Journal of Artificial Intelligence Research 79 (2024) 1237-1280 Submitted 03/2024; published 04/2024

USN: A Robust Imitation Learning Method against Diverse Action Noise

Xingrui Yu YU_XINGRUI@CFAR.A-STAR.EDU.SG
Centre for Frontier AI Research,
Agency for Science, Technology and Research, Singapore
Institute of High-Performance Computing,
Agency for Science, Technology and Research, Singapore

Bo Han BHANML@COMP.HKBU.EDU.HK
Department of Computer Science,
Hong Kong Baptist University, Hong Kong SAR

Ivor W. Tsang IVOR_TSANG@CFAR.A-STAR.EDU.SG
Centre for Frontier AI Research,
Agency for Science, Technology and Research, Singapore
Institute of High-Performance Computing,
Agency for Science, Technology and Research, Singapore
School of Computer Science and Engineering,
Nanyang Technological University, Singapore

Abstract

Learning from imperfect demonstrations is a crucial challenge in imitation learning (IL). Unlike existing works that still rely on the enormous effort of expert demonstrators, we consider a more cost-effective option for obtaining a large number of demonstrations. That is, hire annotators to label actions for existing image records in realistic scenarios. However, action noise can occur when annotators are not domain experts or encounter confusing states. In this work, we introduce two particular forms of action noise, i.e., *state-independent* and *state-dependent* action noise. Previous IL methods fail to achieve expert-level performance when the demonstrations contain action noise, especially the *state-dependent* action noise. To mitigate the harmful effects of action noises, we propose a robust learning paradigm called USN (Uncertainty-aware Sample-selection with Negative learning). The model first estimates the predictive uncertainty for all demonstration data and then selects samples with high loss based on the uncertainty measures. Finally, it updates the model parameters with additional negative learning on the selected samples. Empirical results in Box2D tasks and Atari games show that USN consistently improves the final rewards of behavioral cloning, online imitation learning, and offline imitation learning methods under various action noises. The ratio of significant improvements is up to 94.44%. Moreover, our method scales to conditional imitation learning with real-world noisy commands in urban driving.

1. Introduction

Reinforcement learning (RL) has achieved great success in various domains, including games (Crespo & Wichert, 2020) and robotics controls (Nguyen & La, 2019). Despite these successes, designing hand-crafted reward functions is challenging in many real-world tasks (Sutton & Barto, 2018; Lazaridis et al., 2020). Imitation learning (IL) offers an alternative approach by training an agent to mimic expert demonstrations without the need for hand-crafted rewards

(Russell, 1998; Schaal, 1999; Abbeel & Ng, 2004; Argall et al., 2009). When a large number of high-quality demonstrations are available, imitation learning has achieved impressive performance in playing games and urban driving (Ho & Ermon, 2016; Hussein et al., 2017; Lin et al., 2019; Brown et al., 2019; Codevilla et al., 2018). However, the demonstrations are often imperfect since collecting high-quality demonstrations in real-world tasks is often expensive and challenging (Silver et al., 2013; Wu et al., 2019; Wang et al., 2021b; Cao & Sadigh, 2021). Many previous studies have looked into imitation learning from imperfect demonstrations that are a mixture of optimal and non-optimal demonstrations (Wu et al., 2019; Tangkaratt et al., 2020b, 2020a; Zhang et al., 2021b; Wang et al., 2021b). A limitation of these methods is that they require more optimal demonstrations than non-optimal ones in the dataset. Thus, the generation of such imperfect demonstrations still costs enormous expert efforts.

In this paper, we focus on imitation learning tasks with discrete action space. To maximally reduce the experts' cost for generating demonstrations, one alternative way is to use a crowdsourcing platform to make action annotations for extensive available image records (Audiffren et al., 2015). For example, in the scenario of urban driving, the image records of first-person view can be easily collected, when an expert driver is driving a car on the road. However, using experts to annotate all the data can be costly by selecting correct actions from a set of discrete commands: {*continue*, *left*, *straight*, and *right*}. On the other hand, amateurs are cheaper but unable to give precise action labels (Lei et al., 2022). Annotators with professional domain knowledge are rare, while the amateurs are cheaper and numerous because they are regular crowds like part-time students (Li et al., 2016). Without professional domain knowledge, amateur annotators make mistakes when the data is confusing. This is the primary source of action noise in the annotated demonstrations. As shown in Figure 1, amateur annotators are required to output action labels for the perceptual image records of the car view. Amateur annotators usually output wrong action labels, when the perceptual image records are very confusing at the intersection. We refer to this type of action noise as *state-independent* action noise.

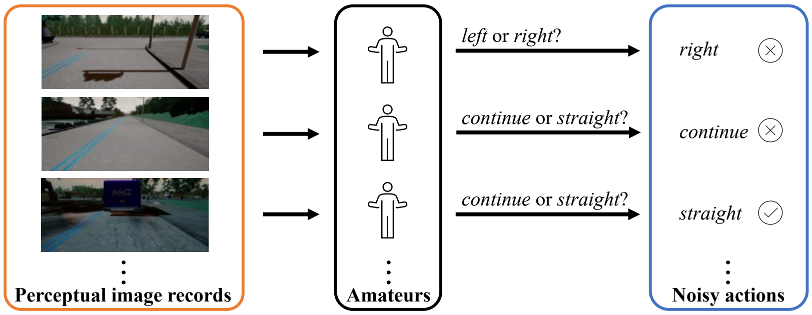


Figure 1: The perceptual image records in urban driving can be very confusing. The amateurs tend to output noisy action labels for the perceptual image records at the intersection of the straight section.

Another cost-effective method for utilizing the numerous image records is to train an action predictor with a few expert annotations, and use the predictor to output pseudo action labels for the rest of the numerous records. Since the training data is rare, the action predictions will not be precise, and action noises depend on the state features. We refer to such action noise as *state-dependent* action noise.

In this paper, we introduce two action noise models, i.e., *state-independent* and *state-dependent* action noise, to study the potential negative impacts of real-world action noises on imitation learning. Existing robust imitation learning methods have yet to consider the existence of such realistic action noises in the demonstrations. They always fail to learn a robust policy with such action noises in the demonstrations. To mitigate the adverse effects of real-world action noise, we propose a new method called Uncertainty-aware Sample-selection with soft Negative learning (USN) that takes into account the correlation between loss and uncertainty estimations (Naeini et al., 2015; Guo et al., 2017) as the noise rate increases.

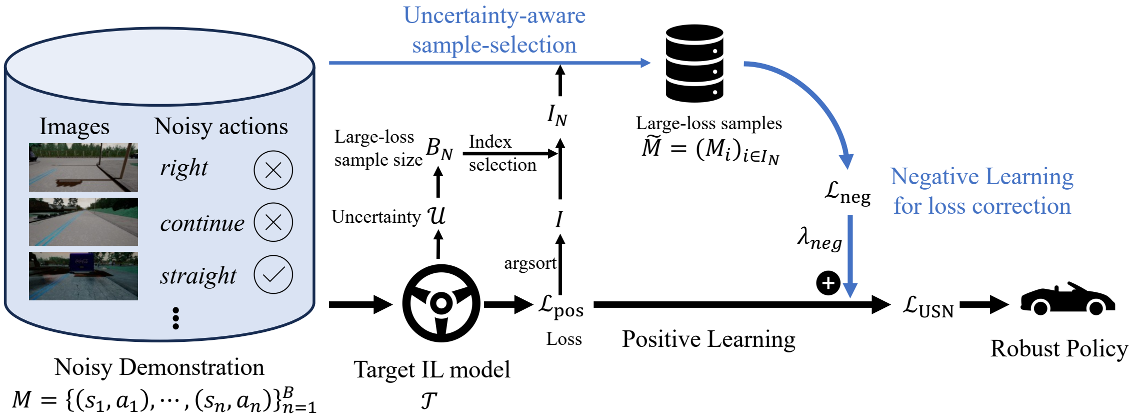

Uncertainty-aware sample-selection
Images
Noisy actions
right
continue
straight
Noisy Demonstration
$M = \{(s_1, a_1), \cdots, (s_n, a_n)\}_{n=1}^{B}$
Large-loss sample size B_N
Index selection
Uncertainty $\mathcal{U}$
I_N
I
argsort
$\mathcal{L}_{\text{pos}}$
Loss
Target IL model
$\mathcal{T}$
Large-loss samples
$\widetilde{M} = (M_i)_{i \in I_N}$
$\mathcal{L}_{\text{neg}}$
λ_{neg}
Negative Learning for loss correction
Positive Learning
$\mathcal{L}_{\text{USN}}$
Robust Policy

Figure 2: The whole procedure of USN includes two steps: (1) uncertainty-aware sample-selection and (2) negative learning for loss correction. Given a batch of noisy demonstration $M = \{(s_1, a_1), \cdots, (s_n, a_n)\}_{n=1}^{B}$, USN first conducts positive learning on the target imitation learning model $\mathcal{T}$ and calculates the uncertainty estimation $\mathcal{U}$. Then, USN uses $\mathcal{U}$ to determine the size B_N and the indexes I_N for selecting large-loss samples $\tilde{M} = (M_i)_{i \in I_N}$. Together with the positive learning, USN performs negative learning on the selected large-loss samples and obtains a robust imitation learning model against action noises. More details of the USN algorithm are summarized in Section 4.

As shown in Figure 2, our method USN trains a policy with an additional process of uncertainty-aware sample-selection and negative learning for loss correction. Specifically, during positive learning (marked in black color), we train an imitation learning model with any noise-tolerant loss function on the noisy demonstration batch M and estimate the uncertainty $\mathcal{U}$ for selecting large-loss samples. This sample-selection step is motivated by recent work in the area of learning with label noise (Angluin & Laird, 1988; Smyth et al., 1994; Kalai & Servedio, 2005; Natarajan et al., 2013; Manwani & Sastry, 2013). Since deep networks learn easy patterns first (Arpit et al., 2017), they would first memorize training data of clean labels and then those of noisy labels with the assumption that clean labels are of the majority in a noisy class. Therefore, large-loss samples are noisy actions with high probability

(Wei et al., 2020). Instead of requiring additional information on noise models (Han et al., 2018; Yu et al., 2019), we conduct a comprehensive study to learn the correlations between loss, uncertainty, and increasing noise rates and design a scalable sample-selection method based on the correlations. Through the study, we observe positive correlations for behavioral cloning, online imitation learning, and offline imitation learning on both *state-independent* action noise and *state-dependent* action noise. We sort the positive learning loss to obtain its index I, and use uncertainty estimation $\mathcal{U}$ to adaptively determine the large loss sample size and indexes I_N. The large-loss samples are selected from the noisy demonstration batch M according to the indexes I_N: $\tilde{M} = (M_i)_{i \in I_N}$. Next, we update the imitation learning model by employing negative learning (marked in blue) with a factor of λ_{neg} (Kim et al., 2019) on the large-loss samples $\tilde{M}$, along with positive learning on the full dataset M. Finally, we obtain a robust policy against action noise with USN.

We conclude our contributions in this paper as follows:

- First, we introduced two action noise models: *state-independent* action noise and *state-dependent* action noise, to study the negative impacts of action noises in imitation learning (Section 3.1).
- Then, we studied the correlations between loss and predictive uncertainty of different imitation learning methods under various action noises (Section 3.2). The study determines the range of action noise where the positive correlation holds. Further analysis in the study laid the basis for our following robust imitation learning method.
- Next, we proposed our method USN (Section 4) to mitigate the harmful effect of action noise in imitation learning. USN is a versatile meta-algorithm that can be applied to online and offline imitation learning. Compared to existing IL methods, our method has multiple advantages, including improved imitation performance, adaptability, and scalability. Our approach can adaptively select large-loss samples for soft negative learning across different noise rates without the need for additional datasets, models, or prior information about the noise model. USN eliminates the need for estimating noise rates and transition matrices as previous noise-robust methods require, avoiding the associated extra efforts and drawbacks.
- Finally, we empirically demonstrated that USN significantly improves the robustness of behavioral cloning (Section 5.1), online imitation learning (Section 5.2), and offline imitation learning (Section 5.3) when learning with *state-independent* and *state-dependent* action noises. USN brings a tangible improvement within the noise range that was determined in the correlation study, i.e., the noise should be above a certain threshold for online and offline imitation learning. Moreover, USN scales up to conditional imitation learning with noisy commands in urban driving (Section 6).

2. Background and Related Work

In this section, we first discuss existing offline and online imitation learning methods. Then, we introduce some related works of learning with noisy labels. We also briefly discuss action noise in deep reinforcement learning and imitation learning.

Behavioral cloning (BC) is probably the simplest offline imitation learning algorithm (Pomerleau, 1988; Bain & Sammut, 1999; Ross et al., 2011). For imitation learning in environments with discrete action space, the BC policy $\pi(a|s)$ is optimized by softmax cross-entropy loss.

2.1 Online Imitation Learning

Generative adversarial imitation learning (GAIL) (Ho & Ermon, 2016) is one of the state-of-the-art online IL methods. GAIL treats imitation learning as a distribution matching problem between the expert policy and the agent policy. Specifically, GAIL uses a discriminator D_ϕ to distinguish expert transitions from agent transitions. In contrast, the agent aims to "fool" the discriminator into treating agent transitions as expert data. Formally, the objective function of GAIL is written as

$$\min_\theta \max_\phi \mathbb{E}_{(s,a)\sim\rho_\theta}[\log D_\phi(s,a)] + \mathbb{E}_{(s,a)\sim\rho_E}[\log(1-D_\phi(s,a))], \tag{1}$$

where ρ_θ and ρ_E denote the occupancy measures of agent policies π_θ and the expert π_E, respectively.

Built on top of GAN (Goodfellow et al., 2014), GAIL and its many (robust) variants (Li et al., 2017; Peng et al., 2019; Tangkaratt et al., 2020b, 2020a; Wang et al., 2021a) have achieved great success in imitation learning in low-dimensional space even with noisy demonstrations. However, GAIL fails to scale to high-dimensional imitation learning tasks (Brown et al., 2019; Tucker et al., 2018). One straightforward solution to alleviate this issue is to initiate the actor of GAIL using behavior cloning.

Selective adversarial imitation learning (SAIL) (Wang et al., 2021a) was proposed to address the imperfect demonstration issue, in which good demonstrations can be adaptively selected for training while bad demonstrations are abandoned. Specifically, a binary weight $w \in \{0,1\}$ is assigned to each expert demonstration to indicate whether to select it for training. The weight is set to be determined by the reward function in Wasserstein GAIL (Xiao et al., 2019) (i.e., higher reward results in higher weight). The resulting algorithm - SAIL-hard is defined as follows:

$$\min_\theta \max_\phi \mathbb{E}_{(s,a)\sim\rho_E}[w(s,a)r_\phi(s,a) - Kw(s,a)] - \mathbb{E}_{(s,a)\sim\rho_\phi}[r_\phi(s,a)] + \lambda\Psi(r_\phi). \tag{2}$$

Besides hard binary weighting, they also propose a soft weighting scheme with the suggested optimal soft weight as $w^* = \frac{1}{1+e^{K-r_\phi(s,a)}}$. The corresponding algorithm - SAIL-soft, is defined as follows:

$$\min_\theta \max_\phi \mathbb{E}_{(s,a)\sim\rho_E}[w(s,a)r_\phi(s,a) + f(w(s,a))] - \mathbb{E}_{(s,a)\sim\rho_\phi}[r_\phi(s,a)] + \lambda\Psi(r_\phi), \tag{3}$$

where $f(w) = w\log(w^{-1}-1) - \log(1-w) - Kw$. We regard GAIL, RIL_CO (Tangkaratt et al., 2020a), and SAIL as baselines for comparing their robustness against action noise.

2.2 Offline Imitation Learning

Batch-constrained Q-learning (BCQ) (Fujimoto et al., 2019) is the first continuous control algorithm capable of learning from arbitrary batch data without exploration. BCQ aims to perform Q-learning while constraining the action space to eliminate actions that are unlikely to be selected by the behavioral policy π_b and are, therefore, unlikely to be contained in the batch. At its core, BCQ uses a state-conditioned generative model $G : s \to a$ to model the data distribution in the batch, $G \approx b$ akin to a behavioral cloning model. As it is easier to sample from $\pi_b(a|s)$ than model $\pi_b(a|s)$ exactly in a continuous action space, the policy is defined by sampling N actions a_i from $G(s)$ and selecting the highest valued action according to a Q-network.

Discrete BCQ (Fujimoto et al., 2019) extends BCQ to discrete action space by computing the probabilities of every action $G \approx \pi_b(a|s)$, and instead utilizing some threshold to eliminate actions. To adaptively adjust this threshold, Fujimoto et al. (2019) scale it by the maximum probability from the generative model over all actions to only allow actions whose relative probability is above some threshold. Specifically, Fujimoto et al. (2019) applies Double DQN (Van Hasselt et al., 2016) to select the max valued action with the current Q-network Q_θ, and evaluate with the target Q-network $Q_{\theta'}$. This results in an algorithm comparable to DQN (Mnih et al., 2015) where the policy is defined by a constrained argmax. The Q-network is trained by swapping the max operation with actions selected by the policy:

$$\mathcal{L}(\theta) = l_\kappa \Bigg(r + \gamma \max_{a|G(a|s)/\max_{\hat{a}} G(\hat{a}|s) > \tau} Q_{\theta'}(s', a') - Q_\theta(s, a) \Bigg), \tag{4}$$

where l_κ defines the Huber loss (Watkins, 1989):

$$l_\kappa(\delta) = \begin{cases} 0.5\delta^2 & \text{if } \delta \leq \kappa \\ \kappa(|\delta| - 0.5\kappa) & \text{otherwise.} \end{cases} \tag{5}$$

With this threshold τ, Fujimoto et al. (2019) maintain the original property of BCQ where setting $\tau = 0$ returns Q-learning and $\tau = 1$ returns an imitator of the actions contained in the batch. The generative model G, effectively a behavioral cloning network, is trained in a standard supervised learning fashion, with a cross-entropy loss. Intrinsic motivation has been used as an alternative reward function in the exploration literature (Pathak et al., 2017; Burda et al., 2019; Colas et al., 2022). In this paper, we adopt the (Discrete) BCQ method for offline IL by training an intrinsic curiosity module (ICM) (Pathak et al., 2017) for generating intrinsic rewards instead of using the true reward data in the demonstration.

Uncertainty weighted actor-critic (UWAC) (Wu et al., 2021) is a recently proposed offline reinforcement learning algorithm that can detect OOD (Out-Of-Distribution) state-action pairs and down-weight their contribution to the training objectives accordingly. UWAC is closely related to our work since it also uses uncertainty estimation to improve the model's robustness. Specifically, UWAC uses Monte Carlo (MC) dropout (Gal & Ghahramani, 2016) to estimate the uncertainty of Q model. The model uncertainty is captured by the approximate predictive variance with respect to the estimated $\hat{Q}$ for T stochastic forward passes:

$$Var[Q(s, a)] \approx \sigma^2 + \frac{1}{T} \sum_{t=1}^{T} \hat{Q}_t(s, a)^\top \hat{Q}_t(s, a) - \mathbb{E}[\hat{Q}(s, a)]^\top \mathbb{E}[\hat{Q}(s, a)]. \tag{6}$$

Then, UWAC down-weighs the Bellman loss for the Q function by inverse the uncertainty of the Q-target $Q_{\theta'}(s', a')$:

$$\mathcal{L}(Q_\theta) = \mathbb{E}_{(s'|s,a)\sim\mathcal{D}} \mathbb{E}_{a'\sim\pi(\cdot|s')} \Big[\frac{\beta}{Var[Q_{\theta'}(s', a')]} Err(s, a, s', a')^2 \Big], \tag{7}$$

where $Err(s, a, s', a') = Q_\theta(s, a) - (R(s, a)) + \gamma Q_{\theta'}(s', a')$. This directly reduces the effects that OOD backups have on the overall training process (Wu et al., 2021). We treat UWAC as a strong baseline in our paper and show that our USN can scale to MC dropout for uncertainty estimation. The original UWAC was built on top of BEAR (Kumar et al., 2019). To enable a fair comparison, we implement UWAC on top of BCQ as follows:

$$\mathcal{L}(\theta) = l_\kappa\Big(r + \gamma \max_{a|G(a|s)/\max_{\hat{a}} G(\hat{a}|s)>\tau} \frac{\beta}{Var[Q_{\theta'}(s', a')]}\big[Q_{\theta'}(s', a') - Q_\theta(s, a)\big]\Big). \tag{8}$$

2.3 Learning with Noisy Labels

Learning with noisy labels aims to learn a robust classifier f by exploiting training samples with only noisy labels $(x_i, \tilde{y}_i)_{i=1}^N$.

Generalized cross-entropy (GCE) (Zhang & Sabuncu, 2018) is a generalization of cross-entropy (CE) and Mean Absolute Error (MAE). To exploit the benefits of both the noise-robustness provided by MAE and the implicit weighting scheme of CCE, they proposed using the negative Box-Cox transformation (Box & Cox, 1964) as a loss function:

$$\mathcal{L}_q(f(x), e_j) = \frac{(1 - f_j(x)^q)}{q}, \tag{9}$$

where $q \in (0, 1]$. It is equivalent to CE for $\lim_{q\to 0} \mathcal{L}_q(f(x), e_j)$, and becomes MAE/unhinged loss when $q = 1$.

Label smoothing has been commonly used to improve the performance of deep learning models (Szegedy et al., 2016; Vaswani et al., 2017; Zoph et al., 2018; Real et al., 2019; Huang et al., 2019). It serves as a regularizer that improves the generalization and model calibration (predictive uncertainty) by using smoothed labels $\tilde{y}$ that mix the original one-hot labels y with a uniform mixture over all possible labels for $\alpha \in [0, 1]$(Li et al., 2020; Pereyra et al., 2017; Müller et al., 2019; Yuan et al., 2020; Zhang et al., 2021a):

$$\tilde{y}_i = (1 - \alpha) \cdot y_i + \frac{\alpha}{K} \cdot \mathbf{1}, \tag{10}$$

where K is the number of classes. Recently, Lukasik et al. (2020) demonstrated that label smoothing is also effective when learning with symmetric label noise.

2.4 Action Noise in Deep Reinforcement Learning

Hollenstein et al. (2022) presented a study on the effect of action noise in off-policy deep reinforcement learning. In continuous control tasks, off-policy deep reinforcement learning algorithms improve their exploration ability by using additive action noise sampled from a Gaussian Distribution and a Qrnstein-Uhlenbeck process (Fujimoto et al., 2018; Lillicrap et al., 2016). This approach is simple yet surprisingly effective. To study the impact of action noise on exploration and performance of off-policy deep reinforcement learning in continuous control domain, Hollenstein et al. (2022) performed an extensive experimental study testing 6 environments, 4 algorithms, 5 noise scales, 3 schedulers, and 2 noise types. The study found that (1) the noise type needs to be chosen to fit the environment, (2) the positive and negative correlation between the state-space coverage and performance should guide the selection of action noise, (3) reducing the impact of action noise improves the performance and increases robustness to action noise choice, and (4) action noise scale appears to be the most important factor.

Action noise appears to positively impact the exploration ability and potentially increase the performance of off-policy deep reinforcement learning in the continuous control domain (Hollenstein et al., 2022). However, action noise in the demonstrations negatively impacts imitation learning performance (Tangkaratt et al., 2020b, 2020a). Inspired by Hollenstein et al. (2022), we perform an extensive experimental study in this paper to understand the impact of action noise in imitation learning. Our study considers the correlations between uncertainty estimation, loss estimation, and the scale of action noise. Similar to Hollenstein et al. (2022), we also consider 2 types of action noise, i.e., *state-independent* action noise and *state-dependent* action noise that occur in the demonstrations. Our experimental study in the following section tests 3 diverse imitation learning algorithms, 3 uncertainty estimation methods, 5 noise scales, and 2 noise types.

3. How does Action Noise Affect Imitation Learning?

In this section, we first introduce *state-independent* action noise and *state-dependent* action noise in imitation learning. Then, we briefly summarize loss estimation and uncertainty estimation methods. Next, we comprehensively study the correlations between loss and uncertainty under multiple action noise.

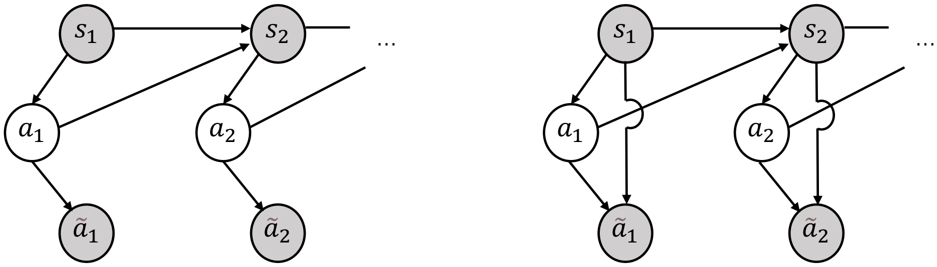
3.1 Action Noise in Imitation Learning

In traditional imitation learning, we consider the expert demonstrations $\mathcal{D} = \{s_i, a_i, s_{i+1}\}_{i=1}^{N}$ are collected from some expert demonstrator with policy π^*. In practice, we may recruit annotators to give action labels for the recorded sequences of expert behaviors. In contrast, in our setting of **imitation learning with action noise**, we assume the demonstrations are imperfect and contain two types of action noise, i.e., *state-independent* action noise and *state-dependent* action noise. In particular, we focus on discrete action space in this work and remain the extension to continuous action space for future investigation.

***State-independent* action noise (SIN):** The *state-independent* action noise occurs when an amateur annotator randomly picks an action for confusing states since they lack professional domain knowledge. Consider imitation learning in a discrete action space $\mathcal{A} = \{1, 2, \cdots, |\mathcal{A}|\}$, where $|\mathcal{A}|$ is the number of action category. We assume a noisy action $\tilde{a}$ is randomly flipped from an expert action $a \sim \pi^*(a|s)$ according to the conditional probability defined in a noise transition matrix $\mathbf{M} \in [0, 1]^{|\mathcal{A}| \times |\mathcal{A}|}$, where $\mathbf{M}_{p,q} = \Pr(\tilde{a} = q|a = p)$ and $p, q \in \mathcal{A}$. The generation process of *state-independent* action noise is borrowed from the two structures of class-conditional label noise (ϵ is the noise rate) (Blum & Mitchell, 1998; Van Rooyen et al., 2015; Patrini et al., 2017; Zhang & Sabuncu, 2018): (1) Symmetry flipping, where a noisy label is randomly flipped from other classes using $\mathbf{M}_{\text{sym}}$; (2) Pair flipping, where a noisy label is flipped from its very similar classes using $\mathbf{M}_{\text{pair}}$ (Natarajan et al., 2013; Han et al., 2018).

$$\mathbf{M}_{\text{sym}} = \begin{bmatrix} \epsilon & \frac{1-\epsilon}{|\mathcal{A}|-1} & \frac{1-\epsilon}{|\mathcal{A}|-1} & \cdots & \frac{1-\epsilon}{|\mathcal{A}|-1} \\ \frac{1-\epsilon}{|\mathcal{A}|-1} & \epsilon & \frac{1-\epsilon}{|\mathcal{A}|-1} & \cdots & \frac{1-\epsilon}{|\mathcal{A}|-1} \\ \frac{1-\epsilon}{|\mathcal{A}|-1} & \frac{1-\epsilon}{|\mathcal{A}|-1} & \epsilon & \ddots & \vdots \\ \vdots & \vdots & \ddots & \ddots & \frac{1-\epsilon}{|\mathcal{A}|-1} \\ \frac{1-\epsilon}{|\mathcal{A}|-1} & \frac{1-\epsilon}{|\mathcal{A}|-1} & \cdots & \frac{1-\epsilon}{|\mathcal{A}|-1} & \epsilon \end{bmatrix}, \mathbf{M}_{\text{pair}} = \begin{bmatrix} \epsilon & 1-\epsilon & 0 & \cdots & 0 \\ 0 & \epsilon & 1-\epsilon & \ddots & \vdots \\ \vdots & 0 & \epsilon & \ddots & 0 \\ 0 & \vdots & \ddots & \ddots & 1-\epsilon \\ 1-\epsilon & 0 & \cdots & 0 & \epsilon \end{bmatrix}.$$

Formally, we define the noisy demonstrations $\tilde{\mathcal{D}} = \{s_i, \tilde{a}_i, s_{i+1}\}_{i=1}^n$ with *state-independent* action noise are drawn from $p(\tilde{\mathcal{D}}|\Pr, \pi^*) = p(s_0)\Pi_{t=0}^{N-1}\Pr(\tilde{a}_t|a_t)p(s_{t+1}|s_t, a_t)$, where Pr is determined by the expertise of the amateur annotator. Using the two structures $\mathbf{M}_{\text{sym}}$ and $\mathbf{M}_{\text{pair}}$, we have two instances of *state-independent* action noise, i.e., symmetric action noise and pairflip action noise, respectively. We present a graphical model for *state-independent* action noise in Figure 3(a) and the whole procedure of *state-independent* action noise generation in the Algorithm 1.



(a) *state-independent* action noise

(b) *state-dependent* action noise

Figure 3: The graphical model of (a) *state-independent* action noise and (b) *state-dependent* action noise.

Algorithm 1 *State-Independent* Action Noise Generation

```
Input: Expert demonstration $\mathcal{D} = \{(s_i, a_i, s_{i+1})\}_{i=1}^N$; Noise rate: $\epsilon$.
Prepare the noise transition matrix $\mathbf{M} = \mathbf{M}_{\text{sym}}$ or $\mathbf{M} = \mathbf{M}_{\text{pair}}$ w.r.t. noise rate $\epsilon$.
Iterations: // Flip action according to the noise transition matrix.
for $n = 1$ to $N$ do
    Draw a multinomial distribution $P_m$ with a probability of $\mathbf{M}_{a_n,:}$.
    Sample the noisy action $\tilde{a}_n$ from the multinomial distribution $P_m$.
end for
Output: Noisy Demonstrations $\tilde{\mathcal{D}} = \{(s_i, \tilde{a}_i, s_{i+1})\}_{i=1}^N$.
```

***State-dependent* action noise (SDN):** We assume the noisy action is dependent on not only the expert action but also the state feature. The resulting noisy demonstrations $\tilde{\mathcal{D}} = \{s_i, \tilde{a}_i, s_{i+1}\}_{i=1}^n$ with *state-dependent* action noise are drawn from

$$p(\tilde{\mathcal{D}}|\tilde{\pi}, \pi^*) = p(s_0)\Pi_{t=0}^{N-1}\tilde{\pi}(\tilde{a}_t|s_t)p(s_{t+1}|s_t, a_t), \tag{11}$$

where $\tilde{\pi}$ is the noisy policy from an amateur annotator. Figure 3(b) shows a graphical model for *state-dependent* action noise. We present the whole procedure of *state-dependent* action noise generation in the Algorithm 2.

Algorithm 2 *State-Dependent* Action Noise Generation

```
1:  **Input:** Expert demonstration $\mathcal{D} = \{(s_i, a_i, s_{i+1})\}_{i=1}^{N}$; Noise rate: $\epsilon$; Size of feature:
    $1 \times |S|$; Number of action category: $|\mathcal{A}|$.
2:  Sample instance flip rates $q_n$ from the truncated normal distribution $\mathcal{N}(\epsilon, 0.1^2, [0,1])$;
3:  Sample $W \in \mathcal{R}^{|S| \times |\mathcal{A}|}$ from the standard normal distribution $\mathcal{N}(0, 1^2)$;
4:  **for** $n = 1$ to $N$ **do**
5:      $p = s_i \cdot W$          // Generate state dependent flip rates. The size of $p$ is $1 \times |\mathcal{A}|$.
6:      $p_{a_i} = -\infty$          // Only consider entries different from the expert actions
7:      $p = q_i \cdot \mathrm{softmax}(p)$          // Let $q_n$ be the probability of getting a wrong action
8:      $p_{a_i} = 1 - q_i$          // Keep expert actions w.p. $1 - q_i$
9:      Randomly choose an action from the action space as noisy action $\tilde{a}_i$ according to $p$;
10: **end for**
11: **Output:** Noisy Demonstrations $\tilde{\mathcal{D}} = \{(s_i, \tilde{a}_i, s_{i+1})\}_{i=1}^{N}$.
```

We simulate the generation of *state-dependent* action noise by following the *instance-dependent* label noise setting (Xia et al., 2020). Note that it is more realistic that different states have different flip rates. It is more challenging to model the action noise and train robust policies without constraining different states to have the same flip rate. In Step 1, to control the global flip rate as ϵ but without constraining all the states to have the same flip rate, we sample their flip rates from a truncated normal distribution $\mathcal{N}(\epsilon, 0.1^2, [0,1])$. Specifically, this distribution limits the flip rates of states in the range [0, 1]. Their mean and standard deviation are equal to the mean ϵ and the standard deviation 0.1 of the selected truncated normal distribution, respectively.

In Step 2, we sample parameters $w_1, w_2, \cdots, w_c$ from the standard normal distribution for generating *state-dependent* action noise. The dimensionality of each parameter is $d \times c$, where d denotes the dimensionality of the state. Learning these parameters is critical to model *state-dependent* action noise. However, it is hard to identify these parameters without any assumptions. Note that a state with expert action a will be flipped only according to the $a-$th row of the transition matrix. Thus, in Steps 5 to 8, we only use the a_i-th row of the state-dependent transition matrix for the state s_i. Specifically, Steps 6 and 8 ensure the diagonal entry of the a_i-th row is $1 - q_i$. Step 7 ensures that the sum of the off-diagonal entries is q_i.

3.2 Correlation Between Loss and Predictive Uncertainty Under Action Noise

In this section, we comprehensively study the correlation between the loss, uncertainty estimation of behavioral cloning (BC), online imitation learning (GAIL), and offline imitation learning (BCQ) models under diverse action noise in the demonstrations. To study the correlations, we pre-train imitation learning models with the expert demonstrations, and then estimate the corresponding loss and uncertainty estimations (ECE, Entropy, and

MC_Dropout) on demonstrations with both *state-independent* action noises (symmetric and pairflip) and *state-dependent* action noise under different noise rates.

Loss estimation $\mathcal{L}_{\text{pos}}$. To study the correlations, we first need to define the target model $\mathcal{T}$ and its corresponding loss estimation for each imitation learning method. The target model $\mathcal{T}$ is the model we perform loss and uncertainty estimations under different action noises. For behavioral cloning, the target model $\mathcal{T}$ is the behavioral cloning model itself, and we use cross-entropy as the loss estimation since we focus on discrete action space in this work. For GAIL, we define the target model $\mathcal{T}$ as the discriminator of GAIL and use the adversarial loss for the demonstrations as the loss estimation. For BCQ, we chose its generative model G as the target model, and we also used cross-entropy for loss estimation.

Uncertainty estimation $\mathcal{U}$. Uncertainty estimation has been widely used in machine learning applications as a complement that reflects the degree of trust in the model predictions (Kotelevskii et al., 2022). Usually, the total uncertainty of a prediction comes from two types of uncertainty: *aleatoric* and *epistemic* (Der Kiureghian & Ditlevsen, 2009; Kendall & Gal, 2017). The *aleatoric* uncertainty, known as data uncertainty, reflects the noise and ambiguity in the data. The *epistemic* uncertainty, known as model uncertainty, is related to the lack of knowledge about model parameters. Quantifying both types of uncertainty is crucial for safe decisions in practical applications (Filos et al., 2020).

One simple *aleatoric* uncertainty is called MaxProb, which uses maximum softmax probabilities of the deep neural network as the uncertainty measure (Nguyen et al., 2015). Entropy is another widely used uncertainty estimation method. MC_dropout is based on Bayesian techniques, capturing both types of uncertainty, and is known to be a more reliable uncertainty estimation method (Gal & Ghahramani, 2016). Recently, a promising direction of uncertainty estimation methods based on a single deterministic neural network has been developed (Lee et al., 2018; Liu et al., 2020). More recently, (Kotelevskii et al., 2022) proposes a nonparametric uncertainty quantification (NUQ) method, which enables uncertainty to be disentangled into *aleatoric* and *epistemic* uncertainty and is scalable to a large dataset.

Expected Calibration Error (ECE) (Naeini et al., 2015; Guo et al., 2017) is widely used to measure the predictive uncertainty (model calibration) of a deep network. To approximate the calibration error in expectation, ECE discretizes the probability interval into a fixed number of bins and assigns each predicted probability to the bin that encompasses it. The calibration error is the difference between the fraction of predictions in the bin that are correct (accuracy) and the mean of the probabilities in the bin (confidence). Intuitively, the accuracy estimates $\mathbb{P}(Y = y|\hat{p} = p)$, and the average confidence is a setting of p. ECE computes a weighted average of this error across bins:

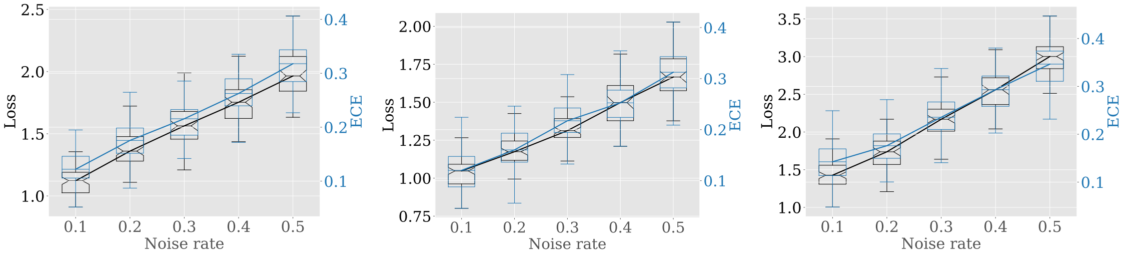
$$\text{ECE} = \sum_{b=1}^{B} \frac{n_b}{N} |\text{acc}(b) - \text{conf}(b)|, \tag{12}$$

where n_b is the number of predictions in bin b, N is the total number of data points, and $\text{acc}(b)$ and $\text{conf}(b)$ are the accuracy and confidence of bin b, respectively. As framed in Naeini et al. (2015), ECE leaves ambiguity in both its binning implementation and how to compute calibration for multiple classes. In Guo et al. (2017), they bin the probability interval [0; 1] into equally spaced subintervals, and they take the maximum probability output for each datapoint (i.e., the predicted class's probability). We use this for our ECE implementation.

Instead of using the uncertainty for OOD detection (Malinin & Gales, 2018), or down-weighting the Bellman loss for Q function (Wu et al., 2021) in RL, we leverage uncertainty for robust imitation learning in a novel direction. Namely, we first study the correlation between loss and uncertainty estimations (ECE, Entropy, and MC_Dropout) and then propose a robust imitation learning paradigm called uncertainty-aware sample selection with negative learning (USN) based on the correlations. Our method, USN, is scalable to behavioral cloning, online imitation learning, and offline imitation learning. We provide details of the correlation studies for behavioral cloning, GAIL, and BCQ in the following parts of this section.

3.2.1 Behavioral Cloning

We first perform a correlation study for the behavioral cloning (BC) model. We use the pre-trained Proximal Policy Optimization (PPO) (Schulman et al., 2017) agent from stable-baselines3[1] to generate ten expert demonstrations as the training data to train BC model on the LunarLander-v2 task. We build the BC model using a 2-layer MLP architecture with 32 neurons on each layer. We use the expert demonstrations to generate noisy demonstrations according to the noise models of *state-independent* action noise and *state-dependent* action noise defined in Section 3.1. For each action noise type, we set the noise rate ϵ as 0.1, 0.2, 0.3, 0.4, and 0.5. We train the BC model on the expert demonstrations and evaluate it on the noisy demonstrations with increasing noise rates to calculate the loss and uncertainty estimations.

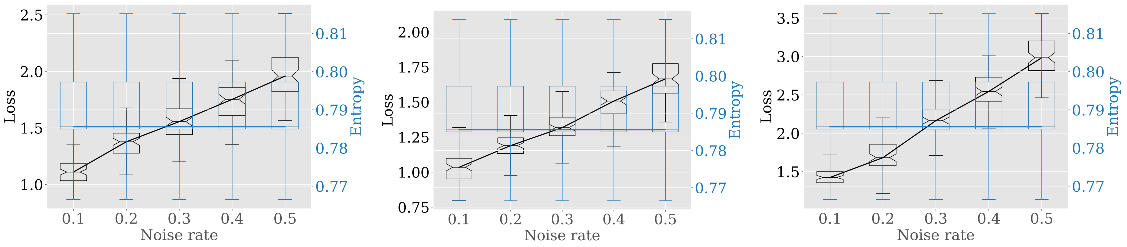


(a) Symmetric (b) Pairflip (c) *State-dependent*

Figure 4: The correlations between loss and uncertainty estimation (ECE) of behavioral cloning model as noise rates increase in the LunarLander-v2 task.

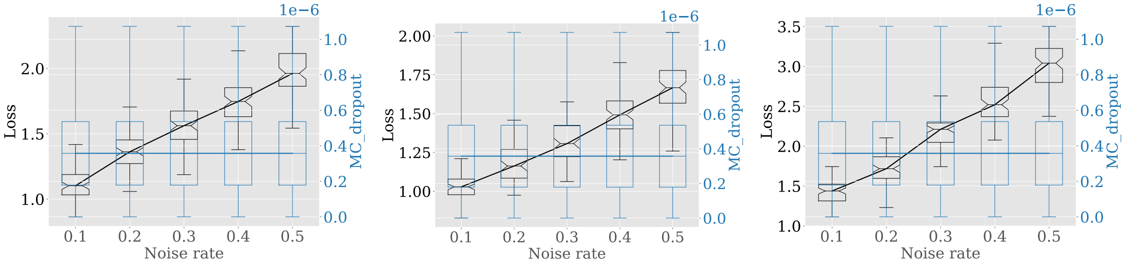
Figure 4, Figure 5 and Figure 6 show the correlations between loss and uncertainty estimations (ECE, Entropy, and MC_Dropout) as the noise rate increases, separately. In each figure, we report the boxplot of loss estimations (in black color) and uncertainty estimations (in light blue color) for *state-independent* action noises ((a) symmetric, (b) pairflip) and *state-dependent* action noise (c). As shown in Figure 4, ECE shows a clearly positive correlation with loss as the noise rate increases. This positive correlation holds for different action noise types. On the contrary, Entropy and MC_Dropout are not sensitive to the changes in noise rates, and thus have no correlations with the loss estimations. We

[1] https://github.com/DLR-RM/rl-baselines3-zoo



(a) Symmetric (b) Pairflip (c) *State-dependent*

Figure 5: The correlations between loss and uncertainty estimation (Entropy) of behavioral cloning model as noise rates increase in the LunarLander-v2 task.



(a) Symmetric (b) Pairflip (c) *State-dependent*

Figure 6: The correlations between loss and uncertainty estimation (MC_Dropout) of behavioral cloning model as noise rates increase in the LunarLander-v2 task.

obtain a rough observation from this study for behavioral cloning, that is, ECE and loss are sensitive to the changes in noise rates, and they are positively correlated with each other.

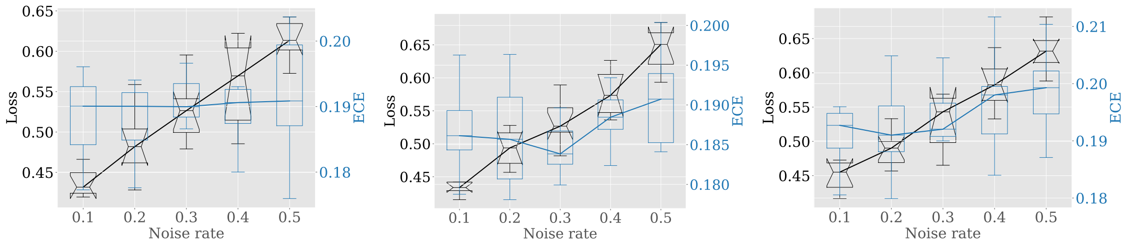
To provide quantitative support to this observation, we calculate the Spearman correlation coefficients $\mathcal{C}$ (Hollenstein et al., 2022) in Table 1. Typically, $\mathcal{C} > 0.5$ means high positive correlation, and $\mathcal{C} < -0.5$ means high negative correlation. From Table 1, we know that only ECE shows high positive correlations with both loss estimation and increasing noise rates. These results indicate that the loss and ECE are good identifiers of both *state-independent* action noise and *state-dependent* action noise for the behavioral cloning model.

	Symmetric			Pairflip			*State-dependent*		
$\mathcal{U}$	$\mathcal{C}(\mathcal{U}, \mathcal{L}_{\text{pos}})$	$\mathcal{C}(\mathcal{U}, \epsilon)$	$\mathcal{C}(\mathcal{L}_{\text{pos}}, \epsilon)$	$\mathcal{C}(\mathcal{U}, \mathcal{L}_{\text{pos}})$	$\mathcal{C}(\mathcal{U}, \epsilon)$	$\mathcal{C}(\mathcal{L}_{\text{pos}}, \epsilon)$	$\mathcal{C}(\mathcal{U}, \mathcal{L}_{\text{pos}})$	$\mathcal{C}(\mathcal{U}, \epsilon)$	$\mathcal{C}(\mathcal{L}_{\text{pos}}, \epsilon)$
ECE	**0.99**	**0.99**	**0.99**	**0.99**	**0.99**	**0.99**	**0.99**	**0.99**	**0.99**
Entropy	0.0	0.0	0.99	0.0	0.0	0.99	0.0	0.0	0.99
MC_Dropout	0.0	0.0	0.99	0.0	0.0	0.99	0.0	0.0	0.99

Table 1: Spearman correlation coefficients on noise rate ϵ for: uncertainty estimation $\mathcal{U}$ and loss estimation $\mathcal{L}_{\text{pos}}$. Results are shown on the LunarLander-v2 environment for the BC model.

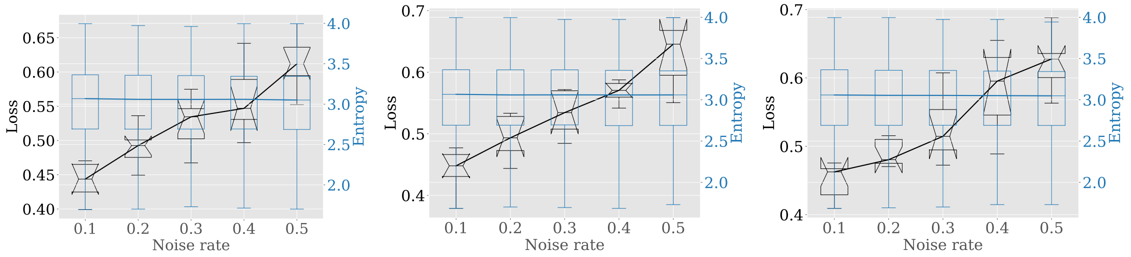
3.2.2 Online Imitation Learning - GAIL.

Then, we perform a correlation study for the GAIL discriminator. We use the open-source implementation[2] for Atari games. Figure 7, Figure 8 and Figure 9 shows the correlations between loss and uncertainty estimations (ECE, Entropy, and MC_Dropout) as the noise rate increases on Q*bert game, separately. From these Figure 8 and Figure 9, we observe that Entropy and MC_Dropout are not sensitive to the increases of noise rate. Figure 7 shows that ECE positively correlates with the loss when the noise rate $\epsilon >= 0.3$; otherwise, ECE is not sensitive to the changes of noise rates.



(a) Symmetric (b) Pairflip (c) *State-dependent*

Figure 7: The correlations between loss and the uncertainty estimation (ECE) of GAIL discriminator as noise rates increase in the Q*bert game.



(a) Symmetric (b) Pairflip (c) *State-dependent*

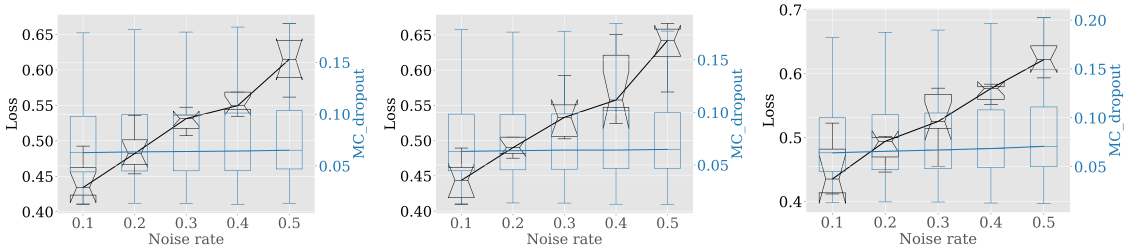
Figure 8: The correlations between loss and the uncertainty estimation (Entropy) of GAIL discriminator as noise rates increase in the Q*bert game.

To confirm this observation, we also calculate the Spearman correlation coefficients in Table 2. ECE shows high positive correlations with loss and the increasing noise rates ($\epsilon >= 0.3$) under all action noise types. Thus, the loss and ECE of the GAIL discriminator are good identifiers of both *state-independent* action noise and *state-dependent* action noise when noise rate $\epsilon >= 0.3$.

3.2.3 Offline Imitation Learning - BCQ.

We finally use the open-source platform[3] to perform the correlation study for BCQ's generative model G. We train the BCQ (Algorithm 3) with an ICM module (Pathak et al., 2017) that produces intrinsic rewards for updating policy. BCQ starts by sampling a mini-batch of

[2] https://github.com/yunke-wang/gail_atari
[3] https://github.com/thu-ml/tianshou



(a) Symmetric (b) Pairflip (c) *State-dependent*

Figure 9: The correlations between loss and the uncertainty estimation (MC_Dropout) of GAIL discriminator as noise rates increase in the Q*bert game.

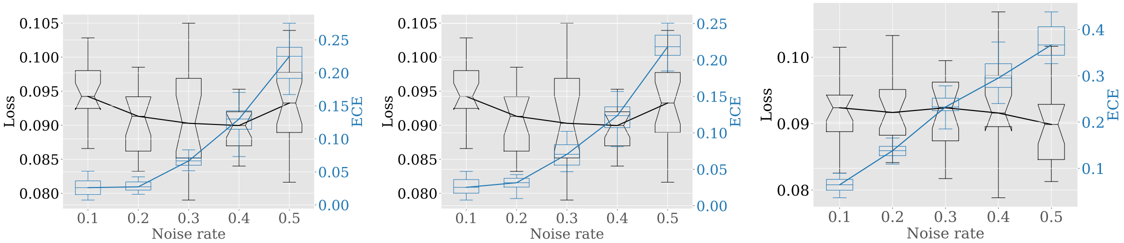
$\epsilon >= 0.3$ $\mathcal{U}$	Symmetric $\mathcal{C}(\mathcal{U}, \mathcal{L}_{\text{pos}})$	 $\mathcal{C}(\mathcal{U}, \epsilon)$	 $\mathcal{C}(\mathcal{L}_{\text{pos}}, \epsilon)$	Pairflip $\mathcal{C}(\mathcal{U}, \mathcal{L}_{\text{pos}})$	 $\mathcal{C}(\mathcal{U}, \epsilon)$	 $\mathcal{C}(\mathcal{L}_{\text{pos}}, \epsilon)$	*State-dependent* $\mathcal{C}(\mathcal{U}, \mathcal{L}_{\text{pos}})$	 $\mathcal{C}(\mathcal{U}, \epsilon)$	 $\mathcal{C}(\mathcal{L}_{\text{pos}}, \epsilon)$
ECE	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**
Entropy	-0.5	-0.5	1.0	-0.5	-0.5	1.0	-1.0	-1.0	1.0
MC_Dropout	**1.0**	**1.0**	**1.0**	0.5	0.5	1.0	**1.0**	**1.0**	**1.0**
$\epsilon < 0.3$ $\mathcal{U}$	Symmetric $\mathcal{C}(\mathcal{U}, \mathcal{L}_{\text{pos}})$	 $\mathcal{C}(\mathcal{U}, \epsilon)$	 $\mathcal{C}(\mathcal{L}_{\text{pos}}, \epsilon)$	Pairflip $\mathcal{C}(\mathcal{U}, \mathcal{L}_{\text{pos}})$	 $\mathcal{C}(\mathcal{U}, \epsilon)$	 $\mathcal{C}(\mathcal{L}_{\text{pos}}, \epsilon)$	*State-dependent* $\mathcal{C}(\mathcal{U}, \mathcal{L}_{\text{pos}})$	 $\mathcal{C}(\mathcal{U}, \epsilon)$	 $\mathcal{C}(\mathcal{L}_{\text{pos}}, \epsilon)$
ECE	-0.99	-0.99	0.99	-0.99	-0.99	0.99	-0.99	-0.99	0.99
Entropy	-0.99	-0.99	0.99	-0.99	-0.99	0.99	-0.99	-0.99	0.99
MC_Dropout	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99

Table 2: Spearman correlation coefficients on noise rate ϵ for: uncertainty estimation $\mathcal{U}$ and loss estimation $\mathcal{L}_{\text{pos}}$. Results are shown on the Q*bert game for the GAIL discriminator.

transitions from the input demonstration data $\mathcal{D}$. Then, BCQ uses a generative model G to eliminate actions (step 6) for updating the Q network (step 7). The generative model G is trained using a cross-entropy loss in step 8.

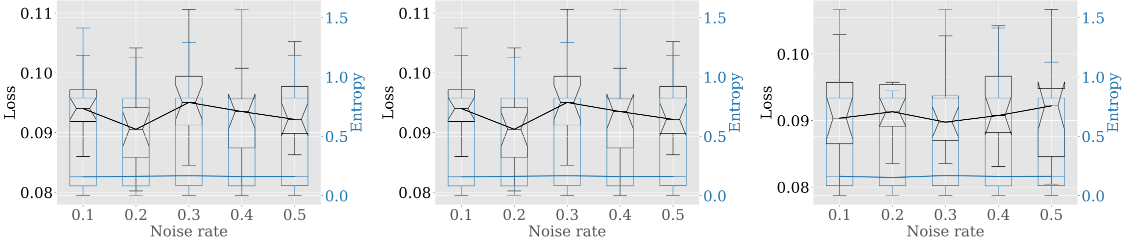
Algorithm 3 BCQ with ICM for offline imitation learning.

```
1: **Input:** Demonstration $\mathcal{D} = \{(s_i, a_i, s_{i+1})\}_{i=1}^{N}$, number of iterations $T$, target_update_rate, mini-batch size $B$, threshold $\tau$.
2: Initialize Q-network $Q_\theta$, generative model $G$ with parameter $\omega$ and target network $Q_{\theta'}$ with $\theta' \leftarrow \theta$.
3: **for** $t = 1$ to $T$ **do**
4:     Sample mini-batch $M$ of $B$ transitions $(s, a, s')$ from $\mathcal{D}$.
5:     Train an ICM module to generate intrinsic reward: $r = \text{ICM}(s, a, s')$.
6:     $a' \leftarrow \arg\max_{a' | G(a'|s') / \max_{\hat{a}} G(\hat{a}|s') > \tau} Q_\theta(s', a')$.
7:     $\theta \leftarrow \arg\min_\theta \sum_{(s,a,s') \in M} k_\kappa(r + \gamma Q_{\theta'}(s', a') - Q_\theta(s, a))$
8:     $\omega \leftarrow \arg\min_\omega - \sum_{(s,a) \in M} \log G(a|s)$.    //Update the generative model $G$ using cross-entropy loss.
9:     If $t$ mod target_update_rate $= 0 : \theta' \leftarrow \theta$.
10: **end for**
```



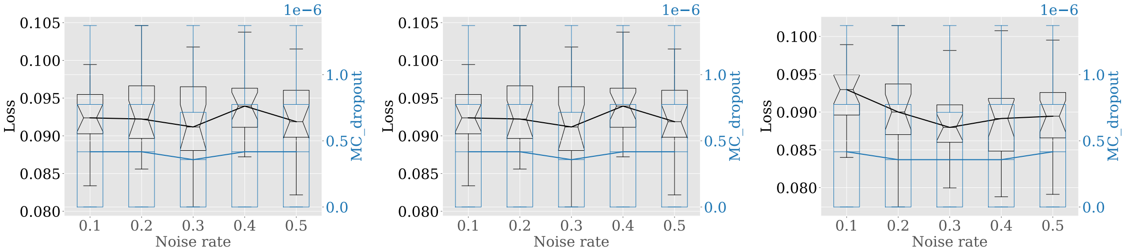
(a) Symmetric (b) Pairflip (c) *State-dependent*

Figure 10: The correlations between loss and the uncertainty (ECE) of BCQ generative model - G as noise rates increase in the AirRaid game.



(a) Symmetric (b) Pairflip (c) *State-dependent*

Figure 11: The correlations between loss and the uncertainty (Entropy) of BCQ generative model - G as noise rates increase in the AirRaid game.



(a) Symmetric (b) Pairflip (c) *State-dependent*

Figure 12: The correlations between loss and the uncertainty (MC_Dropout) of BCQ generative model - G as noise rates increase in the AirRaid game.

Figure 10, Figure 11 and Figure 12 shows the correlations between loss and uncertainty estimations (ECE, Entropy and MC_Dropout) as the noise rate increases, separately. From Figure 11 and Figure 12, we know that Entropy and MC_Dropout again are not sensitive to the increases of noise rates. From Figure 10, we observe that ECE tends to have a positive correlation to the loss estimation when the noise rate $\epsilon >= 0.4$ under *state-independent*

$\epsilon >= 0.4$	Symmetric			Pairflip			*State-dependent*		
$\mathcal{U}$	$\mathcal{C}(\mathcal{U}, \mathcal{L}_{\text{pos}})$	$\mathcal{C}(\mathcal{U}, \epsilon)$	$\mathcal{C}(\mathcal{L}_{\text{pos}}, \epsilon)$	$\mathcal{C}(\mathcal{U}, \mathcal{L}_{\text{pos}})$	$\mathcal{C}(\mathcal{U}, \epsilon)$	$\mathcal{C}(\mathcal{L}_{\text{pos}}, \epsilon)$	$\mathcal{C}(\mathcal{U}, \mathcal{L}_{\text{pos}})$	$\mathcal{C}(\mathcal{U}, \epsilon)$	$\mathcal{C}(\mathcal{L}_{\text{pos}}, \epsilon)$
ECE	**0.99**	**0.99**	**0.99**	**0.99**	**0.99**	**0.99**	-0.99	0.99	-0.99
Entropy	-0.99	0.99	-0.99	-0.99	0.99	-0.99	0.99	0.99	0.99
MC_Dropout	0.0	0.0	-0.99	0.0	0.0	0.99	0.99	0.99	0.99
$\epsilon < 0.4$	Symmetric			Pairflip			*State-dependent*		
$\mathcal{U}$	$\mathcal{C}(\mathcal{U}, \mathcal{L}_{\text{pos}})$	$\mathcal{C}(\mathcal{U}, \epsilon)$	$\mathcal{C}(\mathcal{L}_{\text{pos}}, \epsilon)$	$\mathcal{C}(\mathcal{U}, \mathcal{L}_{\text{pos}})$	$\mathcal{C}(\mathcal{U}, \epsilon)$	$\mathcal{C}(\mathcal{L}_{\text{pos}}, \epsilon)$	$\mathcal{C}(\mathcal{U}, \mathcal{L}_{\text{pos}})$	$\mathcal{C}(\mathcal{U}, \epsilon)$	$\mathcal{C}(\mathcal{L}_{\text{pos}}, \epsilon)$
ECE	-1.0	1.0	-1.0	-1.0	1.0	-1.0	-0.5	1.0	-0.5
Entropy	0.5	1.0	0.5	0.5	1.0	0.5	-1.0	0.5	-0.5
MC_Dropout	0.86	-0.86	-1.0	0.86	-0.86	-1.0	0.86	-0.86	-1.0

Table 3: Spearman correlation coefficients on noise rate ϵ for: uncertainty estimation $\mathcal{U}$ and loss estimation $\mathcal{L}_{\text{pos}}$. Results are shown on the AirRaid environment for the BCQ's generative model.

action noise. For the other noise cases, ECE negatively correlates to the loss estimation of BCQ's generative model G. Table 3 shows the Spearman correlation coefficients on noise rate ϵ for uncertain estimation $\mathcal{U}$ and loss estimation $\mathcal{L}_{\text{pos}}$ of BCQ's generative model. The correlation coefficient values $\mathcal{C}$ of ECE are all close to 1 for the *state-independent* action noises (symmetric and pariflip), which confirms our observation.

Analysis. From the above qualitative and quantitative study on *state-independent* and *state-dependent* action noises, we can conclude that both ECE and the loss estimation have positive correlations, and they are sensitive for detecting noisy actions for the BC model, GAIL's discriminator ($\epsilon >= 0.3$) and BCQ's generative model ($\epsilon >= 0.4$). Table 4 summarizes the application scope of ECE and loss estimation for detecting action noises for BC, GAIL, and BCQ.

Target model $\mathcal{T}$	BC		GAIL's Discriminator D_ϕ		BCQ's Generative Model G	
Action Noise	SIN	SDN	SIN ($\epsilon >= 0.3$)	SDN ($\epsilon >= 0.3$)	SIN ($\epsilon >= 0.4$)	SDN ($\epsilon >= 0.4$)
$\mathcal{U}$(ECE)	✓	✓	✓	✓	✓	✓
$\mathcal{L}_{\text{pos}}$	✓	✓	✓	✓	✓	-

Table 4: The application scope of ECE and loss estimation for detecting action noises for BC, GAIL, and BCQ. SIN denotes *state-independent* action noise, and SDN denotes *state-dependent* action noise. ✓ marks the cases where the criteria (ECE or loss) is sensitive to and shows a high positive correlation to the increase of noise rate and thus is suitable to detect the noisy actions.

Within the application scope of each algorithm, the "large-loss" and the "large-ECE" samples have a high probability of containing noisy actions. Therefore, we can leverage both the loss estimation and ECE as a criterion to detect noisy actions with high probability, and leverage the selected data to improve the robustness of BC, GAIL, and BCQ against diverse action noise in the demonstrations. To this end, we propose a general paradigm called Uncertainty-aware Sample-selection with Negative learning (USN) based on the positive correlations for robust training IL models against action noise. USN uses the ECE uncertainty estimation to determine the number and indexes for “large-loss” sample selection. We will provide details of our method in the following section.

4. Method

Our main goal is to develop a general robust IL paradigm against diverse types of action noise. To achieve this goal, we design USN to be a composite of two main steps: (1) uncertainty-aware sample-selection and (2) negative learning for loss correction. Algorithm 4 summarizes the whole procedure of USN.

Algorithm 4 Uncertainty-aware Sample-selection with Soft Negative learning (USN)

```
1: Input: A mini-batch $M$ of $B$ transitions $(s, a)$ from $\mathcal{D}$, target model $\mathcal{T}$.
2: Initialize the weight of negative learning loss as $\lambda_{neg} = 1.0$.
   Uncertainty-aware sample-selection (steps 3-6):
3: Estimate predictive uncertainty $\mathcal{U}$ and loss $\mathcal{L}_{\text{pos}}$ for the target model $\mathcal{T}$.
4: Use the uncertainty $\mathcal{U}$ to determine the number of large-loss samples for selection:
   $B_N = B \times (1 - \mathcal{U})$, where $\mathcal{U} \in (0, 1)$.
5: Sort the batch loss to obtain the indexes $I = \text{argsort}(\mathcal{L}_{\text{pos}})$.
6: Sample a large-loss batch $\tilde{M} = (M_i)_{i \in I_N}$, where $I_N = (I_i)_{i \in [B-B_N, B)}$.
   Negative learning for loss correction (steps 7-8):
7: Generate complementary actions for $\tilde{M}$: $\bar{a}$ = Randomly select from$\{1, .., |\mathcal{A}|\} \backslash \{a\}$,
   resulting in a complementary batch $\bar{M}$ for negative learning.
8: $\mathcal{L}_{\text{USN}} = \sum_{(s,a) \in M} \mathcal{L}_{\text{pos}}(s, a) + \lambda_{neg} \sum_{(s,a) \in \bar{M}} \mathcal{L}_{\text{neg}}(s, a)$.
9: Output: $\mathcal{L}_{\text{USN}}$.
```

Uncertainty-aware sample-selection aims to select samples that contain noisy actions with high probability. Given a mini-batch of demonstration data with a size of B, we first employ an imitation learning model for positive learning on the full-batch data. As mentioned in Section 3.2, the target model $\mathcal{T}$ is a component of the imitation learning model. This paper defines the target model as the behavioral model itself, the GAIL’s discriminator D_ϕ, and the BCQ’s generative model G. Then in step 3, we estimate uncertainty $\mathcal{U}$ and loss $\mathcal{L}_{\text{pos}}$ for the target model $\mathcal{T}$. According to the qualitative and quantitative analysis in Section 3, we know that the loss estimation $\mathcal{L}_{\text{pos}}$ is a more reliable criterion for indicating noisy actions. The uncertainty estimation $\mathcal{U}$ is also a good indicator, since it usually shows high positive correlations to $\mathcal{L}_{\text{pos}}$ as the noise rate increases. Thus, we propose to use these two criteria jointly to select reliable large-loss samples for robust training imitation learning models. Namely, we use $\mathcal{U}$ to determine the number of large-loss samples for selection (step 4): $B_N = B \times (1 - \mathcal{U})$, where $\mathcal{U} \in (0, 1)$. Since we are selecting large-loss samples, we need to sort the batch data to obtain its indexes $I = \text{argsort}(\mathcal{L}_{\text{pos}})$ (step 5). Then, the indexes of the large-loss samples becomes $I_N = (I_i)_{i \in [B-B_N, B)}$. In step 6, we use the indexes I_N to sample a large-loss batch $\tilde{M} = (M_i)_{i \in I_N}$.

Previous sample selection methods (Hafner et al., 2018; Han et al., 2020; Xia et al., 2022) usually assume that the noise rate is known and design the sample selection threshold using the noise rate. In contrast, we select large-loss samples jointly using the uncertainty estimation $\mathcal{U}$ and the loss estimation $\mathcal{L}_{\text{pos}}$. Since the uncertainty estimation dynamically changes during training, the large-loss batch $\tilde{M}$ is automatically updated and is adaptive to different noise rates and model capabilities. In this way, our method can maximally reduce the harmful effects of action noise using the adaptive threshold without requiring any prior knowledge about the noise model.

Negative learning for loss correction. Positive learning with full-batch data with noisy actions will result in bias in the loss training and misguides the policy to choose the wrong actions. We propose to leverage the selected large-loss sample for negative learning to correct the loss bias. Intuitively, the selected large-loss batch $\tilde{M}$ contains noisy actions with high probability. Its complementary set has more chances to contain true actions. Therefore, we generate complementary actions for $\tilde{M}$ by randomly selecting $\bar{a}$ from $\{1,..,|\mathcal{A}|\}\backslash\{a\}$, resulting a complementary batch $\bar{M}$ for negative learning (step 7). Negative learning on the complementary batch of the selected large-loss samples will correct the loss bias from action noise, improving imitation learning performance. We implement negative learning with label smoothing to further boost the performance, resulting in *Soft Negative learning.* Specifically, we employ the following negative log-likelihood (NLL) loss for negative learning on the "large-loss" samples with label smoothing: $\mathcal{L}_{neg} = \text{NLL}\Big(1-\mathcal{T}(a|s), (1-\alpha)\cdot\bar{a}+\frac{\alpha}{|\mathcal{A}|}\cdot\mathbf{1}\Big)$, where $\bar{a}$ is the complementary action of the "large-loss" samples, and α is the smooth rate.

5. Experiments on Synthetic Benchmarks

In this section, we evaluate the effectiveness of USN on synthetic benchmarks. We want to answer the following questions: *(1) does USN consistently improve the robustness of behavioral cloning, online imitation learning (GAIL), and offline imitation learning (BCQ) under state-independent action noise and state-dependent action noise?* and *(2) are the improvements significant?*

Synthetic benchmarks generation. To answer these questions, we pre-train expert agents on the benchmark tasks and then use the pre-trained experts to generate demonstrations with synthetic *state-independent* action noise (Algorithm 1) and *state-dependent* action noise (Algorithm 2). We denote $\mathcal{D}$ as *standard demonstration* with expert actions and $\tilde{\mathcal{D}}$ as a demonstration with noisy actions, a.k.a. the *noisy demonstration.* Note that we use ground-truth rewards only to train the expert agent, and we discard the rewards afterward. We set the scaling weight $\lambda_{neg} = 1.0$ and the smooth rate $\alpha = 0.01$ across all the experiments.

Statistical analysis. We perform one-way **ANOVA** (Analysis of VAriance) for the experimental results to show the statistical significance of the differences between USN and baselines. We can conclude that there are significant differences among treatments if the p-value obtained from ANOVA analysis is small (p<0.05). However, ANOVA does not tell which treatments are significantly different from each other. To know the pairs of significantly different treatments, we will perform multiple pairwise comparison (post hoc comparison) analyses for all unplanned comparisons using Tukey's honestly significantly differenced (HSD) test. **Tukey's HSD test** accounts for multiple comparisons using the following formula:

$$\text{HSD} = q_{A,\alpha,dof}\sqrt{\frac{\text{MSE}}{n}}, \tag{13}$$

where $q_{A,\alpha,dof}$ denotes the studentized range statistic with A number of groups, α significance level (0.05 or 0.01), and dof degrees of freedom; MSE is the mean square error from ANOVA; n is the sample size in each group, when the sample size is equal in two comparison groups. Note that p-value 0.001 from Tukey's HSD output should be interpreted as <=0.001.

5.1 Behavioral Cloning with Noisy Demonstrations

Experimental setup. For behavioral cloning, we directly replace the cross-entropy loss as our USN loss to obtain the robust version of BC - BC-USN. We conduct experiments on the classic control task from OpenAI Gym (Brockman et al., 2016) - LunarLander-v2. This is a classic rocket trajectory optimization problem. According to Pontryagin's maximum principle (Kopp, 1962), it is optimal to fire the engine at full throttle or turning it off in the LunarLander environment. We pre-train a DQN policy as the expert agent and generate noisy demonstrations with 50K steps. We set the noise rates for both types of action noise as {0.1, 0.2, 0.3, 0.4}. We use a 3-layer MLP architecture with 32 units in each hidden layer as the model backbone for implementing BC, BC-GCE, and BC-USN. We train all the models using the Adam optimizer for 20 epochs. Our implementation is based on the opensource code[4]. We set $\lambda_{neg} = 1.0$ for BC-USN.

Baselines. Besides the original BC, we also compare BC-USN to a baseline robust BC model called BC-GCE that is trained using the generalized cross-entropy loss (GCE) loss (Zhang & Sabuncu, 2018).

Results. Tables 5 and 6 show the performance of BC, BC-GCE, and BC-USN under multiple *state-independent* action noise and *state-dependent* action noise, respectively. The results demonstrate that BC-USN consistently outperforms baselines significantly under both types of action noise.

ϵ	0.1	0.2	0.3	0.4
BC	175.6 ± 74.9	177.4 ± 61.2	37.7 ± 154.4	-72.4 ± 178.1
BC-GCE	-561.4 ± 124.5	-461.3 ± 148.6	-252.3 ± 119.4	-498.6 ± 143.4
BC-USN **(Ours)**	**233.5 ± 28.6**	**233.0 ± 18.6**	**214.8 ± 34.2**	**217.6 ± 19.0**

Table 5: Average return of BC, BC-GCE and our BC-USN on LunarLander-v2 with *state-independent* (symmetric) action noise. The reported results are averaged over five random seeds.

ϵ	0.1	0.2	0.3	0.4
BC	189.2 ± 36.4	113.8 ± 104.2	2.2 ± 118.4	-159.7 ± 101.2
BC-GCE	-12.5 ± 165.8	-239.6 ± 161.7	-466.2 ± 143.8	-631.3 ± 95.9
BC-USN **(Ours)**	**243.6 ± 18.8**	**205.1 ± 56.8**	**232.0 ± 25.6**	**195.9 ± 66.3**

Table 6: Average return of BC, BC-GCE and our BC-USN on LunarLander-v2 with *state-dependent* action noise. The reported results are averaged over five random seeds.

[4] https://github.com/HumanCompatibleAI/imitation

ANOVA. Table 7 shows ANOVA analysis results under *state-independent* action noise and *state-dependent* action noise. For most cases, the p-value obtained from ANOVA analysis is significant (p<0.05). Therefore, we conclude that there are significant differences among treatments (BC, BC-GCE, and BC-USN).

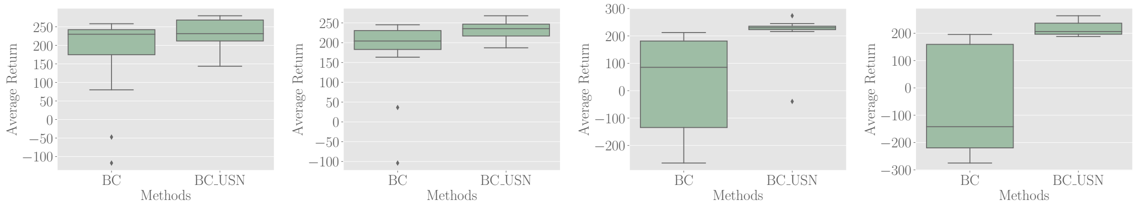
Action noise	p-value			
ϵ	0.1	0.2	0.3	0.4
Symmetric	0.0777	**0.00335**	**7.49e-4**	**2.11e-6**
State-dependent	**7.34e-5**	**0.00963**	**3.947e-7**	**1.16e-10**

Table 7: ANOVA on *state-independent* (symmetric) and *state-dependent* action noises.

Tukey's HSD test. Table 8 summarizes Tukey's HSD results of BC and BC-USN on symmetric action noise and *state-dependent* action noise. The results suggest that except (BC, BC-USN) with Symmetric-0.1 action noise, all other pairwise comparisons from treatments reject the null hypothesis ($p < 0.05$) and indicate statistically significant differences.

Action noise	ϵ	Group1	Group2	Diff	q-value	p-value
Symmetric	0.1	BC	BC-USN	**57.919**	2.589	0.0777
	0.2	BC	BC-USN	**55.541**	3.161	**0.00335**
	0.3	BC	BC-USN	**177.161**	5.349	**0.001**
	0.4	BC	BC-USN	**290.141**	8.41	**0.001**
State-dependent	0.1	BC	BC-USN	**54.390**	6.567	**0.001**
	0.2	BC	BC-USN	**91.362**	3.929	**0.00963**
	0.3	BC	BC-USN	**229.728**	9.299	**0.001**
	0.4	BC	BC-USN	**355.648**	14.021	**0.001**

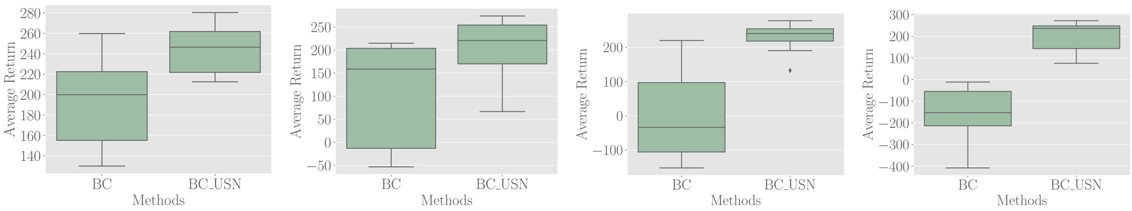
Table 8: Tukey's HSD test on *state-independent* (symmetric) and *state-dependent* action noises.



(a) $\epsilon = 0.1$ (b) $\epsilon = 0.2$ (c) $\epsilon = 0.3$ (d) $\epsilon = 0.4$

Figure 13: Boxplot on symmetric action noise.

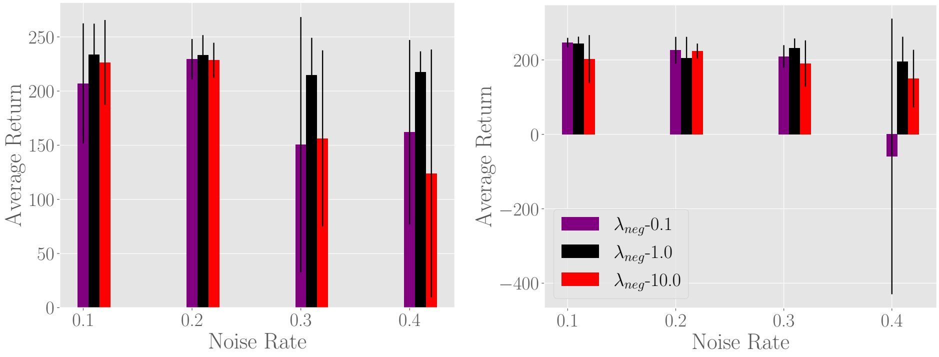
Using the boxplots Figure 13 and Figure 14, we can easily detect the differences between BC and BC-USN under symmetric action noise and *state-dependent* action noise with different noise rates. We can conclude that USN improves the robustness of BC against diverse action noises, and the improvements are usually statistically significant.



(a) $\epsilon = 0.1$ (b) $\epsilon = 0.2$ (c) $\epsilon = 0.3$ (d) $\epsilon = 0.4$

Figure 14: Boxplot on *State-dependent* action noise.

Ablation study on λ_{neg}. We perform an ablation to study the effects of λ_{neg} to the performance of BC-USN under symmetric and *state-dependent* action noises. The ablation study is conducted on the LunarLander-v2 task over three λ_{neg} (0.1, 1.0, and 10.0) and four noise rates (0.1, 0.2, 0.3, and 0.4). From the bar charts in Figure 15 and ablation study results in Table 9, we know that BC-USN achieves the best performance in most cases when $\lambda_{neg} = 1.0$. Therefore, we set $\lambda_{neg} = 1.0$ for the following experiments.



(a) Symmetric (b) *State-dependent*

Figure 15: Ablation study of λ_{neg} on the LunarLander-v2 task.

Action noise	ϵ / λ_{neg}	0.1	0.2	0.3	0.4
Symmetric	0.1	207.1 ± 55.3	229.4 ± 18.5	150.3 ± 117.9	161.9 ± 85.2
	1.0	**233.5 ± 28.6**	**233.0 ± 18.6**	**214.8 ± 34.2**	**217.6 ± 19.0**
	10.0	226.4 ± 39.1	228.4 ± 16.1	156.2 ± 81.3	123.8 ± 114.4
State-dependent	0.1	**246.4 ± 12.8**	**225.9 ± 36.0**	209.3 ± 30.3	-59.3 ± 370.3
	1.0	243.6 ± 18.8	205.1 ± 56.8	**232.0 ± 25.6**	**195.9 ± 66.3**
	10.0	202.0 ± 64.6	223.8 ± 20.1	190.2 ± 62.3	149.7 ± 77.4

Table 9: Average return of our BC-USN on LunarLander-v2 with different λ_{neg} on *state-independent* (symmetric) action noise and *state-dependent* action noise. The reported results are averaged over five random seeds.

5.2 Online Imitation Learning with Noisy Demonstrations

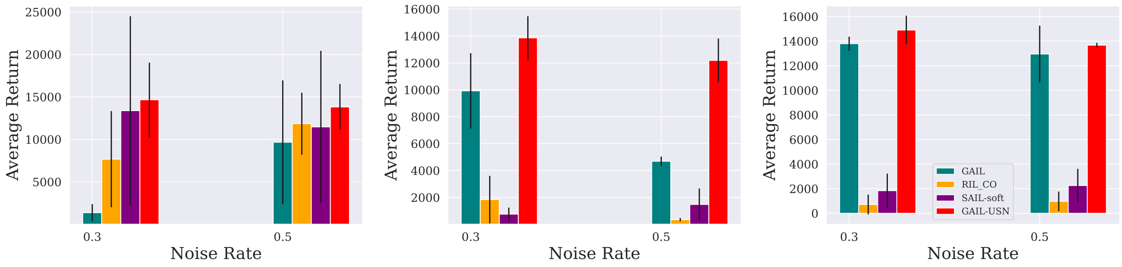
Experimental setup. Since most of the previous robust online imitation learning algorithms (Tangkaratt et al., 2020a; Wang et al., 2021a) are based on GAIL, we also choose GAIL as our base method. However, the original GAIL can not perform well in a high-dimensional environment like Atari games (Brown et al., 2019). Fortunately, we found that using behavioral cloning as an initialization for the actor, GAIL can achieve good performance on some Atari games, e.g., KungFuMaster, Q*bert, and Hero, with only one full-episode demonstration. We use widely used Atari games simulated through Arcade Learning Environment (Bellemare et al., 2013). We generate one full-episode demonstration using pre-trained PPO agents. The PPO is trained with a learning rate of 2.5e-4, a clipping threshold of 0.1, an entropy coefficient of 0.01, a value function coefficient of 0.5, and a GAE parameter of 0.95 (Schulman et al., 2016). We generated noisy demonstrations with *state-independent* action noise and *state-dependent* action noise from the one full-episode demonstration following Algorithm 1 and Algorithm 2, respectively. According to the application scope analysis (Table 4, Section 3.2), we generate demonstrations with noise rate ϵ>=0.3 for this online imitation learning experiment.

Implementation and baselines. We apply USN for training the discriminator of GAIL, resulting in the robust algorithm GAIL-USN. We add an auxiliary fully-connected layer to output logits for using USN loss to achieve this goal. The original discriminator of GAIL can be presented as $D_\phi = f_d \circ h$, where h is the backbone network for learning features from input state-action pairs. We add a fully-connected layer f_l to output logits from the feature via $f_l \circ h$. The whole learning objective of GAIL-USN is as follows:

$$\begin{aligned}\min_{\theta} \max_{\phi} \mathbb{E}_{(s,a)\sim\rho_\theta}&\Big[\log D_\phi(s,a) + \mathrm{USN}\big((s,a), f_l \circ h\big)\Big] \\ &+ \mathbb{E}_{(s,a)\sim\rho_E}\Big[\log\big(1 - D_\phi(s,a)\big) + \mathrm{USN}\big((s,a), f_l \circ h\big)\Big],\end{aligned} \tag{14}$$

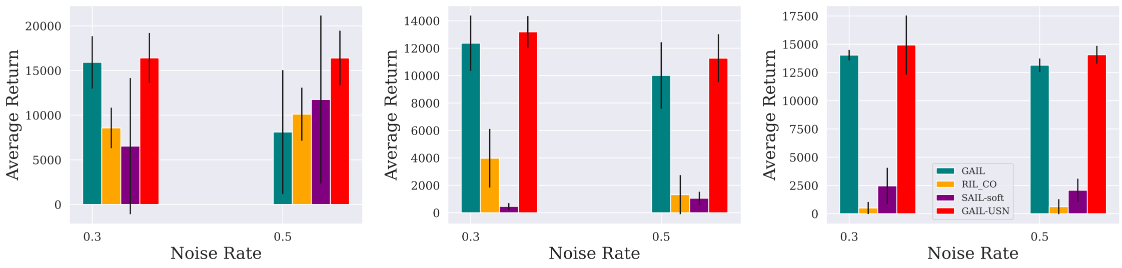
where ρ_θ and ρ_E denote the occupancy measures of agent policies π_θ and the demonstrator π_E, respectively. We compare GAIL-USN to the original GAIL and state-of-the-art robust imitation learning algorithms, i.e., RIL_CO (Tangkaratt et al., 2020a) and SAIL with soft weights (Wang et al., 2021a). We implement all the methods using the open-source code https://github.com/yunke-wang/gail_atari. All the policies are trained using PPO with the same hyper-parameters as the expert agents.

Results. We conduct experiments on KungFuMaster, Q*bert, and Hero games under *state-independent* (symmetric and pairflip) action noise and *state-dependent* action noise with noise rate ϵ of 0.3 and 0.5. We compare GAIL-USN to the original GAIL, RIL_CO, and SAIL-soft. The bar charts in Figure 16 show the performance differences of the algorithms under symmetric action noise over the noise rate of 0.3 and 0.5. GAIL performs badly on the KungFuMaster game, even with a noise rate of 0.3. RIL_CO and SAIL-soft outperform GAIL with a noise rate of 0.3 and 0.5, while still performing worse than our GAIL-USN. In Q*bert and Hero, RIL_CO and SAIL-soft totally fail with both noise rates, while GAIL achieves the second-best performance. Figure 16(b) shows that our GAIL-USN significantly outperforms the original GAIL and other baselines under symmetric action noises. The results comparison under *state-dependent* action noise (Figure 18) are similar to those under Symmetric noise in Figure 16. Our GAIL-USN significantly outperforms GAIL and other



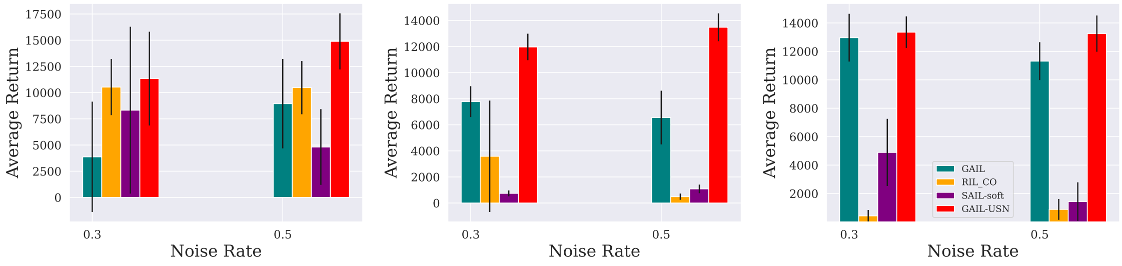
(a) KungFuMaster (b) Q*bert (c) Hero

Figure 16: The performance of GAIL-USN and baselines with noisy demonstration across different *state-independent* (symmetric) action noise on Atari games.



(a) KungFuMaster (b) Q*bert (c) Hero

Figure 17: The performance of GAIL-USN and baselines with noisy demonstration across different *state-independent* action noise (pairflip) on Atari games.



(a) KungFuMaster (b) Q*bert (c) Hero

Figure 18: The performance of GAIL-USN and baselines with noisy demonstration across different *state-dependent* action noise on Atari games.

baselines in the Q*bert game under the *state-dependent* action noise. Figure 17 demonstrates that GAIL-USN consistently outperforms other baselines under the pairflip action noise.

The quantitative comparisons for each game are summarized in Table 10, Table 11 and Table 12. SAI-soft, and RIL_CO usually perform worse than the original GAIL, while our GAIL-USN consistently outperforms the baselines under all the evaluated noise settings.

KungFuMaster	ϵ	0.3	0.5
Symmetric	GAIL	3,873.0 ± 5,264.5	8,944.3 ± 4,263.9
	RIL_CO	10,529.0 ± 2,678.4	10,473.9 ± 2,537.2
	SAIL-soft	8,324.3 ± 7,958.4	4,811.1 ± 3,612.7
	GAIL-USN (ours)	**11,337.0 ± 4,471.4**	**14,887.6 ± 2,675.2**
Pairflip	GAIL	15,915.3 ± 2,932.6	8,110.0 ± 6,933.2
	RIL_CO	8,572.3 ± 2,268.2	10,110.1 ± 2,968.8
	SAIL-soft	6,531.7 ± 7,621.9	11,752.7 ± 9,405.2
	GAIL-USN (ours)	**16,409.8 ± 2,796.1**	**16,399.3 ± 3,060.1**
State-dependent	GAIL	1,369.9 ± 1,010.5	9,664.6 ± 7,281.6
	RIL_CO	7,672.5 ± 5,656.8	11,850.3 ± 3,647.1
	SAIL-soft	13,380.8 ± 11,120.8	11,472.2 ± 8,951.7
	GAIL-USN (ours)	**14,666.6 ± 4,366.6**	**13,819.6 ± 2,708.0**

Table 10: Average return of GAIL, RIL_CO, SAIL-soft, and our GAIL-USN with noisy demonstrations on the KungFuMaster games. The reported results are averaged over five random seeds.

Q*bert	ϵ	0.3	0.5
Symmetric	GAIL	7,776.3 ± 1,184.3	6,555.3 ± 2,058.9
	RIL_CO	3,582.5 ± 4,274.3	484.1 ± 240.5
	SAIL-soft	739.1 ± 225.0	1,082.0 ± 330.0
	GAIL-USN (ours)	**11,972.1 ± 1,014.7**	**13,481.1 ± 1,068.3**
Pairflip	GAIL	11,534.4 ± 1,618.4	10,359.0 ± 2,686.0
	RIL_CO	2,751.2 ± 1,653.7	1,824.8 ± 1,897.7
	SAIL-soft	513.4 ± 173.2	1,136.8 ± 377.4
	GAIL-USN (ours)	**13,066.2 ± 1,144.7**	**10,974.4 ± 1,775.8**
State-dependent	GAIL	9,922.7 ± 2,799.3	4,686.9 ± 350.1
	RIL_CO	1,838.7 ± 1,762.7	346.1 ± 119.1
	SAIL-soft	758.2 ± 484.2	1,478.2 ± 1,185.1
	GAIL-USN (ours)	**13,861.7 ± 1,608.5**	**12,186.6 ± 1,625.4**

Table 11: Average return of GAIL, RIL_CO, SAIL-soft, and our GAIL-USN with noisy demonstrations on the Q*bert games. The reported results are averaged over five random seeds.

Hero	ϵ	0.3	0.5
Symmetric	GAIL	12,962.7 ± 1,680.0	11,314.4 ± 1,333.4
	RIL_CO	427.9 ± 411.4	880.8 ± 735.2
	SAIL-soft	4,889.9 ± 2,362.7	1,431.2 ± 1,357.7
	GAIL-USN (ours)	**13,350.2 ± 1,113.1**	**13,249.6 ± 1,276.6**
Pairflip	GAIL	14,034.2 ± 464.5	13,140.7 ± 597.7
	RIL_CO	502.4 ± 533.7	618.3 ± 664.2
	SAIL-soft	2,459.3 ± 1,600.6	2,071.7 ± 1,022.9
	GAIL-USN (ours)	**14,930.2 ± 2,609.9**	**14,068.0 ± 787.4**
State-dependent	GAIL	13,796.1 ± 570.9	12,954.0 ± 2,306.0
	RIL_CO	710.6 ± 802.6	972.3 ± 794.2
	SAIL-soft	1,836.4 ± 1,387.3	2,256.2 ± 1,356.5
	GAIL-USN (ours)	**14,904.8 ± 1,162.5**	**13,672.8 ± 2,708.0**

Table 12: Average return of GAIL, RIL_CO, SAIL-soft, and our GAIL-USN with noisy demonstrations on the Hero games. The reported results are averaged over five random seeds.

ANOVA. Table 13 shows the ANOVA analysis results for KungFuMaster, Q*bert, and Hero under the symmetric, pairflip, and *state-dependent* action noises with two noise rates. For most cases, the p-value obtained from ANOVA analysis is significant (p<0.05). Therefore, we conclude that there are significant differences among treatments (GAIL, RIL_CO, SAIL-soft, and GAIL-USN).

Environments	p-value					
	Symmetric		Pairflip		State-dependent	
	0.3	0.5	0.3	0.5	0.3	0.5
KungFuMaster	**9.443e-11**	0.155	**1.679e-11**	**8.425e-5**	**3.2e-5**	**3.97e-13**
Q*bert	**5.002e-52**	**3.814e-68**	**3.09e-63**	**1.002e-43**	**4.844e-35**	**2.973e-64**
Hero	**6.485e-82**	**9.067e-63**	**2.011e-61**	**8.032e-75**	**1.643e-56**	**1.022e-64**

Table 13: ANOVA analysis results for KungFuMaster, Q*bert, and Hero under the symmetric, pairflip, and *state-dependent* action noises with two noise rates.

Tukey's HSD test. Table 14 summarizes Tukey's HSD results of GAIL, RIL_CO, SAIL-soft, and GAIL-USN on symmetric noise, pairflip noise and *state-dependent* noise for the KungFuMaster game. Note that p-value 0.001 from Tukey's HSD output should be interpreted as <=0.001. The results suggest that 10 out of 18 pairwise comparisons from treatments reject the null hypothesis ($p < 0.05$) and indicate statistically significant differences. The ratio of significant improvements is 55.55%.

KungFuMaster	ϵ	Group1	Group2	Diff	q-value	p-value
Symmetric	0.3	GAIL	GAIL-USN	13,296.636	9.911	**0.001**
		RIL_CO	GAIL-USN	6,994.060	5.213	**0.00194**
		SAIL-soft	GAIL-USN	1,285.727	0.958	0.9
	0.5	GAIL	GAIL-USN	4,154.999	3.250	0.104
		RIL_CO	GAIL-USN	1,969.393	1.540	0.674
		SAIL-soft	GAIL-USN	2,347.424	1.836	0.558
Pairflip	0.3	GAIL	GAIL-USN	494.484	0.463	0.900
		RIL_CO	GAIL-USN	7,837.424	7.348	**0.001**
		SAIL-soft	GAIL-USN	9,878.090	9.262	**0.001**
	0.5	GAIL	GAIL-USN	8,289.303	6.561	**0.001**
		RIL_CO	GAIL-USN	6,289.242	4.978	**0.003**
		SAIL-soft	GAIL-USN	4,646.575	3.678	0.050
State-dependent	0.3	GAIL	GAIL-USN	7,464.030	6.553	**0.001**
		RIL_CO	GAIL-USN	807.090	0.708	0.900
		SAIL-soft	GAIL-USN	3,012.727	2.645	0.246
	0.5	GAIL	GAIL-USN	5,943.242	7.351	**0.001**
		RIL_CO	GAIL-USN	4,413.697	5.459	**0.001**
		SAIL-soft	GAIL-USN	10,076.424	12.464	**0.001**
Ratio of significant improvements:						55.55%

Table 14: Tukey's HSD test on the KungFuMaster game under *state-independent* (symmetric and pairflip) action noise and *state-dependent* action noise.

Q*bert	ϵ	Group1	Group2	Diff	q-value	p-value
Symmetric	0.3	GAIL	GAIL-USN	3,939.045	10.182	**0.001**
		RIL_CO	GAIL-USN	12,022.995	31.081	**0.001**
		SAIL-soft	GAIL-USN	13,103.513	33.874	**0.001**
	0.5	GAIL	GAIL-USN	7,499.719	32.799	**0.001**
		RIL_CO	GAIL-USN	11,840.553	51.779	**0.001**
		SAIL-soft	GAIL-USN	9,865.300	46.828	**0.001**
Pairflip	0.3	GAIL	GAIL-USN	1,531.833	5.088	**0.00263**
		RIL_CO	GAIL-USN	10,314.992	34.265	**0.001**
		SAIL-soft	GAIL-USN	12,737.530	42.316	**0.001**
	0.5	GAIL	GAIL-USN	615.462	1.531	0.678
		RIL_CO	GAIL-USN	9,149.613	22.761	**0.001**
		SAIL-soft	GAIL-USN	10,185.257	25.337	**0.001**
State-dependent	0.3	GAIL	GAIL-USN	4,195.772	9.268	**0.001**
		RIL_CO	GAIL-USN	8,389.583	18.533	**0.001**
		SAIL-soft	GAIL-USN	11,232.992	24.814	**0.001**
	0.5	GAIL	GAIL-USN	6,925.795	24.613	**0.001**
		RIL_CO	GAIL-USN	12,997.053	46.189	**0.001**
		SAIL-soft	GAIL-USN	12,309.146	44.062	**0.001**
Ratio of significant improvements:						94.44%

Table 15: Tukey's HSD test on the Q*bert game under *state-independent* (symmetric and pairflip) action noise and *state-dependent* action noise.

Table 15 summarizes Tukey's HSD results for the Q*bert game. The results suggest that except the (GAIL, GAIL-USN) under Pairflip-0.5 noise, all other pairwise comparisons from treatments reject the null hypothesis ($p < 0.05$) and indicate statistically significant differences. The ratio of significant improvements is 94.44%. Table 16 summarizes Tukey's HSD results for the Hero game. The ratio of significant improvements is 77.77%. Therefore, we conclude that our GAIL-USN consistently outperforms baselines on KungFuMaster, Q*bert, and Hero games under three types of action noise with two noise rates, and the improvements are usually significant.

Hero	ϵ	Group1	Group2	Diff	q-value	p-value
Symmetric	0.3	GAIL	GAIL-USN	1,108.786	4.554	**0.00890**
		RIL_CO	GAIL-USN	14,194.199	58.298	**0.001**
		SAIL-soft	GAIL-USN	13,068.460	53.674	**0.001**
	0.5	GAIL	GAIL-USN	718.759	2.203	0.407
		RIL_CO	GAIL-USN	12,700.433	38.935	**0.001**
		SAIL-soft	GAIL-USN	11,416.551	34.999	**0.001**
Pairflip	0.3	GAIL	GAIL-USN	895.942	2.393	0.332
		RIL_CO	GAIL-USN	14,427.750	38.544	**0.001**
		SAIL-soft	GAIL-USN	12,470.831	33.316	**0.001**
	0.5	GAIL	GAIL-USN	927.277	3.501	0.069
		RIL_CO	GAIL-USN	13,449.739	50.784	**0.001**
		SAIL-soft	GAIL-USN	11,996.333	45.296	**0.001**
State-dependent	0.3	GAIL	GAIL-USN	387.474	1.109	0.845
		RIL_CO	GAIL-USN	12,922.313	36.997	**0.001**
		SAIL-soft	GAIL-USN	8,460.231	24.222	**0.001**
	0.5	GAIL	GAIL-USN	1,935.165	6.478	**0.001**
		RIL_CO	GAIL-USN	12,368.772	41.409	**0.001**
		SAIL-soft	GAIL-USN	11,818.353	39.566	**0.001**
Ratio of significant improvements:						77.77%

Table 16: Tukey's HSD test on the Hero game under *state-independent* (symmetric and pairflip) action noise and *state-dependent* action noise.

5.3 Offline Imitation Learning with Noisy Demonstrations

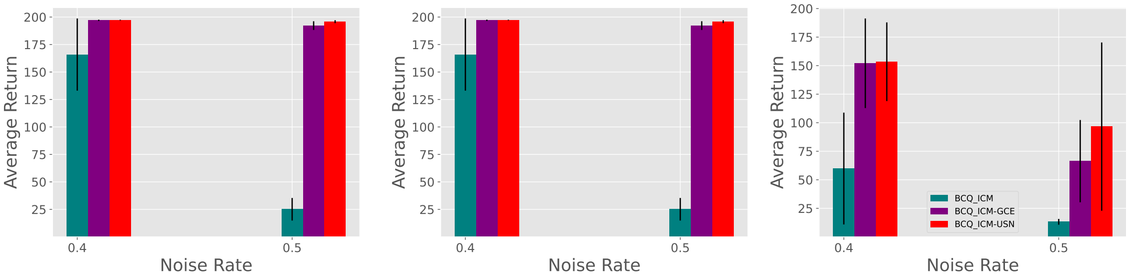
Experimental setup. We first pre-train a QRDQN policy as the expert agents on the Atari games. The expert policy is trained using Adam optimizer with a learning rate of 0.0001 for 100 epochs. Then, we generate 50K-step demonstration datasets with synthetic *state-independent* and *state-dependent* action noises.

Implementation and baselines. We apply USN on the generative model G, resulting in our robust method BCQ_ICM-USN. Algorithm 5 presents an instance of our BCQ_ICM-USN. BCQ_ICM-USN improves BCQ_ICM by updating the generative model G using our USN method (Algorithm 4) in step 8. USN enhances the ability of generative model G to eliminate noisy actions for policy learning. Thus, BCQ_ICM-USN is more robust than BCQ_ICM, achieving better imitation learning performance across diverse levels of action noise. We compare BCQ_ICM-USN with BCQ_ICM, BCQ_ICM-GCE, and UWAC (Wu et al., 2021). We use the open-source platform[5] for implementing the algorithms.

Algorithm 5 BCQ_ICM-USN for robust offline imitation learning against action noise.

```
Input: Demonstration $\mathcal{D} = \{(s_i, a_i, s_{i+1})\}_{i=1}^{N}$, number of iterations $T$, target_update_rate, mini-batch size $B$, threshold $\tau$.
Initialize Q-network $Q_\theta$, generative model $G$ with parameter $\omega$ and target network $Q_{\theta'}$ with $\theta' \leftarrow \theta$.
for $t = 1$ to $T$ do
    Sample mini-batch $M$ of $B$ transitions $(s, a, s')$ from $\mathcal{D}$.
    Train an ICM module to generate intrinsic reward: $r = \mathrm{ICM}(s, a, s')$.
    $a' \leftarrow \arg\max_{a' | G(a'|s') / \max_{\hat{a}} G(\hat{a}|s') > \tau} Q_\theta(s', a')$.
    $\theta \leftarrow \arg\min_\theta \sum_{(s,a,s') \in M} k_\kappa(r + \gamma Q_{\theta'}(s', a') - Q_\theta(s, a))$
    $\omega \leftarrow \arg\min \mathcal{L}_{\mathrm{USN}}(M, \omega)$        // Update $G$ using USN approach in Algorithm 4.
    If $t$ mod target_update_rate $= 0 : \theta' \leftarrow \theta$.
end for
```

Results on control tasks. The bar chats in Figure 19 demonstrate that our method BCQ_ICM-USN outperforms the baselines on CartPole-v0 under *state-independent* (symmetric and pairflip) action noise and *state-dependent* action noise. We can see that the performance of BCQ_ICM drops when the noise rate is larger than 0.3, while our method BCQ_ICM-USN consistently performs the best across diverse noise rates. Figure 20 shows that our method BCQ_ICM-USN consistently outperforms the baselines under the three types of action noise across two noise rates.

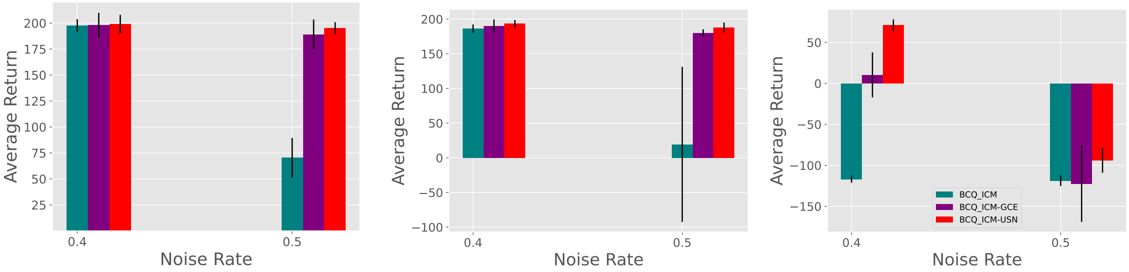


(a) Symmetric noise (b) Pairflip noise (c) State-dependent noise

Figure 19: Average return of BCQ_ICM, BCQ_ICM-GCE and our BCQ_ICM-USN on CartPole-v0 with *state-independent* (symmetric and pairflip) action noise, and *state-dependent* action noise.

The quantitative results of CartPole-v0 and LunarLander-v2 are summarized in Table 17, and Table 18, respectively. The results show that the improvements of our method BCQ_ICM-USN to the baselines are usually significant, especially under the *state-dependent* action noise. In the CartPole-v0 task, BCQ_ICM-USN achieves a performance that is 1.45 times of BCQ_ICM-GCE and 7.3 times of BCQ_ICM under *state-dependent* action noise with a noise rate of 0.5.

[5] https://github.com/thu-ml/tianshou



(a) Symmetric noise (b) Pairflip noise (c) State-dependent noise

Figure 20: Average return of BCQ_ICM, BCQ_ICM-GCE and our BCQ_ICM-USN on LunarLander-v2 with *state-independent* (symmetric and pairflip) action noise, and *state-dependent* action noise.

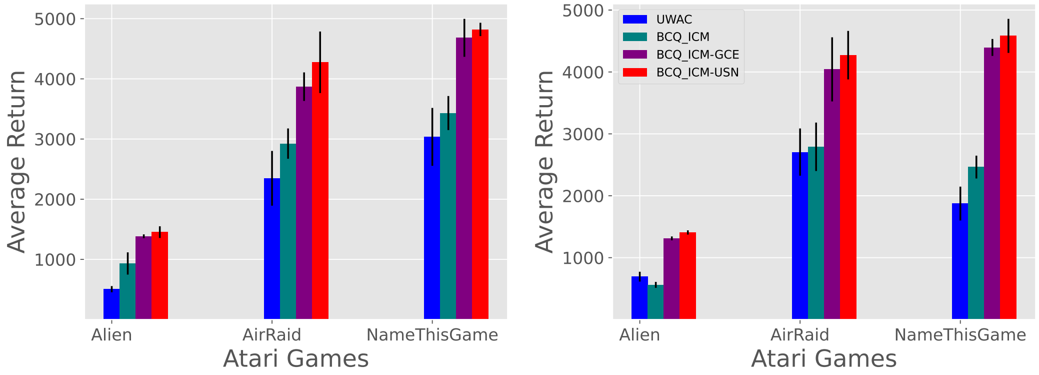
CartPole-v0	ϵ	0.4	0.5
Symmetric	BCQ_ICM	165.8 ± 32.8	25.0 ± 10.2
	BCQ_ICM-GCE	196.9 ± 0.6	192.2 ± 3.2
	BCQ_ICM-USN (ours)	**197.2** ± **0.3**	**195.7** ± **1.3**
Pairflip	BCQ_ICM	165.8 ± 32.8	25.0 ± 10.2
	BCQ_ICM-GCE	196.9 ± 0.6	192.2 ± 3.9
	BCQ_ICM-USN (ours)	**197.2** ± **0.3**	**195.7** ± **1.3**
State-dependent	BCQ_ICM	59.8 ± 48.9	13.1 ± 2.5
	BCQ_ICM-GCE	152.0 ± 39.2	66.3 ± 36.0
	BCQ_ICM-USN (ours)	**153.4** ± **34.4**	**96.5** ± **73.7**

Table 17: Average return of BCQ_ICM, BCQ_ICM-GCE and our BCQ_ICM-USN with noisy demonstrations on the CartPole-v0 task.

LunarLander-v2	ϵ	0.4	0.5
Symmetric	BCQ_ICM	197.7 ± 6.0	70.3 ± 19.0
	BCQ_ICM-GCE	197.9 ± 11.8	189.1 ± 14.3
	BCQ_ICM-USN (ours)	**198.9** ± **8.9**	**195.0** ± **5.8**
Pairflip	BCQ_ICM	186.4 ± 5.8	19.3 ± 5.1
	BCQ_ICM-GCE	189.9 ± 9.6	180.1 ± 5.1
	BCQ_ICM-USN (ours)	**193.3** ± **5.4**	**187.7** ± **7.1**
State-dependent	BCQ_ICM	-116.9 ± 4.2	-118.7 ± 6.4
	BCQ_ICM-GCE	10.3 ± 27.4	-122.5 ± 46.4
	BCQ_ICM-USN (ours)	**70.8** ± **7.2**	**-93.7** ± **15.2**

Table 18: Average return of BCQ_ICM, BCQ_ICM-GCE and our BCQ_ICM-USN with noisy demonstrations on the LunarLander-v2 task.

Results on Atari games. In Figure 21, we show bar charts comparison of UWAC (Gal & Ghahramani, 2016), BCQ_ICM, BCQ_ICM-GCE, and BCQ_ICM-USN for three Atari games under *state-independent* (symmetric) action noise and *state-dependent* action noise with a noise rate of 0.45. The results show that UWAC is the worst among all the algorithms, while our method, BCQ_ICM-USN, consistently outperforms the baselines. The performance improvements of BCQ_ICM-USN compared to other methods are usually significant. Table 19 and Table 20 summarize the quantitative comparison results under the symmetric noise and *state-dependent* action noise, respectively.



(a) Symmetric noise

(b) Pairflip noise

Figure 21: Bar chart comparison of the average return of BCQ_ICM, BCQ_ICM-GCE and BCQ_ICM-USN on Atari games with 45% Symmetric and Piirflip action noise.

Atari Games	Alien	AirRaid	NameThisGame
UWAC	505.1 ± 51.5	2,347.5 ± 455.0	3,035.0 ± 481.0
BCQ_ICM	932.1 ± 184.8	2,924.1 ± 251.6	3,431,8 ± 282.8
BCQ_ICM-GCE	1,384.3 ± 31.3	3,870.4 ± 237.0	4,681.8 ± 315.1
BCQ_ICM-USN **(Ours)**	**1,453.0 ± 97.3**	**4,275.4 ± 511.2**	**4,821.3 ± 110.6**

Table 19: Average return of UWAC, BCQ_ICM, BCQ_ICM-GCE, and BCQ_ICM-USN when learning with *state-independent* (symmetric) action noise with noise rate of 0.45 on four Atari games.

Atari Games	Alien	AirRaid	NameThisGame
UWAC	693.0 ± 79.6	2,705.8 ± 381.6	1,874.1 ± 273.5
BCQ_ICM	560.3± 48.0	2,791.2 ± 391.2	2,463.6 ± 184.3
BCQ_ICM-GCE	1,311.6 ± 34.3	4,042.0 ± 517.9	4,396.1±136.8
BCQ_ICM-USN **(Ours)**	**1,407.3 ± 34.3**	**4,271.6 ± 391.6**	**4,583.0 ± 275.0**

Table 20: Average return of UWAC, BCQ_ICM, BCQ_ICM-GCE, and BCQ_ICM-USN when learning with *state-independent* (pairflip) action noise with noise rate of 0.45 on four Atari games.

6. Experiments on Real-world Benchmarks

In this section, we evaluate the effectiveness of USN on real-world benchmarks.

6.1 Conditional Imitation Learning (CIL) for Autonomous Driving

Imitation Learning is a promising approach for training autonomous driving models. Early imitation learning methods assume that they can infer optimal actions from perceptual inputs only. However, this assumption often does not hold in practice, and thus, they do not scale up to fully autonomous urban driving. Conditional imitation learning (CIL) (Codevilla et al., 2018) was proposed to address this challenge by taking in the perceptual inputs and control commands. In driving scenarios, CIL specializes in a set of four commands: *continue* (follow the road), *left* (turn left at the next intersection), *straight* (go straight at the next intersection), and *right* (turn right at the next intersection). The additional command guidance resolves the ambiguity in perceptuomotor mapping, and enables imitation learning to scale up to the complex urban scenarios.

In the original conditional imitation learning, the training dataset is $\mathcal{D} = \{(\mathbf{o}_i, \mathbf{c}_i, \mathbf{a}_i)\}_{i=1}^{N}$, where $\mathbf{o}$ is the observation, $\mathbf{c}$ is the command, and $\mathbf{a}$ denotes the action. The objective of the original conditional imitation learning is

$$\min_{\theta} \sum_{i} \ell(F(\mathbf{o}_i, \mathbf{c}_i; \theta), \mathbf{a}_i), \tag{15}$$

where F is the function approximator parameterized with θ, will be optimized to map observations to actions. Actions $\mathbf{a} = \langle s, a \rangle$ are two-dimensional vectors that collate steering angle and acceleration. Given the ground truth action $\mathbf{a}_{gt}$, the loss function of the original conditional imitation learning is defined as

$$\ell(\mathbf{a}, \mathbf{a}_{gt}) = \ell(\langle s, a \rangle, \langle s_{gt}, a_{gt} \rangle) = \|s - s_{gt}\|^2 + \lambda_a \|a - a_{gt}\|^2. \tag{16}$$

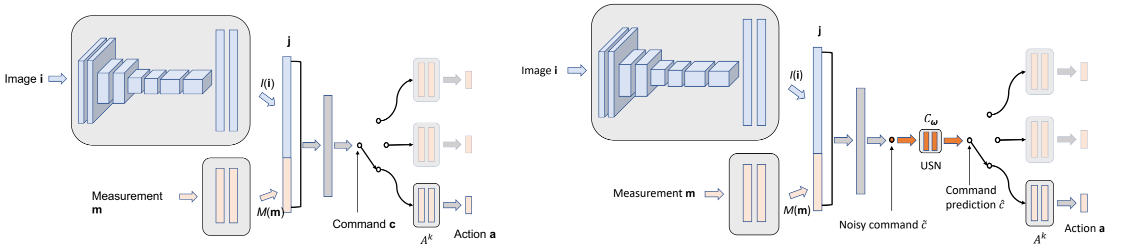
Network architecture. To take the commands into account, the original conditional imitation learning designs a branched architecture as in Figure 22 (a). Given the observation $\mathbf{o} = \langle \mathbf{i}, \mathbf{m} \rangle$ that comprises an image $\mathbf{i}$ and a measurement vector $\mathbf{m}$ (Dosovitskiy & Koltun, 2017) and a discrete set of commands $\mathcal{C} = \{\mathbf{c}^0, \cdots, \mathbf{c}^K\}$, the output of the branched architecture is

$$F(\mathbf{i}, \mathbf{m}, \mathbf{c}^i) = A^i(\mathbf{j}) = A^i(J(\mathbf{i}, \mathbf{m})), \tag{17}$$

where A^i are the branches for learning sub-policies that correspond to different commands, $\mathbf{j} = J(\mathbf{i}, \mathbf{m}) = \langle I(\mathbf{i}), M(\mathbf{m}) \rangle$ is the joint representation that concatenates the two modules' outputs.

6.2 Learning from Noisy Commands

The original conditional imitation learning assumes that high-quality commands are available for all the observations and actions. However, this assumption may not hold in practice because hiring expert annotators to give correct commands for all observations and actions is expensive or intractable. In practice, we usually only obtain a small dataset $\mathcal{D}_l =$



(a) CIL

(b) CIL-USN

Figure 22: The network architectures for conditional imitation learning: (a) the original CIL architecture where the correct command acts as a switch that selects between specialized sub-modules for choosing actions; (b) our CIL-USN architecture, which uses USN to correct the noisy commands before switching an action.

$\{(\mathbf{o}_i, \mathbf{c}_i, \mathbf{a}_i)\}_{i=1}^{M}$ with correct commands, and a larger dataset $\mathcal{D}_u = \{(\mathbf{o}_i, \mathbf{a}_i)\}_{i=M+1}^{N}$ without commands.

To leverage the large dataset, we train an inverse dynamics model $\mathcal{I}(\mathbf{c}_i|\mathbf{o}_i, \mathbf{o}_{i+1}; \eta)$ on $\mathcal{D}_l$ to infer pseudo commands for $\mathcal{D}_u$. Given the command prediction $\hat{\mathbf{c}}$ and the ground truth command $\mathbf{c}$, the inverse dynamics model $\mathcal{I}$ was optimized by minimizing the cross-entropy loss. Then, we can use the trained inverse dynamics model $\mathcal{I}$ to predict pseudo commands $\tilde{\mathbf{c}}$ for $\mathcal{D}_u$. However, due to the limited number of training data, the trained inverse dynamics model is inaccurate in predicting the pseudo commands for $\mathcal{D}_u$, resulting in the noisy demonstration dataset $\tilde{\mathcal{D}}_u = \{(\mathbf{o}_i, \tilde{\mathbf{c}}_i, \mathbf{a}_i)\}_{i=M+1}^{N}$. The noisy commands in the dataset will misguide the imitation learning policy in selecting the incorrect actions and fail urban driving. More details of noisy demonstrations are summarized in Appendix A. Given the noisy demonstration $\tilde{\mathcal{D}}$, we train CIL-USN to correct the commands before switching an action module for urban driving.

CIL-USN. To alleviate the harmful effects of the noisy commands $\tilde{\mathbf{c}}$, we introduce an additional *command-correction module* C_ω to generate corrected command predictions $\hat{\mathbf{c}}$ before switching an *action module* A^k. In Figure 22(b), we regard the noisy commands $\tilde{\mathbf{c}}$ as noisy discrete actions and apply our USN (Algorithm 4) to update C_ω. The resulting robust conditional imitation learning method is called CIL-USN. The objective of the CIL-USN is

$$\min_{\theta} \sum_{i} \ell(F(\mathbf{o}_i, \hat{\mathbf{c}}_i; \theta), \mathbf{a}_i), \tag{18}$$

where $\hat{\mathbf{c}} = C_\omega(J(\mathbf{i}, \mathbf{m}))$. We optimize the command-correction submodule C_ω by applying USN (Algorithm 4) as follows:

$$\min_{\omega} \mathbb{E}_{(\mathbf{o}, \tilde{\mathbf{c}}) \in \tilde{\mathcal{D}}_u} [\mathcal{L}_{\text{USN}}(\{(\mathbf{o}, \tilde{\mathbf{c}})\}, \omega)]. \tag{19}$$

Baselines. We compare CIL-USN with the following two related baselines:

- **CIL-CE**, which trains the *command-correction module* C_ω using cross-entropy loss.
- **CIL-GCE**, which trains the *command-correction module* C_ω using the robust generalized cross-entropy (GCE) loss (Zhang & Sabuncu, 2018).

Experimental setup. The CIL network contains *image module* and *action modules*. The CIL-USN and the baselines contain an additional *command-correction module*. The *image module* of CIL is the same as that of the inverse dynamics model $\mathcal{I}$. The other modules are implemented as standard multilayer perceptrons. The *command-correction module* and *action module* contain three fully connected layers with 256 units each, followed by ReLU nonlinearities and dropout operation. We train CIL-USN and the baselines on the noisy demonstration dataset $\tilde{\mathcal{D}}$ using the Adam optimizer with an initial learning rate of 0.0002.
Results. We measure the performance of CIL-USN and the baselines using the command correction accuracy of C_ω. Figure 23 and Table 21 show that CIL-USN achieves a higher command correction accuracy than other baselines in the cases of $\epsilon = 0.1$ and $\epsilon = 0.4$. The results of this experiment demonstrate that our proposed USN method improves the robustness of conditional imitation learning with noisy commands in complex urban driving tasks.

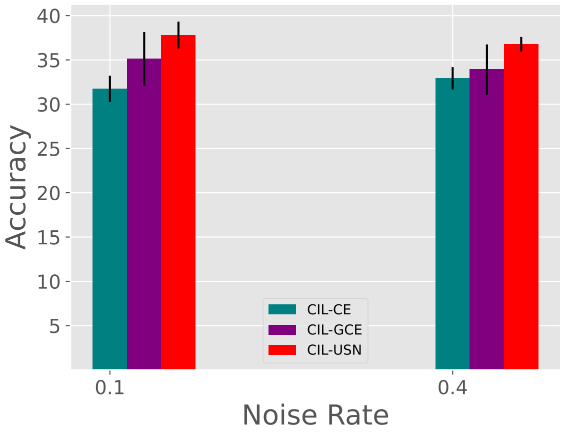


Figure 23: Bar chart comparison of the average return of CIL, CIL-GCE, and CIL-USN on noisy demonstrations with $\epsilon = 0.1$ and $\epsilon = 0.4$.

ϵ	CIL-CE	CIL-GCE	CIL-USN (Ours)
0.1	31.73 ± 1.48 %	35.11 ± 3.02 %	**37.79 ± 1.51%**
0.4	32.92 ± 1.25%	33.90 ± 2.83%	**36.77 ± 0.81%**

Table 21: Command prediction accuracy of CIL-CE, CIL-GCE, and CIL-USN on noisy demonstration dataset with $\epsilon = 0.1$ and $\epsilon = 0.4$. The reported results are the averaged accuracy over five random seeds.

ANOVA. We perform one-way ANOVA for the above results. Table 22 shows ANOVA analysis results for the real-world benchmark with noise rates of 0.1 and 0.4. The p-value obtained from ANOVA analysis is significant (p<0.05) in both cases. Therefore, we conclude that there are significant differences among treatments (CIL-CE, CIL-GCE, and CIL-USN).

ϵ	p-value
0.1	**7.67e-9**
0.4	**4.761e-5**

Table 22: ANOVA on the real-world benchmark with noise rate of 0.1 and 0.4.

Tukey's HSD test. Table 23 summarizes Turkey's HSD results of CIL-CE, CIL-GCE, and CIL-USN. Note that the p-value 0.001 from Tukey's HSD output should be interpreted as <=0.001. The results suggest that all of the pairwise comparisons from treatments reject the null hypothesis ($p < 0.05$) and indicate statistically significant differences. Thus, we conclude that CIL-USN significantly outperforms baselines on the real-world benchmark with noise rates of 0.1 and 0.4.

ϵ	Group1	Group2	Diff	q-value	p-value
0.1	CIL-CE	CIL-USN	6.071	10.957	**0.001**
0.1	CIL-GCE	CIL-USN	2.714	4.898	**0.00347**
0.4	CIL-CE	CIL-USN	3.720	7.019	**0.001**
0.4	CIL-GCE	CIL-USN	2.765	5.217	**0.00208**

Table 23: Tukey's HSD test on the real-world benchmark with noise rate of 0.1 and 0.4.

7. Discussion

Robustness. The experiments conducted in this paper showed that our proposed USN method consistently improves the robustness of behavioral cloning, online imitation learning (GAIL), and offline imitation learning (BCQ) under both *state-independent* (symmetric and pairflip) action noise and *state-dependent* action noise. The ANOVA and Tukey's HSD test results demonstrated that the improvements are usually significant. The ratio of significant improvements is up to 94.44%. Moreover, our USN improves the robustness of conditional imitation learning in the real-world urban driving benchmark with natural noisy commands.
Limitations and future work. The limitations in this work present opportunities for future research. Firstly, this work focuses on imitation learning with discrete action space. Extending our method to more general tasks with continuous action space would be a valuable direction for future investigation. To scale USN to continuous action space, regression uncertainty estimation methods might be helpful for sample selection (Gustafsson et al., 2023; Tohme et al., 2023). Secondly, USN only selects samples based on the positive correlation between loss and uncertainty estimations when the noise rate increases. Moreover, we only perform USN within the application scope (Table 4 of Section 3.2) of different imitation learning methods. It would be interesting to explore how to apply USN beyond the application scope, where negative correlations might exist. Finally, although we have demonstrated that our USN consistently improves the robustness of several traditional imitation learning algorithms across diverse types of action noise, it would be exciting and worthy to explore further the scalability of our method to foundation model-based decision making (Yang et al., 2023).

8. Conclusion

This paper thoroughly examined the correlations between loss estimation and uncertainty estimation in imitation learning models under *state-independent* action noise and *state-dependent* action noise. Based on positive correlations within the corresponding application scope of each IL method, we present a new paradigm for robust imitation learning in the presence of action noise. Our uncertainty-aware sample selection method for negative learning (USN) can be applied to various types of imitation learning, such as behavioral cloning, online imitation learning (GAIL), and offline imitation learning (BCQ), and is agnostic to the type of action noise. USN consistently brings a tangible improvement for diverse IL methods within the noise range determined in the correlation study (Section 3), i.e., $\epsilon >= 0.3$ for GAIL and $\epsilon >= 0.4$ for BCQ. Moreover, our method scales well to autonomous driving scenarios, where the driving policy is trained via conditional imitation learning with real-world noisy commands. Through ANOVA analysis and Tukey's HSD test, we conclude that USN usually significantly outperforms the baselines under diverse types of action noise. We have empirically demonstrated the new findings and discussed the limitations of our work. In addition, we have pointed out several interesting directions for future exploration.

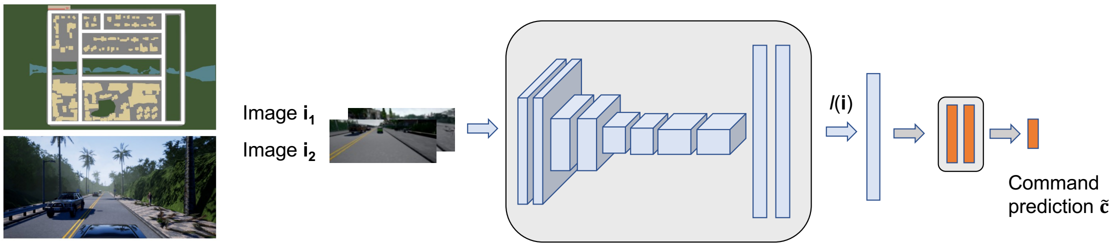
Acknowledgments

XY was supported by China Scholarship Council No. 201806450045, Australian Artificial Intelligence Institute (AAII), University of Technology Sydney (UTS), Australia, and Centre for Frontier AI Research (CFAR), Agency for Science, Technology and Research (A*STAR), Singapore (https://www.a-star.edu.sg/cfar). BH was supported by the NSFC General Program No. 62376235, Guangdong Basic and Applied Basic Research Foundation No. 2022A1515011652, HKBU Faculty Niche Research Areas No. RC-FNRA-IG/22-23/SCI/04, and HKBU CSD Departmental Incentive Scheme. IWT was supported by Australian Artificial Intelligence Institute (AAII), University of Technology Sydney (UTS), Australia. This research was partially supported by the National Research Foundation, Singapore, and the Maritime and Port Authority of Singapore / Singapore Maritime Institute under the Maritime Transformation Programme (Maritime Artificial Intelligence (AI) Research Programme – Grant number SMI-2022-MTP-06). The computational work for this article was partially performed on resources of the National Supercomputing Centre, Singapore (https://www.nscc.sg).

Appendix A. Noisy Demonstrations in the Real-world Benchmark

Dataset with noisy commands. In the experiments on real-world benchmark (Section 6), we use the CARLA dataset[6] to train our models. The dataset was collected using the urban driving simulator CARLA (Dosovitskiy et al., 2017). CARLA is an open-source simulator that contains dynamic urban environments with traffic. Figure 24(a) provides a map and sample views of a simulated urban environment in CARLA. The environment contains buildings, vegetation, traffic signs, and vehicular and pedestrian traffic.

[6] https://github.com/carla-simulator/imitation-learning



(a) CARLA.

(b) Network architecture of the inverse dynamics model $\mathcal{I}$.

Figure 24: (a) Simulated urban environment in CARLA (Codevilla et al., 2018); and (b) network architecture of the inverse dynamics model $\mathcal{I}$.

In the training dataset, the observation $\mathbf{o}$ is the currently observed image at 200×88 pixel resolution. The car's current speed is regarded as the measurement $\mathbf{m}$. Actions $\mathbf{a}$ are two-dimensional vectors that consist of steering angle s and acceleration a. As mentioned, we assume we can only access a small dataset $\mathcal{D}_l$ with correct command labels. Thus, we sample $\mathcal{D}_l$ from the original CARLA dataset with a portion of $\rho \in (0, 0.5)$. The rest $(1-\rho)$ portion of the CARLA dataset is considered as the $\mathcal{D}_u$ by removing the commands. Given the small dataset $\mathcal{D}_l$, we train the inverse dynamics model $\mathcal{I}(\mathbf{c}_i|\mathbf{o}_i, \mathbf{o}_{i+1}; \eta)$ for command prediction.

As shown in Figure 24(b), we consider the observation $\mathbf{o} = \mathbf{i}$ that only contains an image when training the inverse dynamics model. The inverse dynamics model includes an *image module* and a *pseudo command module*. The *image module* consists of 8 convolutional layers with a kernel size of 5 in the first layer and 3 in the following layers. The number of channels increases from 32 to 256 from the first to the last convolutional layer. The convolutional layers are followed by two fully connected layers with 512 units each. ReLU nonlinearities are used for all hidden layers, and batch normalization is applied after convolutional layers. The *pseudo command module* contains three fully connected layers with 256 units each.

We train the inverse dynamics model $\mathcal{I}$ on the sampled dataset $\mathcal{D}_l$ using the Adam optimizer with an initial learning rate of 0.0002. Then, we use the trained inverse dynamics model $\mathcal{I}$ to generate the pseudo commands for $\mathcal{D}_u$, resulting in our noisy demonstration dataset $\tilde{\mathcal{D}}$ with a noise rate of $\epsilon = 1-\rho$. Given $\tilde{\mathcal{D}}$, we train CIL-USN to correct the commands before switching an action module for urban driving. We apply data augmentation for the dataset when training the inverse dynamics model and conditional imitation learning models, which follows the original CIL paper (Codevilla et al., 2018).

References

Abbeel, P., & Ng, A. Y. (2004). Apprenticeship learning via inverse reinforcement learning. In *Proceedings of the International Conference on Machine Learning*, p. 1.

Angluin, D., & Laird, P. (1988). Learning from noisy examples. *Machine Learning*, *2*(4), 343–370.

Argall, B. D., Chernova, S., Veloso, M., & Browning, B. (2009). A survey of robot learning from demonstration. *Robotics and Autonomous Systems*, *57*(5), 469–483.

Arpit, D., Jastrzębski, S., Ballas, N., Krueger, D., Bengio, E., Kanwal, M. S., Maharaj, T., Fischer, A., Courville, A., Bengio, Y., et al. (2017). A closer look at memorization in deep networks. In *Proceedings of the International Conference on Machine Learning*, pp. 233–242. PMLR.

Audiffren, J., Valko, M., Lazaric, A., & Ghavamzadeh, M. (2015). Maximum entropy semi-supervised inverse reinforcement learning. In *Proceedings of the International Joint Conference on Artificial Intelligence*.

Bain, M., & Sammut, C. (1999). A framework for behavioural cloning. In *Machine Intelligence 15, Intelligent Agents [St. Catherine's College, Oxford, July 1995]*, p. 103–129, GBR. Oxford University.

Bellemare, M. G., Naddaf, Y., Veness, J., & Bowling, M. (2013). The arcade learning environment: An evaluation platform for general agents. *Journal of Artificial Intelligence Research*, *47*, 253–279.

Blum, A., & Mitchell, T. (1998). Combining labeled and unlabeled data with co-training. In *Proceedings of the Annual Conference on Computational Learning Theory*, pp. 92–100.

Box, G. E., & Cox, D. R. (1964). An analysis of transformations. *Journal of the Royal Statistical Society: Series B (Methodological)*, *26*(2), 211–243.

Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., & Zaremba, W. (2016). Openai gym. *arXiv preprint arXiv:1606.01540*.

Brown, D. S., Goo, W., Nagarajan, P., & Niekum, S. (2019). Extrapolating beyond suboptimal demonstrations via inverse reinforcement learning from observations. In *Proceedings of the International Conference on Machine Learning*.

Burda, Y., Edwards, H., Pathak, D., Storkey, A., Darrell, T., & Efros, A. A. (2019). Large-scale study of curiosity-driven learning. In *Proceedings of the International Conference on Learning Representations*.

Cao, Z., & Sadigh, D. (2021). Learning from imperfect demonstrations from agents with varying dynamics. *IEEE Robotics and Automation Letters*, *6*(3), 5231–5238.

Codevilla, F., Müller, M., López, A., Koltun, V., & Dosovitskiy, A. (2018). End-to-end driving via conditional imitation learning. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pp. 4693–4700. IEEE.

Colas, C., Karch, T., Sigaud, O., & Oudeyer, P.-Y. (2022). Autotelic agents with intrinsically motivated goal-conditioned reinforcement learning: a short survey. *Journal of Artificial Intelligence Research*, *74*, 1159–1199.

Crespo, J., & Wichert, A. (2020). Reinforcement learning applied to games. *SN Applied Sciences*, *2*, 1–16.

Der Kiureghian, A., & Ditlevsen, O. (2009). Aleatory or epistemic? does it matter?. *Structural Safety*, *31*(2), 105–112.

Dosovitskiy, A., & Koltun, V. (2017). Learning to act by predicting the future. In *Proceedings of the International Conference on Learning Representations*.

Dosovitskiy, A., Ros, G., Codevilla, F., Lopez, A., & Koltun, V. (2017). Carla: An open urban driving simulator. In *Proceedings of the Conference on Robot Learning*, pp. 1–16. PMLR.

Filos, A., Tigkas, P., McAllister, R., Rhinehart, N., Levine, S., & Gal, Y. (2020). Can autonomous vehicles identify, recover from, and adapt to distribution shifts?. In *Proceedings of the International Conference on Machine Learning*, pp. 3145–3153. PMLR.

Fujimoto, S., Conti, E., Ghavamzadeh, M., & Pineau, J. (2019). Benchmarking batch deep reinforcement learning algorithms. In *Workshop on Deep Reinforcement Learning at NeurIPS 2019*.

Fujimoto, S., Hoof, H., & Meger, D. (2018). Addressing function approximation error in actor-critic methods. In *Proceedings of the International Conference on Machine Learning*, pp. 1587–1596. PMLR.

Fujimoto, S., Meger, D., & Precup, D. (2019). Off-policy deep reinforcement learning without exploration. In *Proceedings of the International Conference on Machine Learning*, pp. 2052–2062. PMLR.

Gal, Y., & Ghahramani, Z. (2016). Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *Proceedings of the International Conference on Machine Learning*, pp. 1050–1059. PMLR.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., & Bengio, Y. (2014). Generative adversarial nets. In *Advances in Neural Information Processing Systems*, Vol. 27.

Guo, C., Pleiss, G., Sun, Y., & Weinberger, K. Q. (2017). On calibration of modern neural networks. In *Proceedings of the International Conference on Machine Learning*, pp. 1321–1330. PMLR.

Gustafsson, F. K., Danelljan, M., & Schön, T. B. (2023). How reliable is your regression model's uncertainty under real-world distribution shifts?. *Transactions on Machine Learning Research*.

Hafner, D., Lillicrap, T., Fischer, I., Villegas, R., Ha, D., Lee, H., & Davidson, J. (2018). Learning latent dynamics for planning from pixels. In *Proceedings of the International Conference on Machine Learning*.

Han, B., Niu, G., Yu, X., Yao, Q., Xu, M., Tsang, I., & Sugiyama, M. (2020). Sigua: Forgetting may make learning with noisy labels more robust. In *Proceedings of the International Conference on Machine Learning*, pp. 4006–4016. PMLR.

Han, B., Yao, Q., Yu, X., Niu, G., Xu, M., Hu, W., Tsang, I., & Sugiyama, M. (2018). Co-teaching: Robust training of deep neural networks with extremely noisy labels. In *Advances in Neural Information Processing Systems*, Vol. 31.

Ho, J., & Ermon, S. (2016). Generative adversarial imitation learning. In *Advances in Neural Information Processing Systems*, pp. 4565–4573.

Hollenstein, J., Auddy, S., Saveriano, M., Renaudo, E., & Piater, J. (2022). Action noise in off-policy deep reinforcement learning: Impact on exploration and performance. *Transactions on Machine Learning Research*, *1*. Survey Certification.

Huang, Y., Cheng, Y., Bapna, A., Firat, O., Chen, D., Chen, M., Lee, H., Ngiam, J., Le, Q. V., Wu, Y., et al. (2019). Gpipe: Efficient training of giant neural networks using pipeline parallelism. In *Advances in Neural Information Processing Systems*, Vol. 32.

Hussein, A., Gaber, M. M., Elyan, E., & Jayne, C. (2017). Imitation learning: A survey of learning methods. *ACM Computing Surveys (CSUR)*, *50*(2), 1–35.

Kalai, A. T., & Servedio, R. A. (2005). Boosting in the presence of noise. *Journal of Computer and System Sciences*, *71*(3), 266–290.

Kendall, A., & Gal, Y. (2017). What uncertainties do we need in bayesian deep learning for computer vision?. In *Advances in Neural Information Processing Systems*, Vol. 30.

Kim, Y., Yim, J., Yun, J., & Kim, J. (2019). Nlnl: Negative learning for noisy labels. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 101–110.

Kopp, R. E. (1962). Pontryagin maximum principle. In *Mathematics in Science and Engineering*, Vol. 5, pp. 255–279. Elsevier.

Kotelevskii, N., Artemenkov, A., Fedyanin, K., Noskov, F., Fishkov, A., Petiushko, A., & Panov, M. (2022). Nuq: Nonparametric uncertainty quantification for deterministic neural networks. In *Advances in Neural Information Processing Systems*.

Kumar, A., Fu, J., Soh, M., Tucker, G., & Levine, S. (2019). Stabilizing off-policy q-learning via bootstrapping error reduction. In *Advances in Neural Information Processing Systems*, Vol. 32.

Lazaridis, A., Fachantidis, A., & Vlahavas, I. (2020). Deep reinforcement learning: A state-of-the-art walkthrough. *Journal of Artificial Intelligence Research*, *69*, 1421–1471.

Lee, K., Lee, K., Lee, H., & Shin, J. (2018). A simple unified framework for detecting out-of-distribution samples and adversarial attacks. In *Advances in Neural Information Processing Systems*, Vol. 31.

Lei, J., Zhang, Z., Zhang, L., & Li, X.-Y. (2022). Coca: Cost-effective collaborative annotation system by combining experts and amateurs. In *Proceedings of the International Conference on Data Engineering (ICDE)*, pp. 674–685. IEEE.

Li, G., Wang, J., Zheng, Y., & Franklin, M. J. (2016). Crowdsourced data management: A survey. *IEEE Transactions on Knowledge and Data Engineering*, *28*(9), 2296–2319.

Li, W., Dasarathy, G., & Berisha, V. (2020). Regularization via structural label smoothing. In *Proceedings of the International Conference on Artificial Intelligence and Statistics*, pp. 1453–1463. PMLR.

Li, Y., Song, J., & Ermon, S. (2017). Infogail: Interpretable imitation learning from visual demonstrations. In *Advances in Neural Information Processing Systems*, Vol. 30.

Lillicrap, T. P., Hunt, J. J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., & Wierstra, D. (2016). Continuous control with deep reinforcement learning. In *Proceedings of the International Conference on Learning Representations*.

Lin, X., Adams, S. C., & Beling, P. A. (2019). Multi-agent inverse reinforcement learning for certain general-sum stochastic games. *Journal of Artificial Intelligence Research*, *66*, 473–502.

Liu, J., Lin, Z., Padhy, S., Tran, D., Bedrax Weiss, T., & Lakshminarayanan, B. (2020). Simple and principled uncertainty estimation with deterministic deep learning via distance awareness. In *Advances in Neural Information Processing Systems*, Vol. 33, pp. 7498–7512.

Lukasik, M., Bhojanapalli, S., Menon, A., & Kumar, S. (2020). Does label smoothing mitigate label noise?. In *Proceedings of the International Conference on Machine Learning*, pp. 6448–6458. PMLR.

Malinin, A., & Gales, M. (2018). Predictive uncertainty estimation via prior networks. In *Advances in Neural Information Processing Systems*, Vol. 31.

Manwani, N., & Sastry, P. (2013). Noise tolerance under risk minimization. *IEEE Transactions on Cybernetics*, *43*(3), 1146–1151.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., et al. (2015). Human-level control through deep reinforcement learning. *Nature*, *518*(7540), 529.

Müller, R., Kornblith, S., & Hinton, G. E. (2019). When does label smoothing help?. In *Advances in Neural Information Processing Systems*, Vol. 32.

Naeini, M. P., Cooper, G., & Hauskrecht, M. (2015). Obtaining well calibrated probabilities using bayesian binning. In *Proceedings of the AAAI Conference on Artificial Intelligence*.

Natarajan, N., Dhillon, I. S., Ravikumar, P. K., & Tewari, A. (2013). Learning with noisy labels. In *Advances in Neural Information Processing Systems*, Vol. 26.

Nguyen, A., Yosinski, J., & Clune, J. (2015). Deep neural networks are easily fooled: High confidence predictions for unrecognizable images. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 427–436.

Nguyen, H., & La, H. (2019). Review of deep reinforcement learning for robot manipulation. In *Proceedings of the IEEE International Conference on Robotic Computing (IRC)*, pp. 590–595. IEEE.

Pathak, D., Agrawal, P., Efros, A. A., & Darrell, T. (2017). Curiosity-driven exploration by self-supervised prediction. In *Proceedings of the International Conference on Machine Learning*, pp. 2778–2787.

Patrini, G., Rozza, A., Krishna Menon, A., Nock, R., & Qu, L. (2017). Making deep neural networks robust to label noise: A loss correction approach. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1944–1952.

Peng, X. B., Kanazawa, A., Toyer, S., Abbeel, P., & Levine, S. (2019). Variational discriminator bottleneck: Improving imitation learning, inverse rl, and gans by constraining information flow. In *Proceedings of the International Conference on Learning Representations*.

Pereyra, G., Tucker, G., Chorowski, J., Kaiser, Ł., & Hinton, G. (2017). Regularizing neural networks by penalizing confident output distributions. In *Proceedings of the International Conference on Learning Representations*.

Pomerleau, D. A. (1988). Alvinn: An autonomous land vehicle in a neural network. In *Advances in Neural Information Processing Systems*, Vol. 1.

Real, E., Aggarwal, A., Huang, Y., & Le, Q. V. (2019). Regularized evolution for image classifier architecture search. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 33, pp. 4780–4789.

Ross, S., Gordon, G., & Bagnell, D. (2011). A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the International Conference on Artificial Intelligence and Statistics*, pp. 627–635.

Russell, S. (1998). Learning agents for uncertain environments. In *Proceedings of the Annual Conference on Computational Learning Theory*, pp. 101–103.

Schaal, S. (1999). Is imitation learning the route to humanoid robots?. *Trends in Cognitive Sciences*, *3*(6), 233–242.

Schulman, J., Moritz, P., Levine, S., Jordan, M., & Abbeel, P. (2016). High-dimensional continuous control using generalized advantage estimation. In *Proceedings of the International Conference on Learning Representations*.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., & Klimov, O. (2017). Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*.

Silver, D., Bagnell, J. A., & Stentz, A. (2013). Learning autonomous driving styles and maneuvers from expert demonstration. In *Experimental Robotics*, pp. 371–386. Springer.

Smyth, P., Fayyad, U., Burl, M., Perona, P., & Baldi, P. (1994). Inferring ground truth from subjective labelling of venus images. In *Advances in Neural Information Processing Systems*, Vol. 7.

Sutton, R. S., & Barto, A. G. (2018). *Reinforcement learning: An introduction (2nd Edition)*. MIT press.

Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., & Wojna, Z. (2016). Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2818–2826.

Tangkaratt, V., Charoenphakdee, N., & Sugiyama, M. (2020a). Robust imitation learning from noisy demonstrations. In *Proceedings of the International Conference on Artificial Intelligence and Statistics*, pp. 298–306. PMLR.

Tangkaratt, V., Han, B., Khan, M. E., & Sugiyama, M. (2020b). Variational imitation learning with diverse-quality demonstrations. In III, H. D., & Singh, A. (Eds.), *Proceedings of the International Conference on Machine Learning*, Vol. 119, pp. 9407–9417. PMLR.

Tohme, T., Vanslette, K., & Youcef-Toumi, K. (2023). Reliable neural networks for regression uncertainty estimation. *Reliability Engineering & System Safety*, *229*, 108811.

Tucker, A., Gleave, A., & Russell, S. (2018). Inverse reinforcement learning for video games. In *Workshop on Deep Reinforcement Learning at NeurIPS 2018*.

Van Hasselt, H., Guez, A., & Silver, D. (2016). Deep reinforcement learning with double q-learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 30.

Van Rooyen, B., Menon, A., & Williamson, R. C. (2015). Learning with symmetric label noise: The importance of being unhinged. In *Advances in Neural Information Processing Systems*, Vol. 28.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., & Polosukhin, I. (2017). Attention is all you need. In *Advances in Neural Information Processing Systems*, Vol. 30.

Wang, Y., Xu, C., & Du, B. (2021a). Robust adversarial imitation learning via adaptively-selected demonstrations.. In *Proceedings of the International Joint Conference on Artificial Intelligence*, pp. 3155–3161.

Wang, Y., Xu, C., Du, B., & Lee, H. (2021b). Learning to weight imperfect demonstrations. In *Proceedings of the International Conference on Machine Learning*, pp. 10961–10970. PMLR.

Watkins, C. J. C. H. (1989). *Learning from delayed rewards*. Phd thesis, King's College, Cambridge United Kingdom.

Wei, H., Feng, L., Chen, X., & An, B. (2020). Combating noisy labels by agreement: A joint training method with co-regularization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 13726–13735.

Wu, Y., Zhai, S., Srivastava, N., Susskind, J., Zhang, J., Salakhutdinov, R., & Goh, H. (2021). Uncertainty weighted actor-critic for offline reinforcement learning. In *Proceedings of the International Conference on Machine Learning*, pp. 11319–11328.

Wu, Y.-H., Charoenphakdee, N., Bao, H., Tangkaratt, V., & Sugiyama, M. (2019). Imitation learning from imperfect demonstration. In *Proceedings of the International Conference on Machine Learning*, pp. 6818–6827. PMLR.

Xia, X., Liu, T., Han, B., Gong, M., Yu, J., Niu, G., & Sugiyama, M. (2022). Sample selection with uncertainty of losses for learning with noisy labels. In *Proceedings of the International Conference on Learning Representations*.

Xia, X., Liu, T., Han, B., Wang, N., Gong, M., Liu, H., Niu, G., Tao, D., & Sugiyama, M. (2020). Part-dependent label noise: Towards instance-dependent label noise. In *Advances in Neural Information Processing Systems*, Vol. 33, pp. 7597–7610.

Xiao, H., Herman, M., Wagner, J., Ziesche, S., Etesami, J., & Linh, T. H. (2019). Wasserstein adversarial imitation learning. *arXiv preprint arXiv:1906.08113*.

Yang, S., Nachum, O., Du, Y., Wei, J., Abbeel, P., & Schuurmans, D. (2023). Foundation models for decision making: Problems, methods, and opportunities. *arXiv preprint arXiv:2303.04129*.

Yu, X., Han, B., Yao, J., Niu, G., Tsang, I., & Sugiyama, M. (2019). How does disagreement help generalization against label corruption?. In *Proceedings of the International Conference on Machine Learning*, pp. 7164–7173. PMLR.

Yuan, L., Tay, F. E., Li, G., Wang, T., & Feng, J. (2020). Revisiting knowledge distillation via label smoothing regularization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3903–3911.

Zhang, C.-B., Jiang, P.-T., Hou, Q., Wei, Y., Han, Q., Li, Z., & Cheng, M.-M. (2021a). Delving deep into label smoothing. *IEEE Transactions on Image Processing*, *30*, 5984–5996.

Zhang, S., Cao, Z., Sadigh, D., & Sui, Y. (2021b). Confidence-aware imitation learning from demonstrations with varying optimality. In *Advances in Neural Information Processing Systems*, Vol. 34.

Zhang, Z., & Sabuncu, M. R. (2018). Generalized cross entropy loss for training deep neural networks with noisy labels. In *Advances in Neural Information Processing Systems*.

Zoph, B., Vasudevan, V., Shlens, J., & Le, Q. V. (2018). Learning transferable architectures for scalable image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 8697–8710.