EDGE: Explaining Deep Reinforcement Learning Policies

Wenbo Guo
The Pennsylvania State University
wzg13@ist.psu.edu

Xian Wu*
The Pennsylvania State University
xkw5132@psu.edu

Usmann Khan*
Georgia Institute of Technology
ukhan35@gatech.edu

Xinyu Xing
Northwestern University
The Pennsylvania State University
xinyu.xing@northwestern.edu

Abstract

With the rapid development of deep reinforcement learning (DRL) techniques, there is an increasing need to understand and interpret DRL policies. While recent research has developed explanation methods to interpret how an agent determines its moves, they cannot capture the importance of actions/states to a game's final result. In this work, we propose a novel self-explainable model that augments a Gaussian process with a customized kernel function and an interpretable predictor. Together with the proposed model, we also develop a parameter learning procedure that leverages inducing points and variational inference to improve learning efficiency. Using our proposed model, we can predict an agent's final rewards from its game episodes and extract time step importance within episodes as strategy-level explanations for that agent. Through experiments on Atari and MuJoCo games, we verify the explanation fidelity of our method and demonstrate how to employ interpretation to understand agent behavior, discover policy vulnerabilities, remediate policy errors, and even defend against adversarial attacks.

1 Introduction

Deep reinforcement learning has shown great success in automatic policy learning for various sequential decision-making problems, such as training AI agents to defeat professional players in sophisticated games [74, 65, 24, 37] and controlling robots to accomplish complicated tasks [33, 38]. However, existing DRL agents make decisions in an opaque fashion, taking actions without accompanying explanations. This lack of transparency creates key barriers to establishing trust in an agent's policy and scrutinizing policy weakness. This issue significantly limits the applicability of DRL techniques in critical application fields (*e.g.,* finance [47] and self-driving cars [11]).

To tackle this limitation, prior research (*e.g.,* [9, 13, 73]) proposes to derive an explanation for a target agent's action at a specific time step. Technically, this explanation can be obtained by pinpointing the features within the agent's observation of a particular state that contribute most to its corresponding action at that state. Despite demonstrating great potential to help users understand a target agent's

*Equal Contribution.

35th Conference on Neural Information Processing Systems (NeurIPS 2021), Sydney, Australia.

individual actions, they lack the capability to extract insights into the overall policy of that agent. In other words, existing methods cannot shed light on the general sensitivity of an agent's final reward from a game in regards to the actions/states in that game episode. Consequently, these methods fall short in troubleshooting an agent's policy's weaknesses when it fails its task.

We propose a novel explanation method to derive strategy-level interpretations of a DRL agent. As we discuss later in Section 3, we define such interpretations as the identification of critical time steps contributing to a target agent's final reward from each game episode. At a high level, our method identifies the important time steps by approximating the target agent's decision-making process with a self-explainable model and extracting the explanations from this model. Specifically, given a well-trained DRL agent, our method first collects a set of episodes and the corresponding final rewards of this agent. Then, it fits a self-explainable model to predict final rewards from game episodes. To model the unique correlations in DRL episodes and enable high-fidelity explanations, rather than simply applying off-the-shelf self-explanation techniques, we develop a novel self-explainable model that integrates a series of new designs. First, we augment a Gaussian Process (GP) with a customized deep additive kernel to capture not only correlations between time steps but, more importantly, the joint effect across episodes. Second, we combine this deep GP model with our newly designed explainable prediction model to predict the final reward and extract the time step importance. Third, we develop an efficient inference and learning framework for our model by leveraging inducing points and variational inference. We refer to our method as "Strategy-level **E**xplanation of **D**rl a**GE**nts" (for short `EDGE`). [2]

With extensive experiments on three representative RL games, we demonstrate that `EDGE` outperforms alternative interpretation methods in terms of explanation fidelity. Additionally, we demonstrate how DRL policy users and developers can benefit from `EDGE`. Specifically, we first show that `EDGE` could help understand the agent's behavior and establish trust in its policy. Second, we demonstrate that guided by the insights revealed from our explanations, an attacker could launch efficient adversarial attacks to cause a target agent to fail. Third, we demonstrate how, with `EDGE`'s capability, a model developer could explain why a target agent makes mistakes. This allows the developer to explore a remediation policy following the explanations and using it to enhance the agent's original policy. Finally, we illustrate that `EDGE` could help develop a defense mechanism against a newly emerging adversarial attack on DRL agents. To the best of our knowledge, this is the first work that interprets a DRL agent's policy by identifying the most critical time steps to the agent's final reward in each episode. This is also the first work that demonstrates how to use an explanation to understand agent behavior, discover policy vulnerabilities, patch policy errors, and robustify victim policies.

2 Related Work

Past research on DRL explanation primarily focuses on associating an agent's action with its observation at a particular time step (*i.e.,* pinpointing the features most critical to the agent's action at a specific time). Technically, these methods can be summarized in the following categories.

- **Post-training explanation** is a method that utilizes and extends post-training interpretation approaches (*e.g.,* [56, 28, 36, 35]) to derive explanation from a DRL agent's policy/value network and thus treat it as the interpretation for that DRL agent (*e.g.,* [9, 44, 32, 68, 20, 72]).
- **Model approximation** is an approach that employs a self-interpretable model to mimic the target agent's policy networks and then derives explanation from the self-interpretable model for the target DRL agent (*e.g.,* [13, 22, 55, 14, 59, 58, 87, 85]).
- **Self-interpretable modeling** is an approach different from the model approximation techniques above. Instead of mimicking the target agent's policy network, self-interpretable modeling builds a self-explainable model to replace the policy network. Since the new model is interpretable, one can easily derive an explanation for the target agent (*e.g.,* [92, 62, 82, 42]).
- **Reward decomposition** is a method that re-engineers a DRL agent's reward function to make the reward gained at each time step more meaningful and explainable. With the more meaningful reward in hand, at each time step, one could use the instant reward gain to interpret the agent's action (*e.g.,* [73, 46, 54]).

[2] The source code of `EDGE` can be found in `https://github.com/Henrygwb/edge`.

From the objective perspective, our work is fundamentally different from the above DRL explanation research. Rather than pinpointing the features – in an observation – critical for an agent's action, our work identifies the critical time steps contributing to the agent's final reward. Using our explanation, one can better understand the agent's policy, unveil policy weakness, and patch policy errors (as shown in Section 5). In Supplement S7, we further conduct a user study to demonstrate the superiority of our method against the above explanation approaches in pinpointing good policies and performing policy forensics. Note that there are two other methods that also understand a DRL policy through the agent's previous memories [49, 23]. These works are fundamentally different from ours in two perspectives. First, both methods have a different explanation goals from our work. Specifically, Koul *et al.* [49] focuses on identifying whether the action at each time step depends more on the current observation or the previous states. The method proposed in [23] pinpoints the important steps w.r.t. the subsequent transitions in the FSM extracted from the target agent rather than the final result of an episode. Second, both methods can be applied only to white-box RNN policies, whereas our method is applicable to DRL policies with arbitrary network structures.

3 Key Technique

3.1 Problem Setup

Consider a DRL game with an agent trained with Q-learning [86, 60] or policy gradient [48, 69, 70]. Our work aims to explain this agent's policy by identifying the important steps contributing most to a game episode's final reward/result. To ensure practicability, we allow access only to the environment states, agent's actions, and rewards. We assume the availability of neither the value/Q function nor the policy network. Formally, given N episodes $\mathcal{T} = \{\mathbf{X}^{(i)}, y_i\}_{i=1:N}$ of the target agent, $\mathbf{X}^{(i)} = \{\mathbf{s}_t^{(i)}, \mathbf{a}_t^{(i)}\}_{t=1:T}$ is the $i-$th episode with the length T, where $\mathbf{s}_t^{(i)} \in \mathbb{R}^{d_s}$ and $\mathbf{a}_t^{(i)} \in \mathbb{R}^{d_a}$ are the state and action at the time step t. y_i is the final reward of this episode.[3] Our goal is to highlight the top-K important time steps within each episode $\mathbf{X}^{(i)}$.

Possible Solutions and Limitations. The most straightforward approach of identifying important time steps is to use the output of the value/Q network as indicators and pinpoint the time steps with the top K highest value/Q function's outputs as the top K critical steps. However, since we do not assume the availability of these networks, this method is not applicable to our problem. A more realistic method is to fit a seq2one model (*i.e.,* RNN) that takes as input the state-action pairs in an episode and predicts the final reward of that episode. With this prediction model, one could utilize a post-training explanation method to derive the time step importance. However, existing post-training explanation techniques usually require approximating the target DNN with more transparent models, which inevitably introduces errors. Additionally, many post-training methods are vulnerable to adversarial attacks [61, 30, 94] or generate model-independent explanations [1, 64, 2, 78]. As we will show later in Section 4 and Supplement S3&S5, these limitations jeopardize the post-training explanations' fidelity. A more promising direction is to fit a self-explainable model to predict the final reward. Existing research has proposed a variety of self-explanation methods. Most of them do not apply to our problem because they either cannot derive feature importance as explanations [5, 52, 50, 18], cannot be applied to sequential data [21, 27, 12], or require explanation ground truth [16, 66]. In this work, we consider two self-explainable models that are designed to fit and explain sequential data – an RNN augmented with attention [10, 39, 34] and rationale net [51]. Technically, both models have the form of $g(\theta(\mathbf{x}) \odot \mathbf{x})$, where $\theta(\cdot)$ is a weight RNN or an attention layer and $g(\cdot)$ is the prediction RNN. The output of $\theta(\cdot)$ can be used to identify the important steps in the input sequence. Despite extracting meaningful explanations, recent research [45, 89, 17] reveals that the explanations given by $\theta(\cdot)$ cannot faithfully reflect the associations (*i.e.,* feature importance) learned by the subsequent prediction model $g(\cdot)$, leading to an even lower fidelity than the post-training explanations in some applications. Additionally, these models are not designed to explain an RL agent and cannot fully capture the dependencies within the episodes of that agent. Specifically, the episodes collected from the same agent tend to exhibit two types of dependencies: dependency between the time steps within an episode and the dependency across different episodes. Although they consider the dependency

[3] For the games with delayed rewards, such as MuJoCo [84] and Atari Pong [8], where a non-zero reward r_T is assigned only to the last step of a game, we use r_T as y_i. For the games with instant rewards (*e.g.,* OpenAI CartPole [15]), we compute an episode's total reward as y_i, i.e., $y_i = \sum_t r_t$.

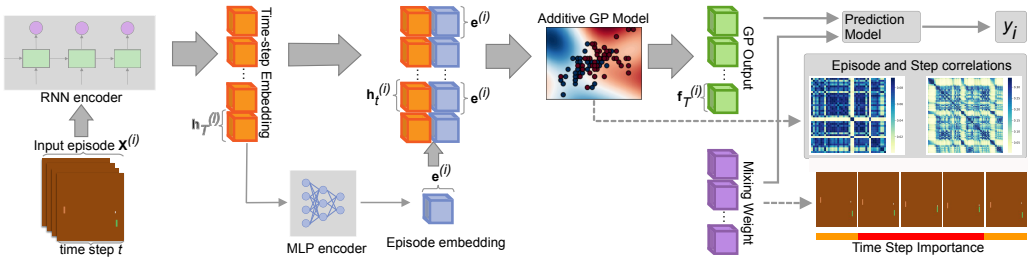


Figure 1: Overview of EDGE with a constant prediction mixing weight.

within each input sequence, these methods cannot capture the correlations between different inputs. As is shown in Section 4 and Supplement S3&S5, this also jeopardizes their explanation fidelity.

3.2 Explanation Model Design of EDGE

In this work, we design a novel self-explainable model by adopting a very different design than existing methods. First, to better capture the associations (*i.e.,* feature importance) learned by the prediction model, we add the explainable module to the final layer rather than the input layer of the prediction model. Formally, our model can be written as $g(f(\mathbf{x}))$, where $f(\cdot)$ is a feature extractor and $g(\cdot)$ is an explainable prediction model. Second, we design a deep Gaussian Process as the feature extractor to capture the correlations between time steps and those across different episodes, which are often exhibited in a set of episodes collected from the same DRL agent. In addition to capturing different levels of correlations, another advantage of deep GPs over typical DNNs is that GPs model the joint distribution of the output signals, enabling access to the output signals' uncertainty. Finally, we design an interpretable Bayesian prediction model to infer the distribution of final rewards and deliver time step importance. Below, we first give an overview of our proposed model. Then, we describe how to adapt the traditional GP model to our problem, followed by the design of the final prediction model.

Overview. As shown in Fig 1, given an episode of the target agent $\mathbf{X}^{(i)}$, EDGE first inputs it into a RNN encoder, which outputs the embedding of each time step in this episode $\{\mathbf{h}_t^{(i)}\}_{t=1:T}$. EDGE also passes the last step's embedding through a shallow MLP to obtain an episode embedding $\mathbf{e}^{(i)}$. Then, EDGE adopts our proposed additive GP framework to process $\{\mathbf{h}_t^{(i)}\}_{t=1:T}$ and $\mathbf{e}^{(i)}$ and obtains a latent representation of the whole episode $\mathbf{f}_{1:T}^{(i)}$. As introduced later, this representation is able to capture the correlations between time steps and those across episodes. Finally, EDGE inputs $\mathbf{f}_{1:T}^{(i)}$ into our prediction model $\mathbf{f}_{1:T}^{(i)}$ and get the predicted final reward of the input episode. As detailed later, our prediction model is designed based on a linear regression, whose regression coefficient can be used to identify important time steps within in the input episode.

Additive GP with Deep Recurrent Kernels. Gaussian Process defines a distribution over an infinite collection of random variables, such that any finite subset of variables follows a multivariate Gaussian distribution [63]. In Statistical modeling, GP defines the prior of a non-parametric function $f : \mathcal{X} \to \mathbb{R}$. Formally, if f has a GP prior, i.e., $f \sim \mathcal{GP}(0, k_\gamma)$,where $k_\gamma(\cdot,\cdot)$ is a positive semi-definite kernel function parameterized by γ, any finite collections of $\mathbf{f} \in \mathbb{R}^N$ follows a multivariate Gaussian distribution $(\mathbf{f}|\mathbf{X}) \sim \mathcal{N}(\mathbf{0}, K_{XX})$. Here, $K_{XX} \in \mathbb{R}^{N\times N}$ is the covariance matrix, with $(K_{XX})_{ij} = k_\gamma(\mathbf{x}_i, \mathbf{x}_j)$. In our model, we adopt the widely applied square exponential (SE) kernel function: $k_\gamma(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(-\frac{1}{2}(\mathbf{x}_i - \mathbf{x}_j)^T\Theta_k(\mathbf{x}_i - \mathbf{x}_j)\right)$, with $\gamma = \Theta_k$. Traditional GP with SE kernel [63] assumes the input space is Euclidean, which is usually invalid for real-world data with high-dimensional inputs [3]. To tackle this challenge, recent research [91, 53] proposes to conduct dimensional reduction via a DNN and then apply a GP to the DNN's latent space. They show that the resultant deep kernel models achieve similar performance to DNNs on complicated datasets.

In our model we capture the sequential dependency within an episode by using an RNN as the deep net inside the kernel function. Specifically, given an episode $\mathbf{X}^{(i)}$, we first concatenate the state and action (*i.e.,* $\mathbf{x}_t^{(i)} = [\mathbf{s}_t^{(i)}, \mathbf{a}_t^{(i)}]$), input them into an RNN h_ϕ, and obtain the latent representation of this episode: $\{\mathbf{h}_t^{(i)}\}_{t=1:T}$, where $\mathbf{h}_t^{(i)} \in \mathbb{R}^q$ is the state-action embedding at the time t. We also

compute an episode embedding by passing the last step's hidden representation through a shallow MLP $e_{\phi_1} : \mathbf{h}_T^{(i)} \to \mathbf{e}^{(i)} \in \mathbb{R}^q$. After obtaining $\{\mathbf{h}_t^{(i)}\}_{t=1:T}$ and $\mathbf{e}^{(i)}$, we then adopt the additive GP framework to capture the correlations between time steps and those across episodes. Formally, an additive GP is the weighted sum of J independent GPs, i.e., $f = \sum_J \alpha_j f_j$. Here, $f_j \sim \mathcal{GP}(0, k_j)$ is the j-th GP component, in which the covariance function k_j is typically applied to a subset of input features. By assigning every component a GP prior, one can ensure that the mixed-signal f also follows a GP prior [25]. Following this framework, we construct our deep GP model as the sum of two components f_t and f_e. Specifically, $f_t \sim \mathcal{GP}(0, k_{\gamma_t})$ models the correlations between time-steps, where the covariance between the t-th steps in episode i and the k-th steps in episode j can be computed by $k_{\gamma_t}(\mathbf{h}_t^{(i)}, \mathbf{h}_k^{(j)})$. Going beyond modeling the correlations between individual steps, $f_e \sim \mathcal{GP}(0, k_{\gamma_e})$ captures a higher level cluster structures within the collected episodes, i.e., the similarity between episodes. Formally, the episode-level covariance between any pair of time steps in episode i and j is given by $k_{\gamma_e}(\mathbf{e}^{(i)}, \mathbf{e}^{(j)})$. Finally, our deep additive GP model can be expressed as: $f = \alpha_t f_t + \alpha_e f_e$, where α_t and α_e are the component weights. Given a set of collected episodes represented by $\mathbf{T} \in \mathbb{R}^{N \times T \times (d_s + d_a)}$, $\mathbf{f} \in \mathbb{R}^{NT}$ is given by: $\mathbf{f}|\mathbf{X} \sim \mathcal{N}(\mathbf{0}, k = \alpha_t^2 k_{\gamma_t} + \alpha_e^2 k_{\gamma_e})$, where $\mathbf{X} \in \mathbb{R}^{NT \times (d_s + d_a)}$ is the flattened matrix of $\mathbf{T}$.

Prediction Model. To ensure explanability, we use a linear regression as the base of our prediction model, where the regression coefficients reflect the importance of each input entity. Specifically, we first convert the flattened response $\mathbf{f}$ back to the matrix form $\mathbf{F} \in \mathbb{R}^{N \times T}$, where the i-th row $\mathbf{F}^{(i)} \in \mathbb{R}^T$ is the i-th episode's encoding given by our GP model. Then, we define the conditional likelihood for the discrete and continuous final reward, respectively. When y_i is continuous, we follow the typical GP regression model [63] and define the $y_i = \mathbf{F}^{(i)} \mathbf{w}^T + \epsilon_1$, where $\mathbf{w} \in \mathbb{R}^{1 \times T}$ is the mixing weight and $\epsilon_1 \sim \mathcal{N}(0, \sigma^2)$ is the observation noise. The conditional likelihood distribution is $y_i|\mathbf{F}^{(i)} \sim \mathcal{N}(\mathbf{F}^{(i)} \mathbf{w}^T, \sigma^2)$. For the discrete final reward with a finite number of possible values, we use the softmax prediction model to perform classification. Formally, we define $y_i|\mathbf{F}^{(i)}$ follows a categorical distribution with $p(y_i = k|\mathbf{F}^{(i)}) = \frac{\exp((\mathbf{F}^{(i)} \mathbf{W}^T)_k)}{\sum_k \exp((\mathbf{F}^{(i)} \mathbf{W}^T)_k)}$. $\mathbf{W} \in \mathbb{R}^{K \times T}$ is the mixing weight, where K is the total number of classes. Finally, we combine all the components together and write our explanation model as (A illustration of our proposed model can be found in Fig. 1.): [4]

$$\mathbf{f}|\mathbf{X} \sim \mathcal{N}(\mathbf{0}, k = \alpha_t^2 k_{\gamma_t} + \alpha_e^2 k_{\gamma_e}), \quad y_i|\mathbf{F}^{(i)} \sim \begin{cases} \text{Cal}(\text{softmax}(\mathbf{F}^{(i)} \mathbf{W}^T)), & \text{If conducting classification} \\ \mathcal{N}(\mathbf{F}^{(i)} \mathbf{w}^T, \sigma^2), & \text{otherwise} \end{cases}, \tag{1}$$

where the mixing weight is constant. This indicates the time step importance derived from the mixing weight is a global explanation. [5] According to the insight that time steps with a high correlation tend to have a joint effect (similar importance) on the game result, we could combine the global explanation with the time step correlations in $K_t(\mathbf{X}, \mathbf{X})$ to gain a fine-grained understanding of each game. Specifically, given an episode and the top important steps indicated by the mixing weight, we can identify the time steps that are highly correlated to these globally important steps and treat them together as the local explanation of that episode. Supplement S1 introduces another way of deriving episode-specific explanations by replacing the constant mixing weight with a weight obtained by a simple DNN. Note that the episode correlations in $K_e(\mathbf{X}, \mathbf{X})$ reveal the cluster structure within a set of episodes, which helps categorize the explanations of similar episodes.

3.3 Posterior Inference and Parameter Learning

Sparse GP with Inducing Points. Direct inference of our model requires computing $(K_{XX} + \sigma^2 I)^{-1}$ over K_{XX}, which incurs $\mathcal{O}(NT^3)$ computational complexity. This cubic complexity restricts our model to only small datasets. To improve scalability, we adopt the inducing points method [91] for inference and learning. At a high level, this method simplifies the posterior computation by reducing the effective number of samples in $\mathbf{X}$ from NT to M, where M is the number of inducing points. Specifically, we define each inducing point at the latent space as $\mathbf{z}_i \in R^{2q}$, and $\mathbf{u}_i$ as the GP output

[4] Note that our model is similar to existing GP-based state-space models [4, 71, 29, 81, 26] in that both use an RNN inside the kernel function. However, these models do not integrate an additive GP. More importantly, their prediction models are not designed for explanation purposes and thus cannot derive time step importance.

[5] The classification model gives K global explanations, one for each class derived from each row of $\mathbf{W}$.

of $\mathbf{z}_i$. Then, the joint prior of $\mathbf{f}$ and $\mathbf{u}$ and the conditional prior $\mathbf{f}|\mathbf{u}$ are given by:

$$\mathbf{f}, \mathbf{u}|\mathbf{X}, \mathbf{Z} \sim \mathcal{N}\left(\begin{bmatrix}\mathbf{0}\\ \mathbf{0}\end{bmatrix}, \begin{bmatrix}K_{XX} & K_{XZ}\\ K_{XZ}^T & K_{ZZ}\end{bmatrix}\right), \mathbf{f}|\mathbf{u}, \mathbf{X}, \mathbf{Z} \sim \mathcal{N}(K_{XZ}K_{ZZ}^{-1}\mathbf{u}, K_{XX} - K_{XZ}K_{ZZ}^{-1}K_{XZ}^T), \quad (2)$$

where K_{XX}, K_{XZ}, K_{ZZ} are the covariance matrices. They can be computed by applying our additive kernel function to the time-step and episode embedding of the training episodes and inducing points. As is shown in Eqn (2), with inducing points, we only need to compute the inverse of K_{ZZ}, which significantly reduces the computational cost from $\mathcal{O}(NT^3)$ to $\mathcal{O}(m^3)$.

Variational Inference and Learning. So far, our model has introduced the following parameters: neural encoder parameters $\Theta_n = \{\phi, \phi_1\}$, GP parameters $\Theta_k = \{\gamma_t, \gamma_e, \alpha_e, \alpha_t\}$, prediction model parameters $\Theta_p = \{\mathbf{w}/\mathbf{W}, \sigma^2\}$, and inducing points $\mathbf{Z} = \{\mathbf{z}_i\}_{i=1:M}$. To learn these parameters, we follow the idea of empirical Bayes [63] and maximize the log marginal likelihood $\log p(y|\mathbf{X}, \mathbf{Z}, \Theta_n, \Theta_k, \Theta_p)$. Maximizing this log marginal likelihood is computationally expensive and, more important, intractable for models with non-Gaussian likelihood. To provide factorized approximation to marginal likelihood and enable efficient learning, we assume a variational posterior over the inducing variable $q(\mathbf{u}) \sim \mathcal{N}(\mu, \Sigma)$ and a factorized joint posterior $q(\mathbf{f}, \mathbf{u}) = q(\mathbf{u})p(\mathbf{f}|\mathbf{u})$, where $p(\mathbf{f}|\mathbf{u})$ is the conditional prior in Eqn. (2). By Jensen's inequality, we can derive the evidence lower bound (ELBO):

$$\log p(y|\mathbf{X}, \mathbf{Z}, \Theta_n, \Theta_k, \Theta_p) \geq \mathbb{E}_{q(\mathbf{f})}[\log p(y|\mathbf{f})] - \mathrm{KL}[q(\mathbf{u})||p(\mathbf{u})], \quad (3)$$

where the first part is the likelihood term. The second KL term penalizes the difference between the approximated posterior $q(\mathbf{u})$ and the prior $p(\mathbf{u})$. Maximizing the ELBO in Eqn. (3) will automatically maximize the marginal likelihood, which is also equivalent to minimizing the KL divergence from the variational joint posterior to the true posterior (See Supplement S1 for more details).

When conducting classification, the categorical likelihood makes the likelihood term in Eqn. (3) intractable. To tackle this challenge, we first compute the marginal variational posterior distribution of $\mathbf{f}$, denoted as $q(\mathbf{f}) = \mathcal{N}(\mu_f, \Sigma_f)$ (See Supplement S1 for detailed computations). Then, we apply the reparameterization trick [57] to $q(\mathbf{f})$. Formally, we define $\mathbf{f} = v(\epsilon_f) = \mu_f + \mathbf{L}_f\epsilon_f$, with $\epsilon_f \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ and $\mathbf{L}_f\mathbf{L}_f^T = \Sigma_f$. With this reparameterization, we can sample from the standard Gaussian distribution and approximate the likelihood term with Monte Carlo (MC) method [83]:

$$\mathbb{E}_{q(\mathbf{f})}[\log p(y|\mathbf{f})] = \mathbb{E}_{p(\epsilon_f)}[\log p(y|v(\epsilon_f))] \approx \frac{1}{B}\sum_b \sum_i \log p(y_i|(\mathbf{F}^{(i)})^{(b)}), \quad (4)$$

where B is the number of MC samples. For the regression model, we directly compute the analytical form of likelihood term and use it for parameter learning (See derivation in Supplement S1). With the above approximations, our model parameters (*i.e.,* Θ_n, Θ_k, Θ_p, $\mathbf{Z}$, and $\{\mu, \Sigma\}$) can be efficiently learned by maximizing the (approximated) ELBO using a stochastic gradient descent method. Implementation details and hyper-parameter choices can be found in Supplement S2.

4 Evaluation

In this section, we evaluate EDGE on three representative RL games (all with delayed rewards) – Pong in Atari, You-Shall-Not-Pass in MuJoCo, and Kick-And-Defend in MuJoCo. Supplement S5 further demonstrates the effectiveness of our method on two OpenAI GYM games (both with instant rewards). For each game, we used a well-trained agent as our target agent (See Supplement S2 for more details about these agents).

Baseline Selection. Recall our goal is to take as input the episode of a target agent and identify the steps critical for the agent's final reward. As is discussed in Section 3.1, to do this, there are two categories of alternative approaches – ❶ fitting an episode through a non-interpretable model and then deriving explanation from that model and ❷ fitting an episode through a self-explainable model and then obtaining interpretation directly from its interpretation component. In this section, we select some representative alternative methods as our baseline and compare them with our proposed method. Below, we briefly describe these baseline approaches and discuss the rationale behind our choice.

With respect to the first type of alternative approaches, we first utilize the RNN structure proposed in [43] to fit the reward prediction model. Then, we apply various gradient-based saliency methods on the RNN model and thus derive interpretation accordingly. We implement three widely used saliency

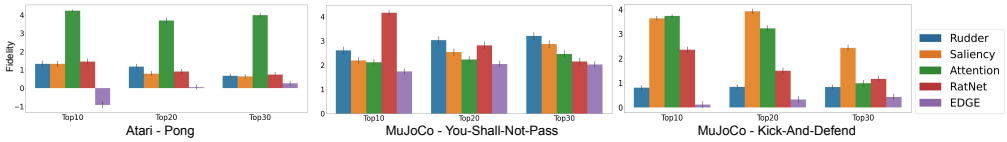


Figure 2: Mean and standard error of the fidelity scores obtained by each explanation method. The x-axis represents the different choices of K. "RatNet" stands for Rationale Net. For our method, we use the global explanations derived from the mixing weight in this evaluation.

methods – Vanilla gradient [76], integrated gradient [80], and SmoothGrad [77] – as well as their variants (ExpGrad [79], VarGrad [40], and integrated gradient with uniform baseline [79]).[6] When comparing RNN+saliency method with our proposed approach, we choose the RNN's interpretation from the saliency method with the best explanation fidelity. For the fidelity comparison between each saliency method, the readers could refer to Supplement S3. In addition to the RNN+saliency method, another method falling into the first kind of alternative approaches is Rudder [7]. Technically speaking, this method also learns an RNN model to predict an agent's final reward. Differently, it derives explanation from decomposed final reward.

Regarding the second kind of alternative approaches, we choose Attention RNN and Rationale Net. Attention RNN [10] is typically treated as a self-interpretable model. From the model's attention layer, one could extract its output and use it as the important scores for the input dimensions. We use these important scores to pinpoint the critical time steps in the input episode. Similar to Attention RNN, Rationale Net is also self-interpretable. In our experiments we use Rationale Net's original model structure [51] rather than the improved model structure proposed in [17]. This is because, going beyond training data, the improved model training requires additional information, which is unavailable for our problem.

Evaluation Metric. An intuitive method to evaluate the fidelity of the various approaches' explanations is to vary the actions at the time steps critical for the final reward and then measure the reward difference before and after the action manipulation. However, this method invalidates the physical realistic property of an episode because the change of an agent's action at a specific time step would inevitably influence its consecutive actions and the state transitions. To address this problem, we introduce a physically realistic method to manipulate episodes. Then, we introduce a new metric to quantify the fidelity of interpretation.

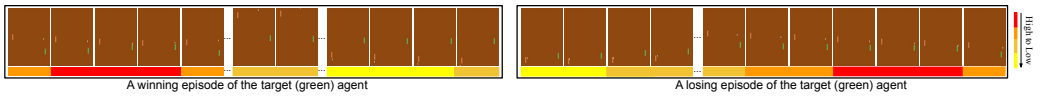
Given the explanation of the i-th episode – $\mathbf{E}_i$, we first identify the top-K important time steps from $\mathbf{E}_i$. From the top-K time steps, we then extract the longest sequence (*i.e.,* the longest continuous time steps), record its length – l, and treat its elements as the time steps most critical to y_i.

To evaluate and compare the fidelity of the interpretation (*i.e.,* the most critical time steps extracted through different interpretation methods), we first replay the actions recorded on that episode to the time step indicated by the longest sequence. Starting from the beginning of the longest continuous time steps to its end (*i.e.,* $t_i \cdots t_{i+l}$), we replace the corresponding actions at these time steps with random actions. [7] Following the action replacement, we pass the state at t_{i+l+1} to the agent's policy. Starting from t_{i+l+1}, we then use the agent's policy to complete the game, gather the final reward, and compute the final reward difference before/after replaying denoted as d. After computing l and d, we define the fidelity score of $\mathbf{E}_i$ as $\log(p_l) - \log(p_d)$. Here, $p_l = l/T$ is the length of the longest sequence normalized by the total length of the episode - T. $p_d = |d|/d_{\text{max}}$ is the absolute reward difference normalized by the maximum absolute reward difference of the game. When the value of the fidelity score $\log(p_l) - \log(p_d)$ is low, it indicates $\mathbf{E}_i$ is illustrated by a short length of sequence. By varying the actions pertaining to this short sequence, we can observe a great change in the agent's final reward. As such, a low score implies high fidelity of an interpretation method.

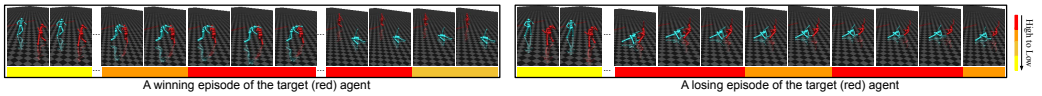
Result. Fig. 2 shows the comparison results of EDGE against the aforementioned alternative explanation approaches. First, we observe that existing self-explainable methods (*i.e.,* Attention and Rational

[6] Note that we select these saliency methods because they pass the sanity check [1, 2]. Besides, it should be noted that we do not consider the perturbation-based methods to derive interpretation from RNN because these methods are mainly designed to explain convolutional networks trained for image recognition tasks.

[7] If the policy network is an RNN, we also fit the observation at time $t_i \cdots t_{i+l}$ into the policy to ensure the RNN policy's memory is not truncated.



(a) Time step importance of the target agent in the Pong game.



(b) Time step importance of the target regular agent in the You-Shall-Not-Pass game.

Figure 3: Illustrations of the critical time steps extracted by EDGE in a winning/losing episode.

Table 1: The target agent's performance in different use cases. "MuJoCo-Y" represents the You-Shall-Not-Pass game and "MuJoCo-K" stands for the Kick-And-Defend game. To demonstrate the statistical significance of our results, we run all the experiments three times with different random seeds and show the mean and standard error of results on each setup. Numbers before the brackets are means and those in the brackets are standard deviations. Supplementary S6 further shows a hypothesis test result.

Applications	Games	Rudder	Saliency	Attention	RatNet	EDGE
Target agent win rate changes before/after attacks	Pong	-19.93 (4.43)	-30.33 (0.47)	-25.27(1.79)	-29.20 (4.24)	**-65.47 (2.90)**
	MuJoCo-Y	-32.53 (4.72)	-29.33 (3.68)	-33.93 (5.77)	-30.00 (1.63)	**-35.13 (2.29)**
	MuJoCo-K	-21.80 (3.70)	-37.87 (6.31)	-41.20 (4.70)	-7.13 (2.50)	**-43.47 (4.01)**
Target agent win rate changes before/after patching	Pong	+1.89 (1.25)	-1.13 (0.96)	-0.58 (1.81)	-3.66 (1.35)	**+2.75 (0.65)**
	MuJoCo-Y	+1.76 (0.17)	+0.92 (0.32)	+0.44 (0.06)	+1.68 (0.50)	**+2.91 (0.32)**
	MuJoCo-K	+0.96 (0.1)	+1.17 (0.17)	+0.57 (0.04)	**+1.21 (0.16)**	**+1.21 (0.13)**
Victim agent win rate changes before/after robustifying	MuJoCo-Y	+8.54 (0.75)	+12.69 (1.46)	+25.10 (1.44)	+25.42 (1.32)	**+35.30 (3.02)**

Net) cannot consistently outperform the post-training explanation approaches (*i.e.,* saliency methods and Rudder). This observation aligns with our discussion in Section 3.1. Second, we discover that our method demonstrates the highest interpretation fidelity across all the games in all settings. As we discuss in Section 3.2, it is because our method could capture not only the inter-relationship between time steps but, more importantly, the joint effect across episodes.

In addition to the fidelity of our interpretation, we also evaluate the stability of our explanation and measure the explainability of each approach with regard to the underlying model. We present the experiment results in Supplement S3, demonstrating the superiority of our method in those dimensions. Along with this comparison, we further describe how well our method could fit given episodes, discuss the efficiency of our proposed approach, and test its sensitivity against the choice of hyper-parameters. Due to the page limit, we also detail these experiments and present experimental results in Supplement S3.

5 Use Cases of Interpretation

Understanding Agent Behavior. Fig. 3 showcases some episode snapshots of the target agent in the Pong and You-Shall-Not-Pass game together with the time-step importance extracted by our method. As we can first observe from Fig. 3(a), in the winning (left) episode, EDGE highlights the time steps when the agent hits the ball as the key steps leading to a win. This explanation implies that our target agent wins because it sends a difficult ball bouncing over the sideline and sailing to the corner where the opponent can barely reach. Oppositely, our method identifies the last few steps that the target agent misses the ball as the critical step in the losing episode. This indicates that the agent loses because it gets caught out of position. Similarly, our method also pinpoints the critical time steps matching human perceptions in the You-Shall-Not-Pass games. For example, in the left episode of Fig. 3(b), our explanations state that the runner (red agent) wins because it escapes from the blocker and crosses the finish line. Overall, Fig. 3 demonstrates that the critical steps extracted by EDGE can help humans understand how an agent wins/loses a game. In Supplement S4, we show more examples of critical time steps and the correlations we extracted from the three games. Supplement S7 further shows user study to demonstrate that our explanation could help user understand agent behaviors and thus perform policy forensics.

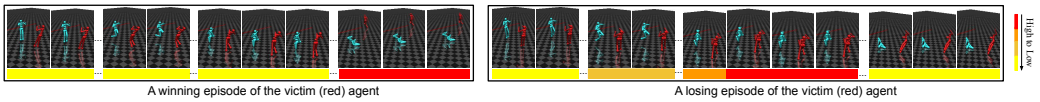


Figure 4: Time step importance of the victim agent in the You-Shall-Not-Pass game. By comparing this figure with Fig. 3(b), we can observe that our method could derive different explanations for different policies in the same game, indicating explanation is policy dependent.

Launching Adversarial Attacks. The qualitative analysis above reveals that an agent usually wins because of its correct moves at the crucial steps. With this finding, we now discuss how to launch adversarial attacks under the guidance of the interpretation. Previous research [41, 93] has proposed various attacks to fail a DRL agent by adding adversarial perturbation to its observations at each time step. We demonstrate that with the help of the explanations, an attacker could defeat an agent by varying actions at only a few critical steps rather than adding physically unrealistic perturbations. Our key idea is intuitive. If an agent's win mainly relies on its actions at a few crucial steps, then the agent could easily lose if it takes sub-optimal actions at those steps. Guided by this intuition, we propose an explanation-based attack that varies the agent's action at the critical steps identified by an explanation method. To test this attack's effectiveness, we first collect 2000 episodes where the target agent wins and explain these episodes with `EDGE` and the baseline approaches. Second, we conclude the top-K commonly critical steps across all the episodes (Here, we set K=30). Finally, we run the agent in the environment and force it to take a random action at the common important steps. We test the agent for 500 rounds and record the changes in its winning rate before/after attacks in Table 1. As we can observe from the 2$\sim$4 row of Table 1, all the explanation models can generate effective attacks that reduce the agent's winning rate. Benefiting from the high explanation fidelity, the attack obtained from our explanations demonstrates the strongest exploitability. Supplement S4 shows the results of different choices of K and discusses the potential alternatives of our attack. Note that this attack is different from the fidelity test in Section 4 in that our attack generalizes the summarized time step importance to unseen episodes while the fidelity test replays the explained episodes.

Patching Policy Errors. We design an explanation-guided policy patch method. The key idea is to explore a remediation policy by conducting explorations at the critical time steps of losing games and use the mixture of the original policy and the remediation policy as the patched policy. Specifically, we first collected a set of losing episodes of the target agent and identified the important time steps with `EDGE` and the baseline approaches above. Then, we explore the remediation policy by replay those episodes with different actions at the critical steps. Here, since we do not assume an oracle knowing the correct actions to take, we perform random explorations. First, we set an exploration budget of 10, representing replaying 10 times for each losing episode. In each replaying, we take a random action at the top 5 consecutive critical steps and record the random actions and corresponding states leading to a win. Finally, we form a look-up table with these collected state-action pairs and use it as the remediation policy. When running in the environment, the target agent will act based on the table if the current state is in the table. [8] Otherwise, the agent will take the actions given by its original policy. To test the effectiveness of our method, we run 500 games and record the changes in the target agent's winning rate before and after patching. As is shown in row 5$\sim$7 of Table 1, overall, the patched policies enhance the target agent's performance, and `EDGE` demonstrates the highest winning rate improvement. Table 1 also shows that in some cases, the patched policy introduces too many false positive that even reduce the winning rate. In Supplement S4, we discuss how to mitigate this problem via a probabilistic mixture of the remediation policy and the original policy. Supplement S4 also experiments the influence of the look-up table size on the patching performance and discusses other alternatives to our patching method.

Robustifying Victim Policies. Finally, we apply our methods to explain the episodes of a victim agent playing against an adversarial opponent in the You-Shall-Not-Pass game. The adversarial policy is obtained by the attack proposed in [31]. Fig. 4 demonstrates the identified important steps. First, the losing episode in Fig. 4 shows the blocker takes a sequence of adversarial behaviors (*i.e.,* intentionally falling on the ground). These malicious actions trick the runner into falling and thus losing the game.

[8] For games with a continuous state space, we compute the l_1 norm difference of the current state s_t and the states s_i in the table. If the state difference is lower than a small threshold (1e-4 in our experiment. We tested 1e-3, 1e-4, and 1e-5 and observed similar results.), we treat s_t and s_i as the same state. Since the games of the same agent usually start from relatively similar states and transition following the same policy, it is possible to observe similar states in different episodes.

Oppositely, the similar adversarial actions in the winning episode cannot trigger the runner to behave abnormally. The explanations reveal that the different focus of the victim causes the different victim actions. In the winning episode, the victim agent focuses less on the steps pertaining to adversarial actions, whereas those steps carry the highest weights in the losing episode. This finding implies that the victim agent may be less distracted by the adversarial actions if it does not observe them. Guided by this hypothesis, we propose to robustify the victim agent by blinding its observation on the adversary at the critical time steps in the losing episode (*i.e.,* the time steps pertaining to adversarial actions). We test the partially blind victim and record the changes in its winning rate before/after blinding. As is shown in the last row of Table 1, blinding the victim based on our explanations significantly improves its winning rate. Table 1 also demonstrates the effectiveness of the baseline approaches in robustifying victim policies. Overall, we demonstrate that the explanations of a victim policy could pinpoint the root cause of its loss and help develop the defense mechanism.

6 Discussion

Scalability. As is discussed in Section 3.3, by using inducing points and variational inference, our model parameters can be efficiently solved by stochastic gradient descents. Supplement S3&S5 show that EDGE imposes only a small training overhead over the existing methods. We can further accelerate the training of EDGE by leveraging more advanced matrix computation methods, such as approximating the covariance matrix with kernel structure interpolation [90] or replacing Cholesky decomposition with Contour Integral Quadrature when computing the K_{ZZ}^{-1} [67].

Other games. Besides the two-party Markov games (*i.e.,* Atari Pong and MuJoCo) studied in this work, many other games also have delayed rewards – mainly multi-player Markov games (*e.g.,* some zero-sum real-time strategy games [88]) and extensive-form games (*e.g.,* Go [74] and chess [75]). Regarding the multi-player Markov games, the associations between the episodes and final rewards will also be more sophisticated, requiring a model with a high capacity to fit the prediction. As part of future work, we will investigate how to increase the capacity of our proposed model for those games, such as adding more GP components or using a more complicated DNN as the mixing weight. For the extensive-form games, only one agent can observe the game state at any given time step and thus take action. As such, these games have a different form of episodes from the Markov games. In the future, we will explore how to extend our model to fit and explain the episodes collected from extensive-form games. Supplement S5 demonstrates our method's explanation fidelity on two games with instant rewards. Future work will evaluate the effectiveness of our attack and patching methods on those games and generalize our method to more sophisticated games with instant rewards.

Limitations and Future Works. Our work has a few limitations. First, we mainly compare EDGE with some existing techniques that have been used to explain sequential data. It is possible that with some adaptions, other explanation methods can also be applied to sequential data. It is also possible that existing methods can be extended to capture the correlations between episodes. As part of future work, we will explore these possibilities and broader solutions to explain a DRL policy. Second, the fidelity evaluation method introduced in Section 4 could be further improved, such as identifying multiple continuous important sequences. Our future work will investigate more rigorous fidelity testing methods and metrics. Third, our current learning strategy provides the point estimate of the mixing weight (explanations). In future work, we will explore how to place a prior on the model parameters and apply Bayesian inference (*e.g.,* MCMC [6]) to output the explanation uncertainty. Finally, our work also suggests that it may be possible to train a Transformer on MDP episodes to analyze offline trajectory data [19], and then add a GP on top to perform ablation studies. As part of future works, we will explore along this direction.

7 Conclusion

This paper introduces EDGE to derive strategy-level explanations for a DRL policy. Technically, it treats the target DRL agent as a blackbox and approximates its decision-making process through our proposed self-explainable model. By evaluating it on three games commonly utilized for DRL evaluation, we show that EDGE produces high-fidelity explanations. More importantly, we demonstrate how DRL policy users and developers could benefit from EDGE to understand policy behavior better, pinpoint policy weaknesses, and even conduct automated patches to enhance the original DRL policy.

Acknowledgments

We would like to thank the anonymous reviewers and meta reviewer for their helpful comments. This project was supported in part by NSF grant CNS-2045948 and CNS-2055320, by ONR grant N00014-20-1-2008, by the Amazon Research Award, and by the IBM Ph.D. Fellowship Award.

References

[1] J. Adebayo, J. Gilmer, M. Muelly, I. Goodfellow, M. Hardt, and B. Kim. Sanity checks for saliency maps. In *Proc. of NeurIPS*, 2018.

[2] J. Adebayo, M. Muelly, I. Liccardi, and B. Kim. Debugging tests for model explanations. In *Proc. of NeurIPS*, 2018.

[3] C. C. Aggarwal, A. Hinneburg, and D. A. Keim. On the surprising behavior of distance metrics in high dimensional space. In *Proc. of ICDT*, 2001.

[4] M. Al-Shedivat, A. G. Wilson, Y. Saatchi, Z. Hu, and E. P. Xing. Learning scalable deep kernels with recurrent structure. *The Journal of Machine Learning Research (JMLR)*, 2017.

[5] D. Alvarez-Melis and T. S. Jaakkola. Towards robust interpretability with self-explaining neural networks. In *Proc. of NeurIPS*, 2018.

[6] C. Andrieu, N. De Freitas, A. Doucet, and M. I. Jordan. An introduction to mcmc for machine learning. *Machine learning*, 2003.

[7] J. A. Arjona-Medina, M. Gillhofer, M. Widrich, T. Unterthiner, J. Brandstetter, and S. Hochreiter. Rudder: Return decomposition for delayed rewards. In *Proc. of NeurIPS*, 2019.

[8] ATARI. Atari games. `https://www.atari.com/`, 2006.

[9] A. Atrey, K. Clary, and D. Jensen. Exploratory not explanatory: Counterfactual analysis of saliency maps for deep reinforcement learning. In *Proc. of ICLR*, 2020.

[10] D. Bahdanau, K. Cho, and Y. Bengio. Neural machine translation by jointly learning to align and translate. In *Proc. of ICLR*, 2015.

[11] D. Balaban. How ai is mishandled to become a cybersecurity risk. `https://www.eweek.com/security/how-ai-is-mishandled-to-become-a-cybersecurity-risk/`, 2021.

[12] C. Bass, M. da Silva, C. Sudre, P.-D. Tudosiu, S. Smith, and E. Robinson. Icam: Interpretable classification via disentangled representations and feature attribution mapping. In *Proc. of NeurIPS*, 2020.

[13] O. Bastani, Y. Pu, and A. Solar-Lezama. Verifiable reinforcement learning via policy extraction. In *Proc. of ICML*, 2018.

[14] T. Bewley and J. Lawry. Tripletree: A versatile interpretable representation of black box agents and their environments. In *Proc. of AAAI*, 2021.

[15] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba. Openai gym. *arXiv preprint arXiv:1606.01540*, 2016.

[16] O.-M. Camburu, T. Rocktäschel, T. Lukasiewicz, and P. Blunsom. e-snli: Natural language inference with natural language explanations. In *Proc. of NeurIPS*, 2018.

[17] S. Chang, Y. Zhang, M. Yu, and T. Jaakkola. Invariant rationalization. In *Proc. of ICML*, 2020.

[18] C. Chen, O. Li, C. Tao, A. J. Barnett, J. Su, and C. Rudin. This looks like that: deep learning for interpretable image recognition. In *Proc. of NeurIPS*, 2019.

[19] L. Chen, K. Lu, A. Rajeswaran, K. Lee, A. Grover, M. Laskin, P. Abbeel, A. Srinivas, and I. Mordatch. Decision transformer: Reinforcement learning via sequence modeling. *arXiv preprint arXiv:2106.01345*, 2021.

[20] X. Chen, Z. Wang, Y. Fan, B. Jin, P. Mardziel, C. Joe-Wong, and A. Datta. Towards behavior-level explanation for deep reinforcement learning. *arXiv preprint arXiv:2009.08507*, 2020.

[21] L. Chu, X. Hu, J. Hu, L. Wang, and J. Pei. Exact and consistent interpretation for piecewise linear neural networks: A closed form solution. In *Proc. of KDD*, 2018.

[22] Y. Coppens, K. Efthymiadis, T. Lenaerts, A. Nowé, T. Miller, R. Weber, and D. Magazzeni. Distilling deep reinforcement learning policies in soft decision trees. In *Proc. of IJCAI Workshop on XAI*, 2019.

[23] M. H. Danesh, A. Koul, A. Fern, and S. Khorram. Re-understanding finite-state representations of recurrent policy networks. In *Proc. of ICML*, 2021.

[24] DeepMind. Alphastar: Mastering the real-time strategy game starcraft ii. `https://en.wikipedia.org/wiki/AlphaStar_(software)`, 2017.

[25] D. Duvenaud, H. Nickisch, and C. E. Rasmussen. Additive gaussian processes. *arXiv preprint arXiv:1112.4394*, 2011.

[26] S. Eleftheriadis, T. Nicholson, M. P. Deisenroth, and J. Hensman. Identification of gaussian process state space models. In *Proc. of NeurIPS*, 2017.

[27] G. F. Elsayed, Q. V. Le, and S. Kornblith. Saccader: Accurate, interpretable image classification with hard attention. In *Proc. of NeurIPS*, 2019.

[28] R. C. Fong and A. Vedaldi. Interpretable explanations of black boxes by meaningful perturbation. In *Proc. of ICCV*, 2017.

[29] R. Frigola, Y. Chen, and C. E. Rasmussen. Variational gaussian process state-space models. In *Proc. of NeurIPS*, 2014.

[30] A. Ghorbani, A. Abid, and J. Zou. Interpretation of neural networks is fragile. In *Proc. of AAAI*, 2019.

[31] A. Gleave, M. Dennis, N. Kant, C. Wild, S. Levine, and S. Russell. Adversarial policies: Attacking deep reinforcement learning. In *Proc. of ICLR*, 2020.

[32] S. Greydanus, A. Koul, J. Dodge, and A. Fern. Visualizing and understanding atari agents. In *Proc. of ICML*, 2018.

[33] S. Gu, E. Holly, T. Lillicrap, and S. Levine. Deep reinforcement learning for robotic manipulation with asynchronous off-policy updates. In *Proc. of ICRA*, 2017.

[34] T. Guo, T. Lin, and N. Antulov-Fantulin. Exploring interpretable lstm neural networks over multi-variable data. In *Proc. of ICML*, 2019.

[35] W. Guo, S. Huang, Y. Tao, X. Xing, and L. Lin. Explaining deep learning models-a bayesian non-parametric approach. *Proc. of NeurIPS*, 2018.

[36] W. Guo, D. Mu, J. Xu, P. Su, G. Wang, and X. Xing. Lemna: Explaining deep learning based security applications. In *Proc. of CCS*, 2018.

[37] W. Guo, X. Wu, S. Huang, and X. Xing. Adversarial policy learning in two-player competitive games. In *Proc. of ICML*, 2021.

[38] T. Haarnoja, V. Pong, A. Zhou, M. Dalal, P. Abbeel, and S. Levine. Composable deep reinforcement learning for robotic manipulation. In *Proc. of ICRA*, 2018.

[39] J. Heo, H. B. Lee, S. Kim, J. Lee, K. J. Kim, E. Yang, and S. J. Hwang. Uncertainty-aware attention for reliable interpretation and prediction. In *Proc. of NeurIPS*, 2018.

[40] S. Hooker, D. Erhan, P.-J. Kindermans, and B. Kim. A benchmark for interpretability methods in deep neural networks. In *Proc. of NeurIPS*, 2019.

[41] S. Huang, N. Papernot, I. Goodfellow, Y. Duan, and P. Abbeel. Adversarial attacks on neural network policies. In *Proc. of ICLR workshop*, 2017.

[42] A. Hüyük, D. Jarrett, C. Tekin, and M. Van Der Schaar. Explaining by imitating: Understanding decisions by interpretable policy learning. In *Proc. of ICLR*, 2021.

[43] A. A. Ismail, M. Gunady, L. Pessoa, H. C. Bravo, and S. Feizi. Input-cell attention reduces vanishing saliency of recurrent neural networks. In *Proc. of NeurIPS*, 2019.

[44] R. Iyer, Y. Li, H. Li, M. Lewis, R. Sundar, and K. Sycara. Transparency and explanation in deep reinforcement learning neural networks. In *Proc. of AIES*, 2018.

[45] S. Jain and B. C. Wallace. Attention is not explanation. In *Proc. of NAACL*, 2019.

[46] Z. Juozapaitis, A. Koul, A. Fern, M. Erwig, and F. Doshi-Velez. Explainable reinforcement learning via reward decomposition. In *Proc. of IJCAI Workshop on XAI*, 2019.

[47] O. Katz. Explainability: The next frontier for artificial intelligence in insurance and banking. `https://www.unite.ai/explainability-the-next-frontier-for-artificial-intelligence-in-insurance-and-banking/`, 2021.

[48] V. R. Konda and J. N. Tsitsiklis. Actor-critic algorithms. In *Proc. of NeurIPS*, 2000.

[49] A. Koul, S. Greydanus, and A. Fern. Learning finite state representations of recurrent policy networks. *arXiv preprint arXiv:1811.12530*, 2018.

[50] G.-H. Lee, W. Jin, D. Alvarez-Melis, and T. Jaakkola. Functional transparency for structured data: a game-theoretic approach. In *Proc. of ICML*, 2019.

[51] T. Lei, R. Barzilay, and T. Jaakkola. Rationalizing neural predictions. In *Proc. of EMNLP-IJCNLP*, 2017.

[52] O. Li, H. Liu, C. Chen, and C. Rudin. Deep learning for case-based reasoning through prototypes: A neural network that explains its predictions. In *Proc. of AAAI*, 2018.

[53] J. Liang, Y. Wu, D. Xu, and V. Honavar. Longitudinal deep kernel gaussian process regression. In *Proc. of AAAI*, 2021.

[54] Z. Lin, K.-H. Lam, and A. Fern. Contrastive explanations for reinforcement learning via embedded self predictions. In *Proc. of ICLR*, 2021.

[55] G. Liu, O. Schulte, W. Zhu, and Q. Li. Toward interpretable deep reinforcement learning with linear model u-trees. In *Proc. of ECML-PKDD*, 2018.

[56] Y. Y. Lu, W. Guo, X. Xing, and W. S. Noble. Dance: Enhancing saliency maps using decoys. In *Proc. of ICML*, 2021.

[57] C. J. Maddison, A. Mnih, and Y. W. Teh. The concrete distribution: A continuous relaxation of discrete random variables. In *Proc. of ICLR*, 2017.

[58] P. Madumal, T. Miller, L. Sonenberg, and F. Vetere. Distal explanations for model-free explainable reinforcement learning. *arXiv preprint arXiv:2001.10284*, 2020.

[59] P. Madumal, T. Miller, L. Sonenberg, and F. Vetere. Explainable reinforcement learning through a causal lens. In *Proc. of AAAI*, 2020.

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

[61] S.-M. Moosavi-Dezfooli, A. Fawzi, O. Fawzi, and P. Frossard. Universal adversarial perturbations. In *Proc. of CVPR*, 2017.

[62] A. Mott, D. Zoran, M. Chrzanowski, D. Wierstra, and D. J. Rezende. Towards interpretable reinforcement learning using attention augmented agents. In *Proc. of NeurIPS*, 2019.

[63] K. P. Murphy. *Machine learning: a probabilistic perspective*. MIT press, 2012.

[64] W. Nie, Y. Zhang, and A. Patel. A theoretical explanation for perplexing behaviors of backpropagation-based visualizations. In *Proc. of ICML*, 2018.

[65] OpenAI. Openai at the international 2017. `https://openai.com/the-international/`, 2017.

[66] T. Pedapati, A. Balakrishnan, K. Shanmugam, and A. Dhurandhar. Learning global transparent models consistent with local contrastive explanations. In *Proc. of NeurIPS*, 2020.

[67] G. Pleiss, M. Jankowiak, D. Eriksson, A. Damle, and J. R. Gardner. Fast matrix square roots with applications to gaussian processes and bayesian optimization. In *Proc. of NeurIPS*, 2020.

[68] N. Puri, S. Verma, P. Gupta, D. Kayastha, S. Deshmukh, B. Krishnamurthy, and S. Singh. Explain your move: Understanding agent actions using specific and relevant feature attribution. In *Proc. of ICLR*, 2020.

[69] J. Schulman, S. Levine, P. Abbeel, M. Jordan, and P. Moritz. Trust region policy optimization. In *Proc. of ICML*, 2015.

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

[71] Q. She and A. Wu. Neural dynamics discovery via gaussian process recurrent neural networks. In *Proc. of UAI*, 2020.

[72] W. Shi, S. Song, Z. Wang, and G. Huang. Self-supervised discovering of causal features: Towards interpretable reinforcement learning. *arXiv preprint arXiv:2003.07069*, 2020.

[73] T. Shu, C. Xiong, and R. Socher. Hierarchical and interpretable skill acquisition in multi-task reinforcement learning. *arXiv preprint arXiv:1712.07294*, 2017.

[74] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *Nature*, 2016.

[75] D. Silver, T. Hubert, J. Schrittwieser, I. Antonoglou, M. Lai, A. Guez, M. Lanctot, L. Sifre, D. Kumaran, T. Graepel, et al. Mastering chess and shogi by self-play with a general reinforcement learning algorithm. *arXiv preprint arXiv:1712.01815*, 2017.

[76] K. Simonyan, A. Vedaldi, and A. Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps. *arXiv:1312.6034*, 2013.

[77] D. Smilkov, N. Thorat, B. Kim, F. Viégas, and M. Wattenberg. Smoothgrad: removing noise by adding noise. *arXiv:1706.03825*, 2017.

[78] S. Srinivas and F. Fleuret. Rethinking the role of gradient-based attribution methods for model interpretability. In *Proc. of ICLR*, 2021.

[79] P. Sturmfels, S. Lundberg, and S.-I. Lee. Visualizing the impact of feature attribution baselines. *Distill*, 2020.

[80] M. Sundararajan, A. Taly, and Q. Yan. Axiomatic attribution for deep networks. In *Proc. of ICML*, 2017.

[81] A. Svensson, A. Solin, S. Särkkä, and T. Schön. Computationally efficient bayesian learning of gaussian process state space models. In *Proc. of AISTATS*, 2016.

[82] Y. Tang, D. Nguyen, and D. Ha. Neuroevolution of self-interpretable agents. In *Proc. of GECCO*, 2020.

[83] S. Thrun. Monte carlo pomdps. In *Proc. of NeurIPS*, 2000.

[84] E. Todorov, T. Erez, and Y. Tassa. Mujoco: A physics engine for model-based control. In *Proc. of ICIRS*, 2012.

[85] N. Topin and M. Veloso. Generation of policy-level explanations for reinforcement learning. In *Proc. of AAAI*, 2019.

[86] H. Van Hasselt, A. Guez, and D. Silver. Deep reinforcement learning with double q-learning. In *Proc. of AAAI*, 2016.

[87] A. Verma, V. Murali, R. Singh, P. Kohli, and S. Chaudhuri. Programmatically interpretable reinforcement learning. In *Proc. of ICML*, 2018.

[88] O. Vinyals, I. Babuschkin, W. M. Czarnecki, M. Mathieu, A. Dudzik, J. Chung, D. H. Choi, R. Powell, T. Ewalds, P. Georgiev, et al. Grandmaster level in starcraft ii using multi-agent reinforcement learning. *Nature*, 2019.

[89] S. Wiegreffe and Y. Pinter. Attention is not not explanation. In *Proc. of EMNLP-IJCNLP*, 2019.

[90] A. Wilson and H. Nickisch. Kernel interpolation for scalable structured gaussian processes (kiss-gp). In *Proc. of ICML*, 2015.

[91] A. G. Wilson, Z. Hu, R. Salakhutdinov, and E. P. Xing. Stochastic variational deep kernel learning. In *Proc. of NeurIPS*, 2016.

[92] V. Zambaldi, D. Raposo, A. Santoro, V. Bapst, Y. Li, I. Babuschkin, K. Tuyls, D. Reichert, T. Lillicrap, E. Lockhart, et al. Deep reinforcement learning with relational inductive biases. In *Proc. of ICLR*, 2018.

[93] H. Zhang, H. Chen, D. Boning, and C.-J. Hsieh. Robust reinforcement learning on state observations with learned optimal adversary. In *Proc. of ICLR*, 2021.

[94] X. Zhang, N. Wang, H. Shen, S. Ji, X. Luo, and T. Wang. Interpretable deep learning under fire. In *Proc. of USENIX Security*, 2020.

EDGE: Explaining Deep Reinforcement Learning Policies

S1 Additional Technical Details

Evidence Lower Bound. In the following, we derive the evidence lower bound (ELBO) in the Eqn. (3) of Section 3.3 and explain why maximizing it is equivalent to minimizing the KL divergence from the variational joint distribution to the true posterior. Specifically, we start with the log marginal likelihood $\log p(y|\mathbf{X}, \mathbf{Z})$ and show how to derive ELBO from it

$$
\begin{aligned}
\log p(y|\mathbf{X}, \mathbf{Z}) &= \log \int\int p(y, \mathbf{f}, \mathbf{u}|\mathbf{X}, \mathbf{Z}) \mathrm{d}\mathbf{u}\mathrm{d}\mathbf{f} \\
&= \log \int\int p(y|\mathbf{f}, \mathbf{X}) p(\mathbf{f}, \mathbf{u}|\mathbf{X}, \mathbf{Z}) \mathrm{d}\mathbf{u}\mathrm{d}\mathbf{f} \\
&= \log \int\int p(y|\mathbf{f}, \mathbf{X}) p(\mathbf{f}, \mathbf{u}|\mathbf{X}, \mathbf{Z}) \frac{q(\mathbf{f}, \mathbf{u})}{q(\mathbf{f}, \mathbf{u})} \mathrm{d}\mathbf{u}\mathrm{d}\mathbf{f} \\
&= \log \int\int p(y|\mathbf{f}, \mathbf{X}) q(\mathbf{f}, \mathbf{u}) \frac{p(\mathbf{f}, \mathbf{u}|\mathbf{X}, \mathbf{Z})}{q(\mathbf{f}, \mathbf{u})} \mathrm{d}\mathbf{u}\mathrm{d}\mathbf{f} \\
&= \log \mathbb{E}_{q(\mathbf{f}, \mathbf{u})}[p(y|\mathbf{f}, \mathbf{X}) \frac{p(\mathbf{f}, \mathbf{u}|\mathbf{X}, \mathbf{Z})}{q(\mathbf{f}, \mathbf{u})}] \\
&\overset{(a)}{\geq} \mathbb{E}_{q(\mathbf{f}, \mathbf{u})}[\log\, (p(y|\mathbf{f}, \mathbf{X}) \frac{p(\mathbf{f}, \mathbf{u}|\mathbf{X}, \mathbf{Z})}{q(\mathbf{f}, \mathbf{u})})] \\
&\geq \mathbb{E}_{q(\mathbf{f}, \mathbf{u})}[\log\, p(y|\mathbf{f}, \mathbf{X}) + \log\, \frac{p(\mathbf{f}, \mathbf{u}|\mathbf{X}, \mathbf{Z})}{q(\mathbf{f}, \mathbf{u})}] \\
&\geq \mathbb{E}_{q(\mathbf{f}, \mathbf{u})}[\log\, p(y|\mathbf{f}, \mathbf{X})] - \mathbb{E}_{q(\mathbf{f}, \mathbf{u})}[\log\, \frac{q(\mathbf{f}, \mathbf{u})}{p(\mathbf{f}, \mathbf{u}|\mathbf{X}, \mathbf{Z})}] \\
&\geq \mathbb{E}_{q(\mathbf{f}, \mathbf{u})}[\log\, p(y|\mathbf{f})] - \mathbb{E}_{q(\mathbf{f}, \mathbf{u})}[\log\, \frac{p(\mathbf{f}|\mathbf{u}) q(\mathbf{u})}{p(\mathbf{f}|\mathbf{u}) p(\mathbf{u})}] \\
&\geq \mathbb{E}_{q(\mathbf{f}, \mathbf{u})}[\log\, p(y|\mathbf{f})] - \int p(\mathbf{f}|\mathbf{u}) \mathrm{d}\mathbf{f} \int [\log\, \frac{q(\mathbf{u})}{p(\mathbf{u})}] q(\mathbf{u}) \mathrm{d}\mathbf{u} \\
&\geq \mathbb{E}_{q(\mathbf{f})}[\log\, p(y|\mathbf{f})] - \mathbb{KL}[q(\mathbf{u})||p(\mathbf{u})]\,,
\end{aligned}
\tag{1}
$$

where we omit the parameters $\{\Theta_n, \Theta_k, \Theta_p\}$ and $q(\mathbf{u})$ is the variational distribution. The (a) step is according to the Jensen's inequality. As we can observe from Eqn. (1), maximizing the ELBO will automatically maximize the marginal likelihood. Below, we derive the ELBO from the KL divergence from $q(\mathbf{f}, \mathbf{u})$ to $p(\mathbf{f}, \mathbf{u}|y)$.

$$
\begin{aligned}
\mathbb{KL}[q(\mathbf{f}, \mathbf{u})||p(\mathbf{f}, \mathbf{u}|y)] &= \int\int q(\mathbf{f}, \mathbf{u}) \log\, \frac{q(\mathbf{f}, \mathbf{u})}{p(\mathbf{f}, \mathbf{u}|y)} \\
&= \int\int q(\mathbf{f}, \mathbf{u}) \log\, \frac{q(\mathbf{f}, \mathbf{u}) p(y)}{p(y|\mathbf{f}, \mathbf{u}) p(\mathbf{f}, \mathbf{u})} \mathrm{d}\mathbf{u}\mathrm{d}\mathbf{f} \\
&= \int\int q(\mathbf{f}, \mathbf{u}) [\log\, \frac{1}{p(y|\mathbf{f}, \mathbf{u})} + \log\, \frac{q(\mathbf{f}, \mathbf{u}) p(y)}{p(\mathbf{f}, \mathbf{u})} + \log\, p(y)] \mathrm{d}\mathbf{u}\mathrm{d}\mathbf{f} \\
&= -\int\int q(\mathbf{f}, \mathbf{u}) [\log\, p(y|\mathbf{f}, \mathbf{u}) - \log\, \frac{q(\mathbf{f}, \mathbf{u}) p(y)}{p(\mathbf{f}, \mathbf{u})}] \mathrm{d}\mathbf{u}\mathrm{d}\mathbf{f} + \log\, p(y) \\
&= -\mathrm{ELBO} + \log\, p(y)\,.
\end{aligned}
\tag{2}
$$

Since $\mathbb{KL}[q(\mathbf{f},\mathbf{u})||p(\mathbf{f},\mathbf{u}|y)] \geq 0$, Eqn (2) shows that ELBO is a lower bound on the log marginal likelihood $\log p(y)$. In addition, since $\log p(y)$ is independent from $q(\mathbf{f},\mathbf{u})$, maximizing the ELBO will automatically minimize $\mathbb{KL}[q(\mathbf{f},\mathbf{u})||p(\mathbf{f},\mathbf{u}|y)]$.

Marginal Variational Posterior with Whitening. In our model, we apply the "whitening" operation proposed in [19]. Specifically, we first define $\mathbf{u} = \mathbf{L}\mathbf{v}$, where $\mathbf{L}\mathbf{L}^T = K_{ZZ}$ and $p(\mathbf{v}) = \mathcal{N}(\mathbf{0},\mathbf{I})$. Instead of directly defining $q(\mathbf{u})$, here, we define a variational distribution for $\mathbf{v}$, denoted as $q(\mathbf{v}) = \mathcal{N}(\mu_v,\mathbf{S})$. Then, we can compute $q(\mathbf{u}) = \mathcal{N}(L\mu_v, L\mathbf{S}L^T)$. Recall that $q(\mathbf{f},\mathbf{u}) = p(\mathbf{f}|\mathbf{u})q(\mathbf{u})$ and $p(\mathbf{f}|\mathbf{u}) = \mathcal{N}(K_{XZ}K_{ZZ}^{-1}\mathbf{u}, K_{XX} - K_{XZ}K_{ZZ}^{-1}K_{XZ}^T)$, we can compute $q(\mathbf{f})$ as

$$q(\mathbf{f}) = \int p(\mathbf{f}|\mathbf{u})q(\mathbf{u})\mathrm{d}\mathbf{u} = \mathcal{N}(K_{XZ}K_{ZZ}^{-1/2}\mu_v, K_{XX} + K_{XZ}K_{ZZ}^{-1/2}(\mathbf{S}-\mathbf{I})K_{ZZ}^{-1/2}K_{XZ}^T)\,. \tag{3}$$

Below, we denote $\mu_f = K_{XZ}K_{ZZ}^{-1/2}\mu_v$ and $\Sigma_f = K_{XX} + K_{XZ}K_{ZZ}^{-1/2}(\mathbf{S}-\mathbf{I})K_{ZZ}^{-1/2}K_{XZ}^T$. Note that, here we use the true marginal variational posterior, in our implementation, we also enable the widely applied SoR approximation [23], i.e., $q(\mathbf{f}|\mathbf{u}) \approx K_{XZ}K_{ZZ}^{-1/2}\mu_v$. With SoR, $q(\mathbf{f}) \approx \mathcal{N}(\mu_f, K_{XZ}K_{ZZ}^{-1/2}\mathbf{S}K_{ZZ}^{-1/2}K_{XZ}^T)$. It should also be noted that, with whitening, the variational parameters change from $\{\mu,\Sigma\}$ to $\{\mu_v,\mathbf{S}\}$ and the KL divergence term in ELBO becomes $\mathbb{KL}[q(\mathbf{v})||p(\mathbf{v})]$.

Expected Conditional Log Likelihood in Regression Model. Recall that our regression model has an analytical form of the likelihood term in the ELBO. Here, we derive this analytical from of the expected conditional log likelihood. Specifically, we first rewrite our regression model as follows:

$$\mathbf{f}|\mathbf{X} \sim \mathcal{N}(\mathbf{0}, k = \alpha_t^2 k_{\gamma_t} + \alpha_e^2 k_{\gamma_e}), \quad y_i|\mathbf{f}^{(i)} \sim \mathcal{N}(\mathbf{f}^{(i)}\mathbf{w}^T, \sigma^2)\,. \tag{4}$$

With the $q(\mathbf{f})$ in Eqn. (3), we then compute the expected conditional log likelihood as

$$\begin{aligned}\mathbb{E}_{q(\mathbf{f})}[\log p(y|\mathbf{f})] = \mathbb{E}_{q(\mathbf{f})}[\log p(y|\mathbf{f})] &= \mathbb{E}_{q(\mathbf{f})}[\frac{-N}{2}[\log\sigma^2 + \log 2\pi + \frac{1}{\sigma^2}(\mathbf{y}-\mathbf{F}\mathbf{w}^T)^T(\mathbf{y}-\mathbf{F}\mathbf{w}^T)]] \\ &= \frac{-N}{2}[\log\sigma^2 + \log 2\pi + \frac{1}{\sigma^2}\mathbb{E}_{q(\mathbf{f})}[(\mathbf{y}-\mathbf{F}\mathbf{w}^T)^T(\mathbf{y}-\mathbf{F}\mathbf{w}^T)]]\,,\end{aligned} \tag{5}$$

where $\mathbb{E}_{q(\mathbf{f})}[(\mathbf{y}-\mathbf{F}\mathbf{w}^T)^T(\mathbf{y}-\mathbf{F}\mathbf{w}^T)]$ can be computed as follows:

$$\begin{aligned}\mathbb{E}_{q(\mathbf{f})}[(\mathbf{y}-\mathbf{F}\mathbf{w}^T)^T(\mathbf{y}-\mathbf{F}\mathbf{w}^T)] &= \mathbb{E}_{q(\mathbf{f})}[\mathbf{y}^T\mathbf{y} - \mathbf{w}\mathbf{F}^T\mathbf{y} - \mathbf{y}^T\mathbf{F}\mathbf{w}^T + \mathbf{w}\mathbf{F}^T\mathbf{F}\mathbf{w}^T] \\ &= \mathbf{y}^T\mathbf{y} - \mathbf{w}\nu_f^T\mathbf{y} - \mathbf{y}^T\nu_f\mathbf{w}^T + \mathbf{w}\mathbb{E}_{q(\mathbf{f})}[\mathbf{F}^T\mathbf{F}]\mathbf{w}^T \\ &= \sum_i (y_i^2 - 2\nu_f^{(i)}\mathbf{w}^T) + \mathbf{w}\mathbb{E}_{q(\mathbf{f})}[\mathbf{F}^T\mathbf{F}]\mathbf{w}^T\,,\end{aligned} \tag{6}$$

where $\nu_f \in \mathbb{R}^{N\times T}$ is the matrix form of μ_f. $\nu_f^{(i)} \in \mathbb{R}^{1\times T}$ is the i-th row of ν_f, representing the mean of the variational posterior of $\mathbf{F}^{(i)}$. After computing the expectation of each element in $\mathbf{F}^T\mathbf{F}$ and combine them together, we have

$$\mathbb{E}_{q(\mathbf{f})}[\mathbf{F}^T\mathbf{F}] = \sum_N [\Sigma_f^{(i)} + (\nu_f^{(i)})^T\nu_f^{(i)}]\,, \tag{7}$$

where $\Sigma_f^{(i)} \in \mathbb{R}^{T\times T}$ is the covariance matrix of the variational posterior of $\mathbf{F}^{(i)}$. Plugging Eqn. (7) into Eqn. (6) and Eqn. (5), we have

$$\mathbb{E}_{q(\mathbf{f})}[(\mathbf{y}-\mathbf{F}\mathbf{w}^T)^T(\mathbf{y}-\mathbf{F}\mathbf{w}^T)] = \frac{-N}{2}[\log\sigma^2 + \log 2\pi + \frac{1}{\sigma^2}\sum_i((y_i - \nu_f^{(i)}\mathbf{w}^T)^2 + \mathbf{w}\Sigma_f^{(i)}\mathbf{w}^T)]\,. \tag{8}$$

With the analytical form in Eqn. (8), we can minimize the exact ELBO for our regression model without any approximation.

Predictive Distributions. Although our model mainly focuses on providing explanations, it can also perform prediction with the predictive distribution. In the following, we derive the predictive distribution of our regression and classification model with the variational distributions. Given a set

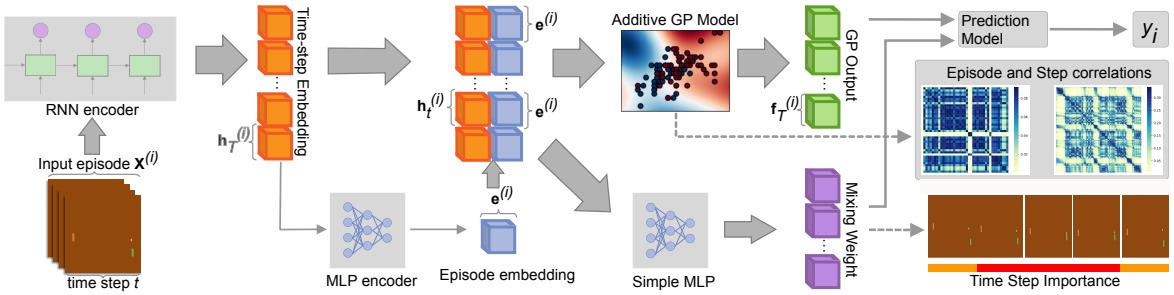


Figure S1: Overview of EDGE with a mixing weight given by an MLP.

of testing episodes $\mathbf{X}_* \in \mathbb{R}^{N_* T \times (d_s + d_a)}$, we first compute the variational posterior of their GP outputs $\mathbf{f}_*$ according to Eqn. (3).

$$q(\mathbf{f}_*) = \int p(\mathbf{f}_*|\mathbf{u}) q(\mathbf{u}) \mathrm{d}\mathbf{u} = \mathcal{N}(K_{X_* Z} K_{ZZ}^{-1/2} \mu_v, K_{X_* X} + K_{X_* Z} K_{ZZ}^{-1/2} (\mathbf{S} - \mathbf{I}) K_{ZZ}^{-1/2} K_{X_* Z}^T), \quad (9)$$

where $\{\mu_v, \mathbf{S}\}$ are the solved variational parameters. Below, we denote the mean and covariance matrix in $q(\mathbf{f}_*)$ as μ_* and Σ_*. After obtaining $q(\mathbf{f}_*)$, we then discuss how to conduct prediction in our regression and classification model. Regarding the regression model, which has a Gaussian likelihood, we can directly compute the marginal likelihood distribution as the predictive distribution, i.e.,

$$p(\mathbf{y}) = \int p(\mathbf{y}|\mathbf{f}_*) q(\mathbf{f}_*) \mathrm{d}\mathbf{f}_* = \mathcal{N}(\mu_y, \Sigma_y), \quad (10)$$

where $\mu_y \in \mathbb{R}^{N_*}$ can be computed as

$$\mu_y = \mathbb{E}[\mathbf{y}] = \mathbb{E}[\mathbf{F}_* \mathbf{w}^T] = \nu_* \mathbf{w}^T, \quad (11)$$

where $\nu_* \in \mathbb{R}^{N_* \times T}$ is the matrix form of μ_*. Then ,we compute $\Sigma_y \in \mathbb{R}^{N_* \times N_*}$ as

$$\Sigma_y = \mathrm{Var}[\mathbf{y}] = \mathrm{Cov}[\mathbf{F}_* \mathbf{w}^T, \mathbf{F}_* \mathbf{w}^T] + \mathbf{I}\sigma^2, \quad (12)$$

where $\mathrm{Cov}[(\mathbf{F}_* \mathbf{w}^T)_i, (\mathbf{F}_* \mathbf{w}^T)_j] = \mathbf{w}(\Sigma_*)_{iT:(i+1)T, jT:(j+1)T} \mathbf{w}^T$. After computing the marginal predictive distribution, we can make prediction using its mean μ_y and access the prediction uncertainty from Σ_y.

For the classification model, the marginal likelihood is intractable due to the non-Gaussian likelihood. We follow the prediction procedure proposed in [8] and use the MC method to make predictions. Specifically, we first sample B samples from $q(\mathbf{f}_*)$ and compute the conditional likelihood distribution $p(y|\mathbf{f}_*)$ using the drawn samples. Then, we compute the mean of the probability in the conditional distributions (*i.e.*, $\frac{1}{B} \sum_b \mathrm{softmax}(F^{(b)} \mathbf{W}^T)$) as the final predictions.

EDGE with an Input-specific Mixing Weight. Recall that our proposed model can provide input-specific explanations by replacing the constant mixing weight with a neural network. As is shown in Fig. S1, we use a simple MLP e_{ϕ_w} with three layers, i.e., a linear layer with T number of neurons, a LeakyReLU activation layer, and a linear layer with TK number of neurons. Given the episode encoding of N episodes $\mathbf{C} \in \mathbb{R}^{N \times T \times 2q}$, in which each element is the concatenation of that time step's unique embedding and the episode embedding of the corresponding episode (*i.e.*, $[\mathbf{h}_t^{(i)}, \mathbf{e}^{(i)}]$). We first sum the last dimension of each element and obtain $\mathbf{C}' \in \mathbb{R}^{N \times T}$ (*i.e.*, $\mathbf{C}' = \sum_c \mathbf{C}_{\cdot,\cdot,c}$). Second, we input $\mathbf{C}'$ into the network and get the corresponding output $e_{\phi_w}(\mathbf{C}) \in \mathbb{R}^{N \times TK}$, where K is the total number of classes in our classification model. Third, we transform $e_{\phi_w}(\mathbf{C})$ into the input-dependent mixing weight $\mathbf{W}_x \in \mathbb{R}^{N \times T \times K}$. Finally, we manipulate the GP output $\mathbf{F} \in \mathbb{R}^{N \times T}$ with $\mathbf{W}_x$ and obtain the predictions $\mathbf{P} \in \mathbb{R}^{N \times K}$. To ensure the explainability and stability, we borrow the idea from [1] and design a local-linear regularization for e_{ϕ_w}. Note that since our feature extractor is non-parametric, the regularization proposed in [1] is not applicable to our model. Specifically, to ensure local linearity, we propose to minimize $\mathcal{L}_e = \|e_{\phi_w}(\mathbf{C}') - e_{\phi_w}(\mathbf{C}' + \epsilon_c)\|_1$ together with the ELBO, where ϵ is a local

Table S1: Hyper-parameter choices of our method and the baseline approaches in the selected games. The numbers in the bracket of "CNN" represent the number of kernels in each layer. The numbers in the bracket of "Embedding", "MLP", and "GRU" refer to the hidden dimensions.

Games	Hyper-parameters shared by our method and the baseline approaches							Hyper-parameters unique to our method	
	Observation/State encoder	Action encoder	RNN encoder/classifier	Batch size	Epochs	Optimizer	learning rate	Number of inducing points	λ
Pong	CNN(32, 32, 32, 16)	Embedding(16)	GRU (q =4)	40	100	Adam	0.01	100	0.1
You-Shall-Not-Pass	MLP(64, 32)	MLP(64, 32)	GRU (q = 8)	40	200	Adam	0.01	600	0.01
Kick-And-Defend	MLP(64, 32)	MLP(64, 32)	GRU (q = 8)	40	200	Adam	0.01	600	0.01
CartPole	MLP(32, 16)	Embedding(4)→MLP(32, 16)	GRU (q = 4)	40	200	Adam	0.01	100	0.01
Pendulum	MLP(32, 16)	MLP(32, 16)	GRU (q = 4)	40	100	Adam	0.01	600	0.01

random perturbation added to the $\mathbf{C}'$. By minimizing $\mathcal{L}_e$, we can let e_{ϕ_w} to be almost a constant around the local area of each input and thus force the prediction model to be local-linear. In this work, we only apply this input-specific mixing weight to the classification model because it will make the exact computation of the expected log-likelihood in the regression model much more complicated and even intractable.

S2 Implementation Details and Experiment Setups

S2.1 Implementations and Hyper-parameter selections

Implementations. We implement `EDGE` using the `pytorch` [22] and the `gpytorch` [8] package. Regarding the baseline approaches, we implemented them based on the codes released in their original paper – Rudder: `https://github.com/ml-jku/rudder`; Input-Cell Attention RNN used in the saliency methods: `https://github.com/ayaabdelsalam91/Input-Cell-Attention`, saliency methods: `https://github.com/PAIR-code/saliency`; Attention: `https://github.com/sarahwie/attention`; Rational Net: `https://github.com/taolei87/rcnn`. A preliminary version of our software system with `EDGE` and all the baseline approaches is attached to the supplementary material.

Hyper-parameters. Table S1 shows the hyper-parameter choices of our experiments. First, we discuss the hyper-parameters that are shared across all the methods – network structures and training hyper-parameters. Regarding network structures, recall that we concatenate the state/observation and action at each step before inputting them to an RNN. More specifically, we apply a state/observation encoder and an action encoder to transform the original states and actions into a hidden representation. Since different games have different forms of states and actions, we use different network architectures for them. In the pong game, state/observation is an image of the current snapshot of the environment, and action is discrete. Here, we use a CNN with 4 layers, in which each layer has the kernel size of 3, the stride size of 2, and the "ReLU" activation function, as the state/observation encoder and an Embedding layer as the action encoder. Regarding the MuJoCo and Pendulum games, both observation/state and action are vectors with continuous values. In these games, we directly concatenate the state and action and input them into an MLP encoder. For the CartPole games, where the state is a continuous vector and action is discrete, we transform the action into a continuous vector using an Embedding layer and input it together with the state into an MLP encoder. With the hidden representations of each time step, we then input it into an RNN with GRU cells except for the RNN in RNN+Saliency (In RNN+Saliency, we follow the original setup in [13] and use LSTM as the RNN cell). We adopt the widely applied "Tanh" attention as the attention layer in our attention+RNN model. For the baseline approaches, we directly use the RNN as the predictor (*i.e.,* Seq2one structure). For our method, as is introduced in Section 3.2, we use the RNN together with a one-layer MLP to derive the time-step embedding and the episode embedding. As for the training hyper-parameters, our method shares the same choices as the baseline approaches except for the learning rate in the Kick-And-Defend game (See Table S1 for detailed values). Second, our method introduces two unique hyper-parameters – number of inducing points (M) and λ, where λ represents the coefficient of the lasso regularization added on the mixing weight (To encourage more understandable explanations, we add a lasso regularization on the mixing weight $\mathbf{W}/\mathbf{w}$). Table S1 shows the choice of these two hyper-parameters in each game. In Section S3, we further study the sensitivity of our method against M and λ.

Table S2: Descriptions of the episode dataset of each game. "Discrete (X)" refers to categorical actions with the X possible choices. "Vector (X)" refers to continuous state/action vectors with the dimensionality of X. "Classification (2)" stands for the classification task with 2 possible classes. To enable batch operation, we pad all the episodes in the same game to the same length T.

Games	Observation/state	Action	T	Training size	Testing size	Task type
Pong	Image (80, 80, 1)	Discrete (6)	200	21500	1880	Classification (2)
You-Shall-Not-Pass	Vector (380)	Vector (17)	200	31900	2000	Classification (2)
Kick-And-Defend	Vector (380)	Vector (17)	200	31500	2000	Classification (2)
CartPole	Vector (4)	Discrete (2)	200	29500	4200	Regression
Pendulum	Vector (3)	Vector (1)	100	28000	4000	Regression

S2.2 Experiment Setups

Selected games, agents, and episodes. In our experiments, we select three games with delayed rewards – Atari pong, MuJoCo You-Shall-Not-Pass, and MuJoCo Kick-And-Defend, and two games with instant rewards – OpenAI gym CartPole and Pendulum. For the descriptions of the Atari and OpenAI gym games, readers could refer to [21, 5]. These three games are single-player games, and we directly use the agent in each game as our target agent. Regarding MuJoCo games, readers could find the introductions of their game environments and reward designs in [3]. Note that these games are two-player games, we select the runner in You-Shall-Not-Pass and kicker in Kick-And-Defend as our target agent. Section 4 mentioned that we download a well-trained policy for each game. Specifically, we download the policy in the Pong game from `https://github.com/greydanus/baby-a3c` and the policies in the MuJoCo games from `https://github.com/openai/multiagent-competition`. For the CartPole and Pendulum game, we get the agent from `https://github.com/DLR-RM/rl-trained-agents/tree/d81fcd61cef4599564c859297ea68bacf677db6b/ppo`. All the agents are trained with policy gradient methods (A3C for the pong policy and PPO for the MuJoCO and OpenAI Gym policies). After obtaining the target agents, we run each agent in the corresponding environment and collect a set of training and testing episodes by varying the random seed. Table S2 shows the descriptions of these episode datasets including training-testing split, state-action dimensions, and episode length. Table S2 also shows the type of prediction task on each game. Regarding the games with delayed rewards, we conduct the classification with 2 possible classes (*i.e.,* the target agent wins or losses a game). For the games with instant rewards, we conduct the regression task. In our experiments, we use the training set to train the explanation models and the testing set for evaluations and use cases. Supplementary material includes the trained explanation models used in evaluations and use cases.

Computational resources. In our experiments, we use a server with 4 NVIDIA RTX A6000 GPUs to train and test the explanation models.

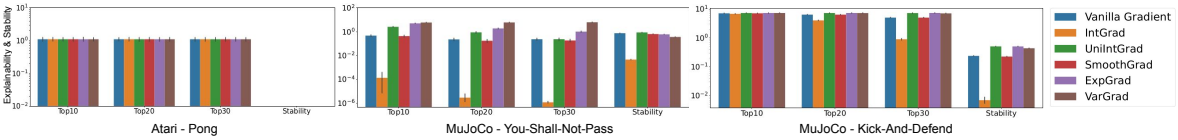
Other Related Issues. As is introduced above, the existing assets used in our experiments are the pretrained agents. We checked the GitHub repositories, from where we download them, and do not find the license information for the Pong and MuJoCO policies. The license for the CartPole and Pendulum is the MIT License. Since all the agents are just neural networks rather than actual data, they do not contains personally identifiable information or offensive content.

S3 Additional Evaluations on Games with Delayed Rewards

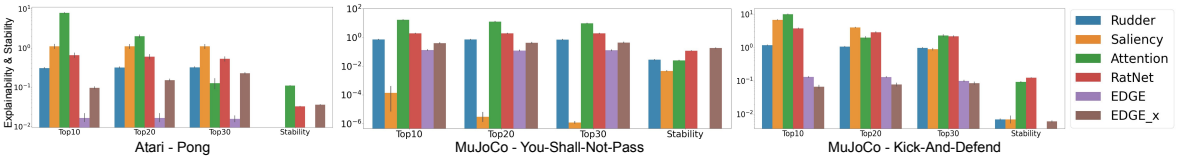
Recall that, besides evaluating the explanation fidelity with regard to the original RL agent and environment, we also compare our method with the baselines approaches from the following dimensions: reward prediction performance, explainability, stability, and efficiency. In this section, we introduce the designs of these experiments on the games with delayed rewards and discuss the corresponding results. Section S5 will present these evaluations on the games with instant rewards. Note that we apply the input-specific mixing weight to our classification model, in this section, we report the results of our method with a constant mixing weight (denoted as `EDGE`) and that with an input-specific mixing weight (denoted as `EDGE`_x).

Table S3: The testing accuracy of each method on the selected games. "EDGE_x" refers to our model with an input-specific mixing weight. Note that "Rudder" is designed only for regression tasks, we use it to directly predict the value of y_i and report the MAE (*i.e.*, $\frac{1}{N}\sum_i |\hat{y}_i - y_i|$).

Games	Rudder	Saliency (%)	Attention (%)	RatNet (%)	EDGE (%)	EDGE_x (%)
Pong	0.012	88.9	99.9	97.3	99.9	99.9
You-Shall-Not-Pass	0.018	99.2	99.1	99.3	99.2	99.1
Kick-And-Defend	0.011	98.3	98.3	98.8	99.7	98.9



(a) Mean and standard error of the explanability and stability scores obtained by each saliency method. "IntGrad" refers to the integrated gradient and "UniIntGrad" stands to the integrated gradient with uniform baseline. The stability of each saliency method on the Pong game is 0.



(b) Mean and standard error of the explanability and stability scores obtained by EDGE and the baseline approaches. The stability of EDGE on all the games is 0. The stability of Rudder and Saliency method on the Pong game is also 0.

Figure S2: Explainability and stability comparison across the selected explanation methods.

Model Performance. To evaluate how well each method could predict the final rewards, we test the prediction model in each method on the testing episodes and record the testing accuracy of each game in Table S3. As we can first observe from the table, all the methods could obtain a decent prediction performance except for the saliency method on the Pong game. We take a closer look at this case and find that the RNN method in the saliency method completely biases towards the winning episodes on the Pong game and cannot recognize the losing ones. Table S3 also shows that overall EDGE establishes the highest testing accuracy. This result confirms the benefits of our GP feature extractor (*i.e.*, capturing the time step and episode correlations). Finally, we observe that the input-specific mixing weight slightly reduces the testing accuracy of our proposed method. We speculate that this is because the additional model capacity introduced by the local linear MLP causes overfitting. This overfitting problem can be mitigated by increasing the regularization strength.

Explainability and Stability. After evaluating the performances of the prediction models, we then evaluate the explainability and the stability of each selected explanation method. Specifically, explainability represents how well an explanation method could explain the prediction model. In our evaluation, we use the metric proposed in [7, 12, 27] to quantify the explainability. Formally, given a normalized explanation $\mathbf{E}_i \in \mathbb{R}^T$ of an input episode $\mathbf{X}_i$, we define the explainability metric as $-\log F^{c_i}(\mathbf{E}_i \odot \mathbf{X}_i)$. Here $\mathbf{E}_i \odot \mathbf{X}_i$ represents multiplying each entity (*i.e.*, the state and action at each step) in $\mathbf{X}_i$ with the corresponding element in $\mathbf{E}_i$, encoding the overlap between the object of interest and the concentration of the explanation. $F^{c_i}(\cdot)$ refers to the model prediction of the true class of $\mathbf{X}_i$. By viewing the explanation as weights of input entities, a faithful explanation should weight important entities more highly than less important ones and thus give rise to a higher predicted class score and a lower metric value. Note that we apply the top-K normalization, i.e., setting the value of the top-K important time steps as 1 and the rest time steps as 0. We do not use the 0-1 normalization because multiplying it with discrete actions will result in invalid actions. It should also be noted that explainability evaluates the faithfulness of the explanation with regard to the prediction

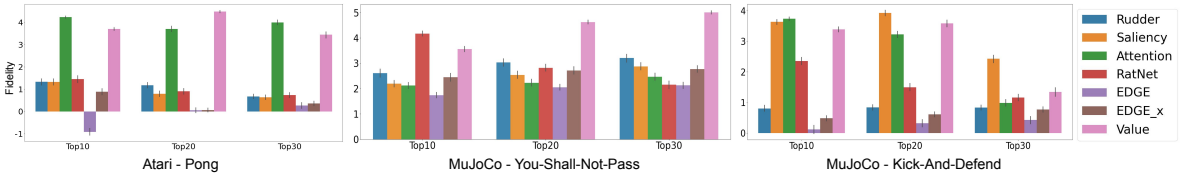


Figure S3: Mean and standard error of the fidelity scores obtained by our method, selected baselines, and the value function. Note that we use the saliency method demonstrates the highest explainability (*i.e.,* integrated gradient) in this experiment (See Fig. S2(a)).

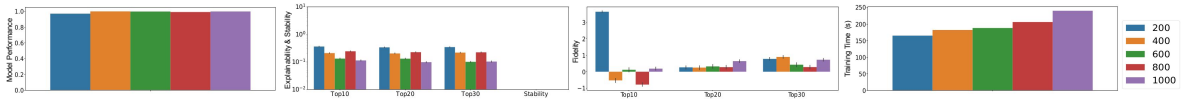
model. We do not need to keep the perturbations (*i.e.,* $\mathbf{E}_i \odot \mathbf{X}_i$) to be physically realistic. In our experiments, we set $K = 10/20/30$. Fig. S2(b) shows the explanability comparisons between EDGE and the baseline approaches. First, we observe that neither Attention nor Rational Net can consistently outperform the post-training explanation approaches (*i.e.,* saliency methods and Rudder). This result confirm the first limitation of the existing self-explainable methods discussed in Section 3.2, i.e., adding the explanation module in front of the prediction model cannot faithfully explain the associations learned by the prediction model. In comparison, with a different explanation module design, EDGE is able to outperform the baseline approaches in most setups. This result verifies the effectiveness of our explanation module design. This result also confirms that by capturing the unique correlations exhibited in the RL episodes, our method could better fit these episodes and thus give rise to higher explainability. We notice a corner case on the You-Shall-Not-Pass game where the saliency method shows a higher explainability than our method. As part of future work, we will investigate the reasons behind this result. Our future work will also explore the reasons behind the different performance of EDGE and EDGE_x on these games. Note that Fig. S2(a) shows the comparison between the six selected saliency methods. Overall, the integrated gradient demonstrates the highest explainability on the selected games. We also observe that all the saliency methods show a similar performance on the Pong games. We speculate this is caused by the model bias mentioned above.

In addition to explainability, we also evaluate the stability of each explanation method against random perturbations added to the input. Specifically, we use the following metric [27] to evaluate the stability: $\mathbb{E}_{\epsilon_s \sim \mathcal{N}(0,\sigma_s^s)} \frac{\|E(\mathbf{X},F^c) - E(\mathbf{X}+\epsilon_s,F^c)\|_2}{\|\epsilon_s\|_2}$, where σ_s^s controls the perturbation strength. In our experiments, we set it as 0.05 times of the maximum value range of input features. A lower metric value represents a more stable explanation. Here, we use the MC method to estimate the expectation, i.e., sampling ϵ_s 10 times and computing the mean of the 10 metric values obtained from the sampled ϵ_s. Note that to ensure the legitimacy of the perturbation, we only add ϵ_s to the states and actions with continuous values. Fig. S2 shows the results of our methods and all the comparison baselines. As shown in the figure, EDGE demonstrates the lowest the metric value (*i.e.,* 0). This is because the explanation of EDGE is a global explanation, which is robust to input perturbation. In comparison, replacing the constant mixing weight with a neural network jeopardizes the explanation stability. We can further improve the stability of EDGE_x by increasing the regularization strength or training set size.

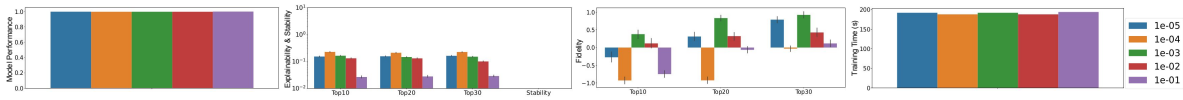
Fidelity. Section 4 shows the fidelity comparison between EDGE and the selected baseline approaches. Here, we further show the fidelity of two other methods – EDGE_x and the explanations drawn from the value function. Regarding the second method, we use the value function of the target agent and treat the time steps with the top-K value function outputs as the top-K important features. Fig. S3 shows the comparison across all the methods. We first observe that the value function cannot faithfully reflect the associations between input episodes and the final rewards. We believe there are two reasons behind this result. First, the policy training inevitably introduces errors to the value function approximation, resulting in inaccurate explanations. More importantly, value function expresses the expected total return of a state, rather than the contribution of the current state to a game's final reward. In other words, it is not designed to capture the specific associations between episodes and their final rewards. Consequentially, the explanations drawn from the value function cannot faithfully represent the above associations. Note that since all of the target agent are trained by policy gradient, we use their

Table S4: Training/explanation time of each method on the selected games. Regarding the saliency method, we record the explanation time of the integrated gradient method, which demonstrates the highest explanability (See Fig. S2(a)).

Games	Rudder	Saliency	Attention	RatNet	EDGE	EDGE_x
Pong	7:05min/0.002s	16:15min/0.02s	6:49min/0.002s	7:26min/0.002s	7:20min/0.002s	07:42min/0.007s
You-Shall-Not-Pass	2:49min/0.0002s	4:41min/0.172s	2:43min/0.0003s	2:50min/0.0002s	3:40min/0.0002s	03:50min/0.011s
Kick-And-Defend	1:57min/0.0001s	2:54min/0.132s	1:55min/0.0001s	2:02min/0.0002s	3:08min/0.0002s	03:31min/0.009s



(a) Evaluation performance comparison between the models trained with different number of inducing points.



(b) Evaluation performance comparison between the models trained with different λ.

Figure S4: Hyper-parameter sensitivity test results of EDGE on the Kick-And-Defend game. EDGE refers to our model with a constant mixing weight.

value networks in our evaluation. Our future work will evaluate the effectiveness of using Q network (function) to derive explanation. Second, Fig. S3 also shows that EDGE_x has a slight worse fidelity than EDGE . We speculate this is caused by its worse model performance and lower explainability. Finally, we notice that the fidelity in Fig. S3 is not strictly aligned with the explanability in Fig. S2. In other words, there are some cases where an explanation with a high explanability cannot achieve a high fidelity to the RL agent. As part of future works, we will take a closer look into these corner cases and investigate the reasons behind their results.

Efficiency. Table S4 shows the training/explanation time of each method. For training, we record the run time of one training epoch. Regarding explanation, we record the run time of deriving one explanation. As we can first observe from the Table, our methods (*i.e.,* EDGE and EDGE_x) introduce only a slight training overhead compared to the baseline approaches. This confirms the efficiency of our parameter learning method. In addition, since our explanations can be directly drawn from the mixing weight, the explanation process takes negligible time. Similarly, the explanation process of all the other methods is also very fast, except for the integrated gradient method, which is an ensemble method requiring multiple gradient computations in one explanation.

Hyper-parameter Sensitivity. Finally, we test the sensitivity of our model performance against the different choices of the unique hyper-parameters introduced by our method (*i.e.,* number of inducing points - M and lasso regularization coefficient - λ). Specifically, we first fix $\lambda = 0.01$ and vary $M = 200/400/600/800/1000$. For each choice of M, we train our explanation model, run the above evaluations, and record the corresponding results in Fig. S4(a). We can roughly observe two trends from the figure – (1) the model performance, explainability, and fidelity get better as M increases; (2) the training time becomes longer as M increases. These trends reflect the general model performance and training efficiency trade-off introduced by inducing points, i.e., using more inducing points will improve the model performance but reduce the training efficiency. Despite the existence of this trade-off, Fig. S4(a) also demonstrates that the model is already able to achieve a decent performance with $M = 600$ and the training time of $M = 600$ is acceptable (introducing about 13.9% extra training time compared to $M = 200$). This result matches with the finding in [25]. That is, a small number of inducing points is enough for decent model performance, and the training time increases slowly in a range of M. Overall, this experiment shows that by choosing M within a reasonable range, our method could achieve a superior performance without introducing too much overhead compared to the baseline approaches. This property escalates the practicability of our method in that users do not need to

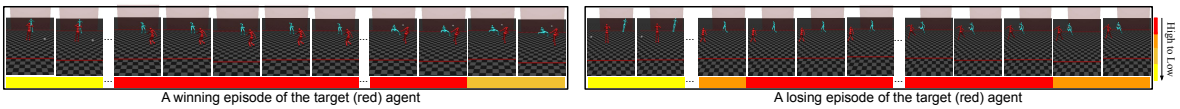


Figure S5: Time step importance of the target agent in the Kick-And-Defend game.

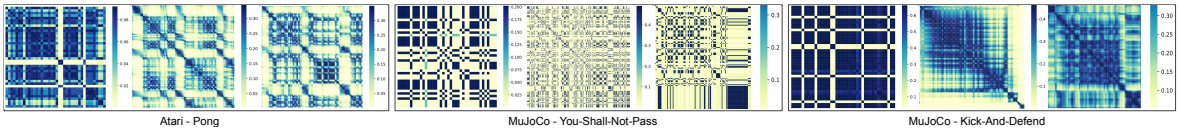


Figure S6: Illustrations of the time step correlations and episode correlations extracted by EDGE. For each game, we show the correlations between 40 episodes (left figure), the time step correlations of a winning episode (middle figure), and the time step correlations of a losing episode (right figure).

exhaustively search for the optimal hyper-parameter choices to achieve a decent performance. Second, we fix $M = 600$ and vary $\lambda = 0.1/0.01/0.001/0.0001/0.00001$. Fig. S4(b) shows the results of each model. As we can first observe from the figure, different choices of λ have a minor influence upon the model performance and the training time. Fig. S4(b) also shows a general trend that the explainability and fidelity get higher as λ increases. This result confirms the benefits of lasso regularization to variable selection in our model. The results of varying λ suggest that users could choose a relatively large regularization strength when applying our method.

S4 Additional Use Cases on Games with Delayed Rewards

In this section, we show more experiment results and discussions of the use cases introduced in Section 5. Note that, following Section 5, we use EDGE with a constant mixing weight in the use cases.

Understanding agent behaviour. Section 5 visualizes the time steps importance obtained by our method in the Pong and You-Shall-Not-Pass game. Here, we show the explanations in the Kick-And-Defend game and the correlations extracted by our method. Fig. S5 shows the game episodes and the time step importance extracted by our method. Similar to the Pong and You-Shall-Not-Pass game, EDGE provides human-understandable explanations in the Kick-And-Defend game. For example, in the winning game, our explanation highlights the time steps when the kicker shoots the ball. This explanation indicates that the kicker (red agent) wins because it shoots a difficult ball that the keeper fails to defend. Fig. S6 visualizes the episode correlations $K_e(\mathbf{X}, \mathbf{X})$ and time step correlations $K_t(\mathbf{X}, \mathbf{X})$ in the selected games. As we can first observe from the figure, the episode correlations demonstrate clear cluster structures. More specifically, in the Pong and Kick-And-Defend game, the episodes form two clusters. After checking the episodes in each cluster, we surprisingly find that the episodes in the same cluster are all winning/losing episodes. In other words, our method successfully groups the games with the same result into one cluster. In the You-Shall-Not-Pass game, despite the episodes have more than two clusters, we also find that each cluster is very pure in terms of the game results with only a few outliers. This result further validates the effectiveness of our method in modeling the joint efforts between episodes. Second, we can also observe that the winning and losing episodes have different time step correlations. Specifically, the losing episodes tend to have more concentrated correlations at the last few steps, whether the time step correlations of the winning episodes are more scattered. As is discussed in Section 3.2, these time step correlations can be used to generated episode-specific explanations from the global ones.

Launching Adversarial Attacks. As is discussed in Section 5, we also experiment with the influence of the number of commonly critical steps K upon the attack performances. Specifically, we set $K = 10/20/30$ and record the corresponding attack results of each explanation method in Table S5. Table S5 first shows that the exploitability of each method improves as perturbing more actions (*i.e.,* enlarging K). We also observe that overall EDGE triggers the most significant performance

Table S5: The changes in target agent's win rate before/after attacks with different choices of K. We ran each experiment three times and report the mean and standard error. Section S6 further shows the result of a hypothesis test.

Games	K	Rudder	Saliency	Attention	RatNet	Our
Pong	10	-4.33 (5.13)	-12.33 (2.62)	-5.60 (1.18)	-13.20 (1.88)	**-61.87 (3.92)**
	20	-5.60 (1.64)	-22.13 (1.23)	-16.27 (1.27)	-23.20 (4.33)	**-64.00 (2.45)**
	30	-19.93 (4.43)	-30.33 (0.47)	-25.27 (1.79)	-29.20 (4.24)	**-65.47 (2.90)**
You-Shall-Not-Pass	10	**-9.73 (3.80)**	-6.40 (1.78)	-7.60 (1.77)	-3.54 (2.04)	-9.60 (2.95)
	20	-21.47 (5.28)	-21.53 (3.49)	-24.27 (6.67)	-13.40 (5.39)	**-29.33 (6.55)**
	30	-32.53 (4.72)	-29.33 (3.68)	-33.93 (5.77)	-30.00 (1.63)	**-35.13 (2.29)**
Kick-And-Defend	10	-6.60 (2.95)	-6.00 (6.56)	-5.93 (2.00)	-2.40 (3.08)	**-8.67 (4.73)**
	20	-14.13 (0.81)	-26.00 (3.46)	-27.13 (6.02)	-4.20 (1.59)	**-33.40 (7.53)**
	30	-21.80 (3.70)	-37.87 (6.31)	-41.20 (4.70)	-7.13 (2.50)	**-43.47 (4.01)**

Table S6: Additional results of our patch methods. The upper table shows the patching results of varying the mixing probability P on the Pong game. The low tables shows the patching result of changing the exploration budget B on three games. We ran each experiment three times and report the mean and standard error. Section S6 further shows the result of a hypothesis test.

Games	Setups	Rudder	Saliency	Attention	RatNet	EDGE
Pong	P=0.14	+0.68 (0.39)	0.08 (0.06)	+0.13 (0.12)	-1.77 (0.53)	**+1.28 (0.24)**
	P=1	+1.89 (1.25)	-1.13 (0.96)	-0.58 (1.81)	-3.66 (1.35)	**+2.75 (0.65)**
Pong	B=10	+1.89 (1.25)	-1.13 (0.96)	-0.58 (1.81)	-3.66 (1.35)	**+2.75 (0.65)**
	B=20	+4.48 (0.72)	-1.91 (1.40)	-4.00 (1.06)	-2.58 (3.61)	**+4.84 (1.91)**
	B=30	**+3.28 (0.88)**	-2.38 (1.74)	-1.23 (1.00)	-5.50 (0.84)	+0.80 (0.57)
You-Shall-Not-Pass	B=10	+1.76 (0.17)	+0.92 (0.32)	+0.44 (0.06)	+1.68 (0.50)	**+2.91 (0.32)**
	B=20	+1.66 (0.15)	+0.47 (0.12)	+0.31 (0.15)	+1.56 (0.25)	**+3.01 (0.34)**
	B=30	+1.34 (0.24)	+0.13 (0.11)	+0.20 (0.06)	+1.42 (0.09)	**+2.79 (0.18)**
Kick-And-Defend	B=10	+0.96 (0.1)	+1.17 (0.17)	+0.57 (0.04)	**+1.21 (0.16)**	**+1.21 (0.13)**
	B=20	+3.16 (0.49)	+3.21 (0.18)	+2.09 (0.06)	+2.43 (0.39)	**+4.02 (0.31)**
	B=30	+3.11 (0.28)	+2.90 (0.30)	+1.84 (0.26)	+3.57 (0.32)	**+3.92 (0.65)**

drop in all the setups except one case. We defer to future work to study the reason behind that case. Besides summarizing the most critical time steps, an attacker could also record the states of the critical time steps and launch attacks at the most important states (if the total number of the state is within a reasonable range). Our future work will compare the exploitability between the attack based on time steps and the attack based on states. Future works could also explore combining the explanation with the existing adversarial attacks (*e.g.,* manipulating the target agent's observation only at the critical time steps identified by the explanation methods).

Patching Policy Errors. As shown in Table 1 in Section 5, our patching method jeopardizes the agent's winning rate on the Pong game when using Attention and Rationale Net as the explanation method. This motivates us to explore a probabilistic mixing policy. That is, when the current state is in the look-up table, the agent chooses the corresponding action in the table with a probability P and the action given by its original policy with the probability $1 - P$. To decide the value of P, we run the agent's original policy in the Pong environment for N_a games and record the number of losing games that encounter the states stored in the look-up table (*i.e.,* N_l). We compute $P = N_l/N_a$, representing the probability of the agent running into the states in the look-up table and eventually losing the corresponding game. In our experiment, we set $N_a = 500$ and compute $P = 0.14$. We use this probability to rerun the patching method on the Pong game and record the changes in the agent's winning rate before/after patching in the upper table of Table S6. As we can observe from the table, a lower mixing probability indeed alleviates the false positive introduced by Attention and Rationale Net. At the same time, it also decreases the effectiveness of the other explanation methods. This result indicates that users could start with a conservative patching policy by setting a small value for P and

Table S7: The testing MAE of each method on games with instant rewards.

Games	Rudder	Saliency	Attention	RatNet	EDGE
Cartpole	0.01	0.03	0.006	0.035	7e-05
Pendulum	0.006	0.008	0.005	0.03	0.009

increase the probability if aiming for a better patching performance. Besides the probabilistic mixing policy, we also study the influence of look-up table size on the patching performance. The look-up table size is decided by the exploration budget B and the number of continuous critical steps. In this experiment, we vary the look-up table size by setting different B. Specifically, we set $B = 10/20/30$ and report the corresponding patching performance in the lower table of Table S6. Overall, we observe that the patch performance improves as B increases from 10 to 20 and drops as B reaches 30. This result indicates that as the look-up table size increases, it includes more losing games and their corresponding remediation policies. As a result, the patched policy is able to correct more errors and thus achieve a higher winning rate. Oppositely, adding more states into the look-up table will also introduce more new errors. This is because a state may occur both in a winning game and a losing game, and changing the actions in an original winning game may, unfortunately, result in a loss. When the table size reaches a certain point (30 in our experiment), the new errors start to dominate the patched episodes, causing a winning rate drop. This result implies that the users may need to search for an optimal choice of B to get the highest patching performance. As part of future work, we will study the corner cases in this experiment and explore how to search for an optimal look-up table size more systemically and efficiently. Note that the patching results of the saliency method are all zeros on the Pong game because it fails to search for any successful remediation. As a result, the look-up table is empty in those setups. We defer to future work to study the reason behind this result.

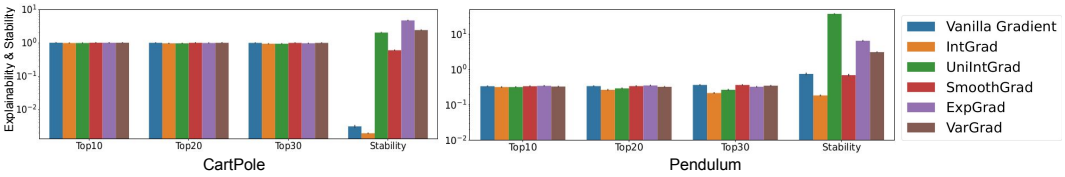
Rather than using a mixed policy, we also tried to enhance the original policy via behavior cloning (*i.e.,* fine-tune the policy network with the collected state-action pairs). We found this strategy barely works because the policy network oftentimes "forgets" its learned policy after the behavior cloning. Note that we perform random exploration in our method. If an oracle or a better policy is available, we can perform more efficient exploration by mimicking their actions at the critical steps. Going beyond mixed policy or learning from oracle, a more general patch solution would be to redesign the reward function based on the explanation results (*e.g.,* adding some intermediate rewards to guide the agent taking correct actions at the important steps). As part of future work, we will investigate how to design such intermediate rewards.

S5 Evaluation on Games with Instant Rewards

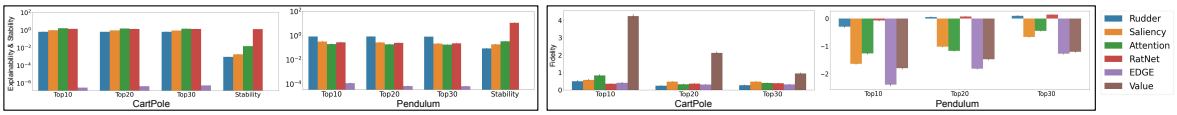
In this section, we evaluate our proposed method on two games with instant rewards, i.e., CartPole and Pendulum. Similar to the evaluations on the games with delayed rewards, we also compare EDGE with the baseline approaches from the following five dimensions: model performance, explainability, stability, fidelity, and efficiency. In the following, we introduce the setup of each experiment and discuss the corresponding experiment results. As discussed above, we do not apply the input-specific mixing weight to the regression method in this paper. As a result, we only report the results of our model with a constant mixing weight (denoted as EDGE).

Model Performance. Table S7 shows the testing MAE (*i.e.,* $\frac{1}{N}\sum_i |\hat{y}_i - y_i|$) of the prediction model in each explanation method. Note that since we conduct regression on these games, we use the MAE instead of accuracy as the metric. As we can observe from the table, EDGE demonstrates the lowest MAE on both games, indicating the best prediction performance. This result further confirms the effectiveness of our model design in fitting RL episodes.

Explainability and Stability. In this evaluation, we use a different explainability metric from the classification tasks. Specifically, we define the explainability metric of the regression tasks as $|F(\mathbf{E}_i \odot \mathbf{X}_i) - F(\mathbf{X}_i)|$. This metric is the MAE between the model prediction of the original input and the input weighted by the explanation. A faithful explanation should highlight the important entities



(a) Mean and standard error of the explanability and stability scores obtained by each saliency method.



(b) Mean and standard error of the explanability, stability, and fidelity scores obtained by EDGE and the baseline approaches. The stability of EDGE on all the games is 0.

Figure S7: Explainability, stability, and Fidelity comparison across the selected explanation methods.

Table S8: Training/explanation time of each method on the games with instant rewards. Regarding the saliency method, we record the explanation time of the integrated gradient method, which demonstrates the highest explanability (See Fig. S7(a)).

Games	Rudder	Saliency	Attention	RatNet	EDGE
Cartpole	37s/3e-05s	1:52min/0.02s	32s/3e-05s	37s/3e-05s	1:44min/0.001s
Pendulum	33s/7.6e-05s	2:24min/0.057s	35s/7.5e-05s	37s/6.8e-05s	1:20min/0.003s

and thus keep the original prediction value and a lower metric value. Similarly, we also apply the top-K normalization and set $K = 10/20/30$. The left two figures in Fig. S7(b) show the explainability comparisons between EDGE and the baseline approaches. The results have a similar trend as those in Fig. S2(b). This result further demonstrates the superiority of our method in explainability and confirms the benefits of our explanation model design. Fig. S7(a) shows the explainability comparison between the six selected saliency methods. Similar to the results in Fig. S2(a), the integrated gradient also demonstrates the highest explainability on these two games. In this experiment, we apply the same setup and metric as the experiment in Section S3 for the stability comparison. As shown in the left two figures of Fig. S7(b), EDGE demonstrates lowest metric value, indicating the highest stability.

Fidelity. We apply the fidelity evaluation method introduced in Section 4 for the fidelity evaluation on the CartPole and Pendulum game. The right two figures in Figure 7(b) show the fidelity comparison between EDGE and the baseline approaches (we also use the integrated gradient as the saliency method). We observe that benefiting from the superior model prediction performance and explainability, our method demonstrates the highest fidelity in most setups. This result further demonstrates the advantage of our self-explainable model over the comparison baselines.

Efficiency. Table S8 shows the training/explanation run time comparison between our method and the baseline approaches. Since the games are simpler than the above games with delayed rewards. The run times in Table S8 are smaller than those in Table S4. Similar to the results in Table S4, our method introduces a small training overhead over the existing methods (up to 72s). The explanation times are negligible for all the explanation methods. In summary, with the results in Section S3 and S5, we can safely conclude that with our newly designed self-explainable model and the parameter learning procedure, EDGE improves the baseline approaches from multiple dimensions (*i.e.,* explainability, stability, and fidelity) without introducing too much extra computational cost.

S6 Hypothesis Test

To further demonstrate the statistical significance of our results, we conducted a paired t-test on the experimental results in Table 1, S5, and S6. Specifically, given a set of results of our method

Table S9: P-value of each experiment in Table 1.

Applications	Games	Rudder vs. EDGE	Saliency vs. EDGE	Attention vs. EDGE	RatNet vs. EDGE
Target agent win rate changes before/after attacks	Pong	<0.001	0.003	0.001	0.009
	MuJoCo-Y	0.19	0.13	0.35	0.04
	MuJoCo-K	0.02	0.08	0.28	0.003
Target agent win rate changes before/after patching	Pong	0.29	0.02	0.09	<0.001
	MuJoCo-Y	0.005	0.006	0.006	0.009
	MuJoCo-K	0.12	0.43	0.006	0.49
Victim agent win rate changes before/after robustifying	MuJoCo-Y	0.002	0.001	0.006	0.008

Table S10: P-value of each experiment in Table S5.

Games	K	Rudder vs. EDGE	Saliency vs. EDGE	Attention vs. EDGE	RatNet vs. EDGE
Pong	10	<0.001	<0.001	<0.001	<0.001
	20	<0.001	0.002	<0.001	0.006
	30	<0.001	0.003	0.001	0.009
You-Shall-Not-Pass	10	0.51	0.12	0.21	0.05
	20	0.05	0.05	0.21	0.05
	30	0.19	0.13	0.35	0.04
Kick-And-Defend	10	0.21	0.36	0.12	0.15
	20	0.02	0.05	0.06	<0.001
	30	0.02	0.08	0.28	0.003

($O = \{O_1, O_2, O_3\}$) and that of a baseline ($B = \{B_1, B_2, B_3\}$), we first compute their difference $D = \{O_i - B_i\}$, for $i = 1, 2, 3$. Our non hypothesis for attacks experiments is $H0_1 : \mathbb{E}[D] \geq 0$. The non-hypothesis for policy patch and adversarial defense experiments is $H0_2 : \mathbb{E}[D] \leq 0$. Given these non-hypothesis, we compute the p-value for the performance difference of each group of comparison and show the values in Table S9, S10, and S11. For attacks, if p is small, we should reject the $H0_1$, indicating our attack triggers a higher winning rate drop than the comparison baseline and thus has a better exploitability. Regarding the patch and defense experiments, if p is small, we should reject the $H0_2$, indicating our method enables a higher winning rate increase than the comparison baseline and thus has a better performance. Overall, the results in Table S9, S10, and S11 are aligned with those in Table 1, S5, and S6.

S7 User Study

Recall that Section 2 discusses that previous DRL explanation methods derive interpretations of individual actions by identifying the observation's feature importance regarding the agent's policy network/value function output. Whereas our work highlights the time steps critical to an agent's final result in each episode (*e.g.,* win or loss). In previous research, some researchers conducted user studies to demonstrate the utility of their explanation methods from human perspectives (*e.g.,* [9, 2] uses interpretation to distinguish well-trained (good) and overfitted (bad) agents). In this work, we also performed a user study to demonstrate the utility of our explanation method. Below, we describe the design and results of our user study.

We obtained IRB approval and conducted a user study to compare our proposed explanation method with a representative explanation method [9] that pinpoints the input features essential to the agent's individual actions via a saliency method. Specifically, we first recruited 30 participants with different backgrounds in DRL and DRL explanations (4 participants have published paper(s) in DRL explanation; 6 participants have read some papers about DRL explanation; 10 participants have a general understanding of DRL explanation; 10 participants have never heard of DRL explanations.). Then, we presented an online survey to these participants. This survey aims to compare our explanation method with [9] from two perspectives. (1) How well can the explanations generated from the two approaches help a user to pinpoint a good policy? (2) How well can the explanations help a user perform episode forensics and thus understand why an agent fails or succeeds? We briefly describe the

Table S11: P-value of each experiment in Table S6.

Games	Setups	Rudder vs. EDGE	Saliency vs. EDGE	Attention vs. EDGE	RatNet vs. EDGE
Pong	P=0.14	0.05	0.005	0.004	<0.001
	P=1	0.29	0.02	0.09	<0.001
Pong	B=10	0.29	0.02	0.09	<0.001
	B=20	0.37	0.05	0.008	0.09
	B=30	0.96	0.04	0.09	0.006
You-Shall-Not-Pass	B=10	0.005	0.006	0.006	0.009
	B=20	0.03	0.002	0.006	0.004
	B=30	<0.001	0.002	0.001	0.002
Kick-And-Defend	B=10	0.12	0.43	0.006	0.49
	B=20	0.01	0.01	0.004	0.02
	B=30	0.14	0.12	0.02	0.15

design of our user study and the study results below.

(1) Identifying a good policy: Given the representative episodes gathered from two agents of the You-Shall-Not-Pass game (one well-trained and the other overfitted to one specific opponent, i.e., an adversarial agent), we first derived explanations for each of these episodes using the aforementioned interpretation methods. Second, we randomly partitioned the 30 participants into two equally-sized groups and presented the episodes to each group. For Group-A, we also presented the explanations that our method generates. For Group-B, we provided the interpretations that the other method [9] generates. Based on the episodes and their corresponding interpretation, we asked each subject to pinpoint the well-trained agent and asked whether the explanations help identify the good policy. We first discovered that 11 out of 15 participants in Group-A correctly identified the well-trained agent, and 63.6% of them found the explanation helpful. Regarding group-B, 10 out of 15 participants identified the good policy, and 50% of them found the explanation helpful. From the above results, we can get that 7 (11×0.636) out of 15 participants in Group-A correctly identified the good policy with the help of our explanations, and 5 (10×0.5) out of 15 participants in Group-B correctly identified the good policy with the help of the explanations given by the baseline approach [9]. To compare the ability of two explanation methods in facilitating good policy identification, we conducted a two population proportion test [11]. Specifically, we first set the null hypothesis H_0 as $p_1 = p_2$, where p_1 and p_2 are the probability of correctly identifying the good policy according to our explanations and the explanations given by the baseline approach [9]. Then, we computed the sample probability for each group – Group-A: $\hat{p}_1 = 7/15$; Group-B: $\hat{p}_2 = 5/15$ and the z statistic, i.e., $z = \frac{(\hat{p}_1 - \hat{p}_2)}{\sqrt{\hat{p}_c(1-\hat{p}_c)(\frac{1}{n}+\frac{1}{n})}}$, in which $\hat{p}_c = (7+5)/30$ is the pooled sample proportion and $n = 15$ is the number of participants in each group. Plugging in $\hat{p}_1$ and $\hat{p}_2$, we computed $z = 0.745$. Finally, we computed the percentile r of z in the standard normal distribution and obtained the p-value as $2(1-r) = 0.456$. Since the p-value is not that small (*e.g.,* ≤ 0.05), we fail to reject H_0. This result shows that our method demonstrates approximately the same utility as the existing explanations in identifying good/bad policies.

(2) Performing forensics: Given a set of representative episodes gathered from one agent, we used the two above explanation methods to derive explanations for each episode. Then, we present to participants the episodes along with the explanations generated by two different methods. We asked the participants which explanation methods are more beneficial in helping the subject understand why the agent fails/succeeds. We discovered that 21 participants ($70\% = 21/30$) chose our method. This discovery implies that interpreting by highlighting critical time steps could better facilitate episode forensics than explaining by highlighting critical input to the action at each step. We further conducted a binomial test [11] to demonstrate that our explanation method significantly outperforms the existing method [9] in helping policy forensics. In this test, our null hypothesis H_0 is $p \leq 0.5$, where p is the probability of choosing our explanation method as more helpful for policy forensics. Then, we computed the percentile r of Y in the binomial distribution $B(30, 0.5)$, where $Y = 21$ is the number of participants that chose our method as the more helpful one. Finally, we compute the p-value as $1-r = 0.008$. This small p-value means we should reject H_0, indicating that our method is significantly better than the baseline [9] in facilitating policy forensics.

Table S12: Mean/Standard error/P-value of the attack performance of our method and the action preference attack with different choices of K. Note that since the action preference attack is not applicable to the You-Shall-Not-Pass and the Kick-And-Defend games, we only conducted the experiment on the Pong game.

	K	Action preference attack	EDGE
Pong	10	-50.34/0.94/0.04	**-61.87/3.92**
	20	-61.33/1.25/0.19	**-64.00/2.45**
	30	-62.33/0.94/0.17	**-65.47/2.90**

More details about our user study: Survey questions for Group-A: `https://forms.gle/SfUCRCWhZEag47gj9`; Survey questions for Group-B: `https://forms.gle/Kkj4z4wapCTDqXN76`. Questions 1-3 are the same in both surveys. They ask about the participant's background and whether he/she understands the game rule and two types of explanations (We found all 30 participants correctly answer Questions 2&3). Questions 4-7 are about identifying good/bad policies. We present four episodes of the well-trained/overfitted agent together with different explanations to the participants (We count a participant as correctly selecting the good policy only if the participant correctly answered all questions.). Questions 9-10 are about forensic evaluation. They are the same in both surveys, where the participants are presented with the same videos. Each video shows an episode of the same agent together with explanations derived by our method and [9]. The participants are asked to choose which explanation is more helpful. As mentioned above, we found that 21 out of 30 participants chose our explanations in both questions.

S8 Comparison of Our Attack with An Existing Attack

As mentioned in Section 5, existing research has developed various attacks against DRL policies (*e.g.,* [10, 26, 17]). In this section, we compared our attack with the attack proposed in [17]. This attack manipulates the observations at the selected time steps, whereas our attack changes the actions at the important steps identified by our method. They cannot be directly compared due to the different attack spaces (observation space vs. action space). To enable the comparison, we applied the time step selection method developed in [17] to choose time steps and modify the actions at the selected steps. Specifically, the time step selection method in [17] first computes the action preference at each step as $\pi(s, a_{\max}) - \pi(s, a_{\min})$. Here, π is the target policy, which outputs the probability of each action. $a_{\max}$ and $a_{\min}$ refers to the action with highest/lowest probability at the state s. Then, it ranks the action preference and selects the time steps with high preference scores to launch its attack. As mentioned above, in our experiment, we rank the action preference scores of states in each episode and randomly change the actions at the states with top-K action preference scores (marked as Action preference attack). We also conducted three groups of experiments and reported the mean/stand deviation/p-value of the paired t-test with $H0_1 : \mathbb{E}[D] \geq 0$ as the null hypothesis. The results are shown in Table S12. We observe from Table S12 that our attack has a stronger exploitability, confirming the advantage of our method in identifying important steps. Note that the time step selection method in [17] cannot be applied to the policy networks that directly output the action rather than the action probability (*e.g.,* the policy networks trained by the PPO algorithm). In our experiment, the policies of the You-Shall-Not-Pass and the Kick-And-Defend games are trained by PPO. As such, [17] cannot be applied to these two games and thus we only compared it with our method on the Pong game, where the agent is trained with the A3C method.

S9 Potential Social Impact

Any work focusing on general statistical methods, machine learning models included, runs the risk of those methods being used for purposes the authors did not consider. As one of these general purpose tools, EDGE is designed to allow for new modes of understanding and improving Reinforcement Learning

(RL) agents in particular.

Reinforcement learning has recently enjoyed successful application in many areas of computer science. For example, gaming, robotics, natural language processing, and computer vision have all seen advancements from RL [16]. With such wide areas of application, RL methods have naturally extended to several business domains; some examples include business management [15], finance [14], healthcare [6], and education [18]. This is to say, our work focuses on a general purpose tool whose applications cannot be well forecast in advance. For any set of these applications, there will inevitably be subsets considered harmful, and others considered beneficial; furthermore, the stated harms and benefits will vary depending on the individuals surveyed.

In considering the potential benefits and harms that may result from our work, we turn to the many such discussions focusing on other general purpose tools. Recently there has been a wide array of literature published on the fairness, benefits, biases, and harms of machine learning. These works focus on aspects such as the biases and discriminatory behavior present in machine learning models, in addition to mitigation strategies that should be employed in deployment and commercialization [4] [24] [20].

`EDGE` , as a tool to understand and improve RL methods, can be used to amplify biased or unfair models. Conversely, it can also be used to understand the decision-making process of such systems and enable the construction of strategies to mitigate these harmful factors. We consider this situation analogous to that of open source tools; open source software enables hackers and criminals but simultaneously provides benefits to all of us, including visibility into the methods of those employing such tools. In this way, we also believe it is important to construct tools such as `EDGE`so that we can further understand and improve RL methods that have come before and after it.

References

[1] David Alvarez-Melis and Tommi S Jaakkola. Towards robust interpretability with self-explaining neural networks. In *Proc. of NeurIPS*, 2018.

[2] Dan Amir and Ofra Amir. Highlights: Summarizing agent behavior to people. In *Proc. of AAMAS*, 2018.

[3] Trapit Bansal, Jakub Pachocki, Szymon Sidor, Ilya Sutskever, and Igor Mordatch. Emergent complexity via multi-agent competition. In *Proc. of ICLR*, 2018.

[4] Reuben Binns. Fairness in machine learning: Lessons from political philosophy. In Sorelle A. Friedler and Christo Wilson, editors, *Proceedings of the 1st Conference on Fairness, Accountability and Transparency*, volume 81 of *Proceedings of Machine Learning Research*, 2018.

[5] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym. *arXiv preprint arXiv:1606.01540*, 2016.

[6] Bibhas Chakraborty and Susan A Murphy. Dynamic treatment regimes. *Annual review of statistics and its application*, 2014.

[7] Piotr Dabkowski and Yarin Gal. Real time image saliency for black box classifiers. In *Proc. of NeurIPS*, 2017.

[8] Jacob R Gardner, Geoff Pleiss, David Bindel, Kilian Q Weinberger, and Andrew Gordon Wilson. Gpytorch: Blackbox matrix-matrix gaussian process inference with gpu acceleration. In *Proc. of NeurIPS*, 2018.

[9] Samuel Greydanus, Anurag Koul, Jonathan Dodge, and Alan Fern. Visualizing and understanding atari agents. In *Proc. of ICML*, 2018.

[10] Wenbo Guo, Xian Wu, Sui Huang, and Xinyu Xing. Adversarial policy learning in two-player competitive games. In *Proc. of ICML*, 2021.

[11] Robert V Hogg, Elliot A Tanis, and Dale L Zimmerman. *Probability and statistical inference*. Pearson/Prentice Hall Upper Saddle River, NJ, USA:, 2010.

[12] Sara Hooker, Dumitru Erhan, Pieter-Jan Kindermans, and Been Kim. A benchmark for interpretability methods in deep neural networks. In *Proc. of NeurIPS*, 2019.

[13] Aya Abdelsalam Ismail, Mohamed Gunady, Luiz Pessoa, Hector Corrada Bravo, and Soheil Feizi. Input-cell attention reduces vanishing saliency of recurrent neural networks. In *Proc. of NeurIPS*, 2019.

[14] Amir E Khandani, Adlar J Kim, and Andrew W Lo. Consumer credit-risk models via machine-learning algorithms. *Journal of Banking & Finance*, 2010.

[15] Lihong Li, Wei Chu, John Langford, and Robert E. Schapire. A contextual-bandit approach to personalized news article recommendation. In Michael Rappa, Paul Jones, Juliana Freire, and Soumen Chakrabarti, editors, *Proceedings of the 19th International Conference on World Wide Web, WWW 2010, Raleigh, North Carolina, USA, April 26-30, 2010*, 2010.

[16] Yuxi Li. Deep reinforcement learning. *CoRR*, abs/1810.06339, 2018.

[17] Yen-Chen Lin, Zhang-Wei Hong, Yuan-Hong Liao, Meng-Li Shih, Ming-Yu Liu, and Min Sun. Tactics of adversarial attack on deep reinforcement learning agents. *arXiv preprint arXiv:1703.06748*, 2017.

[18] Travis Mandel, Yun-En Liu, Sergey Levine, Emma Brunskill, and Zoran Popovic. Offline policy evaluation across representations with applications to educational games. In *Proc. of AAMAS*, 2014.

[19] Alexander Graeme de Garis Matthews. *Scalable Gaussian process inference using variational methods*. PhD thesis, University of Cambridge, 2017.

[20] Ninareh Mehrabi, Fred Morstatter, Nripsuta Saxena, Kristina Lerman, and Aram Galstyan. A survey on bias and fairness in machine learning. *CoRR*, 2019.

[21] OpenAI. Openai gym. `https://gym.openai.com/`, 2016.

[22] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *arXiv preprint arXiv:1912.01703*, 2019.

[23] Bernhard W Silverman. Some aspects of the spline smoothing approach to non-parametric regression curve fitting. *Journal of the Royal Statistical Society: Series B (Methodological)*, 1985.

[24] Jenna Wiens, Suchi Saria, Mark Sendak, Marzyeh Ghassemi, Vincent X Liu, Finale Doshi-Velez, Kenneth Jung, Katherine Heller, David Kale, Mohammed Saeed, et al. Do no harm: a roadmap for responsible machine learning for health care. *Nature medicine*, 2019.

[25] Andrew Gordon Wilson, Zhiting Hu, Ruslan Salakhutdinov, and Eric P Xing. Stochastic variational deep kernel learning. In *Proc. of NeurIPS*, 2016.

[26] Xian Wu, Wenbo Guo, Hua Wei, and Xinyu Xing. Adversarial policy training against deep reinforcement learning. In *Proc. of USENIX Security Symposium*, 2021.

[27] Chih-Kuan Yeh, Cheng-Yu Hsieh, Arun Sai Suggala, David I Inouye, and Pradeep Ravikumar. On the (in) fidelity and sensitivity for explanations. In *Proc. of NeurIPS*, 2019.