noah Siegel, Caglar Gulcehre, Nicolas Heess, and Nando de Freitas. Critic regularized regression. In *Advances in Neural Information Processing Systems*, pages 7768–7778, 2020.

[Yang *et al.*, 2023] Chan-Yun Yang, Chamani Shiranthika, Chung-Yih Wang, Kuo-Wei Chen, and Sagara Sumathipala. Reinforcement learning strategies in cancer chemotherapy treatments: A review. *Computer Methods and Programs in Biomedicine*, 229:107280, 2023.

[Yuchen *et al.*, 2016] Zhang Yuchen, Xi Chen, Dengyong Zhou, and Michael I. Jordan. Spectral methods meet em: A provably optimal algorithm for crowdsourcing. *JMLR*, 17(1):3537–3580, 2016.

[Zhang and Jiang, 2021] Siyuan Zhang and Nan Jiang. Towards hyperparameter-free policy selection for offline reinforcement learning. In *Advances in Neural Information Processing Systems*, 2021.

[Zhang *et al.*, 2020] Zihao Zhang, Stefan Zohren, and Roberts Stephen. Deep reinforcement learning for trading. *The Journal of Financial Data Science*, 2020.

[Zhu *et al.*, 2021] Chen Zhu, Wei Ping, Chaowei Xiao, Mohammad Shoeybi, Tom Goldstein, Anima Anandkumar, and Bryan Catanzaro. Long-short transformer: Efficient transformers for language and vision. In *Advances in Neural Information Processing Systems*, pages 17723–17736, 2021.