CANOR COACH: Towards Noise-Robust Human-in-the-Loop Reinforcement Learning

by

Yuxuan Li

A thesis submitted in partial fulfillment of the requirements for the degree of

Master of Science

Department of Computing Science

University of Alberta

Abstract

Reinforcement learning has been widely applied in different control tasks. However, its performance often faces the challenge of low sample efficiency. Introducing human prior knowledge is often seen as a possible solution, such as behaviour cloning, learning from advice, and inverse reinforcement learning. Learning from feedback is an example of exploiting human knowledge and it is a method to enable the agent to learn from binary feedback, which describes the teacher's attitude towards the agent's action. Compared to traditional learning from demonstration methods, learning from feedback does not require expert-level knowledge. But this can also be a demerit as non-expert feedback comes with inevitable noise. In this thesis, we investigate how and to which extent noise impacts the learning performance. We also propose a series of methods to de-noise the feedback data online and achieve noise-robust human-in-the-loop reinforcement learning with different amounts of prior knowledge.

Preface

Part of this thesis is published in Adaptive and Learning Agents Workshop at AAMAS 24 in collaboration with Srijita Das, Qinglin Liu and Matt Taylor.

To my parents

For their unconditional support of my study

I think there is a world market for maybe five computers.

– Thomas J. Watson, IBM Chairman, 1943.

Acknowledgements

I want to first express my gratitude for my supervisor, Matthew E. Taylor for his support throughout my study at University of Alberta. Furthermore, I want to thank Srijita Das for providing constructive feedback and guidance during our fruitful collaboration and Antonie Bodley for writing assistance. Lastly, I want to thank my parents, for their never-failing support even when I choose to leave for somewhere that is ten thousand kilometres away from my home.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

In this chapter, we introduce the motivation of this thesis and present a summary of the contributions.

1.1 Reinforcement Learning with human's prior knowledge

Empowering machines with human-like intelligence has been one of the most important goals of artificial intelligence. Playing Go [33], Solving Rubik's cube with a robot arm [2], or cancer treatment [41], all involve an agent that tries to make a decision in its environment. As shown in Figure 1.2, Reinforcement Learning is a category of machine learning algorithms inspired by animal learning theory. It aims to achieve automatic learning through interaction within an environment, via updates of its policy by the agent's observations, actions, and the received reward signals.

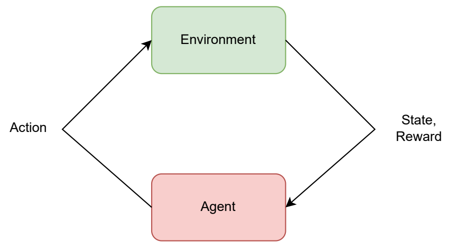


Figure 1.1: The general framework of reinforcement learning, the agent continuingly takes actions according to its current state and receives rewards and next state.

While reinforcement learning can learn with no prior knowledge to the environment, its application easily leads to dilemma of inaccessible reward. Sometimes rewards cannot be easily acquired or defined. And even when they are accessible, reward signals can be too sparse to learn. Go is an example of sparse rewards, where a single trajectory only receives win or lose. Besides rewards that might not be available, reinforcement learning also frequently faces the challenge of low sample frequency[44] as the agent needs to explore and optimise its policy with a larger amount of interactions.

Reward engineering can be a solution to mitigate this problem. Inverse reinforcement learning [27] is an example of using teacher's expert demonstration to infer a reward function. As inverse reinforcement learning methods essentially require humans' knowledge of the domain, it comes naturally that introducing more prior knowledge, in different modalities, amounts, with different depths of understanding towards the domain, can further speed up reinforcement learning.

Like a teacher teaches its students, different frameworks and methods have been introduced to enable agents to directly learn from human's prior knowledge. From learning from demonstration [1] to learning from feedback [8], all these methods directly or indirectly involves querying process from a human teacher or a scripted teacher. Learning from demonstration methods requires expert demonstration trajectories, which is relatively expensive to collect if available. For example, a complicated device is required in some certain domains like robots [34]. In contrast, learning from feedback is easier, and it only requires the teacher to provide binary feedback by observing the agent's behaviour.

1.2 Motivation

Reinforcement learning often faces the challenge of low sample efficiency. Introducing human advice and knowledge in different modalities can help accelerate the learning process. Human advice can be found in different modalities, such as action advice [39], demonstration [34], preference[11][21] and feedback

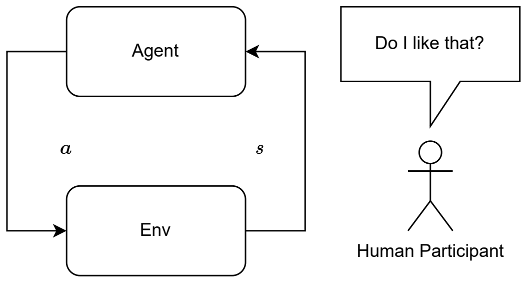


Figure 1.2: Human in the loop Learning to provide feedback. A teachers will observe agent's behaviour and provide signals based on their own judgement.

[8]. Yet, optimal advice and expert demonstration are not always accessible. In fact, they are often hard to collect [34], costly, or even not accessible at all. Furthermore, current human-in-the-loop reinforcement learning often requires a large amount of data[21]. Repeated queries to human participants might pose exhaustion and thus lead to incorrect advice, i.e. noise. Therefore, enabling agents to learn in the presence of noise will improve the agent's performance and stabilise the training process. In this thesis, our goal is to achieve a robust learning from feedback framework to address the instability from noise.

1.3 Thesis Contribution

In this thesis, a new learning framework and a method to calibrate noise in the dataset is proposed to enable agents to learn against noisy teacher advice. The thesis contributions are summarised as:

1. An anomaly-detection-based method to detect noise in the teacher advice dataset

2. A noise calibration method that requires zero prior knowledge

3. A new human-in-the-loop reinforcement learning framework that successfully learns against noise with the potential to be extended to other human-in-the-loop reinforcement learning algorithms

1.4 Thesis Outline

This thesis starts with an introduction of the general motivation and a summary of related work in Chapter 2 Background. Then, Chapter 3 Methodology introduces the proposed learning framework CANOR COACH and its details, followed by Chapter 4 and Chapter 5 on Experimental Design and results analysis. The conclusion and future work are discussed in Chapter 6, which is the end of this thesis.

Chapter 2

Background

In this chapter, we introduce the background and related work of this thesis, from basic reinforcement learning to learning with noisy labels.

2.1 Reinforcement Learning

Reinforcement learning (RL) is a branch of machine learning that enables agents to find a well-performing policy in a certain environment. It is usually defined under a Markov decision process (MDP). An MDP is denoted by a quintuple as $M = \{\mathcal{S}, \mathcal{A}, \mathcal{T}, r, \gamma\}$, where $\mathcal{S}$ denotes the agent's state space, $\mathcal{A}$ is the agent's action space, $\mathcal{T} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$ is the environmental dynamics transition probability, $r : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}$ is the function that gives an immediate reward, γ is a discount factor. In a typical MDP setting, the reinforcement learning agent will start at state s_0 and take an action a_0 following its policy π, which leads the agent to its next state s_1 and receives reward R_0. The interaction will repeat for T steps until the episode ends. The goal of reinforcement learning is to optimise the policy π through maximising the expected accumulated reward $\mathbf{E}_{\tau\sim\pi}[\Sigma_{t=0}^{T} R_t]$.

RL has been used in building many autonomous agents, such as Go [32], Dota [6], and Robotics [2] in different domains. As a result of adapting RL to different domains, there exists several forms of reinforcement learning or closed-related methodologies, like behaviour cloning, reinforcement learning under partially observable Markov decision process (POMDP) [36], offline learning [20] and so on. In the following sections, we will discuss reinforcement learning problems that involve learning from teacher advice.

2.2 Reinforcement Learning with Teacher's Advice

In this section, we discuss different forms of learning with a teacher's advice.

2.2.1 Learning from demonstration

Learning from demonstration methods often involves query demonstrations from human teachers or expert agent teachers to provide advice in the form of trajectories. More specifically, the most frequent form is to learn from $\mathbf{D} = \{\tau_1, \tau_2, ...\tau_N | \tau = \langle s_o, a_o, s_1, a_1, ..., s_{T-1}, a_{T-1}, s_T \rangle\}$, where N trajectories are collected from the teacher.

Behaviour cloning [7], [13] and inverse reinforcement learning [3] are the most often-seen learning from demonstration methods. Behaviour cloning approaches adopt supervised learning to imitate the experts' trajectories by minimising the action prediction error in demonstrations [30] or aggregating the gradient with reinforcement learning [29].

Apart from using demonstration to learn a map from states to actions, inverse reinforcement learning approaches learn an inferred reward function to convert the problem into a traditional RL [12]. Several of the most recent generative adversarial imitation learning practices involve adversarial learning to minimize the discrepancy between imitator and demonstrator [15]. Although learning from demonstration with RL can improve learning efficiency, using it to solve sophisticated manipulation tasks requires a great number of expert samples and suffers from the distribution shift problem. DAgger [30] addresses this problem by adding an online expert to aggregate the dataset, but it also raises the data requirement.

2.2.2 Learning from action advising

Some researchers further investigate it and formulate the problem in a student-teacher framework [39] through an action advising framework with limited budget. With a student-teacher framework, the key problem shifts to how to acquire teacher's action advising with best efficiency. Ilhan et al. [17] achieved this framework in the setting of deep reinforcement learning and also proposed to query advice based on novelty [16] .

2.2.3 Learning from feedback

Learning from feedback refers to methods where agents learn from another modality of teacher advice. Assuming the teacher is observing the agent's behaviour, feedback is $f \in \{-1, 1\}$, representing the teacher's attitude towards the behaviour, or more specifically, a pair of state and action or a piece of trajectory. Knox et al. proposed TAMER [19] to exploit simple scalar human feedback signals. In TAMER, human feedback is collected by letting humans watch the agent and is used to learn a reward model. Macglashan et al. [24] proposed COACH, which assumed feedback to be dependent on the agent's current policy and used it as advantage function. These methods have also been extended in Deep RL settings like Deep TAMER [42] and Deep COACH [4], where evaluations are done on image-based domains with high dimensional state and action space. Loftin et al. [23] proposed methods that take teacher's strategy into account through inferring missing human feedback.

2.2.4 Learning from preferences

Another comparatively easier way of giving advice is by giving preference over an action or trajectory segment as compared to another segment [43]. In this method, the reward model is learnt from teacher-specified preferences without access to the environment reward and this usually involves inverse reinforcement learning. Christiano et al. [11] extended learning from preference to Deep RL, where an ensemble of neural networks was used to model the reward function learnt from rankings over pairwise video clips of trajectories. Lee et al. [21] propose a method that uses unsupervised pretraining to generate diverse experiences, which further helps generate informative queries for soliciting preference from a scripted teacher. This method was successful in solving several robotic manipulation and control tasks. Similar to the learning from feedback methods, learning from preferences methods do not necessarily require expert-level knowledge. However, it requires a considerable amount of preferences to learn a proper policy [21].

2.3 Anomaly detection and learning with noisy labels

Learning with noise has been a challenge in recent applications with larger and larger datasets and the growing size increases the risk of introducing noise. It is reported that the real dataset faces noise from 8.0% to 38.5% [35]. In supervised learning problems, especially those with deep neural networks, noise can be easily memorised by the deep networks [45], thus hurting the generalisation performance. Noticeably, different types of noise exist, such as symmetrical noise and non-symmetrical noise. Symmetrical noise means the data can be wrongly labelled to other categories with equal probability and the non-symmetrical suggests the other way: there exists a unique, non-uniform distribution for the wrongly labelled data. Different anomaly detection techniques have been introduced to address the problem. They can be categorised as three types: supervised, semi-supervised, and unsupervised. Supervised anomaly detection refers to the methods that have access to all the groundtruth (noisy label v.s. correct label) and thus practice anomaly detection in a supervised learning fashion. The semi-supervised and unsupervised anomaly detection are more challenging, as they assume access to only one type of labels (normal or outlier) or no prior knowledge is available at all.

Several methods have been proposed to learn within these scenarios. MentorNet [18] uses a separate mentor network to guide the classifier with a curriculum facing noisy labels. For semi-supervised anomaly detection, Chen et al.[9] use SVDD as a one-class classifier to detect outliers. Han et al. [14] proposed Co-teaching for the supervised learning, being an example of unsupervised anomaly detection method. Co-teaching adds another neural network and lets the two supervised models select data batches based on the losses and feed into each other reciprocally. Following the intuition that the model first tends to learn the easier patterns in data and overtime gradually overfits to the dataset, including noisy labels, the co-teaching models selects the minibatch that gives the smallest loss and hence filter out noisy labels based on each other's model learning ability.

Nevertheless, all of these methods require some extent of prior knowledge of the domain. For example, these methods require users to know the noise scale to establish a decision boundary.

Chapter 3

Methodology

In this chapter, we first introduce the preliminaries and then introduce our proposed algorithm as well as its variations.

3.1 Preliminaries

In this section, we introduce the learning from feedback reinforcement learning algorithm COACH, which is our baseline. Then, we discuss the small loss trick and classifier pretraining.

3.1.1 Reinforcement Learning and Policy Gradient

Reinforcement learning is a method where agents learns through interaction within the environment via maximising the expected rewards. One of the classic reinforcement learning methods, policy gradient [38] is shown in Equation 3.1. $\pi_{\theta_t}(s, a)$ is the policy and $\tau = \langle s_0, a_0, s_1, a_1, ..., s_T \rangle$ is the agent's trajectory. The advantage function $A : S \times A \to \mathbb{R}$ represents the expected benefits of taking action a_t at s_t when compared to other actions, defined as $A_\pi(s, a) = Q_\pi(s, a) - V_\pi(s)$. $Q : S \times A \to \mathbb{R}$ is defined as the value of state and action pair following π as $\mathbf{E}_{\tau \sim \pi}[G_t | S_t = s, A_t = a]$, where G_t is the discounted return. $V : S \to \mathbb{R}$ is value function, defined as $\mathbf{E}_{\tau \sim \pi}[G_t | S_t = s]$.

$$\nabla_{\theta_t} J(\theta_t) = \mathbf{E}_{\tau \sim \pi_{\theta_t}(\cdot | s_t)}[\nabla_{\theta_t} log(\pi_{\theta_t}(a_t | s_t)) \cdot A^{\pi_{\theta_t}}(s_t, a_t)]. \tag{3.1}$$

3.1.2 Learning from feedback: COACH

Feedback is often defined as a scalar value describing the teacher's judgment of the agent's current behaviour, and the teacher can be a human teacher or a scripted teacher, Like COACH [42] and TAMER [19], we define feedback as $f \in \{-1, 0, 1\}$, where -1 means the teacher discourages certain behaviour, 0 means the teacher is indifferent, and 1 encourages the agent's behaviour, which is a pair of action and state $\langle s_t, a_t \rangle$, at time step t. Learning from feedback has been most effective in smaller domains like Mountain Car, Atari and Tetris [4], [8], [19], [42], with limited cases of its application to complicated high-dimensional tasks. In COACH, the feedback is deemed as an estimate of the advantage value and is used to update the policy. Therefore, it has a slightly modified version of policy gradient: feedback f_t as a proxy for advantage function in Deep-COACH [4] are used to update policy as Equation 3.2:

$$\nabla_{\theta_t} J(\theta_t) = \mathbf{E}_{a \sim \pi^h_{\theta_t}(\cdot|s_t)}[\nabla_{\theta_t} log(\pi_{\theta_t}(a_t|s_t)) \cdot f_t]. \tag{3.2}$$

, because feedback resembles the advantage function in that it similarly represents which actions are better or worse than other actions. Furthermore, human teachers may give feedback based on not only current time step but also historical information and the teacher will be very unlikely to provide feedback at every time step. Therefore, eligibility trace[5] is proposed to allow smoothed policy gradient updates for a piece of past history trajectory. The eligibility trace is defined as in Equation 3.3, where λ is the exponential decaying factor. And with the eligibility trace, the policy will be updated as in Equation 3.4, where α is the learning rate.

$$e_\lambda \leftarrow \lambda e_\lambda + \nabla_{\theta_t} log(\pi_{\theta_t}(a_t|s_t)) \tag{3.3}$$

$$\theta_{t+1} \leftarrow \theta_t + \alpha e_\lambda \tag{3.4}$$

3.1.3 Classifiers and the small loss trick

In supervised learning, classification is one crucial problem that applies to a wide range of real-world scenarios, like spam email detection [25] and credit scam detection [37]. In the classification problem, we need to predict which class does the data belongs to in a group of categories based on the input feature vectors. A classification problem can be formulated as follows: with input of feature vector x in feature space $\mathbf{X}$, with ground truth label of $y \in \mathbf{Y}$, we intend to train a classifier $C_\phi : \mathbf{X} \to \mathbf{Y}$ that correctly predicts the label so that $\hat{y} = y$. In order to train a neural network as a classifier, cross entropy loss is widely used [14], [26]. As shown in Equation 3.5, $p_y(x)$ is the actual probability of x being in class y and $q_y(x)$ is the predicted probability of x being in class y.

$$L_{CEL} = \sum_{y \in \mathbf{Y}} p_y(x) log(q_y(x)) \tag{3.5}$$

However, when dealing with noisy labels, the problem will be different. Specifically, the label dataset Y contains two kinds of labels: correct labels $y_{correct}$ and y_{noisy}. Hence it becomes important to ascertain if a label is noisy or not. The small loss trick is one solution to detect noisy labels based on such an observation: a properly trained classifier will have smaller losses on correct labels and, in comparison, will have larger losses on noisy labels. Therefore, cherry-picking the labels that gives smaller cross entropy losses will be more likely to give us correct labels.

3.2 CANOR COACH : Classifier Augmented Noise Robust COACH

In this thesis, we will first practice supervised style anomaly detection along with reinforcement learning, assuming we have access to a small dataset of correct labels. The naive way to achieve this is to introduce a properly pre-trained classifier $C : S \times A \to [0, 1]^2$ on the feedback dataset, which maps the state, action pair to probability distribution of positive and negative feedback.

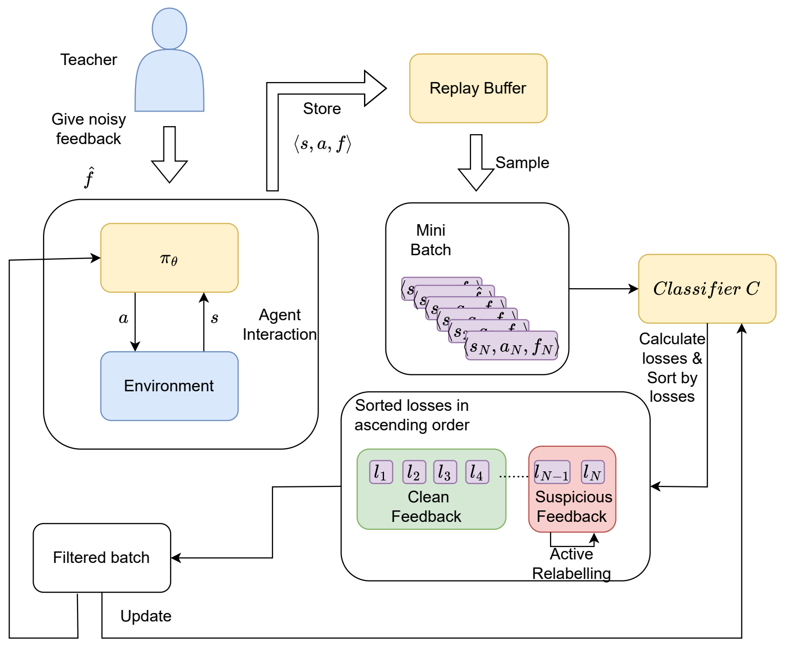


Figure 3.1: The overview of CANOR COACH, where a pretrained classifier C is introduced to filter noise based on the small loss trick.

With the small loss trick, we can select the correct labels and use only them to train the agent, as illustrated in Figure 3.1.

The **C**lassifier **A**ugmented **NO**ise **R**obust COACH (abbreviated as CANOR COACH) is presented as shown in Algorithm 1. With a pretrained classifier C, the agent will interact with the environment and query the teacher for feedback at a certain frequency. The feedback as well as agent's action and observation will be stored in a replay buffer. We will first sample a minibatch B from replay buffer and feed the minibatch into classifier to calculate the cross entropy loss of each data entry.

Then, we will practice the small loss trick and only train the agent on the selected batch $B_{filtered}$. Intuitively, as long as we have a well-performing classifier, the agent will be mostly trained with correct labels and therefore able to stand against the noise.

Algorithm 1 <u>C</u>lassifier <u>A</u>ugmented <u>**NO**</u>ise <u>**R**</u>obust COACH

Input:Pretrained Classifier C, Teacher T, Noise Percentage p_{noise}, Maximum Episode Length l, Maximum episode N_e, Size N Replay Buffer $R = \{\langle s_0, a_0, f_0\rangle, ...\langle s_N, a_N, f_N\rangle\}$, Batch size b

```
 1: Initialise policy $\pi$
 2: for $i \leftarrow 1, 2, ..., N_e$ do
 3:     for $j \leftarrow 1, 2, ..., l$ do
 4:         Agent with policy $\pi$ interacts with the environment and query $T$ for
            feedback $f$
 5:         Sample batch $B = \{\langle s_0, a_0, f_0\rangle, ...\langle s_{b-1}, a_{b-1}, f_{b-1}\rangle\}$ from $R$
 6:         Use Classifier $C$ to predict on state-action pairs in $B$
 7:         Calculate the cross entropy loss $L = \{l_0, ...l_{b-1}\}$ of each prediction of
            $B$
 8:         Sort $L$ in ascending order
 9:         Pick $(1 - p_{noise}) \cdot b$ items from $B$ based on the small loss trick as
            $B_{filtered}$
10:         Update $\pi$ with $B_{filtered}$ following Equation 3.2
11:     end for
12: end for
```

3.3 Classifier pretraining settings

The classifiers are trained on a feedback dataset following either cross entropy loss (Equation 3.5) or focal loss [22], depending on whenther the dataset is imbalanced or not. The focal loss is a modified cross entropy loss function that is designed for unbalanced dataset. The focal loss is defined in Equation 3.6.

$$L_{focal} = \sum_{y \in \mathbf{Y}} -(1 - p_y(x))^{\gamma} p_y(x) log(p_y(x)) \tag{3.6}$$

The focal loss adds a modulating factor $(1 - p_t)^{\gamma}$ to the traditional cross entropy loss, where γ is a hyperparameter that is larger than 0. When $\gamma = 0$, the focal loss downgrades to standard cross entropy loss. By adding the modulating factor, the focal loss reduces the relative loss for samples that the classifier can predict well and, therefore, will focus on misclassified samples. The focal loss can be further updated by adding a weighting factor $a : Y \rightarrow$

$[0, 1]$ as shown in Equation 3.7.

$$L_{focal} = \sum_{y \in \mathbf{Y}} -\alpha(y)(1 - p_y(x))^{\gamma} p_y(x) log(p_y(x)) \tag{3.7}$$

The class weights α is set accordingly to the inverse of the class label frequency to focus more on the classes that have fewer examples [22].

In pretraining the classifier, we first collect the dataset. More coverage of the state space is better. The dataset is collected by uniformly sampling in the state space if possible. But in domains like CartPole, there are no limits on the velocity of the cart and therefore we have to manually set a threshold for sampling. More details can be found in Appendix A.2.

Noticeably, since we are working with binary feedback that is either -1 or $+1$ (the indifferent feedback 0 is ignored since it does not effect the policy update), we can safely aggregate the feedback dataset by labelling the actions to be -1 if they are different from the action with positive feedback. For example, in CartPole, when we know that at s accelerating to the right is the correct action with positive feedback $+1$, we automatically know that accelerating to the left should receive negative feedback -1. Our reported dataset size is before such a data augmentation process. With such a process to augment the dataset, the pretraining dataset may come with highly imbalanced labels of feedback depending on the action space. For an imbalanced dataset, we train the classifier with the focal loss (Equation 3.7).

The summarised classifier pretraining process is described in Algorithm 2. More details of this can be found in Appenfix A.3.

3.4 Online Training of Classifier

The previously propose CANOR cannot handle the possible state distribution shift during training. When the agent explores to different regions, the classifier will inevitably face unseen states and actions and, therefore, can not properly filter noises. We can avoid this challenge by assuming access to a perfect classifier. However, a perfect classifier requires a huge amount of feedback dataset to fully cover the state and action space, and if we had such a

Algorithm 2 Pretraining the classifier

Input:Classifier C, Learning rate lr, Focal loss class weights $\{\alpha_i|, i = 0, 1, ..., N-1\}$, Focusing parameter γ, expert policy π^*, training epochs N, feedback amount k

Output:Pretrained classifier C

```
1: Uniformly sample $k$ states $\{s_0, s_1, ..., s_{k-1}\}$ from state space $\mathbf{S}$
2: Query expert policy $\pi^*$ optimal actions $\{a_0, a_1, ..., a_{k-1}\}$
3: Form the feedback dataset as $\mathbf{D} = \{\langle s_i, a_i, +1\rangle | i = 0, ..., k-1\}$
4: Aggregate dataset $\mathbf{D}$ by adding negative feedback to non-optimal actions
5: Initialise classifier $C$
6: for $i \leftarrow 1, 2, ..., N$ do
7:     Update $C$ following Equation 3.5 or Equation 3.7
8: end for
```

dataset, the agent can learn from this dataset only and nullify the significance of this work. Therefore, using an imperfect classifier to de-noise is our next goal.

In order to address this, online training of the classifier is introduced to improve the CANOR COACH. By updating the classifier with the selected batch $B_{filtered}$, the classifier will gradually adapt to the new state distribution. The algorithm with online training is shown in Algorithm 3.

3.5 Active Relabelling

The last piece to improve the performance of the CANOR COACH lies in exploitation of the noisy labels. Intuitively, if the classifier can efficiently differentiate correct feedback against noisy labels, we can easily aggregate the correct labels by simply flipping the label, thanks to the fact that the teacher provides binary feedback. With active relabelling, the final version of the algorithm is shown in Algorithm 4.

3.6 Estimate the noise scale

All the previously described methods require prior knowledge of the noise percentage as a hyperparameter. But unfortunately noise percentage is difficult to acquire in real world applications with human teachers. Therefore, a proper

Algorithm 3 Introducing online training of classifier: CANOR COACH(OT)

Input:Pretrained Classifier C, Teacher T, Noise Percentage p_{noise}, Maximum Episode Length l, Maximum episode N_e, Size N Replay Buffer $R = \{\langle s_0, a_0, f_0\rangle, ...\langle s_N, a_N, f_N\rangle\}$, Batch size b

```
1: Initialise policy $\pi$
2: for $i \leftarrow 1, 2, ..., N_e$ do
3:     for $j \leftarrow 1, 2, ..., l$ do
4:         Agent with policy $\pi$ interacts with the environment and query $T$ for feedback $f$
5:         Sample batch $B = \{\langle s_0, a_0, f_0\rangle, ...\langle s_{b-1}, a_{b-1}, f_{b-1}\rangle\}$ from $R$
6:         Use Classifier $C$ to predict on state-action pairs in $B$
7:         Calculate the cross entropy loss $L = \{l_0, ...l_{b-1}\}$ of each prediction of $B$
8:         Sort $L$ in ascending order
9:         Pick $(1 - p_{noise}) \cdot b$ items from $B$ based on the small loss trick as $B_{filtered}$
10:        Update $\pi$ with $B_{filtered}$ following Equation 3.2
11:        Update $C$ with $B_{filtered}$ following Equation 3.5
12:    end for
13: end for
```

estimation of noise scale is needed to allow our methods work in unknown quality of feedback data. In this subsection, a newly proposed method for calibrating the noise percentage that requires almost no prior knowledge is presented. The method is supported by two essential assumptions:

1. Consistency Assumption: The feedback should remain the same for two similar/same states.

2. Single-Optimal Action Assumption: For a series similar/same states, the teacher should only give one positive feedback for the optimal action.

Noticeably, the Single-Optimal Action Assumption does not hold for many scenarios and is therefore optional. Based on these assumptions, we can count and establish the distribution of conflicts and fit a map from the distribution to the estimated noise scale. The proposed method consists of two steps: establish state clusters (Algorithm 5) and find conflicts (Algorithm 6). We will first simply practice clustering based on states' Euclidean distance following

Algorithm 4 Introducing Active Relabelling: CANOR COACH (AR+OT)

Input:Pretrained Classifier C, Teacher T, Noise Percentage p_{noise}, Maximum Episode Length l, Maximum episode N_e, Size N Replay Buffer $R = \{\langle s_0, a_0, f_0\rangle, ...\langle s_N, a_N, f_N\rangle\}$, Batch size b, Label flipping rate r_{flip}

```
 1: Initialise policy $\pi$
 2: **for** $i \leftarrow 1, 2, ..., N_e$ **do**
 3:     **for** $j \leftarrow 1, 2, ..., l$ **do**
 4:         Agent with policy $\pi$ interacts with the environment and query $T$ for
            feedback $f$
 5:         Sample batch $B = \{\langle s_0, a_0, f_0\rangle, ...\langle s_{b-1}, a_{b-1}, f_{b-1}\rangle\}$ from $R$
 6:         Use Classifier $C$ to predict feedback with state-action pairs in $B$ as
            input
 7:         Calculate the cross entropy loss $L = \{l_0, ...l_{b-1}\}$ of each prediction of
            $B$
 8:         Sort $L$ in ascending order
 9:         Pick $(1 - p_{noise}) \cdot b$ items from $B$ based on the small loss trick as
            $B_{filtered}$
10:         Pick $p_{noise} \cdot b \cdot r_{flip}$ items with largest cross entropy loss as $B_{suspicious}$
11:         Flip the feedback labels of $B_{suspicious}$
12:         Concatenate $B_{filtered}$ and $B_{suspicious}$ as $B_{aggregated}$
13:         Update $\pi$ with $B_{aggregated}$ following Equation 3.2
14:         Update $C$ with $B_{aggregated}$ following Equation 3.5
15:     **end for**
16: **end for**
```

the Consistency Assumption. Then, based on the idea that those with more conflicts are more likely to be noise, we separate data entries into different bins with different counts of conflicts and then calculate an estimated noise scale by assuming the bin with highest conflict counts are noisy, as shown in Algorithm 6.

Algorithm 5 Establish state clusters by Euclidean Distance

Input:Trajectory dataset $\mathcal{D} = s_0, a_o, s_1, a_1, ...s_{T-1}, a_{T-1}, s_T$, Threshold l
Output:State cluster map $C : S \times S \rightarrow \{0, 1\}$

```
1: Initialise state cluster array $C[0 : T][0 : T]$ to be zeros
2: for $i \leftarrow 0, 1, 2, ..., |\mathcal{D}| - 1$ do
3:     for $j \leftarrow i + 1, 1, 2, ..., |\mathcal{D}| - 1$ do
4:         if $||s_i - s_j||_2 < l$ then
5:             $C[i][j] = 1$
6:             $C[j][i] = 1$
7:         end if
8:     end for
9: end for
10: return $C$
```

Algorithm 6 Estimate Feedback Noise Percentage

Input:Feedback dataset *Feedback*, Action dataset *Action*, State Cluster C, Bin count b
Output:Noise Percentage p_{noise}

```
1: Initialise conflicts counting array $A[0...|\mathcal{D}| - 1]$
2: for $i \leftarrow 0, 1, 2, ..., |\mathcal{D}| - 1$ do
3:     for $j \leftarrow i + 1, 1, 2, ..., |\mathcal{D}| - 1$ do
4:         if $C[i][j] == 1$ then
5:             if $Feedback[i] == 1 \& Feedback[j] == 1 \& Action[i] != Action[j]$
               then
6:                 $A[i] \leftarrow A[i] + 1$
7:             else if $Feedback[i] != Feedback[j] == 1 \& Action[i] == Action[j]$
               then
8:                 $A[i] \leftarrow A[i] + 1$
9:             end if
10:        end if
11:    end for
12: end for
13: Get bin counts $B$ of conflicts counting array $A$
14: return $\frac{B[0]}{|\mathcal{D}|}$
```

Chapter 4

Experimental Design and Research Questions

In this chapter, we examine the performance of our proposed methods on different noise scales. We will start with the research questions and then introduce the domains in which we performed experiments and the metrics to evaluate the performance, as well as the experiments settings.

4.1 Research questions

We intend to answer these research questions:

RQ1: How will feedback noise influence the performance of COACH?

RQ2: Can CANOR COACH learn robustly with noisy feedback? How does noise effect its performance?

RQ3: Can online training and active relabelling further improve the performance of CANOR COACH?

4.2 Experiment design and domain

In this section, we discuss our three evaluation domains, as well as experimental design details.

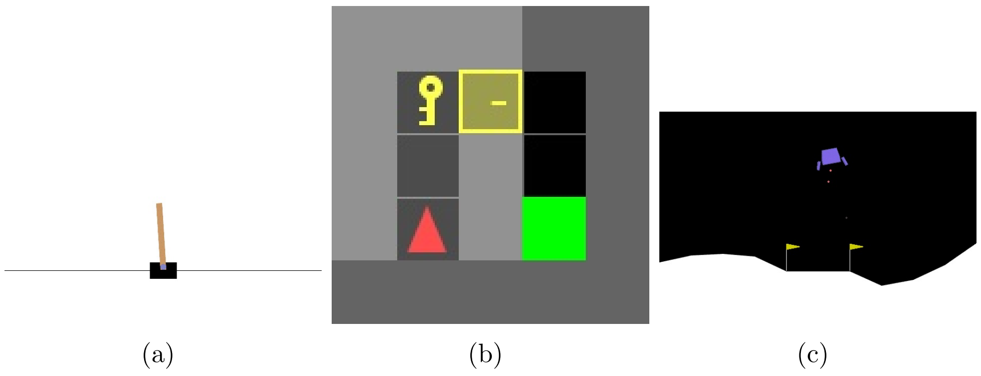


Figure 4.1: Our domains for evaluation: CartPole(a), Minigrid Doorkey(b) and LunarLander(c)

4.2.1 Domains

We conduct our experiments in Gymnasium [40] and the three domains are CartPole, LunarLander, and Minigrid Doorkey [10], as shown in Figure 4.1.

CartPole and LunarLander are two classical control tasks. In CartPole, the agent needs to control a moving cart to avoid the attached rod from falling down. The state space is represented with a vector of four dimensions, the cart's position, the cart's velocity, the pole's angle, and the pole's angular velocity. The action space consists of two discrete actions: applying a force to the left or to the right of the cart. The reward function is simple: a reward of +1 for every time step the pole remains upright, encouraging the agent to keep the pole balanced for as long as possible. The episode ends when the pole falls over or the cart moves too far from the center.

In LunarLander, the agent will challenge the problem of safely landing a spacecraft on the moon, where the agent must control its position and velocity to avoid crashing. The state space is represented by six scalar values: the lander's position (x and y), its velocity (x and y), its angle, its angular velocity, and another two boolean values indicating whether its left and right leg is in contact with the ground. The action space consists of four discrete actions: do nothing, fire the left orientation engine, fire the main engine or fire the right orientation engine. The reward function is more complex, providing rewards for successful landing, penalties for crashing, and rewards for reducing speed and landing smoothly. Additionally, there are small penalties for each fuel unit used, encouraging efficient use of resources. An episodic reward larger than two hundred is considered a success.

In Minigrid Doorkey, the agent needs to explore around to find a key, unlock the door, and reach the destination. The state space is an RGB imgae array with discrete positions for the agent, key, door, and goal, as well as the agent's orientation. The action space includes seven discrete actions: turn left, turn right, move forward, pick up an object, drop the object (unused), toggle an object and done (unused). The reward function provides a positive reward +1 for reaching the goal, which is then discounted by the time steps.

4.2.2 Feature extraction in Minigrid

The Minigrid environments provide flexibility to different types of observation. In our setting, the agent can only receive full observations in the form of an RGB image array. A CNN-based feature extractor from the expert policy (scripted teacher) is reused for agent training to reduce the observation space to 5. More details and hyperparameters can be found in Appendix A.1.

4.2.3 Metrics

We evaluate the agent mainly by the episodic reward. Noting in the learning from feedback reinforcement learning algorithms, the agent does not receive reward signals, here we only use the reward function to evaluate the agent's performance.

Furthermore, we reveal the deeper relationship between noise and performance by showing the pure ratio. The pure ratio is defined as the percentage of correct feedback to update the policy.

4.2.4 General experimental settings

In our settings, the agent receives feedback on fixed intervals of time steps, which is called feedback frequency. The feedback comes with symmetric noise, i.e., all feedback labels are randomly flipped by a fixed probability independently. Naturally, we assume the noise will be smaller than 50%, as agents will never be able to learn from data that contain more errors than correct labels.

The feedback is provided by a scripted teacher. The scripted teacher is a pretrained expert policy in a domain. The teacher provides negative feedback when the agent fails to choose the optimal action. If the agent chooses the same action as the teacher, the teacher provides positive feedback. Furthermore, there exists a maximum number of feedback labels that the agent can receive, which is the budget. In unlimited budget experiments, there is no limit for the amount of budget and the agent can receive feedback until the end of a training run, following feedback frequency.

Noticeably, the scripted teacher is set to be deterministic and therefore only considers the most probable action as optimal action. More details and hyperparameters can be found in the Appendix A.1

Chapter 5

Experimental Results Analysis

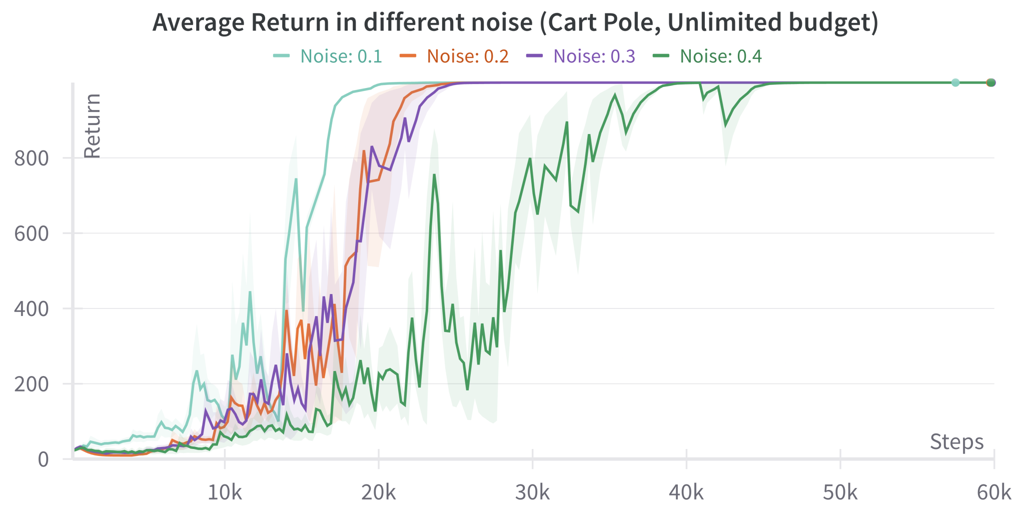
In this chapter, we will discuss experiments regarding different variations of the proposed algorithm and then practice ablation study to show the effectiveness of our algorithm.

5.1 Noise Evaluation

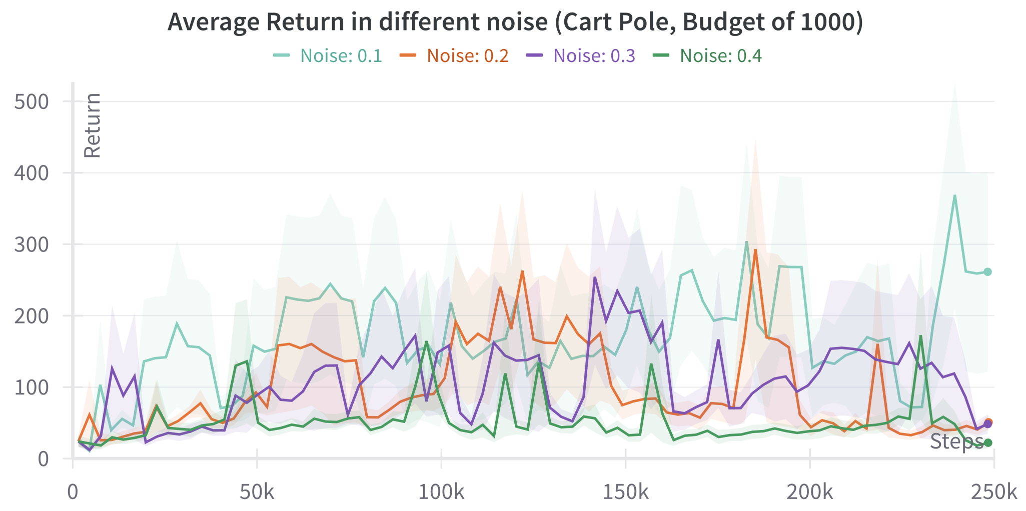
In this section, we conduct experiments to evaluate COACH with different noise scales and answer **RQ1**: How will feedback noise influence the performance of COACH?

We tested COACH from noise of 10% up to 40% in our three domains. As shown in Figure 5.1(a) and 5.1(b), a natural pattern is observed: the higher the noise, the worse the performance. Furthermore, with an unlimited budget, the agent is still able to learn against 40% noise. Statistically, as long as there are more correct labels than the wrong labels, the agent will eventually learn the correct policy given infinite amounts of feedback. However, the situation changes remarkably while we have a limited feedback budget. As illustrated in Figure 5.1(b), the performance significantly deteriorates and even unlearns over time. This shows that the learning performance is very sensitive to noise. The similar results can also be found in the other two domains (Lunar Lander and Minigrid Doorkey), as shown in Figure 5.2 and Figure 5.3.

We here answer **RQ1** firmly that noise poses a significant negative influence on COACH and its performance is worsened in limited budget scenarios. The limited budget setting is more realistic since query teachers in real life can be costly. Therefore, in all of our following experiments, we will test CANOR COACH with a limited budget.

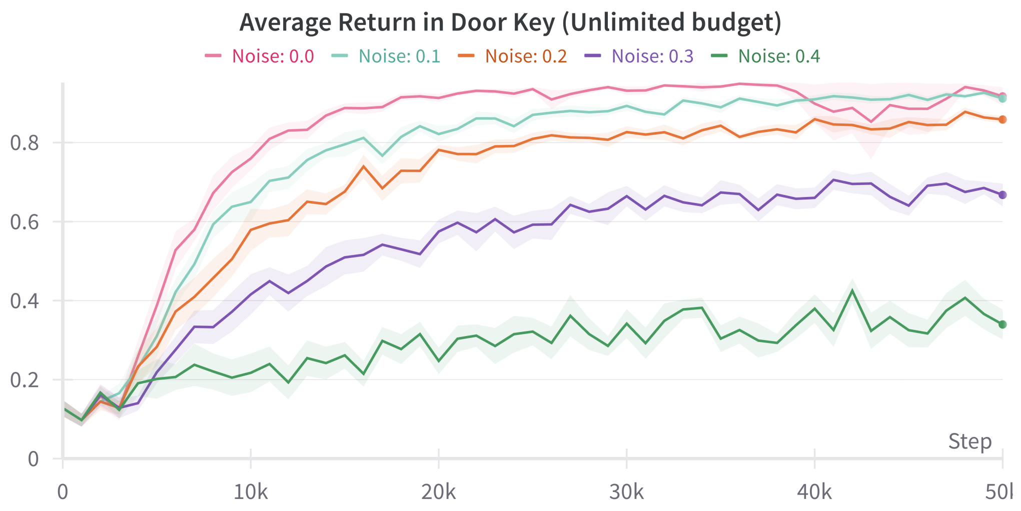


(a) Cart Pole with unlimited budget of feedback

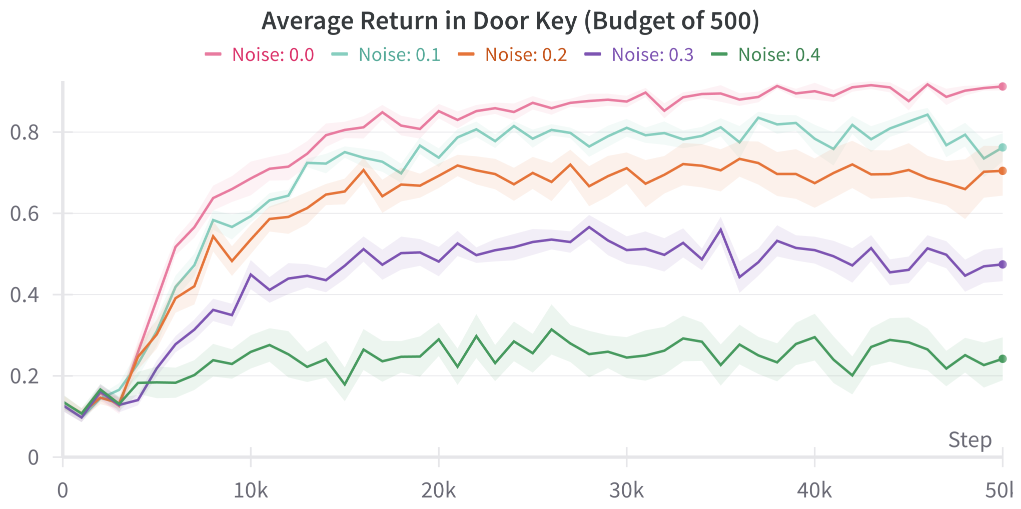


(b) Cart Pole with limited budget (1000) of feedback

Figure 5.1: Performance of Deep COACH under different scales of noise. While with unlimited budget Deep COACH is able to learn against 40% noise slowly, the performance of Deep COACH significantly deteriorates with limited budget.

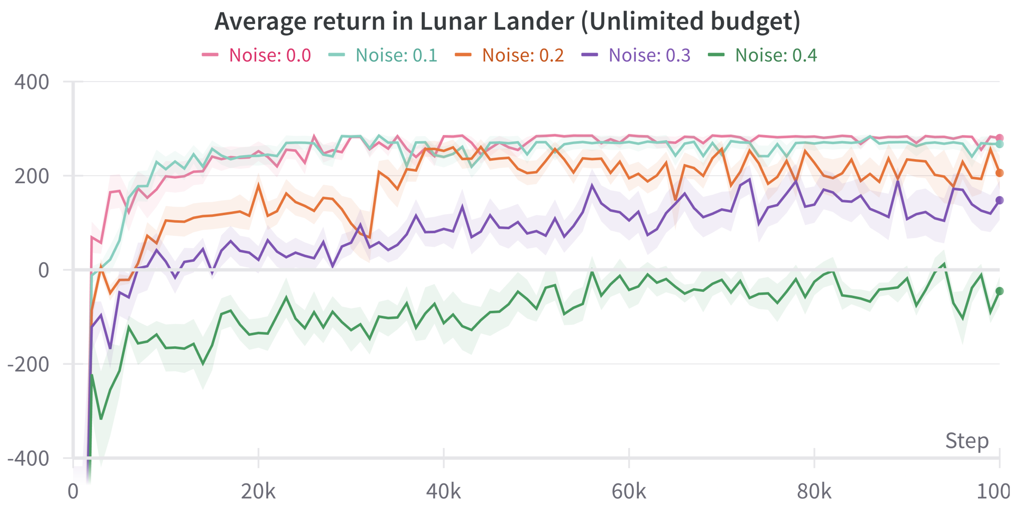


(a) Door Key with unlimited budget of feedback

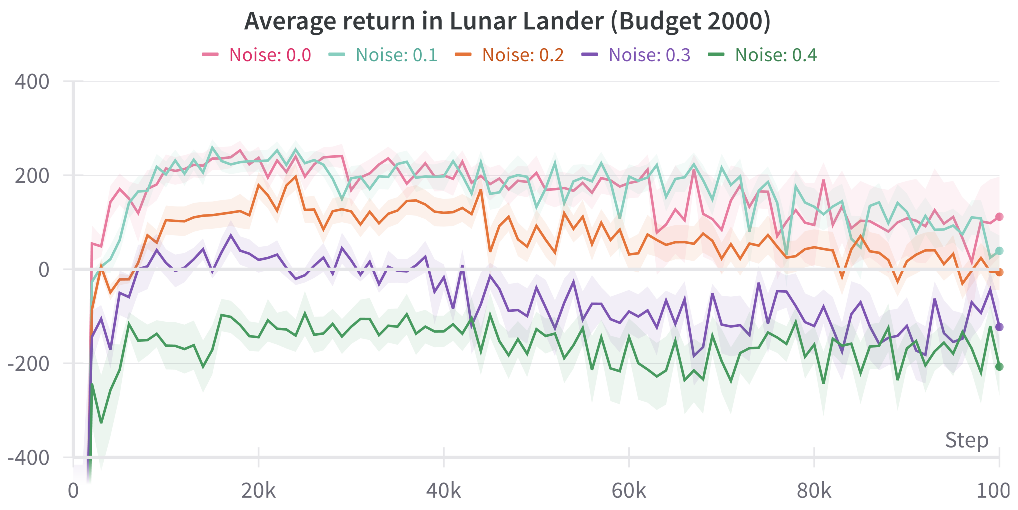


(b) Door Key with limited budget (500) of feedback

Figure 5.2: Performance of Deep COACH under different scales of noise in Door Key with unlimited budget (a) and limited budget (b).



(a) Lunar Lander with unlimited budget of feedback



(b) Lunar Lander with limited budget (2000) of feedback

Figure 5.3: Performance of Deep COACH under different scales of noise in Lunar Lander with unlimited budget (a) and limited budget (b). With a limited budget of feedback, the performance even starts to decrease due to the existence of noise.

5.2 CANOR COACH evaluation

In this section, we evaluate the performance of the original CANOR COACH with an unlimited budget and answer **RQ2** in this setting. We conducted experiments with different sizes of pretraining datasets for the classifier.

As seen from Figure 5.4, CANOR COACH is able to outperform when we have a pretraining datasize of 30 with 40% noise. Though both of the COACH and CANOR COACH is able to learn and reach almost perfect performance eventually, CANOR COACH is able to learn faster than the baseline, with unlimited budget of feedback, Furthermore, when the classifier does not have enough data for pretraining and cannot select correct labels for agent updates, the performance will be downgraded to a very low level. Here, we answer **RQ2** that with an unlimited budget, CANOR COACH is able to outperform Deep COACH by learning faster with a small amount of feedback dataset, But with unlimited feedback budget, even our baseline is able to learn well against high noise. In the next section, we will discuss limited budget scenarios.

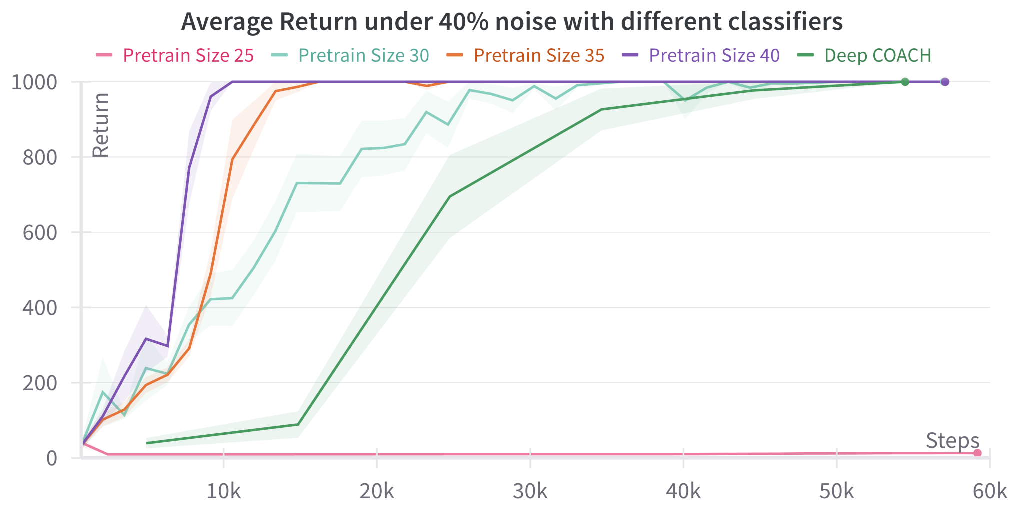


Figure 5.4: Performance of CANOR COACH under 40% noise, with different amount of pretraining dataset size.

5.3 Limited budget CANOR COACH evaluation

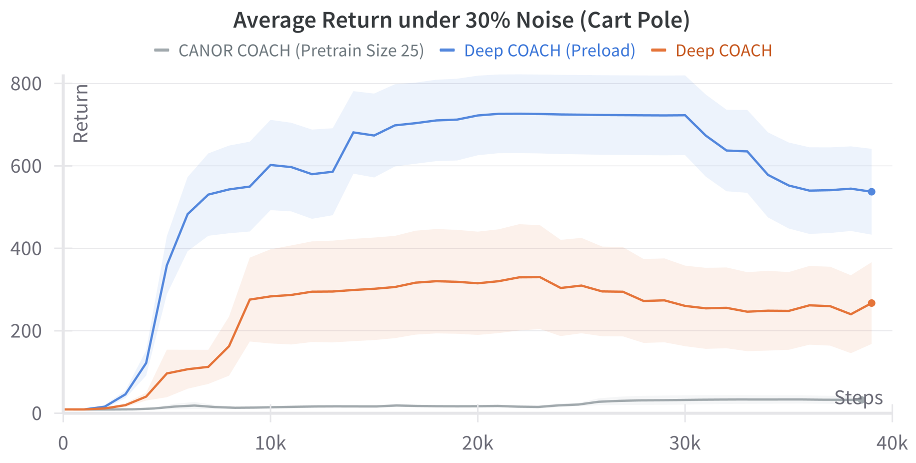
In this section, we evaluate the performance of the original CANOR COACH with a limited budget and answer **RQ2** in limited budget setting. As shown in Section 5.1 that COACH is more sensitive to noise with limited budget, we will discuss CANOR COACH's performance in comparison with COACH. Noticeably, with limited budget, our baseline (COACH) will always have the same budget of noisy feedback to ensure a fair comparison. Therefore, our two baselines are Deep COACH, which is allowed the same amount of noise free feedback budget, and Deep COACH (Preload), which loads the same dataset for classifier pretraining into the replay buffer.

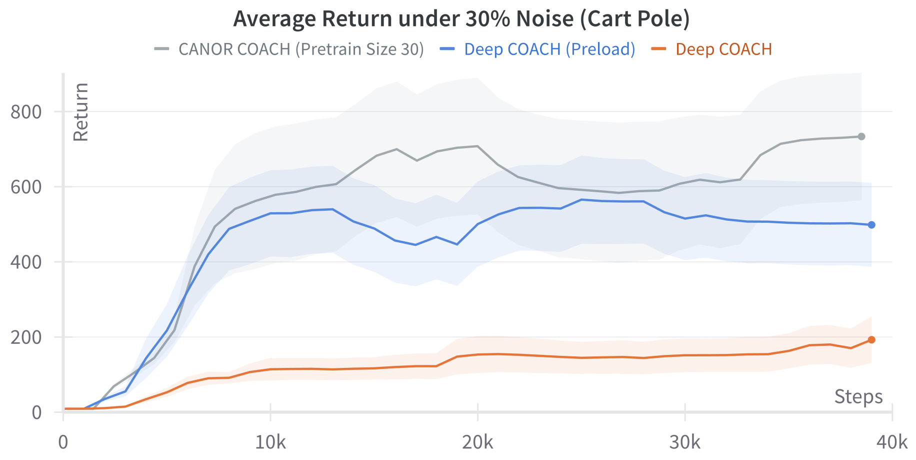
5.3.1 Cart Pole

As shown in Figure 5.5, our method can learn well against 30% noise, while Deep COACH shows highly unstable performance. In fact, even with different amounts of preloaded pure dataset, under same settings, Deep COACH shows significantly different learning curve due to the very unpredictability due to the noise.

Furthermore, it is revealed in Figure 5.6 that a classifier pretrained with datset of size 25 barely filters noise successfully as the pure ratio hovers around 70% under 30% noise, while the other two classifiers successfully improves the pure ratio up to 80% and 85%. Although CANOR COACH is able to learn against noise in the limited budget setting, it is also noticeable that the pure ratio is going down steadily. It is due to the fact that the agent is learning and will naturally explore different states and actions. The shift in state and action distribution leads to performance deterioration of the fixed pretrained classifier.

We also tested CANOR COACH on 40% and 45% noise, as shown in Figure 5.7 and Figure 5.7. With extremely high noises, CANOR COACH requires more pretraining data (35 in both 40% and 45% noise) to succeed. We can also observe a similar pattern in pure ratio which keeps decreasing and then





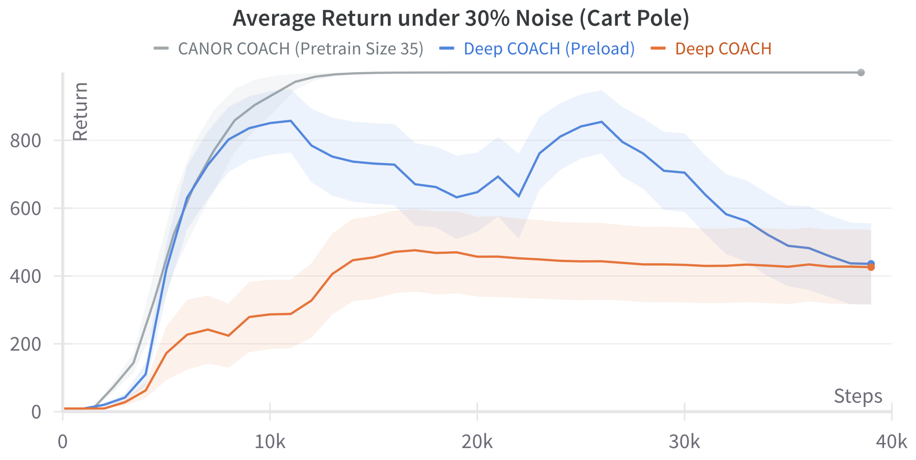


Figure 5.5: Average episode return of CANOR COACH and Deep COACH in CartPole under 30% noise. Three figures show the same experiment with different amounts of pretraining feedback. It can be seen that CANOR COACH needs at least 30 to succeed and 35 to perform well.

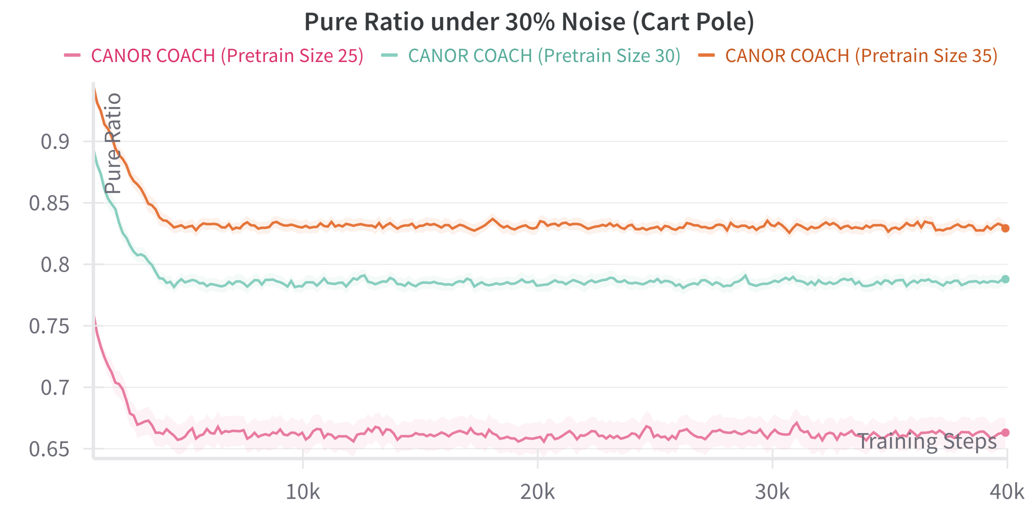


Figure 5.6: Average pure ratio of CANOR COACH in CartPole under 30% noise. While the agent explores new states and actions, the distribution of state and action changes and therefore a fixed classifier predicts less accurately over time.

becomes stable.

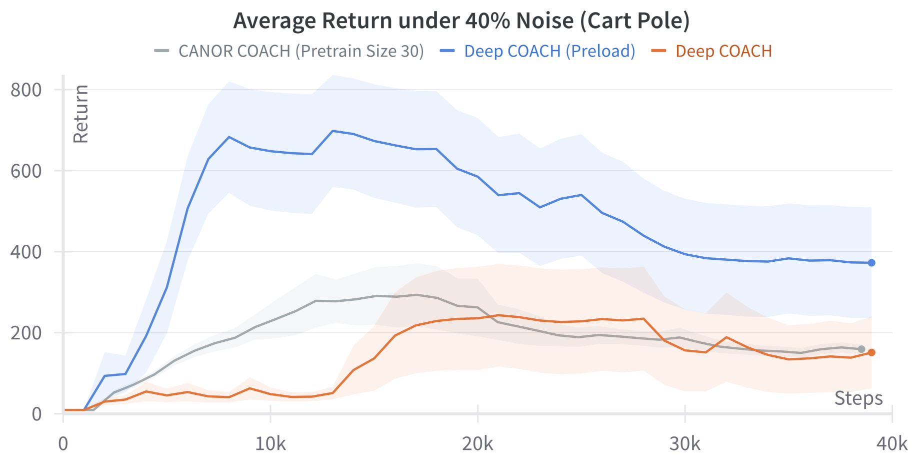
While our CANOR COACH shows its potential to be noise robust, it requires a big enough pretraining dataset and it suffers from the state distribution shift.

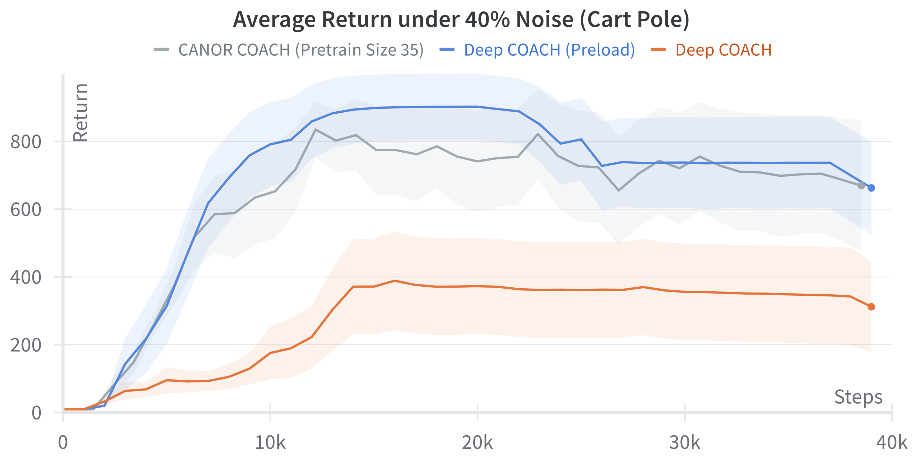
5.3.2 Door Key

In Minigrid Doorkey, the results are slightly different for CANOR COACH. CANOR COACH suffers more from noise and we observe that the evaluation reward will rise firstly but then stably going down and shows no evidence in robustness against noise, as shown in Figure 5.9.

5.3.3 Lunar Lander

In LunarLander, the results for CANOR COACH are shown in Figure 5.10. Like Doorkey, our method (CANOR COACH) does not perform well and shows lower performance than our baselines.





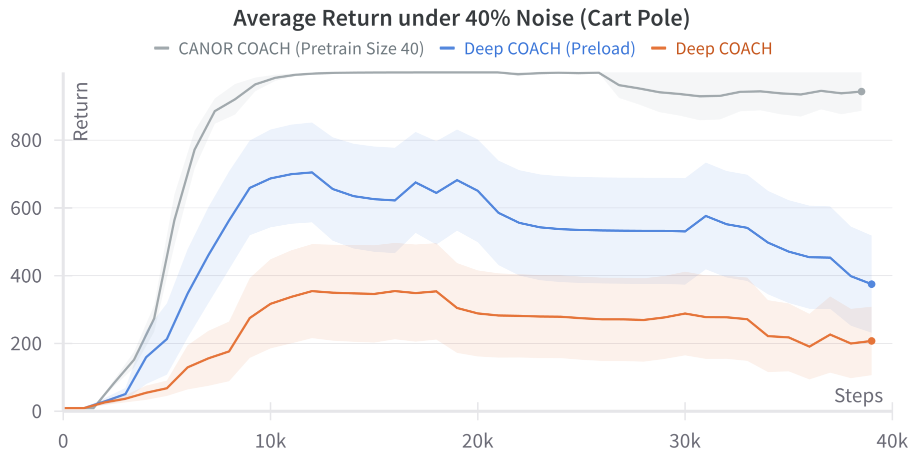
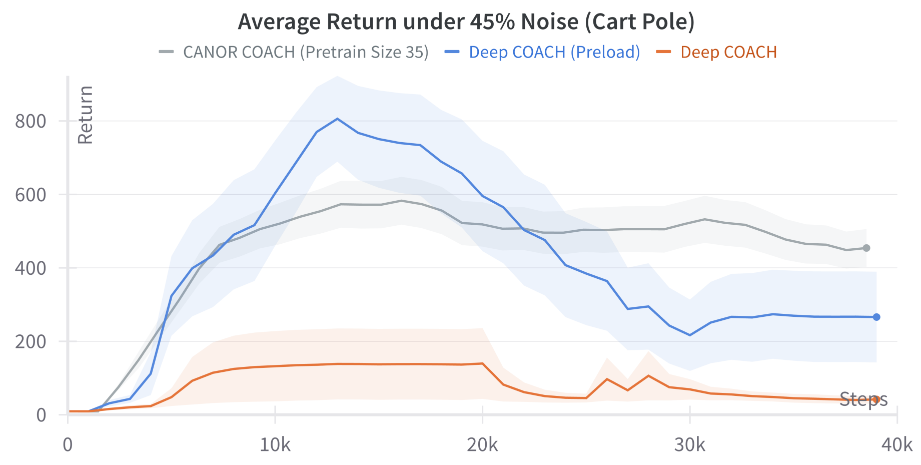


Figure 5.7: Average episode return of CANOR COACH and Deep COACH in CartPole under 40% noise. Three figures show the same experiment with different amounts of pretraining feedback. 40% is much harder and our CANOR COACH will also suffer from noise with pretraining size of 30 and CANOR COACH needs 40 to reach the maximum episodic return (1000).



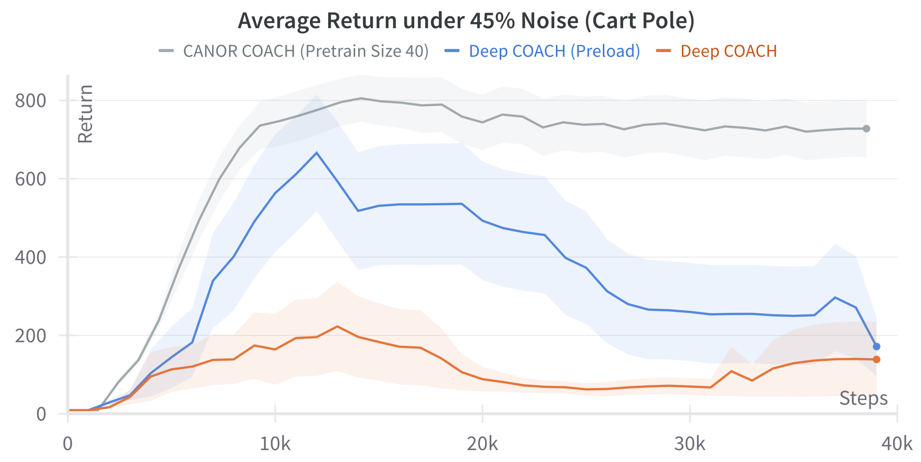


Figure 5.8: Average episode return of CANOR COACH and Deep COACH in CartPole under 45% noise. The first two figures show the same experiment with different amounts of pretraining size. CANOR COACH needs 35 to perform well and be stable against noise. The third figure shows the pure ratio under 45% noise, and a similar decreasing pattern can be observed.

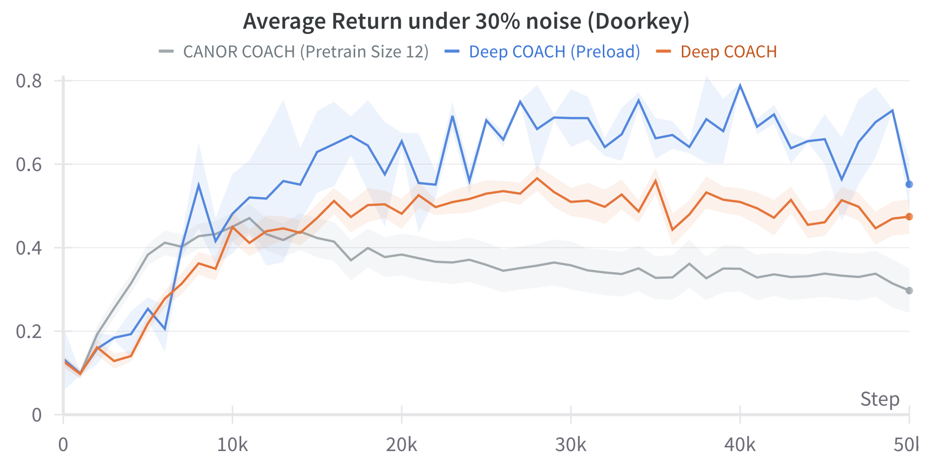


Figure 5.9: Average episode return of CANOR COACH and Deep COACH in Door Key under 30% noise.

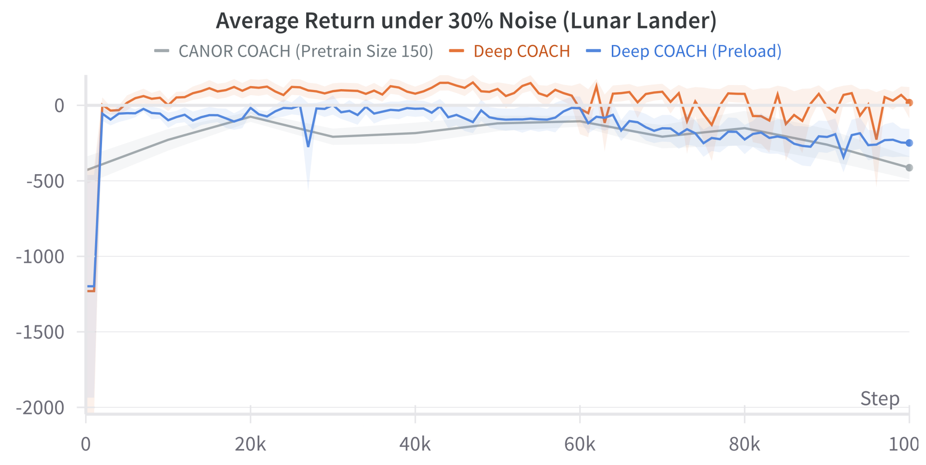


Figure 5.10: Average episode return of CANOR COACH and Deep COACH in Lunar Lander under 30% noise.

5.3.4 Summary

In this section, we saw that for simpler domains (like Cart Pole), just pre-training a classifier to denoise is enough. But for more complicated domains like Doorkey and Lunar Lander, it might require more data to work or just does not show much improvement. In the next section, we analyze CANOR COACH with online training and demonstrate whether this can improve performance.

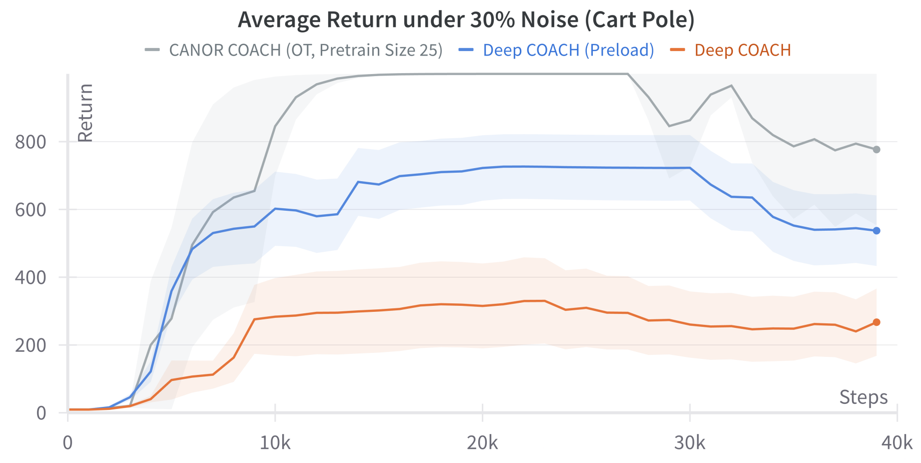
5.4 Online training evaluation

In this section, we evaluate the performance of Online Training CANOR COACH (CANOR COACH with OT) and **RQ2** in this setting. Since the classifier is pretrained on a fixed small pure datatset and is never updated in CANOR COACH , we see that it suffers from state and action distribution shift in Section 5.3. In this section, we introduce online training (abbreviated as OT) of the classifier to mitigate this problem. Online training essentially faces such a trade-off: to suffer from state action distribution with a fixed classifier or to suffer from learning with noisy labels with online training. In this section, we show that it is worthwhile to introduce online training.

5.4.1 Cart Pole

The results of CANOR COACH with online training in 30% noise can be found in Figure 5.11. Recall that in the previous section, CANOR COACH fails to learn with pretraining dataset of size 25. With online training, we observe that the pure ratio can gradually increase, which means that the classifier is learning and is well adapted to the new state and action distribution. Eventually, the pure ratio stabilises around 95% and as a result, CANOR COACH with online training is able to learn robustly against 30% noise with pretraining dataset of size 25.

However, online training does not always show such a great improvement when noise increases. The results in 40% noise can be found in Figure 5.12. Under 40% noise, CANOR COACH with online training fails with 25 pretrain-



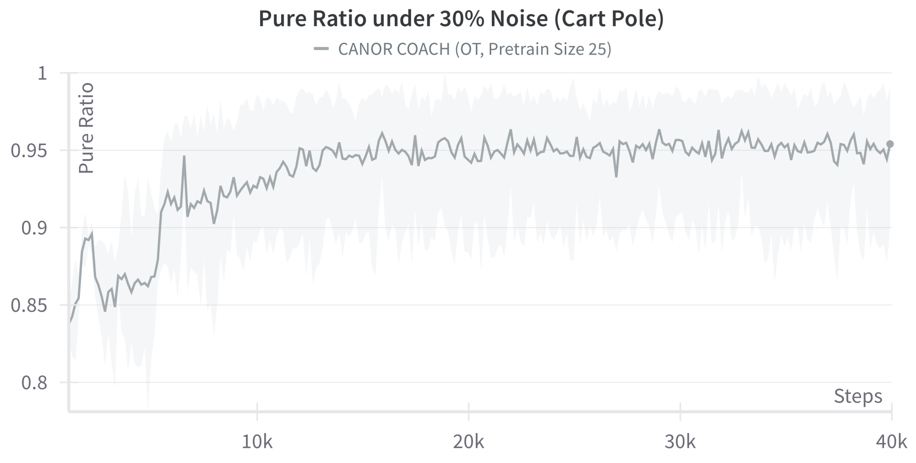
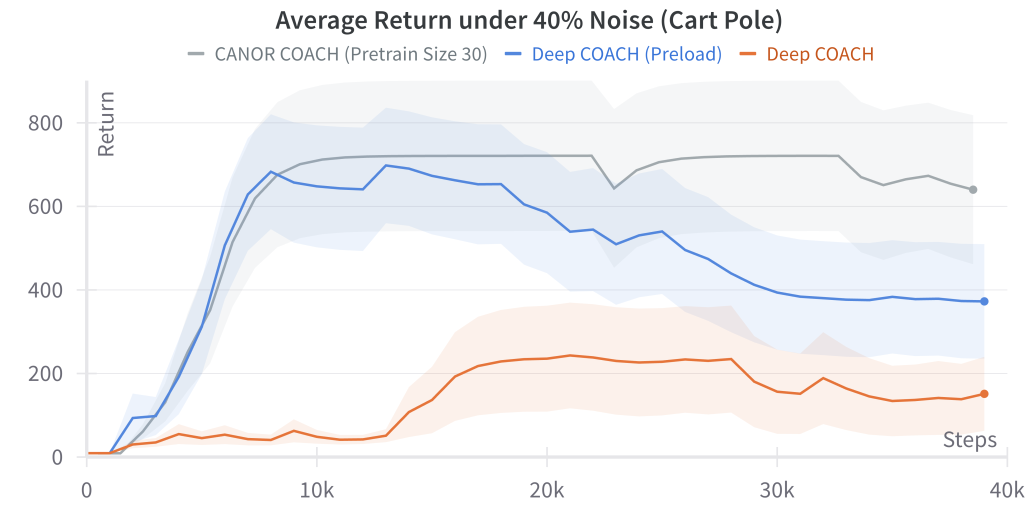


Figure 5.11: Performance of CANOR COACH with online training in 30% noise. With online training, CANOR COACH is able to learn against 30% noise with merely a pretraining dataset of size 25. Furthermore, its pure ratio successfully increases over time and reaches 95%, while CANOR COACH without online training decreases over time.

ing data and needs 30 to succeed. We also observe that the pure ratio under extremely high noise will go down and then fluctuate, which differs from the pattern in Figure 5.11 and Figure 5.6, which shows that the online training mechanism tends to perform worse under high noise. However, this is still better than CANOR COACH. With online training, CANOR COACH requires less data for pretraining and results in a higher pure ratio during training.



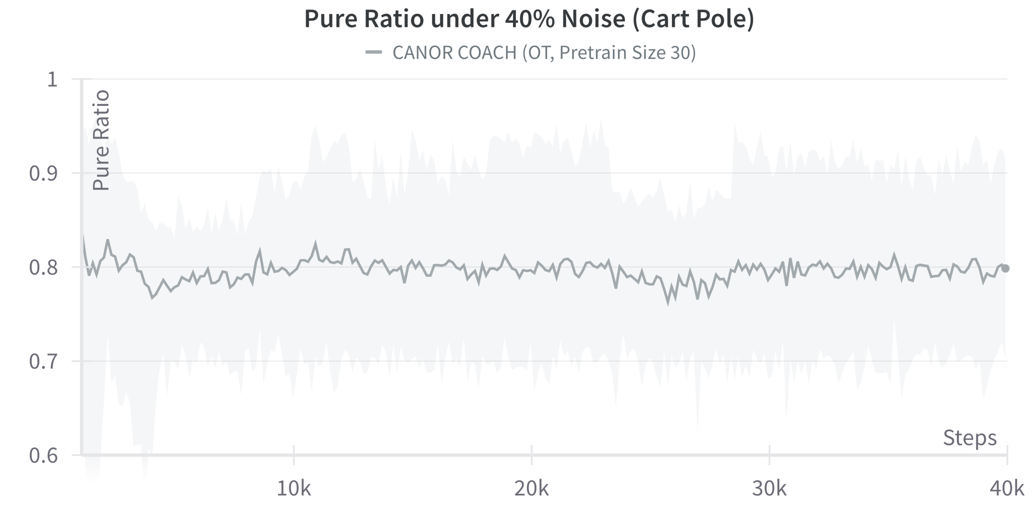


Figure 5.12: Performance of CANOR COACH with online training in 40% noise.

5.4.2 Minigrid Doorkey

Results in 30% and 40% noise in Minigrid Doorkey can be found in Figure 5.15 and Figure 5.16. CANOR COACH with online training is able to outperform two baselines in 30% noise. Furthermore, CANOR COACH with online training is able to learn faster than our baselines and keeps its performance stable while Deep COACH (preload)'s performance starts decline as times goes by. However, when the noise scale increases to 40%, CANOR COACH with online training cannot perform as well and only surpasses Deep COACH in performance. This again suggests that high noise can hurt performance of our algorithm.

5.4.3 Lunar Lander

Lunar Lander is our hardest domain because of its relatively high dimensions in observation space. As suggested by Figure 5.18, CANOR COACH with Online Training is not able to learn at all and completely fails to learn. CANOR COACH is only able to achieve similar performance with Deep COACH under 20% noise as shown in Figure 5.17 and learns still slower with less stability. Although both of the baselines fail to achieve satisfactory performance, CANOR COACH with Online Training is hurt even more by noise, due to its classifier also receives negative influence from noises during training.

5.4.4 Summary

In this section, we evaluated the results with CANOR COACH with Online Training. It generally shows better performance than CANOR COACH against high noises. However, its learning is less reliable and stable for more complicated domains like Lunar Lander due to its high observation space, which makes the classifier more difficult to learn a correct pattern to de-noise. In the next section, we will show that active relabelling can be a relief to this.

5.5 Active relabeling evaluation

In this subsection, we discuss how active relabelling will influence the algorithm's performance. If we had a perfect classifier that can detect all noisy labels, we may flip their labels to augment the dataset since the feedback is binary. However, it is very hard and even unrealistic to acquire such a classifier. Therefore, it remains questionable if flipping detected noisy labels can help. We conduct the following experiments to see whether, with a properly fine-tuned flipping rate, active relabelling can further improve CANOR COACH's performance.

5.5.1 Cart Pole

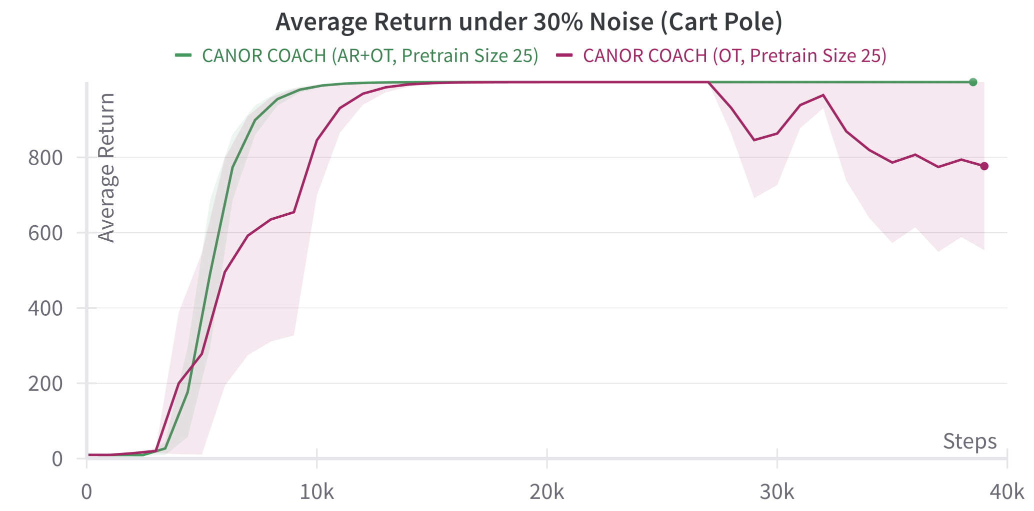
The performance of CANOR COACH with online training and active relabelling in 30% noise is shown in Figure 5.13. It is clearly seen that adding active relabelling not only achieves better performance in 30% noise but also significantly increases the pure ratio compared to CANOR with only online training. In 40% noise, as shown in Figure 5.14, CANOR COACH with active relabelling cannot learn stably against such a high level of noise. But it already outperforms CANOR with only online training, which completely fails to learn and shows episodic returns of zero.

5.5.2 Door key

In Minigrid Doorkey, the best performance is achieved with active relabeling, as shown in Figure 5.15 for 30% noise and Figure 5.16 for 40% noise. With active relabelling, CANOR COACH successfully outperforms our two baselines. Furthermore, active relabeling also results in a higher pure ratio, which means the agent is learning with fewer noisy labels.

5.5.3 Lunar Lander

The experimental results of CANOR COACH with active relabelling and online training can be seen in Figure 5.17 (20% noise), Figure 5.18 (30% noise) and Figure 5.19 (40% noise). We can see in Figure 5.17 and Figure 5.18 that



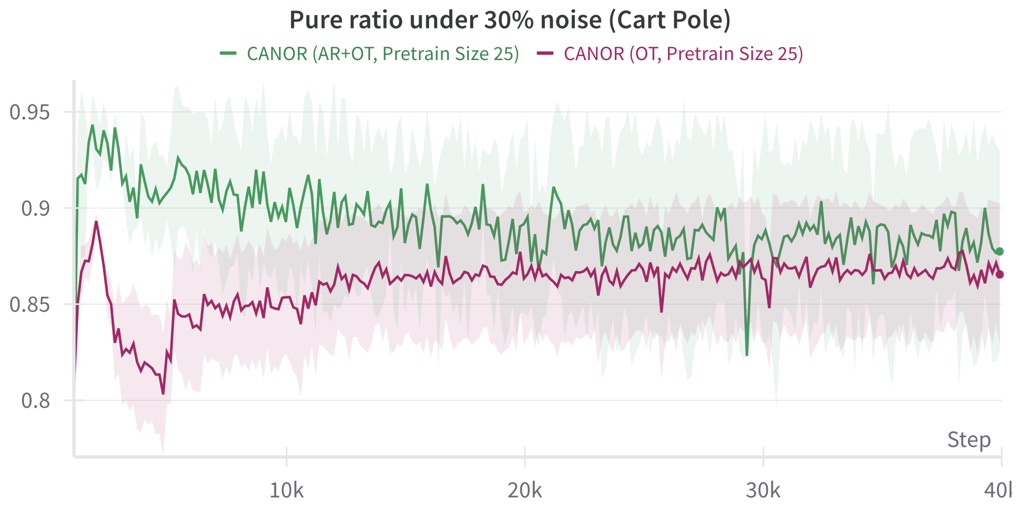
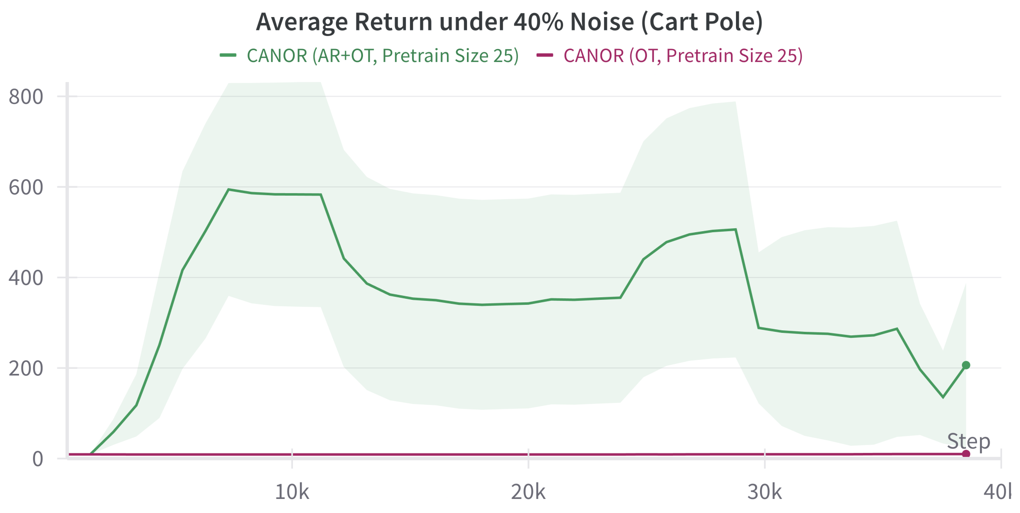


Figure 5.13: Performance of CANOR COACH with online training and active relabelling in 30% noise in Cart Pole



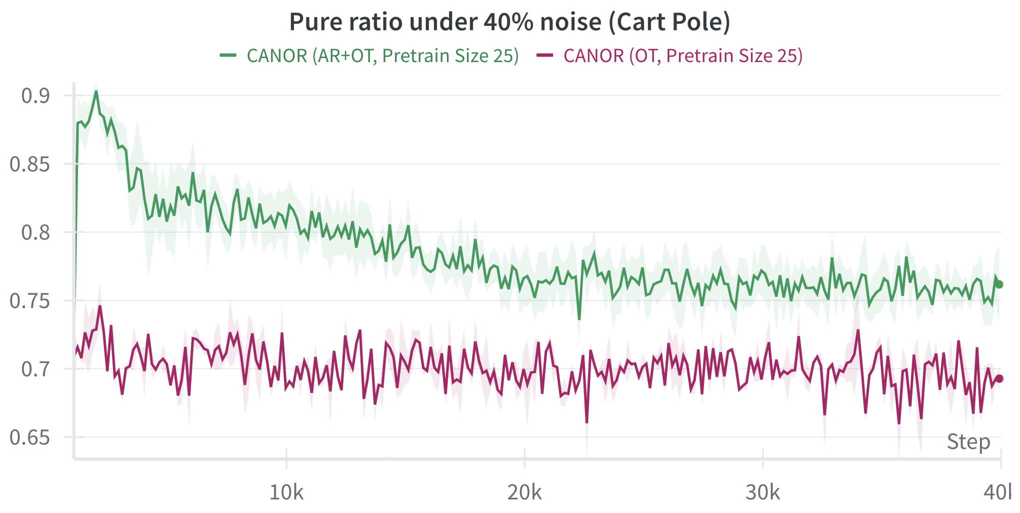
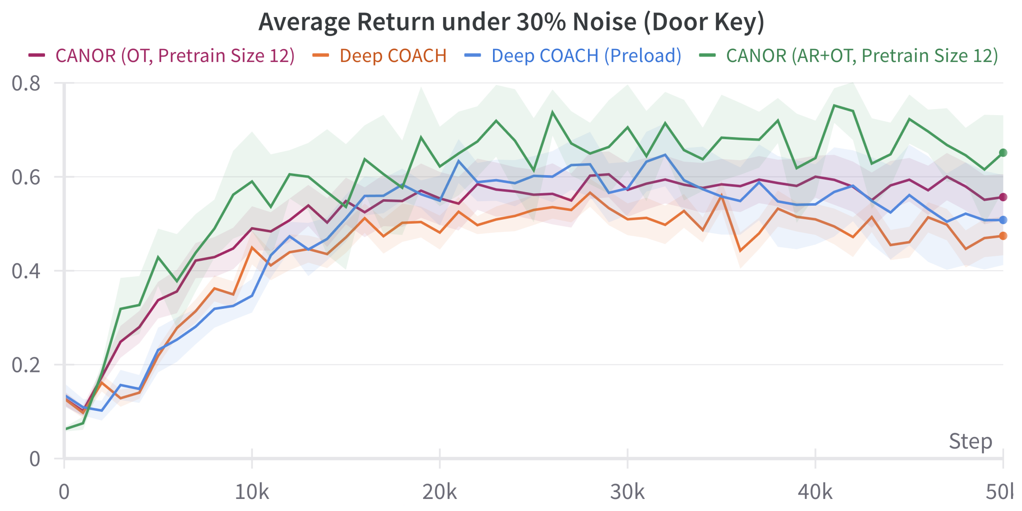


Figure 5.14: Performance of CANOR COACH with online training and active relabelling in 40% noise in Cart Pole



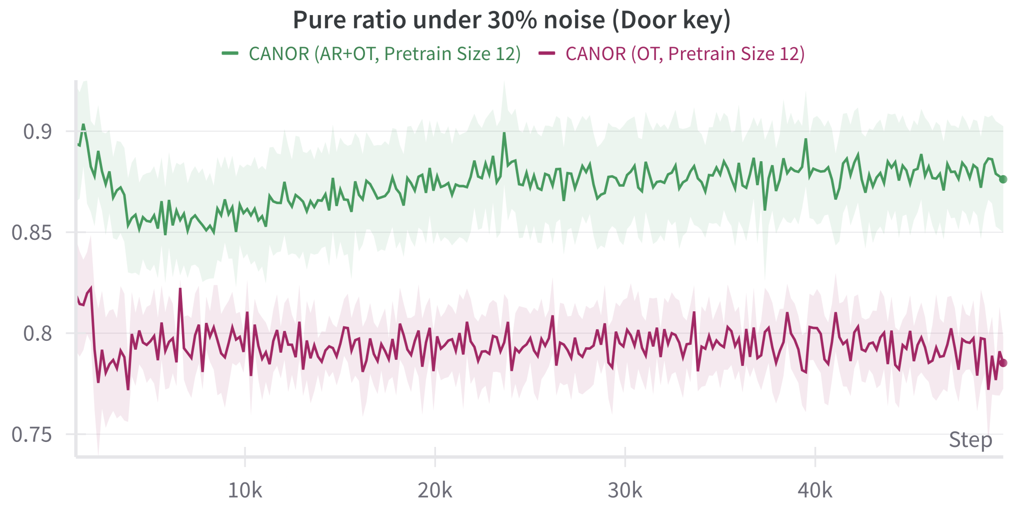
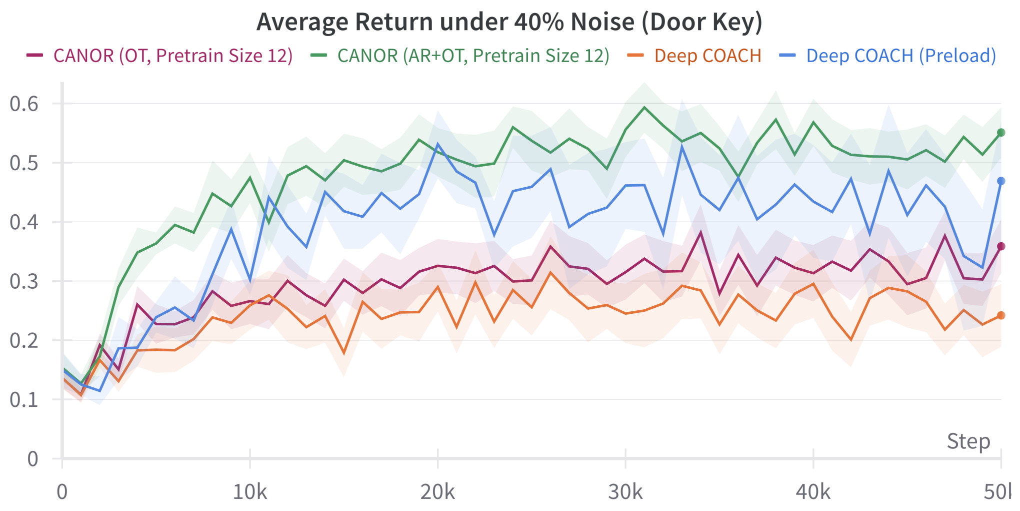


Figure 5.15: Performance and pure ratio of CANOR COACH with online training and active relabelling in 30% noise in Doorkey



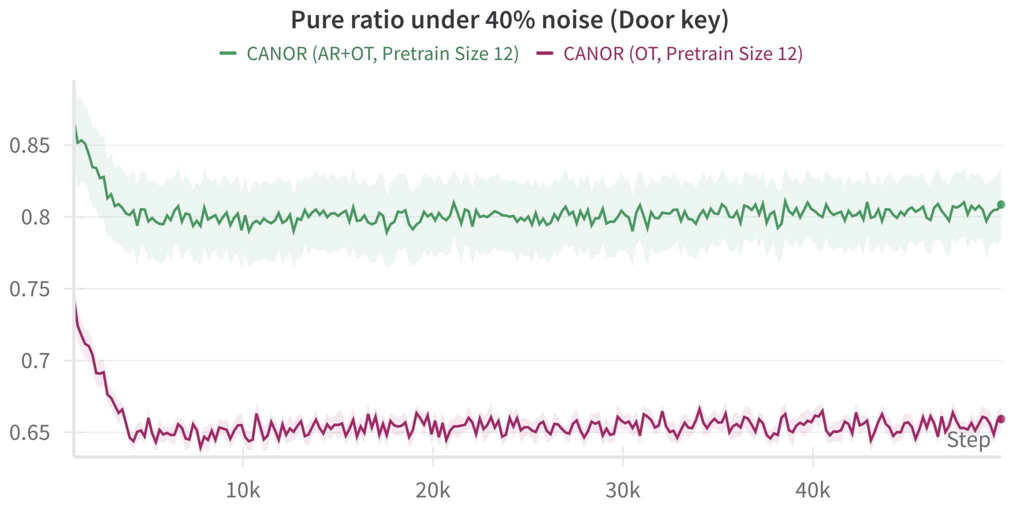


Figure 5.16: Performance and pure ratio of CANOR COACH with online training and active relabelling in 40% noise in Doorkey

CANOR COACH show excellent performance against 20% and 30% noise, while our baselines fail to reach an episode return of 200. However, our most complicated domain, Lunar lander, is still challenging under extremely high noise like 40%, as shown in Figure 5.19. CANOR COACH with active relabelling and online training achieves the best episodic return, but it still suffers from noise and fails to reach 200 in average episodic return.

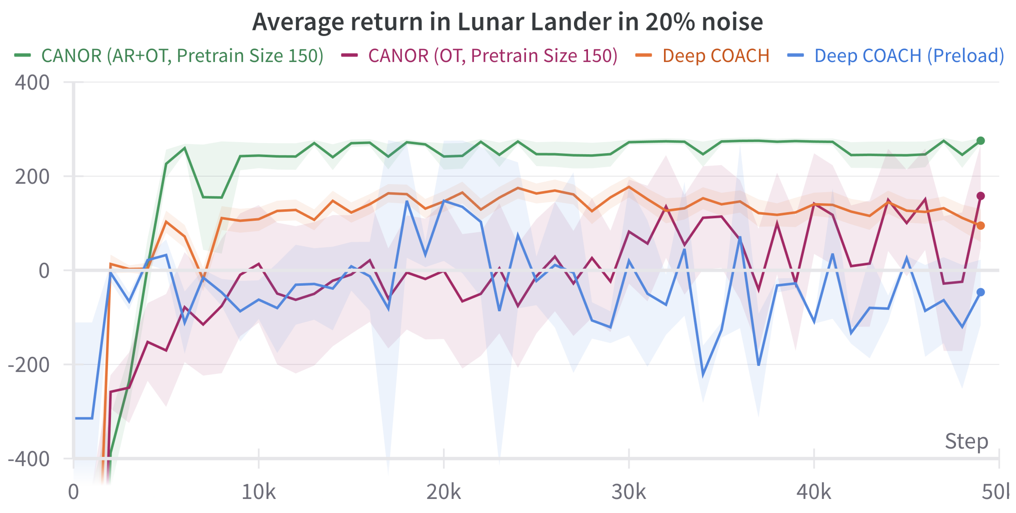
5.6 Noisy pretraining dataset

Recalling in the previous sections, we allow a small amount of noise-free feedback dataset for pretraining the classifier. In this section, we test if our algorithm can still work with a noisy pretraining dataset, i.e., achieve unsupervised anomaly detection.

The experimental results in Cart Pole can be found in Figure 5.20. Figure 5.20 shows that CANOR COACH with active relabelling and online training can still perform well against low amounts of noise, such as 10% and 20%, while CANOR COACH with only online training cannot outperform our baselines (Deep COACH). However, if the noise continues to rise, we see our algorithms fail to outperform Deep COACH at 30% noise. In fact, pretraining with noisy dataset already deteriorates the performance compared to the performance reported in previous sections, where a noise-free pretraining dataset is available. To summarise, CANOR COACH still works with noisy pretraining dataset but it will only show promising results and noise robustness under low noise levels, such as 20% or 10% noise.

5.7 Ablation study on learning from agreement

Our algorithm design is based on the small loss trick and the classifier is trained to predict the correct feedback for a state action pair. Therefore, it comes very naturally that we can also directly learn from the feedback that the classifier agrees, i.e. the received feedback is the same as the classifier's prediction. We call this learning from agreement. In this section, we conduct an experiment



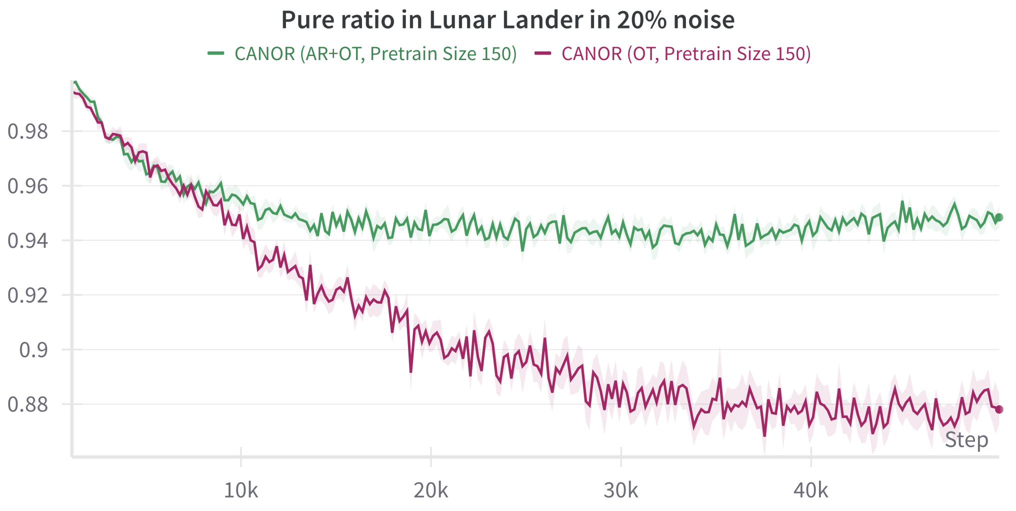
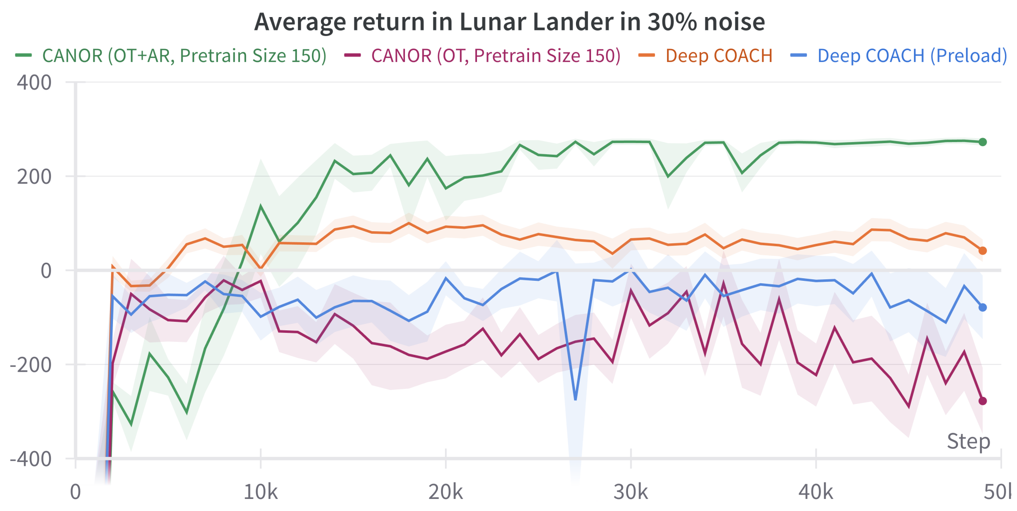


Figure 5.17: Performance and pure ratio of CANOR COACH with online training and active relabelling in 20% noise in Lunar lander



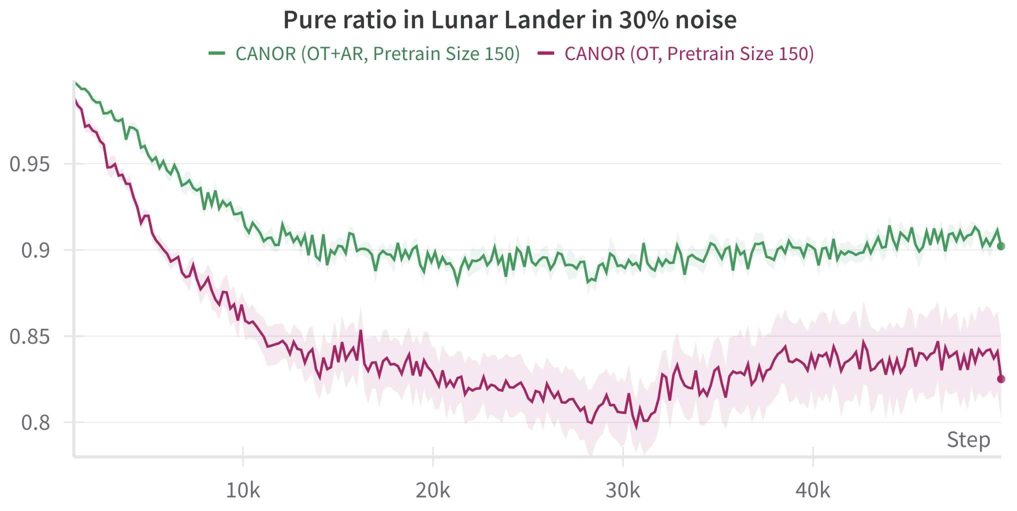
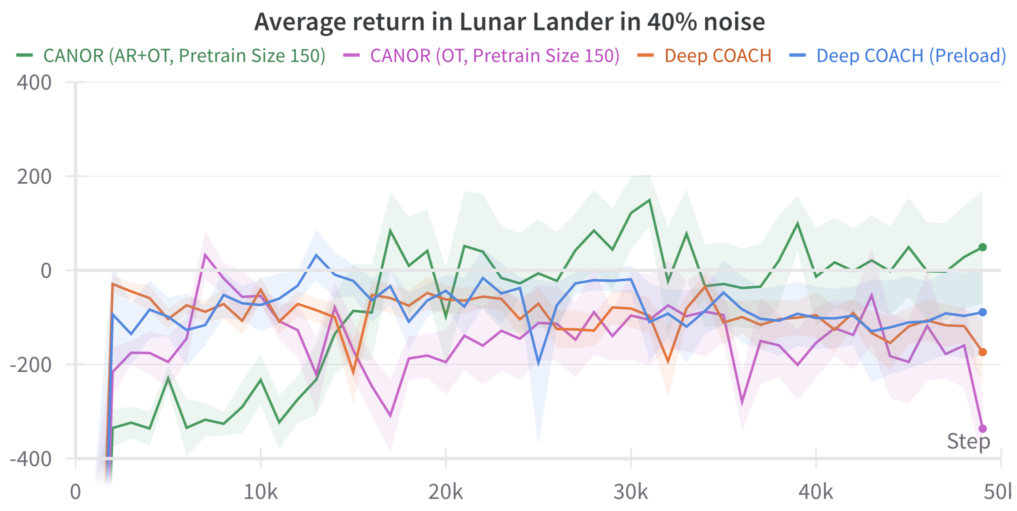


Figure 5.18: Performance and pure ratio of CANOR COACH with online training and active relabelling in 30% noise in Lunar lander



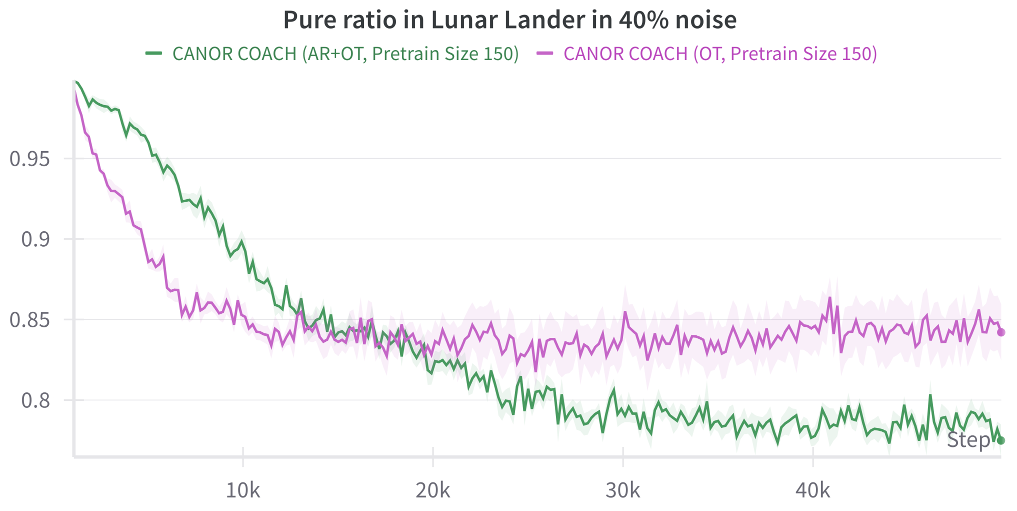
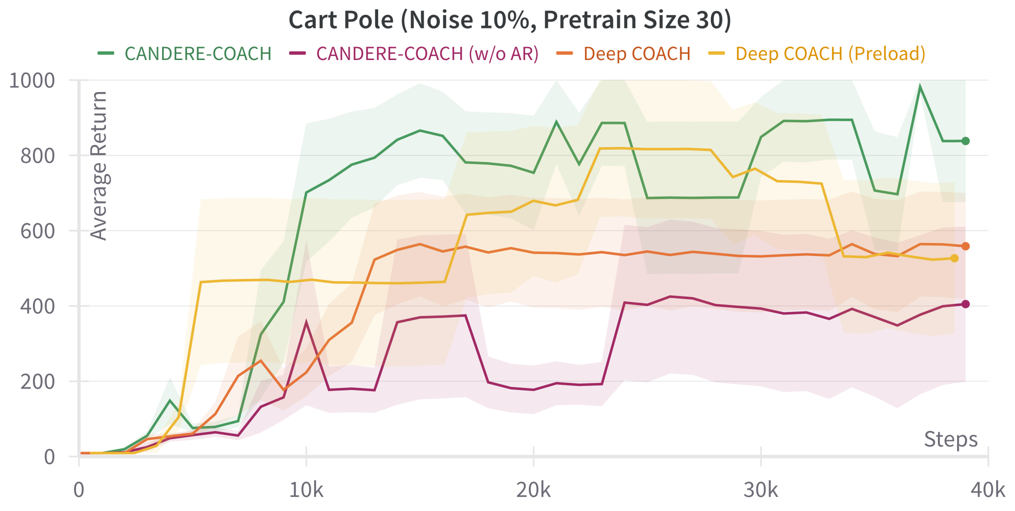
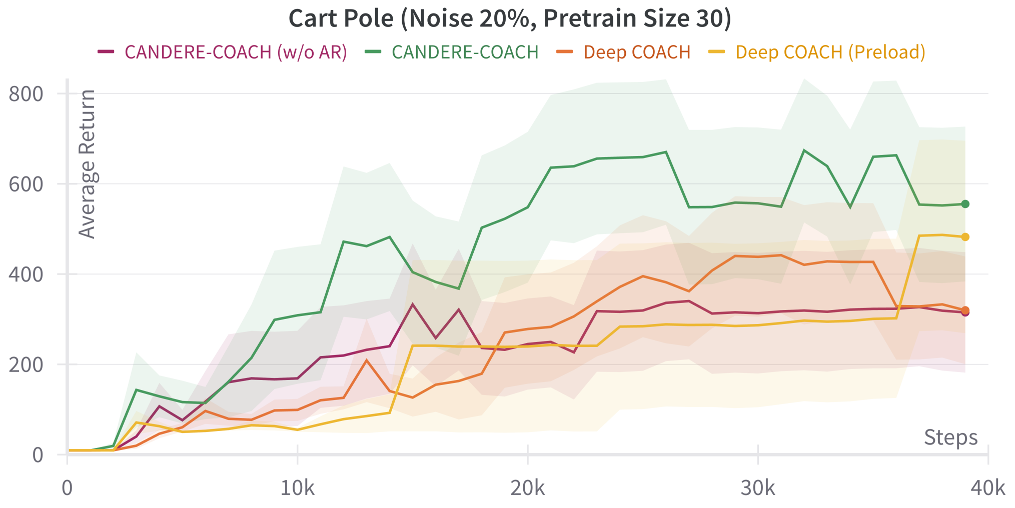


Figure 5.19: Performance and pure ratio of CANOR COACH with online training and active relabelling in 40% noise in Lunar lander





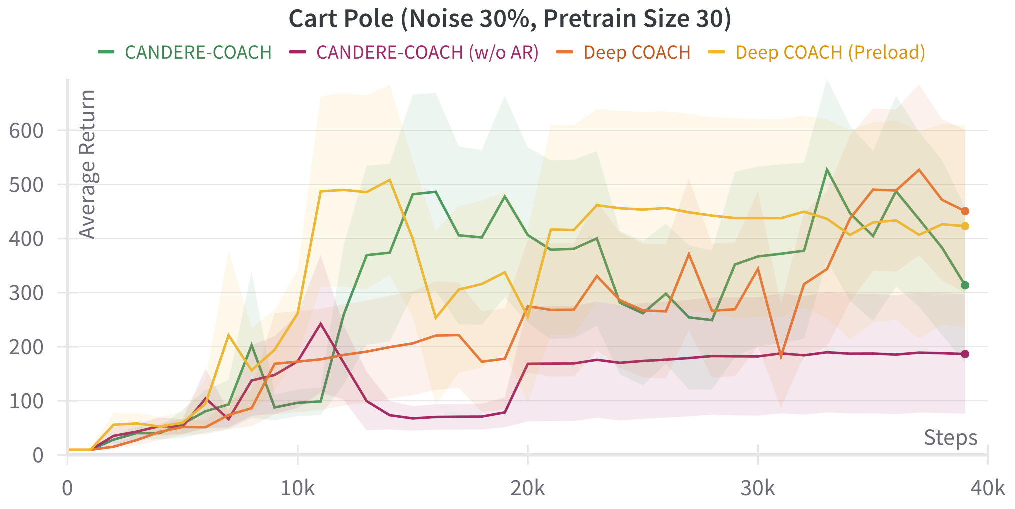


Figure 5.20: Performance CANOR COACH with noisy pretraining dataset in different noise scales in Cart Pole

to show that learning from agreement cannot successfully filter noisy labels and enable agent to learn to solve the task.

We can see from Figure 5.21, that the learning from agreement does not allow the agent to reach high episodic return compared to CANOR COACH, because we only have access to a small feedback dataset for pretraining, and the classifier cannot generalise well on this limited amount of data in terms of predicting accurate feedback. However, its cross entropy loss can still suggest the trend of being correct or not, and therefore, using the small loss trick here allows better performance in filtering noise.

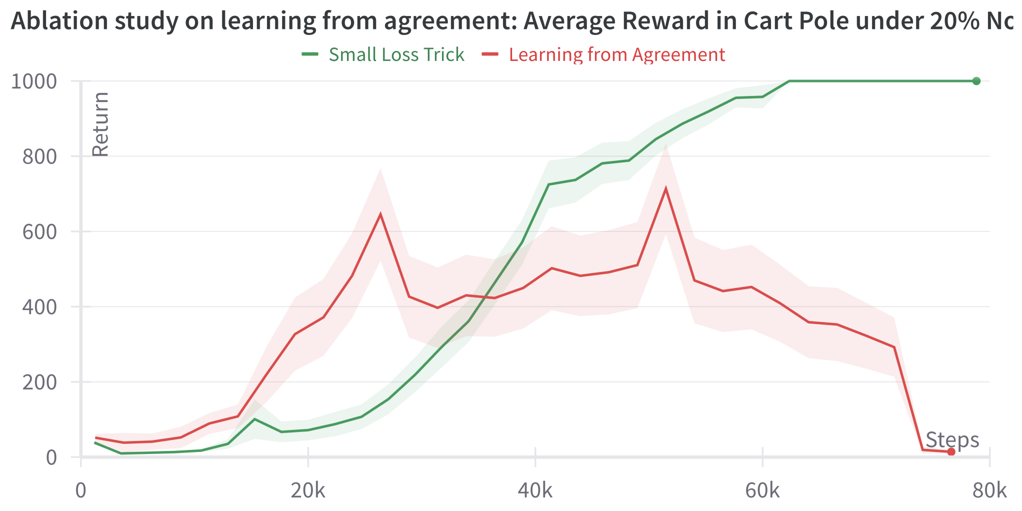


Figure 5.21: Ablation study on learning from agreement

5.8 Study on Noise Scale Estimation

Noticeably, all our experiments are conducted assuming the actual noise scale is known. However, in real-world scenarios, this might not be available. In this section, we conduct experiments to see if the proposed algorithm in Section 3.6 can correctly estimate the noise scale.

Table 5.2 shows that the estimated noise is close to the ground truth noise, with an estimation error less than 4%, when we have 5000 feedback. However, the proposed algorithm is not as reliable when we have a smaller dataset. The results of same experiment with 500 feedback can be seen in Table 5.1. With only 500 feedback, the estimation error rises up to 12% when we have budget of 500. This shows that the proposed noise estimation method requires a large amount of feedback data to achieve a rather accurate estimate, more than the whole budget we can have in some of our previous experiments.

Noise	Average Estimated Noise	Standard Variance
0.1	0.084	0.0129
0.2	0.146	0.0192
0.3	0.218	0.0236
0.4	0.284	0.0197

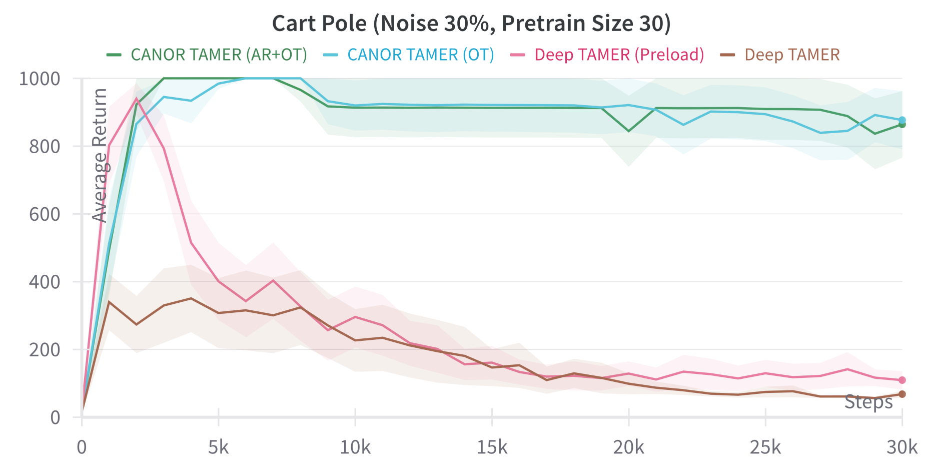
Table 5.1: Noise estimation with dataset of 500 feedback in Cart Pole

Noise	Average Estimated Noise	Standard Variance
0.1	0.140	0.0196
0.2	0.235	0.0297
0.3	0.324	0.0280
0.4	0.382	0.0232

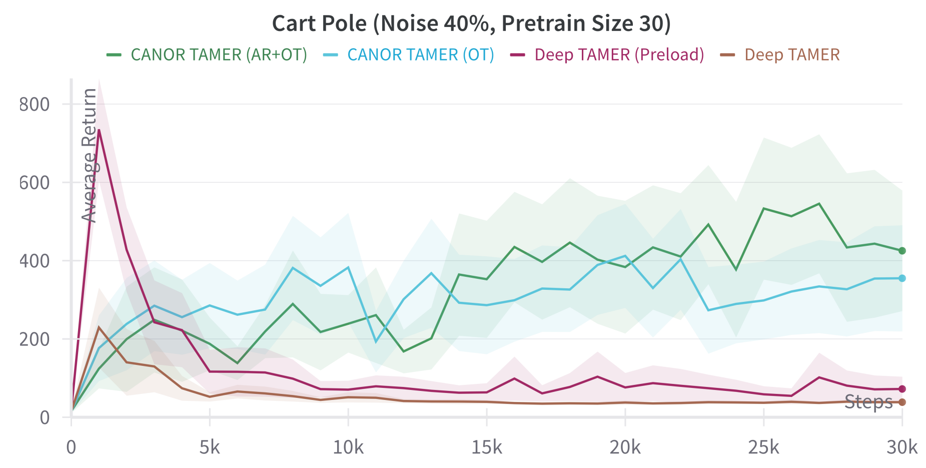
Table 5.2: Noise estimation with dataset of 5000 feedback in Cart Pole

5.9 Extending to TAMER: CANOR TAMER

There is no fundamental challenge to expand our proposed de-noising mechanism to other learning from feedback algorithms. Here, we present CANOR TAMER, which is similar to CANOR COACH but built based on Deep TAMER [42]. The results are shown in Figure 5.22. We observe a similar pattern in which our algorithms outperform our baselines (Deep TAMER) and learn successfully against 30% noise, while performance of baselines degrade poorly with noise. In 40% noise, CANOR TAMER still shows better average return and outperforms Deep TAMER; however it's performance is significantly worse than that with 30% noise. This experiment shows the potential of our approach to be used as plug and play inside any human-in-the-loop RL algorithm which will be explored as future work.



(a) Average return in 30% Noise



(b) Average return in 40% Noise

Figure 5.22: Performance camparison of CANOR TAMER in Cart Pole

Chapter 6

Conclusion and Future work

In this chapter, we first summarise the conclusion and then discuss possible future work.

6.1 Conclusion

In this thesis, a brand-new noise-robust learning framework (CANOR COACH) is proposed for learning from feedback reinforcement learning algorithms. Through experiments in different settings and domains, we show that the proposed CANOR COACH is able to handle up to 40% noise, with only a small noise-free feedback dataset, while the baseline (Deep COACH) fails to learn and shows high instability. We also show that if the noise is small enough, our proposed CANOR COACH can work with a noisy pretraining dataset. An ablation study and further experiments show that online training and active relabelling successfully improve the performance and reduce the requirement of pretraining dataset. Furthermore, the proposed noise detection mechanism shows potential to be extended to other human-in-the-loop reinforcement learning algorithms, like TAMER. To summarise, a first step is made towards noise-robust human-in-the-loop reinforcement learning by introducing a denoising classifier for noisy feedback.

6.2 Discussion and future work

Noise remains a challenge in human-in-the-loop reinforcement learning, even though CANOR COACH shows excellent performance against extreme noise.

First, because this work draws inspiration from previous work in supervised learning with noisy labels and anomaly detection, CANOR COACH similarly requires knowledge of the percentage of noise. However, in real-world applications to human teachers, there are no well-established methods to calibrate the noise, and our proposed algorithm to calibrate the noise shows low data efficiency. Improving calibration of the percentage of noise can be an interesting future work.

Second, while we conduct experiments with static, symmetric noise, we still need to challenge to a more realistic setting: the noise can be observation-dependent and also changing over time. Future work will be done to study different types of noise.

Third, the current CANOR COACH works only in domains with discrete action spaces. While we do not see fundamental challenge to an expansion of CANOR COACH to continuous action space, a continuous action space will increase the requirement of pretraining dataset size. Therefore, improving the classifier's data efficiency can be a very interesting future work.

Lastly, in this thesis, we focus on learning from feedback algorithms. To the best of our knowledge, this current design of CANOR COACH can be a plug-in to any learning from preferences algorithms, such as PEBBLE [21]. We aim to expand our work to learning from preferences in the future.

References

[1] 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.

[2] I. Akkaya, M. Andrychowicz, M. Chociej, *et al.*, “Solving rubik’s cube with a robot hand,” *arXiv preprint arXiv:1910.07113*, 2019.

[3] B. D. Argall, S. Chernova, M. Veloso, and B. Browning, “A survey of robot learning from demonstration,” *Robotics and autonomous systems*, vol. 57, no. 5, pp. 469–483, 2009.

[4] D. Arumugam, J. K. Lee, S. Saskin, and M. L. Littman, “Deep reinforcement learning from policy-dependent human feedback,” *arXiv preprint arXiv:1902.04257*, 2019.

[5] A. G. Barto, R. S. Sutton, and C. W. Anderson, “Neuronlike adaptive elements that can solve difficult learning control problems,” *IEEE transactions on systems, man, and cybernetics*, no. 5, pp. 834–846, 1983.

[6] C. Berner, G. Brockman, B. Chan, *et al.*, “Dota 2 with large scale deep reinforcement learning,” *arXiv preprint arXiv:1912.06680*, 2019.

[7] A. Billard, Y. Epars, G. Cheng, and S. Schaal, “Discovering imitation strategies through categorization of multi-dimensional data,” in *IROS*, vol. 3, 2003, pp. 2398–2403.

[8] C. Celemin and J. Ruiz-del-Solar, “Coach: Learning continuous actions from corrective advice communicated by humans,” in *2015 International Conference on Advanced Robotics (ICAR)*, IEEE, 2015, pp. 581–586.

[9] G. Chen, X. Zhang, Z. J. Wang, and F. Li, “Robust support vector data description for outlier detection with noise or uncertain data,” *Knowledge-Based Systems*, vol. 90, pp. 129–137, 2015.

[10] M. Chevalier-Boisvert, B. Dai, M. Towers, *et al.*, “Minigrid & miniworld: Modular & customizable reinforcement learning environments for goal-oriented tasks,” *CoRR*, vol. abs/2306.13831, 2023.

[11] P. F. Christiano, J. Leike, T. Brown, M. Martic, S. Legg, and D. Amodei, “Deep reinforcement learning from human preferences,” *Advances in neural information processing systems*, vol. 30, 2017.

[12] N. Das, S. Bechtle, T. Davchev, D. Jayaraman, A. Rai, and F. Meier, “Model-based inverse reinforcement learning from visual demonstrations,” Proceedings of Machine Learning Research, vol. 155, pp. 1930–1942, 2020.

[13] A. Giusti, J. Guzzi, D. C. Cireşan, *et al.*, “A machine learning approach to visual perception of forest trails for mobile robots,” *IEEE RAL*, vol. 1, no. 2, pp. 661–667, 2015.

[14] B. Han, Q. Yao, X. Yu, *et al.*, “Co-teaching: Robust training of deep neural networks with extremely noisy labels,” *Advances in neural information processing systems*, vol. 31, 2018.

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

[16] E. Ilhan, J. Gow, and D. Perez, “Student-initiated action advising via advice novelty,” *IEEE Transactions on Games*, vol. 14, no. 3, pp. 522–532, 2021.

[17] E. Ilhan, J. Gow, and D. Perez-Liebana, “Action advising with advice imitation in deep reinforcement learning,” *arXiv preprint arXiv:2104.08441*, 2021.

[18] L. Jiang, Z. Zhou, T. Leung, L.-J. Li, and L. Fei-Fei, “Mentornet: Learning data-driven curriculum for very deep neural networks on corrupted labels,” in *International conference on machine learning*, PMLR, 2018, pp. 2304–2313.

[19] W. B. Knox and P. Stone, “Interactively shaping agents via human reinforcement: The TAMER framework,” in *Proceedings of the fifth international conference on Knowledge capture*, 2009, pp. 9–16.

[20] A. Kumar, A. Zhou, G. Tucker, and S. Levine, “Conservative q-learning for offline reinforcement learning,” *Advances in Neural Information Processing Systems*, vol. 33, pp. 1179–1191, 2020.

[21] K. Lee, L. Smith, and P. Abbeel, “Pebble: Feedback-efficient interactive reinforcement learning via relabeling experience and unsupervised pretraining,” *arXiv preprint arXiv:2106.05091*, 2021.

[22] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, “Focal loss for dense object detection,” in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 2980–2988.

[23] R. Loftin, B. Peng, J. MacGlashan, *et al.*, “Learning behaviors via human-delivered discrete feedback: Modeling implicit feedback strategies to speed up learning,” *Autonomous agents and multi-agent systems*, vol. 30, pp. 30–59, 2016.

[24] J. MacGlashan, M. K. Ho, R. Loftin, *et al.*, “Interactive learning from policy-dependent human feedback,” in *International Conference on Machine Learning*, PMLR, 2017, pp. 2285–2294.

[25] R. Mansoor, N. D. Jayasinghe, and M. M. A. Muslam, "A comprehensive review on email spam classification using machine learning algorithms," in *2021 International Conference on Information Networking (ICOIN)*, IEEE, 2021, pp. 327–332.

[26] A. Mao, M. Mohri, and Y. Zhong, "Cross-entropy loss functions: Theoretical analysis and applications," in *International Conference on Machine Learning*, PMLR, 2023, pp. 23 803–23 828.

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

[28] A. Raffin, *Rl baselines3 zoo*, `https://github.com/DLR-RM/rl-baselines3-zoo`, 2020.

[29] A. Rajeswaran, V. Kumar, A. Gupta, *et al.*, "Learning complex dexterous manipulation with deep reinforcement learning and demonstrations," *Robotics: Science and Systems*, 2018. DOI: `10.15607/RSS.2018.XIV.049`.

[30] S. Ross, G. Gordon, and D. Bagnell, "A reduction of imitation learning and structured prediction to no-regret online learning," in *AISTATS*, JMLR Workshop and Conference Proceedings, 2011, pp. 627–635.

[31] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.

[32] D. Silver, A. Huang, C. J. Maddison, *et al.*, "Mastering the game of go with deep neural networks and tree search," *Nature*, 2016.

[33] D. Silver, J. Schrittwieser, K. Simonyan, *et al.*, "Mastering the game of go without human knowledge," *nature*, vol. 550, no. 7676, pp. 354–359, 2017.

[34] R. Singh, S. Mozaffari, M. Akhshik, M. J. Ahamed, S. Rondeau-Gagné, and S. Alirezaee, "Human–robot interaction using learning from demonstrations and a wearable glove with multiple sensors," *Sensors*, vol. 23, no. 24, p. 9780, 2023.

[35] H. Song, M. Kim, D. Park, Y. Shin, and J.-G. Lee, "Learning from noisy labels with deep neural networks: A survey," *IEEE Transactions on Neural Networks and Learning Systems*, 2022.

[36] M. T. Spaan and N. Spaan, "A point-based pomdp algorithm for robot planning," in *IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA'04. 2004*, IEEE, vol. 3, 2004, pp. 2399–2404.

[37] C. Sudha and D. Akila, "Credit card fraud detection system based on operational & transaction features using svm and random forest classifiers," in *2021 2nd International Conference on Computation, Automation and Knowledge Management (ICCAKM)*, IEEE, 2021, pp. 133–138.

[38] R. S. Sutton, D. McAllester, S. Singh, and Y. Mansour, "Policy gradient methods for reinforcement learning with function approximation," *Advances in Neural Information Processing Systems*, vol. 12, 1999.

[39] L. Torrey and M. Taylor, "Teaching on a budget: Agents advising agents in reinforcement learning," in *Proceedings of the 2013 international conference on Autonomous agents and multi-agent systems*, 2013, pp. 1053–1060.

[40] M. Towers, J. K. Terry, A. Kwiatkowski, *et al.*, *Gymnasium*, Mar. 2023. DOI: `10.5281/zenodo.8127026`. [Online]. Available: `https://zenodo.org/record/8127025` (visited on 07/08/2023).

[41] H.-H. Tseng, Y. Luo, S. Cui, J.-T. Chien, R. K. Ten Haken, and I. E. Naqa, "Deep reinforcement learning for automated radiation adaptation in lung cancer," *Medical physics*, vol. 44, no. 12, pp. 6690–6705, 2017.

[42] G. Warnell, N. Waytowich, V. Lawhern, and P. Stone, "Deep TAMER: Interactive agent shaping in high-dimensional state spaces," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 32, 2018.

[43] A. Wilson, A. Fern, and P. Tadepalli, "A Bayesian approach for policy learning from trajectory preference queries," in *NIPS*, 2012.

[44] Y. Yu, "Towards sample efficient reinforcement learning.," in *IJCAI*, 2018, pp. 5739–5743.

[45] C. Zhang, S. Bengio, M. Hardt, B. Recht, and O. Vinyals, "Understanding deep learning (still) requires rethinking generalization," *Communications of the ACM*, vol. 64, no. 3, pp. 107–115, 2021.

Appendix A

Experimental design

In this chapter, we discuss the experimental details and hyperparameters we used.

A.1 Scripted teacher

Our scripted teacher is trained with PPO [31] following hyperparameters of RL Zoo [28]. More details are described in the following subsections for each domain.

A.1.1 Cart Pole

In Cart Pole, the expert is trained with PPO with hyperparameters shown in Table A.1.

Hyperparameter	Value
Time steps	10^5
Rollout steps	32
Gae lambda	0.8
Gamma	0.9
Learning rate	$1e^{-3}$
Clip range	0.2
Batch size	256
Hidden units	64
Layers	2
Activation	ReLU

Table A.1: Hyperparamters of Cart Pole expert training

A.1.2 Door Key

In Door Key, the expert is trained with PPO with hyperparameters shown in Table A.2. Furthermore, Minigrid Doorkey is set to be full observable and we use a CNN-based feature extractor to reduce the observation space to 5.

Hyperparameter	Value
Time steps	10^5
Rollout steps	128
Gae lambda	0.95
Gamma	0.99
Learning rate	2.5^{-4}
Clip range	0.2
Batch size	64
Hidden units	64
Layers	2
Activation function	ReLU
CNN channels	[16, 32, 64]
CNN kernel size	$2, 2$

Table A.2: Hyperparamters of Door Key expert training

A.1.3 Lunar Lander

In Lunar Lander, the expert is trained with PPO with hyperparameters shown in Table A.3.

Hyperparameter	Value
Time steps	1e6
Rollout steps	1024
Gae lambda	0.98
Gamma	0.999
Learning rate	1e-3
Clip range	0.2
Batch size	64
Hidden units	64
Layers	2
Activation	ReLU

Table A.3: Hyperparamters of Lunar Lander expert training

A.2 Collecting the pretraining dataset

The pretraining dataset is collected with our scripted teacher to label with state-action pairs positive or negative feedback. We first collect states following a certain distribution to ensure better coverage of the state space. Then we label the states and optimal actions to be positive and the collected dataset will be randomly shuffled. Lastly, we practice data augmentation by labeling all the non-optimal actions to be negative. The details of the state sampling distribution of each domain is described in the following subsections.

A.2.1 Cart Pole

In Cart Pole, the state is sampled uniformly in a clipped observation space. The state space of Cart Pole consists of four dimensions and two of them is not bounded. Therefore, we properly choose a suitable clip range to sample the states. We set the sampling range of position and velocity based on the fact that an episode will be terminated if the cart leaves the $(-2.4, 2.4)$ range. Details can be seen in Table A.4.

Dimension	Original Min	Clipped Min	Original Max	Clipped Max
Cart Position	-4.8	-2.4	4.8	2.4
Cart Velocity	-Inf	-2.4	+Inf	2.4
Pole Angle	-0.418	-0.418	0.418	0.418
Pole Angular Velocity	-Inf	-0.418	+Inf	0.418

Table A.4: Sampling space of Cart Pole

A.2.2 Door Key & Lunar Lander

In Door Key and Lunar Lander, the dataset is sampled from trajectories following a sampling policy that takes expert action by 50% chance and random action by 50% chance. The reason of this is different for these domains. In Door Key, we cannot practice uniform sampling in the RGB image array space. In Lunar Lander, the observation space is significantly larger and if we practice uniform sampling, most sampled states will never be visited by the agent and therefore brings low performance of the classifier.

A.3 COACH Hyperparameters

In this section, we show the hyperparameters used in our experiments for CANOR COACH and Deep COACH. Deep COACH shares the same hyperparameters with CANOR COACH if applicable.

A.3.1 Cart Pole

The hyperparameters in Cart Pole can be seen in Table A.5.

Hyperparameter	Value
Actor learning rate	0.00005
Batch size	256
Budget	1000
Eligibility trace windows size	10
Eligibility trace decay factor	0.35
Classifier learning rate	0.01
Feedback frequency	10
Actor hidden units	1024
Actor layers	2
Actor activation	ReLU
Q function hidden units	1024
Q function layers	2
Q function activation	ReLU
Classifier hidden units	64
Classifier layers	2
Classifier activation	ReLU
Classifier pretraining epochs	100
Classifier pretraining learning rate	0.001
Classifier pretraining loss	Cross entropy loss
Active relabelling rate	0.6

Table A.5: Hyperparamters of CANOR COACH in Cart Pole

A.3.2 Door Key

The hyperparameters in Door Key can be seen in Table A.6.

A.3.3 Lunar Lander

The hyperparameters in Lunar Lander can be seen in Table A.7.

Hyperparameter	Value
Actor learning rate	0.00005
Batch size	256
Budget	500
Eligibility trace windows size	10
Eligibility trace decay factor	0.35
Classifier learning rate	0.001
Feedback frequency	10
Actor hidden units	1024
Actor layers	2
Actor activation	ReLU
Q function hidden units	1024
Q function layers	2
Q function activation	ReLU
Classifier hidden units	64
Classifier layers	2
Classifier activation	ReLU
Classifier pretraining epochs	100
Classifier pretraining learning rate	0.001
Classifier pretraining loss	Focal loss
Active relabelling rate	0.8

Table A.6: Hyperparamters of CANOR COACH in Cart Pole

Hyperparameter	Value
Actor learning rate	0.00005
Batch size	256
Budget	5000
Eligibility trace windows size	10
Eligibility trace decay factor	0.35
Classifier learning rate	0.001
Feedback frequency	10
Actor hidden units	1024
Actor layers	2
Actor activation	ReLU
Q function hidden units	1024
Q function layers	2
Q function activation	ReLU
Classifier hidden units	64
Classifier layers	2
Classifier activation	ReLU
Classifier pretraining epochs	100
Classifier pretraining learning rate	0.001
Classifier pretraining loss	Focal loss
Active relabelling rate	0.6

Table A.7: Hyperparamters of CANOR COACH in Lunar Lander

A.3.4 Summary on variations of CANOR COACH

We summarise the aforementioned CANOR COACH and its variations' domain-wise performance and required amount of pretraining dataset size, as shown in Table A.3.4.

Algorithm / Domain &Noise	CANOR COACH	CANOR COACH (OT)	CANOR COACH (AR+OT)
Cart Pole, 30%	Outperform with pretrain size 30	Outperform with pretrain size 25	Outperform with pretrain size 20
Cart Pole, 40%	Outperform with pretrain size 30	Outperform with pretrain size 30	Outperform with pretrain size 25
Door Key, 30%	Fail with pretrain size 12	Outperform with pretrain size 12	Outperform with pretrain size 12
Door Key, 40%	Fail with pretrain size 12	Fail with pretrain size 12	Outperform with pretrain size 12
Lunar Lander, 30%	Fail with pretrain size 150	Fail with pretrain size 150	Outperform with pretrain size 150
Lunar Lander, 40%	Fail with pretrain size 150	Fail with pretrain size 150	Outperform with pretrain size 150

Table A.8: Summary of results of CANOR COACH and its variations