Can Differentiable Decision Trees Enable Interpretable Reward Learning from Human Feedback?

Akansha Kalra
Kahlert School of Computing
University of Utah
akanshak@cs.utah.edu

Daniel S. Brown
Kahlert School of Computing
University of Utah
daniel.s.brown@utah.edu

Abstract

Reinforcement Learning from Human Feedback (RLHF) has emerged as a popular paradigm for capturing human intent to alleviate the challenges of hand-crafting the reward values. Despite the increasing interest in RLHF, most works learn black box reward functions that while expressive are difficult to interpret and often require running the whole costly process of RL before we can even decipher if these frameworks are actually aligned with human preferences. We propose and evaluate a novel approach for learning expressive and interpretable reward functions from preferences using Differentiable Decision Trees (DDTs). Our experiments across several domains, including CartPole, Visual Gridworld environments and Atari games, provide evidence that the tree structure of our learned reward function is useful in determining the extent to which the reward function is aligned with human preferences. We also provide experimental evidence that not only shows that reward DDTs can often achieve competitive RL performance when compared with larger capacity deep neural network reward functions but also demonstrates the diagnostic utility of our framework in checking alignment of learned reward functions. We also observe that the choice between soft and hard (argmax) output of reward DDT reveals a tension between wanting highly shaped rewards to ensure good RL performance, while also wanting simpler, more interpretable rewards. Videos and code, are available at: `https://sites.google.com/view/ddt-rlhf`

1 Introduction

The reward function is central to reinforcement learning (RL) algorithms (Sutton and Barto, 2018); however, it is difficult to manually specify a good reward function for many tasks (Ng et al., 1999; Krakovna et al., 2020), motivating learning reward functions from human input (Jeon et al., 2020). We focus on the problem of learning interpretable reward functions.

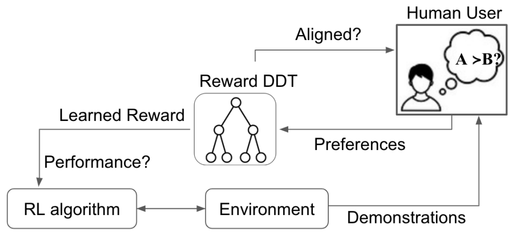


Figure 1: We propose an end-to-end differentiable approach for training reward functions using differentiable decision trees via trajectory preference labels to enable interpretability and identification of misalignment of the learned reward function.

Most modern reward learning methods use deep neural networks (Finn et al., 2016; Christiano et al., 2017; Ibarz et al., 2018; Hejna and Sadigh, 2022; Tien et al., 2023). However, despite the growing interest in explaining black box models trained via deep learning (Gilpin et al., 2018; Zhang and Zhu, 2018; Heuillet et al., 2021; Räukur et al., 2022), deep neural networks remain extremely difficult to interpret. In the context of reward learning, it is especially critical that we can interpret the learned objective—if we cannot understand the objective that a robot or AI system has learned, then it is difficult to know if the AI system's behavior will be aligned with human preferences and intent (Russell et al., 2015;

Leike et al., 2018; Brown et al., 2021). This is particularly significant in tasks where human safety is on the line, for example in healthcare, autonomous navigation, and assistive robots.

Thus, we are faced with a problem: we want highly accurate and expressive reward models, but we also want to be able to interpret the learned reward function. In particular, we seek to integrate structural and interpretability constraints into the reinforcement learning from human feedback (RLHF) pipeline to improve diagnostic capabilities for misalignment issues. A natural step towards both of these goals is to combine the expressiveness of neural networks with an architecture choice that is easier for a human to interpret, such as a decision tree. To tackle the the aforementioned problems, we propose a novel reward learning approach that uses an end-to-end differentiable decision tree model for learning interpretable reward functions from pairwise preferences. We evaluate our approach on three different domains: CartPole (Brockman et al., 2016), a novel set of Visual MNIST Gridworld environments, and two Atari games from the Arcade Learning Environment (Bellemare et al., 2013). We investigate the ability to learn expressive and interpretable reward functions from both low- and high-dimensional state inputs.

Learning a reward model as a differentiable decision tree has the advantage that the tree structure explicitly breaks the reward prediction for a state into a finite number of routing decisions within the tree. This provides the potential to understand how the reward predictions are being made. Leveraging the tree structure, we can provide global explanations across both low- and medium-dimensional environments such as CartPole and visual MNIST gridworlds. For high-dimensional visual state spaces, such as Atari, we propose a novel form of hybrid explanation that seeks to provide global explanations by leveraging aggregations of individual input states.

Our paper makes the following contributions: (1) We introduce a reward learning framework (Fig 1) that employs differentiable decision trees (DDTs) to learn human intent using trajectory preference labels without necessitating any hand-crafting of the input feature space. *To the best of our knowledge, our framework is the first interpretable tree-based method for reward learning that can be applied in visual domains.* (2) We propose hybrid explanations for internal nodes that approximate global explanations by leveraging aggregations of individual input states. (3) We study the ability of DDTs to learn interpretable rewards on visual-control tasks and find that Reward DDTs can often learn interpretable reward functions. We also provide evidence that reward DDTs can be used to identify reward misalignment. (4) We find that the policies obtained by optimizing our reward DDTs via RL often perform comparably to policies trained with black-box neural network reward functions.

2 Related Work

Preference Learning Reinforcement learning from human feedback (RLHF), is a common approach for learning reward functions and corresponding RL policies (Wirth et al., 2016). It has been shown that preference learning allows generalizing to various domains, even when sub-optimal demonstrations are provided without any explicit preferences and can achieve better-than-demonstrator performance (Brown et al., 2019). Preference learning is also applicable across multiple forms of human input: prior work has shown that demonstrations (Brown et al., 2020), e-stops (Ghosal et al., 2023a), rankings (Ouyang et al., 2022), and corrections (Mehta and Losey, 2022), can all be represented in terms of pairwise preferences. Thus, our approach is also applicable in these other settings. Prior work on RLHF typically either assumes access to a set of hand-designed reward features (Sadigh et al., 2017; Biyik et al., 2020; Mehta and Losey, 2022; Ghosal et al., 2023a) or uses deep convolutional or fully connected networks for reward learning (Christiano et al., 2017; Brown et al., 2019; Lee et al., 2021a; Hejna and Sadigh, 2022; Ouyang et al., 2022; Liu et al., 2023; Karimi et al., 2024). By contrast, we study the extent to which we can learn expressive, but also interpretable reward functions via differentiable decision trees (Frosst and Hinton, 2017).

Explaining and Interpreting Reward Functions In the past few years, various attempts have been made to understand learned reward functions. Prior work compares learned reward functions to a ground truth reward using pseudometrics (Gleave et al., 2021), saliency maps and counterfactuals (Brown et al., 2019; Michaud et al., 2020; Mahmud et al., 2023; Tien et al., 2023). Other work leverages human teaching strategies (Lee et al., 2021b; Booth et al., 2022) or uses human-centric evaluation methods for reward explanation (Sanneman and Shah, 2022). Prior work has also looked at using expert-driven reward design techniques to incorporate structural and interpretability constraints (Jiang et al., 2021; Devidze et al., 2021; Icarte et al., 2022). We seek to investigate to what extent differentiable decision trees enable interpretable reward functions.

Differentiable Decision Trees Differential decision trees (DDTs) seek to combine the flexibility of neural networks with the logical and interpretable structure of decision trees (Quinlan, 1986; Jordan, 1994). DDTs have been previously applied to supervised learning tasks (Frosst and Hinton, 2017; Tanno et al., 2019; Hazimeh et al., 2020) and unsupervised tasks (Zantedeschi et al., 2021). Recent work has also investigated using DDTs for reinforcement learning tasks (Silva et al., 2020; Coppens et al., 2019; Tambwekar et al., 2023; Ding et al., 2021; Pace et al., 2022), but focuses on *policy learning* using DDTs. Compared to prior work, the primary objective of our work is to *learn interpretable reward functions* using DDTs. While policy explanations are important, they only show what triggers an agent to take a certain action, rather than explaining the underlying reason why the policy has learned to take take an action. By understanding agent's reward function, we gain insight into the agent's value alignment (Leike et al., 2018; Fisac et al., 2020; Brown et al., 2021). Importantly, understanding an agent's reward function can enable an understanding of how that agent would act across different embodiments and dynamics Fu et al. (2018); Zakka et al. (2022), unlike policies which are tied to the specifics of the MDP transition dynamics and action space. Furthermore, prior work using DDTs for policy learning only considers low-dimensional, non-visual inputs (Silva et al., 2020; Coppens et al., 2019). By contrast, we study DDTs applied to high-dimensional image observations.

Decision Trees for Reward Learning There has been very little prior work on using decision trees for reward learning. Bewley and Lecue (2022) recently pioneered the idea of a tree-based reward function. However, their approach to learning a tree-based reward requires a complex, non-differentiable, multi-stage optimization procedure. By contrast, our approach is end-to-end differentiable and trainable using a simple cross entropy loss. Bewley and Lecue (2022) also only consider low-dimensional inputs where internal nodes in tree have the form $(s, a)_d \geq c$ for each dimension d of the state-action space and threshold c. This approach divides state-action space into axis aligned hyperrectangles, which often works for lower-dimensional spaces, but does not scale to higher-dimensional state and action spaces. Follow-on work (Bewley et al., 2023) uses a differentiable loss function but is not end-to-end differentiable as it requires reward tree to regrow at each update and requires hand crafting input features per decision node in the tree, making it intractable to scale to the types of visual inputs we consider. We seek to extend the state-of-the-art in interpretable tree-based reward learning by learning reward function DDTs that are end-to-end differentiable, do not require hand-crafted features, and scale easily to high dimensional pixel inputs.

3 Reward Learning using Differentiable Decision Trees

Classical decision trees are often interpretable and easy to tune (Kotsiantis, 2013; Molnar, 2020); however, they require feature engineering which can result in lower performance and less generalization compared with other machine learning approaches (Frosst and Hinton, 2017; Hazimeh et al., 2020). In this section, we discuss our proposed approach for learning interpretable but expressive reward functions via differentiable decision trees (DDTs).

While classical decision trees consist of internal nodes that deterministically route inputs, we want our reward function tree to be easily trained using backpropagation. Thus, we need a differentiable soft routing function that retains the expressiveness of a neural network by learning the routing function for each non-leaf node. We define an internal node in the DDT as a sequence of one or more parameterized functions applied, to the input to the DDT to determine probability of routing left or right. To facilitate interpretability, each internal node depends directly on the input—this is a common design choice in DDTs (Frosst and Hinton, 2017) and serves our purpose well by allowing us to easily trace each routing decision in the tree to the raw input features. Thus, the differentiable decision tree learns a hierarchy of decision boundaries that determine the routing probabilities for each input. We describe two variants of an internal node below:

3.1 Internal Nodes

Simple Internal Node Proposed by Frosst and Hinton (2017), a simple internal routing node, i, has a linear layer with learnable parameters $\mathbf{w}_i$ and a bias term b upon which a sigmoid activation function, σ, is applied to derive the routing probability given an input $\mathbf{x}$ (Fig 2). Thus, the probability at node i of routing to the left branch is defined as $p_i(\mathbf{x}) = \sigma(\beta(\mathbf{x} \cdot \mathbf{w}_i + b))$. The inverse temperature parameter, β, controls the degree of soft decisions.

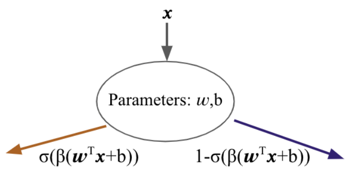


Figure 2: Routing probability of an internal node in a DDT.

Sophisticated Internal Node For higher-dimensional inputs we propose an alternative internal node architecture, which consists of a single convolutional layer with Leaky ReLU as the non-linearity followed by a fully connected linear layer, as before. The probability of going to the leftmost branch at an internal node i is defined as $p_i(\mathbf{x}) = \sigma((\text{LeakyReLU}(\text{Conv2d}(\mathbf{x}))) \cdot \mathbf{w}_i + b)$.

3.2 Leaf Nodes

Following prior work that uses DDTs for classification problems (Frosst and Hinton, 2017), we parameterize each leaf node, l, with a learnable parameter vector $\boldsymbol{\phi}^l$, that defines a softmax distribution over a discrete number of classes c. The probability distribution, $\boldsymbol{Q}^l$, over outputs at a leaf is defined as $\boldsymbol{Q}^l_i = \exp(\boldsymbol{\phi}^l_i)/(\sum_{j=0}^{c} \exp(\boldsymbol{\phi}^l_j))$. We propose two ways to obtain rewards at the leaf nodes:

Multi-Class Reward Leaf (CRL) This formulation of leaf node performs *multi-class classification* and assumes that the user specifies a set of c unique discrete reward values that the DDT can output in the form of a vector $\mathbf{R} = (r_1, r_2, \dots, r_c)$, where c denotes the number of classes for the DDT, and each class index i is assigned reward value r_i. Thus, the learnable parameters, $\boldsymbol{\phi}^l$, at multi-class reward leaf l form the logit values of a classification problem over the possible reward values in $\mathbf{R}$.

Min-Max Reward Interpolation Leaf (IL) As an alternative to the classification approach, we also propose to model the reward of a DDT as *regression problem*, that only requires the user to specify the minimum and maximum range of possible reward values as opposed to requiring finite set of possible reward values as in CRL. Thus, $c = 2$ and the reward vector is of the form $\mathbf{R} = (R_{\min}, R_{\max})$, where $R_{\min}$ and $R_{\max}$ correspond to minimum and maximum desired reward output, respectively. Given this parameterization, we interpret the reward output of a DDT leaf node as a convex combination of $R_{\min}$ and $R_{\max}$ based on the learned parameters $\boldsymbol{\phi}^l$.

3.3 Training DDTs for Reward Learning using Human Preferences

As we want our reward DDT to be end-to-end differentiable when learning a reward function from preference labels, we need to find a way to formulate soft reward prediction. Given a tree of depth $d \geq 1$, we have $\sum_{k=0}^{d-1} 2^k$ internal nodes and 2^d leaves. To formulate a differentiable objective, we first denote the path probability, given an input $\mathbf{x}$, from the root node to a leaf ℓ by $P^\ell(\mathbf{x})$. The soft reward prediction of the tree is given by the sum over all leaves, ℓ, of the path probability of reaching each leaf, $P^\ell(\mathbf{x})$, multiplied with the soft reward output at that leaf:

$$r_\theta(\mathbf{x}) = \sum_\ell P^\ell(\mathbf{x})(\mathbf{Q}^\ell \cdot \mathbf{R}) \ .$$

To train our reward function DDT, we propose to leverage pairwise preference labels over trajectories. Given preferences over trajectories of the form $\tau_i \prec \tau_j$, where $\tau = (\mathbf{x}_1, \mathbf{x}_2, ...\mathbf{x}_T)$, we can train our entire differentiable decision tree via the following cross entropy loss resulting from the Bradley Terry model of preferences (Bradley and Terry, 1952; Christiano et al., 2017; Brown et al., 2019):

$$\mathcal{L}(\theta) = -\sum_{\tau_i \prec \tau_j} \log \frac{\exp \sum_{\mathbf{x} \in \tau_j} r_\theta(\mathbf{x})}{\exp \sum_{\mathbf{x} \in \tau_i} r_\theta(\mathbf{x}) + \exp \sum_{\mathbf{x} \in \tau_j} r_\theta(\mathbf{x})} \ .$$

3.4 Using a Trained Reward DDT for Reward Prediction

Given a trained reward DDT, we want to optimize the learned reward using RL. One option is to use the soft reward (averaged across all leaf nodes weighted by routing probability); however, this loses interpretability since we cannot trace the predicted reward to a small number of discrete decisions. To enable interpretable reward predictions, we can alternatively output a single reward prediction by first finding the leaf node with maximum routing probability for a given input $\mathbf{x}$:

$$l^* = \arg\max_{\ell \in L} P^\ell(\mathbf{x}) \ ,$$

where L denotes set of all leaf nodes in the DDT. The test-time output of a reward DDT with a multi-class reward leaf (CRL) nodes is given as $r_{max}(\mathbf{x}) = r_i, \text{ for } \ i = \arg\max_i \mathbf{Q}_i^{\ell^*}$; while for a reward DDT with min-max interpolation leaf (IL) nodes the reward output is given as $r_{max}(\mathbf{x}) = \mathbf{Q}^{\ell^*} \cdot (R_{\min}, R_{\max})$.

3.5 Hybrid Explanations of Learned Reward DDT

Depending on the dimensionality of the state space in a given environment, our framework allows us to create global explanations across all inputs in form of node activation heatmaps (discussed in further detail later). As an alternative, we also investigate hybrid explanations that approximate global explanations by leveraging aggregations of input states to visually understand the routing probability of each internal node. Inspired by Bobu et al. (2022), we do this by visualizing a synthetic trace at each internal node. The synthetic trace is a sequence of states sorted by the probability of being routing left in decreasing order—the trace begins with the state that has maximum probability of being routed left and ends with the state that has minimum probability of being routed left.

4 Experiments and Results

We designed our experiments to investigate the following questions: (1) Can we detect misalignment in reward function by learning the reward function as a DDT? (2) How does modeling a reward function as a DDT influence downstream RL performance? (3) How does the choice of leaf node (multi-class reward leaf (CRL) or min-max reward interpolation leaf (IL)) affect performance? (4) How does an increase in the environment complexity impact our design choices as well as our ability to interpret the learned reward function? To explore and address these questions, we perform evaluations on three different types of environments: CartPole, a novel set of MNIST Gridworld environments, and Atari 2600 games (Bellemare et al., 2013).

We use CartPole to perform an initial assessment of our framework and provide an example of how interpreting a learned reward DDT enables detection of a *silent misalignment problem*—the reward function is misaligned but the policy still performs well. To evaluate our framework's ability on visual domains, we explore two MNIST Gridworld environments of increasing complexity, where the gridworlds have image based observations. Finally we examine our framework on Atari where the true score is masked and the agent must learn a reward function by interpreting high-dimensional pixel observations derived from video frames. The Atari domains provide evidence in higher-dimensional environments of the ability to detect reward misalignment.

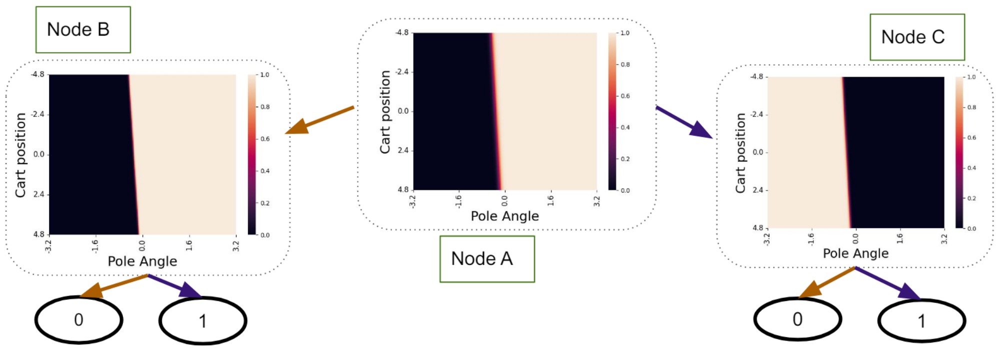


Figure 3: **Identifying Misalignment in the CartPole Reward DDT.** The heatmap for each internal node depicts the learned routing probability. Leaf nodes are depicted as circular nodes with their soft reward values. The tree learns that small magnitude pole angles are good and should be routed to a +1 reward but there is no learned decision boundary that clearly captures the preference that cart position stay within the range $[-2.4, 2.4]$ showing that learned reward is misaligned due to the bias in the training dataset—the cartpole usually falls over long before the cart runs off the track.

4.1 CartPole

The CartPole environment comprises a cart with a pole attached to it, sliding on a friction-less track (Brockman et al., 2016). For this task, we wish to teach the agent to balance the pole on the cart for as long as possible while cart moves to left and right along the track without letting pole fall beyond $\pm 12°$ from the upright position and without letting the cart move beyond ± 2.4 units along the track. We assume no access to the true reward and must learn this from trajectory preferences.

Setup To train a reward function DDT, we generate trajectories by running a random policy in the environment for 200 steps for each trajectory. Following the advice of Freire et al. (2020), we remove the standard terminal or done flag to avoid leaking information about the true reward. The terminal flag normally is triggered in CartPole when either the pole falls or the cart goes off the track. Instead, we make CartPole a fixed horizon task by always accumulating states in each trajectory for 200 timesteps—even if the pole falls over. We design a synthetic preference labeler that returns pairwise preferences based on the true (but unobserved) reward of +1 only if the cart position $x \in [-2.4, 2.4]$ and the pole angle $\theta \in [-12°, +12°]$ and 0 otherwise. Pairwise preferences are assigned based on the true reward for each trajectory.

Given pairwise preference labels over suboptimal trajectories, we train a reward DDT with 3 internal nodes and 4 leaf nodes. We use multi-class reward leaf (CRL) nodes with 2 classes: $\mathbf{R} = (0.0, 1.0)$ (for more details, refer to Appendix B). It is important to note that even though the ground truth preferences are based on both cart position and pole angle, the pole usually falls past the desirable range long before the cart leaves the desirable range. Thus, our dataset is biased and may lead to a misaligned reward function. We evaluate RL performance of the learned reward DDT, by running PPO on the learned reward function to obtain the final policy and then evaluate this learned policy on the ground-truth reward function. We also compute the performance of a PPO policy trained on the same dataset using a neural network reward function. To unveil the fact that our learned reward functions (using both DDT and neural network) are biased, we run RL experiments in two settings: (1) *In-Distribution* uses the default starting cart position in the range $[-0.05, 0.05]$ as in our training dataset and (2) *Out-Of-Distribution* where the starting cart position is in the range $[2.35, 2.45]$ (the boundary of the range of desired track positions).

Results The In-Distribution results in Table 1 show that RL performance of a simple reward DDT is comparable to that of a neural network made up of fully-connected layers as well as to RL policy learned under ground truth reward, irrespective of whether the policy is learned using soft rewards or using the maximum probability path across the learned reward DDT. This primarily gives us

		DDT		Baselines	
		CRL Soft	CRL Argmax	Neural Network	Ground Truth
In-Distribution	Mean (Std)	190.9 (28.1)	200.0 (0.0)	156.3 (59.0)	200.0 (0.0)
	IQM	200.0	200.0	179.5	200.0
Out-Of-Distribution	Mean (Std)	8.8 (3.7)	7.7 (2.1)	20.7 (39.2)	172.0 (45.6)
	IQM	8.3	7.9	8.8	185.3

Table 1: **Silent Misalignment in CartPole.** CRL denotes Class Reward Leaf nodes. For In-Distribution, DDTs with soft outputs and argmax rewards perform on par with a non-interpretable fully connected 2-layer reward network baseline and with RL policy learned under ground truth reward. For Out-Of-Distribution, the RL policy of learned reward models, both DDT and neural network fails to learn to balance pole while moving along the track while RL policy under ground truth reward learns to balance pole as it moves on track. The table shows Mean and Standard deviation across 10 seeds averaged over 100 rollouts as well as the Interquartile Mean (IQM).

the evidence that our framework can achieve relatively competitive performance as that of a neural network for state based observations,before we move on to image-based observations.

Fig 3 shows learned reward DDT. Because the input space to the reward function is 2-dimensional (cart position and pole angle) we visualize the heatmap of routing probability at each internal node (as a function of cart position and pole angle) along with leaf distributions. From DDT it is clear that most of the routing decisions are made based on pole angle, rather than cart position. A nice feature of the reward DDT is that we can easily visually interpret the learned reward just by looking at the tree. From Fig 3 we see that while the tree learns that small magnitude pole angles are good and should be routed to a +1 reward, there is no learned decision boundary that clearly captures the preference that cart position stay within the range $[-2.4, 2.4]$. We call this a *silent misalignment problem*. Similar to a silent bug in programming, it is not obvious by running RL that anything is wrong with the learned reward function—it turns out that trying not to tip the pole is a decent surrogate reward function that works well in the standard CartPole environment. Thus, the agent has learned the right policy for the wrong reason, something that is only clear by interpreting the learned reward. While this poses no serious issues in the standard CartPole environment, silent alignment problems could lead to unwanted behavior under distribution shifts and detecting these silent alignment problems is an open challenge in AI safety and alignment research (Ji et al., 2023).

Indeed, the Out-Of-Distribution results in Table 1 demonstrate this silent misalignment in the learned reward functions, where the policies learned from the reward DDTs as well as neural network reward learn to balance the pole, but fail to stay in the desired track range. In contrast, the RL policy under ground truth reward learns to balance pole correctly while moving along the track, starting from any state. Our DDT framework makes it easier to detect this misalignment in learned reward function prior to running RL, but with non-interpretable black box neural network's learned reward function we had to incur cost of running RL before we could uncover the bias in the learned reward.

4.2 MNIST Gridworlds

Next we evaluated our reward DDT framework on two novel MNIST gridworld environments of increasing difficulty. In each environment the agent can move in the 4 cardinal directions and each state is associated with a 28×28 grey-scale image of the MNIST digit and the value of the digit determines the true unobserved reward at that state (for more details, refer to Appendix C) .

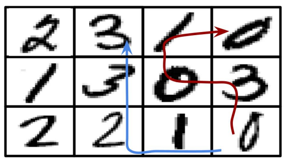
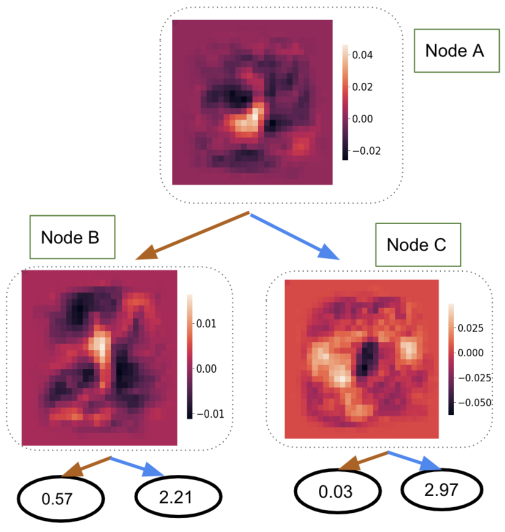
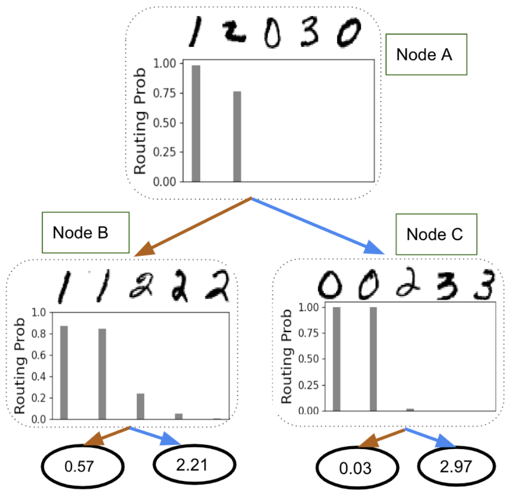


Figure 4: MNIST Gridworld with a pair of trajectories where the blue trajectory is preferred.

The true reward is unobserved and must be inferred from preferences over pairwise preferences over trajectories. To interpret the learned reward DDT, we construct a pixel-level activation heatmap for each internal node by starting with a blank image and iteratively toggling on and off each pixel and computing the resulting difference in routing



(a)



(b)

Figure 5: **Interpreting the MNIST (0-3) Interpolated Leaf Reward DDT**. Leaf nodes are depicted as circular nodes with their learned reward values. **(a)** Visualization of activation maps give insight into the learned routing of the DDT. **(b)** Visualization of synthetic traces along with their respective routing probabilities.

	Reward DDT				Baselines	
	CRL Soft	CRL Argmax	IL Soft	IL Argmax	NNet	Random
MNIST 0-3	71.7%	71.7%	98.9%	97.8%	99.5%	7.6%
MNIST 0-9	79.6%	79.6%	97.3%	92.9%	97.7%	7.9%

Table 2: RL Performance as the percentage of expected return obtained relative to the performance of an optimal policy on the ground-truth reward. Results are averaged across 100 different MDPs. We find evidence that reward DDTs with Interpolated Leaf nodes (IL) perform similar to neural network reward functions, while using Class Reward Leaf nodes (CRL) results in much lower performance, but still outperforms a random policy (Random). These results provides evidence that DDTs can learn both interpretable reward functions without causing a large degradation in RL performance.

probabilities for each internal node. We compare the performance of a policy optimized using the learned DDT reward function against the optimal policy under the true reward, a random policy, and a policy learned by optimizing a black-box neural network reward function trained on the same preference dataset. We also report accuracy of the learned reward models on validation set of pairwise preferences over trajectories in Appendix C.

4.2.1 MNIST (0-3) Gridworld

Setup We begin by examining our framework for image based inputs on a simple 5x5 gridworld where each state in the MDP corresponds to a MNIST digit 0, 1, 2, or 3 (see Fig 4 for an example pairwise trajectory comparison). We trained reward DDTs of depth 2 with 3 simple internal nodes and 4 leaf nodes either all of type CRL with $\mathbf{R} = (0, 1, 2, 3)$ and or all of type IL with $R_{\min} = 0$ and $R_{\max} = 3$ using a learning rate of 0.001 and weight decay 0.005 and the Adam optimizer.

Results We visualized and compared the reward DDTs with CRL and IL leaf nodes and found that in CRL formulation the leaf nodes fail to specialize and the argmax output of the leaf nodes is either 0 or 3, despite investigating several regularization techniques (see Appendix E for details and visualizations). This provides evidence that using IL leaf nodes is better when learning complicated reward functions where we wish to output more than two possible rewards. Table 2 also provides empirical evidence supporting the user of IL nodes. IL nodes are also simpler, as they only require specifying a range of desired reward output values, $[R_{\min}, R_{\max}]$. Thus, we focus on our analysis on the interpretability of the IL reward DDT.

Interestingly, we see in Fig 5a that the activation heatmaps isolate pixel features that are maximally discriminative and aid in understanding of what the reward DDT has learned. These heatmaps show that DDT learns to route based on visual representations of each digit: Node B has learned to discriminate between 1's and 2's by assigning a high routing probability left for vertical pixels in the center (corresponding to the vertical stroke of the digit 1), while using upper and lower curves of digit 2 to route 2's right (note the black shadow that looks like a 2). Node C discriminates between digits 0 and 3 based on the middle cusp of 3 and left curve of the 0. Finally, node A learns to route 1's and 2's left and 0's and 3's right based on the presence of central lower pixels—the highest activation for node A is intersection of the 1 and 2 which falls between middle and lower cusps of 3 and inside digit 0. Despite the lack of fine-grained feedback and no explicit reward labels, when using min-max reward interpolation between $R_{min} = 0$ and $R_{max} = 3$, the DDT learns a close approximation to the actual state rewards and the learned rules in DDT are visually interpretable.

We also interpret the same reward DDT using synthetic traces as shown in Fig 5b. As described in Section 3.5, these traces approximate global explanation by leveraging aggregations of input states for each internal node. Each trace is a sequence of states sorted by the probability of being routed left in decreasing order. We find visual evidence that the DDT has learned to route digits with a vertical stroke to Node B which then discriminates between 1 and 2s, while digits with a circular curve form often get routed to Node C which then discriminates between digits 0, 2 and 3s.

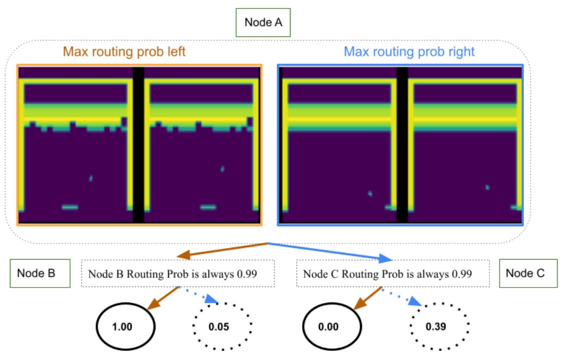
Row 1 of Table 2 shows that RL performance of using a reward DDT trained with IL leaf nodes exceeds the performance when using the classification-based CRL leaf nodes, both when running RL using soft reward outputs and when using the output of the maximum probability path in the tree (argmax). We also found that the learned reward DDT with CRL leaf nodes learns very high/low routing probabilities at each internal node and thus yields nearly identical reward values in both soft and argmax reward setting. Moreover, RL performance of IL reward DDT using soft reward is only slightly lower than the performance of a deep neural network reward function. In Appendix E, we compare the reward DDT in Fig 5a, that is learned from pairwise preferences, with a DDT trained with explicit reward labels and a classification loss and find no significant degradation in interpretability from using pairwise preferences.

4.2.2 MNIST (0-9) Gridworld

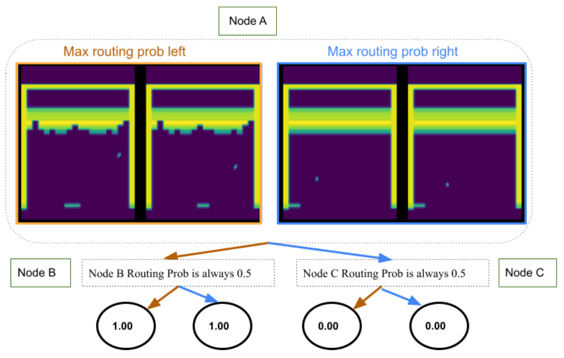
Setup To assess the scalability of our framework, we next explored a 10x10 gridworld with state space comprising of MNIST digits 0 to 9. To further study the effects of leaf node type, we used reward DDTs of depth 4 with simple internal nodes and trained them with one of two types of leaf nodes: either CRL nodes with $\mathbf{R} = (0, 1, .., 9)$ or IL nodes with $R_{\min} = 0$ and $R_{\max} = 9$.

Results Row 2 of Table 2 shows the IL soft reward performance is very similar to the performance of a black-box ConvNet learned reward. However, we find that performance of CRL softmax and argmax is significantly degraded, but much better than a random policy. This provides further evidence simply framing DDT learning as a classification problem is in sufficient for learning good reward function and that the flexibility of interpolation leaf nodes (IL) to learn real valued reward outputs helps with both interpretability and downstream RL performance.

This provides evidence that our framework maintains high performance for much longer horizon and more difficult tasks when using interpolated leaf nodes (IL). Even though tree structures can help with interpretability, the deeper the tree, the harder it is to understand what is going on. In Figure 13 in the Appendix, we visualize the learned IL Reward DDT. While there are some noticeable trends, it is also hard to interpret exactly how the DDT has learned to route nodes. Thus, while our results provide evidence that high-performing policies can be learned via RLHF using reward DDTs, the more complex the DDT, the more difficult it is to interpret.

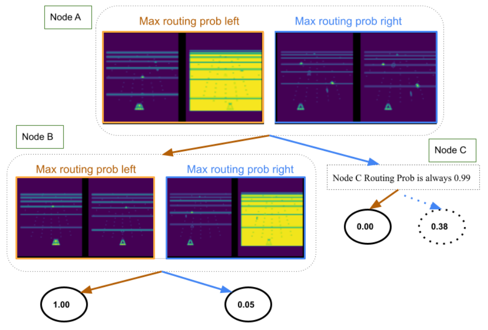


(a) DDT without activation penalty regularization

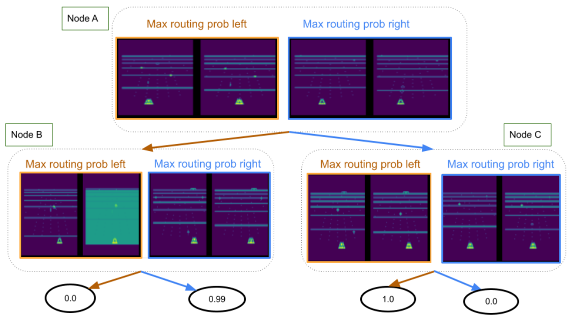


(b) DDT with activation penalty regularization

Figure 6: **Visualization of Breakout Reward DDTs**. We plot the DDTs trained without (a) vs with (b) a regularization penalty on the internal node routing probabilities. Dashed lines denote leaf nodes that are never reachable.



(a) DDT without activation penalty regularization



(b) DDT with activation penalty regularization

Figure 7: **Visualization of Beam Rider Reward DDTs**. We plot the DDTs trained without (a) vs with (b) a regularization penalty on the internal node routing probabilities.

4.3 Atari

As a final test of the efficacy and scalability of learning interpretable rewards via DDTs, we trained reward DDTs on the Beam Rider and Breakout Atari games (Bellemare et al., 2013). Learning rewards for these games is challenging as the states are high-dimensional pixel inputs consisting of stacks of four 84×84 video frames and many prior works have used Atari games to study reward function learning (Christiano et al., 2017; Tucker et al., 2018; Ibarz et al., 2018; Brown et al., 2019).

Setup To train our reward DDT, we used the open-source offline preference datasets collected by Brown et al. (2019)[1]. We then examine whether a reward DDT can match the RL performance of T-REX, a deep convolutional neural network offline RLHF approach proposed by Brown et al. (2019), while also being interpretable. Because of the complexity of the task, we use sophisticated internal nodes and IL leaf nodes with $R_{\min} = 0$ and $R_{\max} = 1$ (see Appendix G for full details).

Because generated heatmaps for Atari, have been shown to have mixed results (Brown et al., 2019), we opt to use traces for interpreting the learned reward DDTs. As before, the trace for an internal node begins with the state that has maximum probability of being routed left and ends with the state that has minimum probability of being routed left. For ease of visualization, we show the first and last state in the trace and find that they still provide useful information about the internals of the learned reward function.

[1]https://github.com/hiwonjoon/ICML2019-TREX

	DDT				Baseline
Game	¬penalty ¬argmax	¬penalty argmax	penalty ¬argmax	penalty argmax	T-REX
Breakout: Mean(Std)	20.2 (34.4)	50.0 (105.3)	83.5 (130.9)	51.5 (100.6)	58.3 (42.4)
Breakout: IQM	12.8	16.6	29.3	15.0	48.9
Beam Rider: Mean (Std)	237.2 (322.2)	189.4 (284.6)	39.9 (40.7)	107.6 (288.2)	323.1 (335.9)
Beam Rider: IQM	94.9	64.1	30.2	7.6	254.9

Table 3: **Reinforcement learning using reward DDTs.** We report mean, standard deviation (Std) and inter-quartile mean (IQM) across 10 different seeds of RL evaluated for 100 epsiodes each.

For each learned reward, we optimized a policy by training an A2C (Mnih et al., 2016) agent using Stable Baselines 3 for 10 million timesteps. As done in our previous experiments, when training the RL agent, we utilize each learned reward DDT in two ways: we either obtain a soft reward over all leaves from tree or we choose the path with maximum routing probability and the reward in this case is obtained by argmaxing over the maximum probability path. We report mean and standard deviation across 10 seeds evaluated for 100 episodes each as well as the inter-quartile mean (IQM), which has been proposed as a better alternative when evaluating smaller numbers of seeds as recommended by prior works (Patterson et al., 2023; Agarwal et al., 2021).

Results

While trying to create synthetic traces, we discovered that some of the leaf nodes were never reachable (e.g., the right child of Node B and C in case of Breakout Fig 6a and the right child of Node C in case of BeamRider Fig 7a). We re-trained the sophisticated reward DDT with the same hyperparameters, but with an added penalty regularization to ensure that, on average across many inputs, each internal node routes left and right equally often across both environments (see Appendix A for details).

We create a synthetic trace for the unregularized reward DDT for Breakout Fig 6a by visualizing states that are routed with maximum and minimum probability to left and found evidence that states that have more bricks missing are routed left to Node B while the states in Node A that have few bricks missing have a lower routing probability and thus are routed right to Node C. Both child nodes of the root node only use their respective left leaves and do not route any state to their respective right leaves, thus a synthetic trace could not be visualized for either Node B or Node C. Fig. 6b shows a similar trend, where the DDT learns to reward missing bricks. Interestingly, we did not find any evidence that the reward DDT learned to recognize the event of the ball hitting a brick. Instead of learning the causal ground truth reward that provides a reward each time a brick is hit, the reward DDT exhibits causal confusion (Tien et al., 2023) by learning to reward missing bricks. Similar to CartPole, we describe this as a “silent misalignment problem". The reward function has learned to reward the wrong thing but this actually leads to behavior that appears aligned based on RL performance in distribution.

We similarly visualize traces for each internal node in the sophisticated reward DDT trained for the Beam Rider game without penalty (Fig 7a) and compare it against that of reward DDT trained using penalty (Fig 7b). In Fig 7a we see that Node A routes states where agent hits an enemy ship to the left and states where it misses enemy ships to right. Then Node B routes states where it looks like it will hit an enemy ship to a reward of 1.0 but interestingly routes states where it has hit an enemy ship to a reward of 0 (yellow flash indicates an enemy being destroyed). This allows us to see a misalignment in the learned reward function. We investigated this further and found that when the agent loses a life, this also triggers a flashing yellow screen. Thus, the agent appears to be misinterpreting the yellow flash and associating it with losing a life, when it should be associated with a good reward for destroying an enemy ship. We also created traces for reward DDT trained with regularization penalty on routing probabilities for Breakout and BeamRider. We observe similar trends of misalignment (Node B) of the learned regularized reward DDT for BeamRider (Fig 7b).

In Table 3 we summarize learned policy performance under 4 different scenarios (without penalty and without argmax (returning soft reward averaged over all leaf nodes), without penalty and with argmax, with penalty and without argmax, with penalty and with argmax) for both Beam Rider and Breakout along with T-REX performance on each of these games. Our results in terms of performance are mixed. We find evidence that using the soft reward output (¬argmax) of a DDT leads to the best RL performance. Interestingly, we observe that penalty regularization helps RL performance in case of Breakout but leads to degradation in RL performance in case of Beam Rider. However, in terms of IQM, the RL performance when optimizing the DDT rewards is not able to match the performance of the end-to-end neural network baseline.

For Beam Rider, we examined the learned RL policies for both reward DDTs and TREX and found that the misaligned reward did lead to misaligned behavior: agents across various seeds for both DDT and TREX move to one end of a screen, learn to stay alive, but never fire at the enemy ship, thus avoiding getting hit but also avoiding scoring points. Notably, in case of reward DDTs, both with and without penalty, we could detect this misalignment before running RL using our synthetic traces, but for TREX which use a dense black box neural network for learning reward, we could not diagnose this misalignment in reward function prior to running RL.

5 Discussion and Future Work

Our work provides mixed results regarding the utility of reward DDTs. On one hand, we provide evidence that reward DDTs are a viable alternative to end-to-end deep network rewards and can sometimes perform on-par with their deep neural network reward counterparts; however, for complex domains like Atari, the best performance comes at the cost of using DDT in a way that is not interpretable: using a soft reward output that is a weighted sum of outputs of all leaf nodes. Ideally, we could use reward DDTs with hard (argmax) reward outputs—the reward output during policy optimization would come from a single leaf node, allowing us to trace the reward output to a small number of binary routing decisions at the internal nodes. While optimizing this kind of hard output (argmax) process works well for the simpler domains we studied (e.g., CartPole and MNIST Gridworlds); it seems to hurt performance on more complex domains. We hypothesize this might be a result of the reward function being too sparse. Thus, our results reveal a tension between wanting highly shaped rewards to ensure good RL performance, while also wanting simple, non-shaped rewards to afford interpretability. Future work should investigate this trade-off in more depth.

In terms of interpretability, we find that for low dimensional tasks such as Cartpole and MNIST GridWorld environments, our framework is capable of providing global explanations that reveal interesting insights into the learned reward. For higher dimensional tasks such as Atari, we approximate global explanations by leveraging aggregations of local explanations by finding the input states that maximally and minimally activate the routing probability of each internal node. However, we also find that the deeper the DDT, the harder it becomes to interpret the learned reward. We also present evidence that demonstrates the practicality of using reward DDTs as a kind of *alignment debugger tool* to inspect learned reward functions for alignment with human intent. In particular, we provide evidence that reward DDTs can reveal cases of silent misalignment. By running policy optimization, we also find that baseline black-box neural network rewards are also misaligned. Importantly, the interpretability of a reward DDT reveals the silent misalignment without needing to run RL.

Future work should investigate using our framework to understand and interpret existing pre-trained neural network reward models that are known to lead to unintended consequences (Christiano et al., 2017; Ibarz et al., 2018; Javed et al., 2021; Tien et al., 2023) by distilling these networks into reward DDTs. Future work also includes investigating how to fix a known misaligned reward DDT by fine-tuning leaf and internal nodes based on human feedback, perhaps by using human-in-the-loop representation and feature learning (Bobu et al., 2022; 2023) or using methods for identifying causal features using small amounts of human annotations (Ghosal et al., 2023b). Future work shouldalso explore how to extend the ideas in this paper to transformer-based reward functions used in LLMs (Ouyang et al., 2022) and investigate the effects of increasing tree depth on interpretability.

Acknowledgments

This work has taken place in the Aligned, Robust, and Interactive Autonomy (ARIA) Lab at The University of Utah. ARIA Lab research is supported in part by the NSF (IIS-2310759), the NIH (R21EB035378), Open Philanthropy, and the ARL STRONG program.

References

Rishabh Agarwal, Max Schwarzer, Pablo Samuel Castro, Aaron C Courville, and Marc Bellemare. Deep reinforcement learning at the edge of the statistical precipice. *Advances in neural information processing systems*, 34:29304–29320, 2021.

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

Marc G Bellemare, Yavar Naddaf, Joel Veness, and Michael Bowling. The arcade learning environment: An evaluation platform for general agents. *Journal of Artificial Intelligence Research*, 47:253–279, 2013.

Tom Bewley and Freddy Lecue. Interpretable preference-based reinforcement learning with tree-structured reward functions. In *Proceedings of the 21st International Conference on Autonomous Agents and Multiagent Systems*, pages 118–126, 2022.

Tom Bewley, Jonathan Lawry, Arthur Richards, Rachel Craddock, and Ian Henderson. Reward learning with trees: Methods and evaluation, 2023. URL https://openreview.net/forum?id=xl2-MIX2DCD.

Erdem Biyik, Malayandi Palan, Nicholas C Landolfi, Dylan P Losey, Dorsa Sadigh, et al. Asking easy questions: A user-friendly approach to active reward learning. In *Conference on Robot Learning*, pages 1177–1190. PMLR, 2020.

Andreea Bobu, Marius Wiggert, Claire Tomlin, and Anca D Dragan. Inducing structure in reward learning by learning features. *The International Journal of Robotics Research*, 41(5):497–518, 2022.

Andreea Bobu, Yi Liu, Rohin Shah, Daniel S. Brown, and Anca D. Dragan. Sirl: Similarity-based implicit representation learning. In *Proceedings of the 2023 ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, 2023.

Serena Booth, Sanjana Sharma, Sarah Chung, Julie Shah, and Elena L Glassman. Revisiting human-robot teaching and learning through the lens of human concept learning. In *2022 17th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, pages 147–156. IEEE, 2022.

Ralph Allan Bradley and Milton E Terry. Rank analysis of incomplete block designs: I. the method of paired comparisons. *Biometrika*, 39(3/4):324–345, 1952.

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

Daniel S. Brown, Wonjoon Goo, Prabhat Nagarajan, and Scott Niekum. Extrapolating beyond suboptimal demonstrations via inverse reinforcement learning from observations. In *International conference on machine learning*, pages 783–792. PMLR, 2019.

Daniel S. Brown, Wonjoon Goo, and Scott Niekum. Better-than-demonstrator imitation learning via automatically-ranked demonstrations. In *Conference on robot learning*, pages 330–359. PMLR, 2020.

Daniel S. Brown, Jordan Schneider, Anca Dragan, and Scott Niekum. Value alignment verification. In *International Conference on Machine Learning*, pages 1105–1115. PMLR, 2021.

Paul F Christiano, Jan Leike, Tom B Brown, Miljan Martic, Shane Legg, and Dario Amodei. Deep reinforcement learning from human preferences. In *NIPS*, 2017.

Youri Coppens, Kyriakos Efthymiadis, Tom Lenaerts, Ann Nowé, Tim Miller, Rosina Weber, and Daniele Magazzeni. Distilling deep reinforcement learning policies in soft decision trees. In *Proceedings of the IJCAI 2019 workshop on explainable artificial intelligence*, pages 1–6, 2019.

Rati Devidze, Goran Radanovic, Parameswaran Kamalaruban, and Adish Singla. Explicable reward design for reinforcement learning agents. *Advances in Neural Information Processing Systems*, 34: 20118–20131, 2021.

Zihan Ding, Pablo Hernandez-Leal, Gavin Weiguang Ding, Changjian Li, and Ruitong Huang. Cdt: Cascading decision trees for explainable reinforcement learning, 2021.

Chelsea Finn, Sergey Levine, and Pieter Abbeel. Guided cost learning: Deep inverse optimal control via policy optimization. In *International conference on machine learning*, pages 49–58. PMLR, 2016.

Jaime F Fisac, Monica A Gates, Jessica B Hamrick, Chang Liu, Dylan Hadfield-Menell, Malayandi Palaniappan, Dhruv Malik, S Shankar Sastry, Thomas L Griffiths, and Anca D Dragan. Pragmatic-pedagogic value alignment. In *Robotics Research: The 18th International Symposium ISRR*, pages 49–57. Springer, 2020.

Pedro Freire, Adam Gleave, Sam Toyer, and Stuart Russell. Derail: Diagnostic environments for reward and imitation learning. *arXiv preprint arXiv:2012.01365*, 2020.

Nicholas Frosst and Geoffrey Hinton. Distilling a neural network into a soft decision tree. *arXiv preprint arXiv:1711.09784*, 2017.

Justin Fu, Katie Luo, and Sergey Levine. Learning robust rewards with adverserial inverse reinforcement learning. In *International Conference on Learning Representations*, 2018.

Gaurav R Ghosal, Matthew Zurek, Daniel S Brown, and Anca D Dragan. The effect of modeling human rationality level on learning rewards from multiple feedback types. *AAAI Conference on Artificial Intelligence*, 2023a.

Gaurav Rohit Ghosal, Amrith Setlur, Daniel S. Brown, Anca Dragan, and Aditi Raghunathan. Contextual reliability: When different features matter in different contexts. In *International Conference on Machine Learning (ICML)*, 2023b.

Leilani H Gilpin, David Bau, Ben Z Yuan, Ayesha Bajwa, Michael Specter, and Lalana Kagal. Explaining explanations: An overview of interpretability of machine learning. In *2018 IEEE 5th International Conference on data science and advanced analytics (DSAA)*, pages 80–89. IEEE, 2018.

Adam Gleave, Michael Dennis, Shane Legg, Stuart Russell, and Jan Leike. Quantifying differences in reward functions. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=LwEQnp6CYev.

Hussein Hazimeh, Natalia Ponomareva, Petros Mol, Zhenyu Tan, and Rahul Mazumder. The tree ensemble layer: Differentiability meets conditional computation. In *International Conference on Machine Learning*, pages 4138–4148. PMLR, 2020.

Donald Joseph Hejna and Dorsa Sadigh. Few-shot preference learning for human-in-the-loop RL. In *6th Annual Conference on Robot Learning*, 2022. URL https://openreview.net/forum?id=IKC5TfXLuW0.

Alexandre Heuillet, Fabien Couthouis, and Natalia Díaz-Rodríguez. Explainability in deep reinforcement learning. *Knowledge-Based Systems*, 214:106685, 2021.

Borja Ibarz, Jan Leike, Tobias Pohlen, Geoffrey Irving, Shane Legg, and Dario Amodei. Reward learning from human preferences and demonstrations in atari. *arXiv preprint arXiv:1811.06521*, 2018.

Rodrigo Toro Icarte, Toryn Q Klassen, Richard Valenzano, and Sheila A McIlraith. Reward machines: Exploiting reward function structure in reinforcement learning. *Journal of Artificial Intelligence Research*, 73:173–208, 2022.

Zaynah Javed, Daniel S. Brown, Satvik Sharma, Jerry Zhu, Ashwin Balakrishna, Marek Petrik, Anca D. Dragan, and Ken Goldberg. Policy gradient bayesian robust optimization. In *International Conference on Machine Learning (ICML)*, 2021.

Hong Jun Jeon, Smitha Milli, and Anca Dragan. Reward-rational (implicit) choice: A unifying formalism for reward learning. *Advances in Neural Information Processing Systems*, 33:4415–4426, 2020.

Jiaming Ji, Tianyi Qiu, Boyuan Chen, Borong Zhang, Hantao Lou, Kaile Wang, Yawen Duan, Zhonghao He, Jiayi Zhou, Zhaowei Zhang, et al. Ai alignment: A comprehensive survey. *arXiv preprint arXiv:2310.19852*, 2023.

Yuqian Jiang, Suda Bharadwaj, Bo Wu, Rishi Shah, Ufuk Topcu, and Peter Stone. Temporal-logic-based reward shaping for continuing reinforcement learning tasks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 7995–8003, 2021.

Michael I. Jordan. A statistical approach to decision tree modeling. In *Proceedings of the Eleventh International Conference on International Conference on Machine Learning*, ICML'94, page 363–370, San Francisco, CA, USA, 1994. Morgan Kaufmann Publishers Inc. ISBN 1558603352.

Zohre Karimi, Shing-Hei Ho, Bao Thach, Alan Kuntz, and Daniel S Brown. Reward learning from suboptimal demonstrations with applications in surgical electrocautery. *International Symposium on Medical Robotics (ISMR)*, 2024.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Sotiris B Kotsiantis. Decision trees: a recent overview. *Artificial Intelligence Review*, 39:261–283, 2013.

Victoria Krakovna, Jonathan Uesato, Vladimir Mikulik, Matthew Rahtz, Tom Everitt, Ramana Kumar, Zac Kenton, Jan Leike, and Shane Legg. Specification gaming: the flip side of ai ingenuity. *DeepMind Blog*, 2020.

Kimin Lee, Laura Smith, and Pieter Abbeel. Pebble: Feedback-efficient interactive reinforcement learning via relabeling experience and unsupervised pre-training. *arXiv preprint arXiv:2106.05091*, 2021a.

Michael S Lee, Henny Admoni, and Reid Simmons. Machine teaching for human inverse reinforcement learning. *Frontiers in Robotics and AI*, 8:693050, 2021b.

Jan Leike, David Krueger, Tom Everitt, Miljan Martic, Vishal Maini, and Shane Legg. Scalable agent alignment via reward modeling: a research direction. *arXiv preprint arXiv:1811.07871*, 2018.

Yi Liu, Gaurav Datta, Ellen Novoseller, and Daniel S Brown. Efficient preference-based reinforcement learning using learned dynamics models. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2921–2928. IEEE, 2023.

Saaduddin Mahmud, Sandhya Saisubramanian, and Shlomo Zilberstein. Reveale: Reward verification and learning using explanations. 2023.

Shaunak A Mehta and Dylan P Losey. Unified learning from demonstrations, corrections, and preferences during physical human-robot interaction. *arXiv preprint arXiv:2207.03395*, 2022.

Eric J Michaud, Adam Gleave, and Stuart Russell. Understanding learned reward functions. *arXiv preprint arXiv:2012.05862*, 2020.

Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *International conference on machine learning*, pages 1928–1937. PMLR, 2016.

Christoph Molnar. *Interpretable machine learning*. Lulu. com, 2020.

Andrew Y Ng, Daishi Harada, and Stuart Russell. Policy invariance under reward transformations: Theory and application to reward shaping. In *Icml*, volume 99, pages 278–287, 1999.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. *Advances in Neural Information Processing Systems*, 35: 27730–27744, 2022.

Alizée Pace, Alex J. Chan, and Mihaela van der Schaar. Poetree: Interpretable policy learning with adaptive decision trees, 2022.

Andrew Patterson, Samuel Neumann, Martha White, and Adam White. Empirical design in reinforcement learning. *arXiv preprint arXiv:2304.01315*, 2023.

J. Ross Quinlan. Induction of decision trees. *Machine learning*, 1:81–106, 1986.

Tilman Räukur, Anson Ho, Stephen Casper, and Dylan Hadfield-Menell. Toward transparent ai: A survey on interpreting the inner structures of deep neural networks. *arXiv preprint arXiv:2207.13243*, 2022.

Stuart Russell, Daniel Dewey, and Max Tegmark. Research priorities for robust and beneficial artificial intelligence. *Ai Magazine*, 36(4):105–114, 2015.

Dorsa Sadigh, Anca D Dragan, Shankar Sastry, and Sanjit A Seshia. Active preference-based learning of reward functions. In *Robotics: Science and Systems*, 2017.

Lindsay Sanneman and Julie A. Shah. An empirical study of reward explanations with human-robot interaction applications. *IEEE Robotics and Automation Letters*, 7(4):8956–8963, 2022. doi: 10.1109/LRA.2022.3189441.

Andrew Silva, Matthew Gombolay, Taylor Killian, Ivan Jimenez, and Sung-Hyun Son. Optimization methods for interpretable differentiable decision trees applied to reinforcement learning. In *International conference on artificial intelligence and statistics*, pages 1855–1865. PMLR, 2020.

Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

Pradyumna Tambwekar, Andrew Silva, Nakul Gopalan, and Matthew Gombolay. Natural language specification of reinforcement learning policies through differentiable decision trees. *IEEE Robotics and Automation Letters*, pages 1–8, 2023. doi: 10.1109/LRA.2023.3268593.

Ryutaro Tanno, Kai Arulkumaran, Daniel Alexander, Antonio Criminisi, and Aditya Nori. Adaptive neural trees. In *International Conference on Machine Learning*, pages 6166–6175. PMLR, 2019.

Jeremy Tien, Jerry Zhi-Yang He, Zackory Erickson, Anca Dragan, and Daniel S Brown. Causal confusion and reward misidentification in preference-based reward learning. In *International Conference on Learning Representations*, 2023.

Aaron Tucker, Adam Gleave, and Stuart Russell. Inverse reinforcement learning for video games. *arXiv preprint arXiv:1810.10593*, 2018.

Christian Wirth, Johannes Fürnkranz, and Gerhard Neumann. Model-free preference-based reinforcement learning. In *Thirtieth AAAI Conference on Artificial Intelligence*, 2016.

Kevin Zakka, Andy Zeng, Pete Florence, Jonathan Tompson, Jeannette Bohg, and Debidatta Dwibedi. Xirl: Cross-embodiment inverse reinforcement learning. In *Conference on Robot Learning*, pages 537–546. PMLR, 2022.

Valentina Zantedeschi, Matt J Kusner, and Vlad Niculae. Learning binary trees by argmin differentiation. *ICML*, 2021.

Quan-shi Zhang and Song-Chun Zhu. Visual interpretability for deep learning: a survey. *Frontiers of Information Technology & Electronic Engineering*, 19(1):27–39, 2018.

A DDT Routing Penalty Regularization

We take inspiration from [19] for adding penalty regularization and we first explain how penalty is defined at each internal node and then elaborate on calculating penalty for a single state over the whole DDT.

The cross-entropy between desired routing probability distribution of an internal node such that it's children nodes are equally used and the actual routing probability distribution is referred to as Penalty and is given by

$$\alpha_i = \frac{\sum_{\mathbf{x}} P^i(\mathbf{x}) p_i(\mathbf{x})}{\sum_{\mathbf{x}} P^i(\mathbf{x})}$$

where the probability of a current internal node is $p_i(\mathbf{x})$ and path probability from root node to an internal node is $P^i(\mathbf{x})$.

Penalty over the whole DDT for a single state is defined as sum over all internal nodes for the given input $\mathbf{x}$

$$C = -\lambda \sum_{i \in \text{ Inner Nodes}} 0.5 \log(\alpha_i) + 0.5 \log(1 - \alpha_i)$$

where hyper-parameter λ controls the strength of penalty λ in reward DDT so that the penalty strength is proportional to 2^{-d} and decays exponentially with depth of tree. Finally the penalty term for learning reward tree from pairwise preferences is calculated by taking the mean over all penalties for all states in the pairwise demonstrations.

B Cartpole

The baseline neural network is comprised of 2 fully connected layers, each of dimension 16 to learn the reward function. For reward models, both DDT and neural network, we use 2000 training and 200 validation pairwise preference demonstrations, each of length 20, with Adam optimizer and $lr = 0.001$ and weight decay= 0.

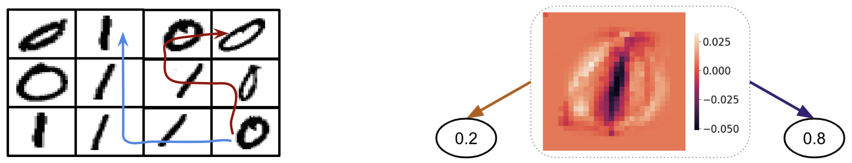
For running RL on ground truth reward as well as under learned reward models, we use Stable Baselines3 PPO with $batchsize = 1024, lr = 0.001, gaelambda = 0.8, gamma = 0.98, nepochs = 20, nsteps = 2048$ for 500000 total timesteps across 5 environments for both In-Distribution and Out-Of-Distribution starting cart positions.

In case of In-Distribution starting cart positions, we found that out of 10 seeds that we report results on ,3 seeds of RL policy learned under the neural network reward function seem to suffer from catastrophic forgetting leading a high standard deviation , with Mean and IQM, marginally lower than RL performance under ground truth reward function as well as performance of policices learned under soft reward and maximum probability path of DDT.

C MNIST Gridworld Additional Details

In this environment,the action space a contains 4 main actions: go left, go right, move up, move down. The transition function is stochastic and moves the agent in the direction chosen with an 80% probability as long as the action does not take it off of the grid. Actions that would result in leaving the grid result in a self transition.

And the neural network used to learn reward from pairwise human preferences consisted 2 convolutional layers with kernel size 7 and 5 respectively and stride 1 with LeakyRelu as the non-linearities followed by 2 fully connected layers.



(a) Pairwise trajectory preference (b) Visualization of Learned Reward DDT

Figure 8: **MNIST (0/1) Gridworld.** (a) A pair of trajectories with the same starting state, where the blue trajectory (which visits more 1's) is preferred over the red trajectory. (b) Heatmap of Learned Reward DDT : The dark pixels at center of heatmap form an approximate shape of digit 1 and are routed to right as the dark colors in heatmap mean that those pixels are turned off, while lighter pixels represent shape of digit 0 and routed to left as those pixels are turned on. Leaf nodes are depicted as circular nodes with their soft reward values.

In Table 4, we report the accuracy of learned reward DDTs with CRL and IL leaf nodes over pairwise preferences generated using the held-out validation dataset, both when using soft reward outputs and when using the output of the maximum probability path in the tree (argmax) for both type of our DDT and compare it against that of a convolutional neural network. Our results show that our IL DDTs can often achieve high accuracy despite not using any convolutional filters even on held out data. Using soft reward with IL leaf nodes offers comparable accuracy to that of CNN while using the reward from maximum probability path leads to a small decrease in accuracy, but still performs better than a DDT with CRL leaf nodes in both soft and argmax settings.

	Reward DDT				Baselines
	CRL Soft	CRL Argmax	IL Soft	IL Argmax	NNet
MNIST 0-3	77.57%	77.10%	97.75%	94.88%	99.21%
MNIST 0-9	72.71%	68.46%	90.83%	81.78%	92.40%

Table 4: Accuracy of the learned reward models on the 25000 pairwise preferences generated using the held-out validation dataset. Since MNIST 0-3 Gridworld is of size 5x5, we use trajectory length of 5 in the pairwise preferences while MNIST 0-9 Gridworld has size 10x10 , thus we use trajectory length 10.

D Additional domain: MNIST (0/1) Gridworld

We also show here an even simpler version of MNIST gridworld where there are only two possible digits. For training the reward DDT with simple internal nodes and CRL leaf nodes, we use a learning rate of 0.001, weight decay of 0.05, and the Adam optimizer (Kingma and Ba, 2014).

	Reward DDT				Baselines	
	CRL Soft	CRL Argmax	IL Soft	IL Argmax	NNet	Random
MNIST 0-1	92.37%	82.27%	99.98	100%	98.2%	7.38%

Table 5: RL Performance as the percentage of expected return obtained relative to the performance of an optimal policy on the ground-truth reward.

Setup We begin by examining our framework for image based inputs on the simplest gridworld environment. In this 5x5 gridworld each state in the MDP corresponds to a MNIST digit 0 or 1. To test whether we can learn an interpretable reward function from pairs of preference demonstrations over trajectories (see Fig 8a for an example), we modeled the reward as a DDT of depth 1 with one simple internal node as the root node and 2 CRL leaf nodes with reward vector $\mathbf{R} = (0.0, 1.0)$. We also compare the RL performance of the same reward DDT but with IL leaf nodes.

Results The resulting heatmap in Fig 8b provides evidence that the reward DDT learns to branch based on visually interpretable features that correspond to a hand-written 0 (routes to left leaf node) and a hand-written 1 (routes to right leaf node). The RL performance using the Soft Reward from CRL Leaf DDT on MNIST 0-1 environment is shown in Table 5 is comparable to a deep neural network reward function trained on pairwise preferences. We observe that taking the maximum probability path across the learned reward tree results in a small decrease in performance relative to when we take soft reward from the learned DDT. The RL performance of IL Leaf DDT outperforms that of both CRL DDT and a deep neural network, hence it provides evidence that our DDT framework is capable of learning an interpretable and useful reward function.

E MNIST (0-3) Gridworld Additional Results and Analysis

In this section we provide detailed analysis about interpretability of different DDTs, beginning from comparison between Reward DDT and Classification DDT, then comparing Reward DDTs constructed using two different leaf node formulations, followed by comparison of different regularization on a reward DDT.

Note that for both reward DDTs with different leaf nodes CRL and IL, we trained using a learning rate of 0.001 and weight decay 0.005 and the Adam optimizer. And the neural network details are same as defined above in Appendix D.

E.1 Min-Max Reward Interpolation Tree vs Classification tree

We train a DDT with explicit reward labels and a classification loss, as in, we re-produce the classification DDT from [19] and compare it to reward tree learned using preferences(refer to Sec 4.2.2 of main paper).

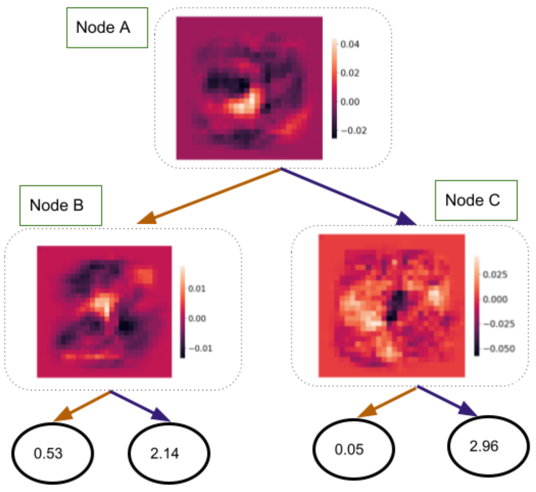
For comparison of reward tree against the classification tree trained using ground truth labels, we plot the heatmaps of internal nodes in both the trees and our results in Figure 9 give evidence that reward tree can capture visual features without any loss in interpretability when compared to the one learnt from simple ground truth labels, even though preferences used here are weaker supervision than ground truth label since preferences used in our experiments are binary as compared to ground truth labels which are 0,1,2,3 corresponding to each actual digit image. This is particularly important in cases where explicit labels are either missing or are hard to be specified or require intensive user-input efforts.

In Figure 9b Node A activates strongly for pixels in the middle of 1s and 2s, routing them left, while and 0s and 3s are routed right. Node B routes left for vertical pixels in the center and sends 2's left and 1's right (note the light shadow looks like a 2 while the darker shadow in the middle that looks like a 1). Node C learns to distinguish between 0s and 3s, routing 3s left and 0s right. This is comparable to the activation heatmaps of the node probability distribution at each of the internal node described for reward tree(in Sec 4.2.2 of main paper).

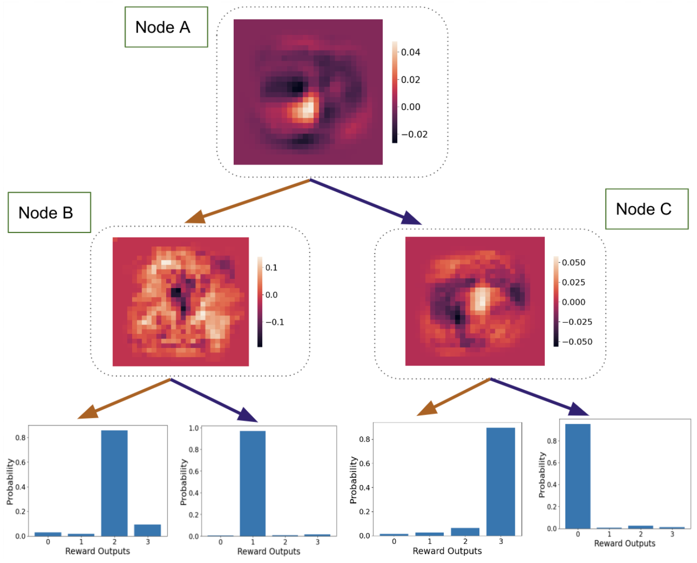
E.2 Min-Max Reward Interpolation Leaf DDT vs Multi-Class Reward Leaf DDT

We train and compare two reward DDTs with simple internal node architecture but with different leaf formulations using the same Bradley-Terry loss over preference demonstration in Figure 10 by visualizing the activation heatmaps of routing probability distributions for the internal nodes and the leaf distribution for each leaf node.

In Figure 10b, each internal node learns to capture almost the same visual feature while the leaf nodes fail to specialize as the argmax output from first two leaf nodes is always a 0 and last two leaf nodes always return a 3. Multi-class Leaf DDT fails to pick up on individual digit in the trajectory, despite requiring the user to input discrete reward vector whereas in the Min-Max Interpolation Leaf DDT each internal node captures different visual attributes and each of the leaf nodes in the



(a) Reward Tree trained using preferences



(b) Classification Tree trained on ground truth label

Figure 9: **Visualization of MNSIT (0-3) Reward vs Classification Tree**

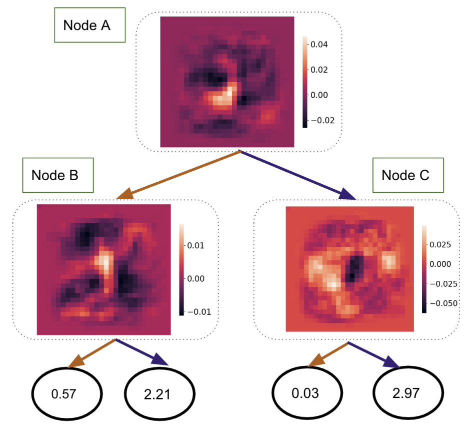
interpolated reward DDT is specialized, even though no discrete reward values were given as an input.

This shows that Min-Max Reward Interpolation Leaf DDT is beneficial over Multi-Class Reward Leaf DDT with respect to interpretability and also in terms of human-input efforts. for all states in the pairwise demonstrations.

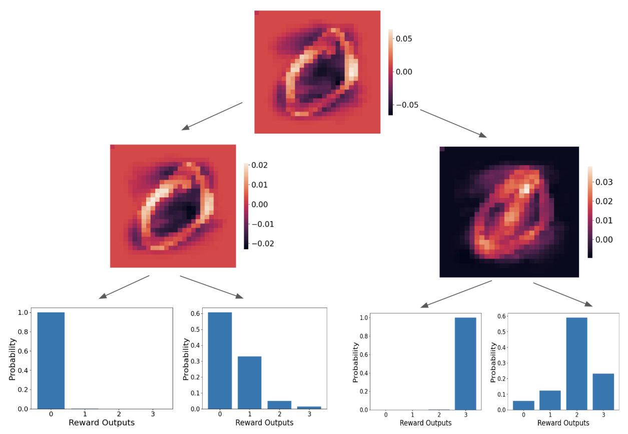
E.3 Min-Max Reward Interpolation DDTs with Simple Internal Nodes vs Sophisticated Internal Node

We compare our 2 methods of constructing internal nodes for a reward DDTs.

Since Min-Max Reward Interpolation Leaf DDT outperforms Multi-Class Reward Leaf DDT, hence we train two different Min-Max Reward Interpolation Leaf DDTs, first one with simple internal nodes

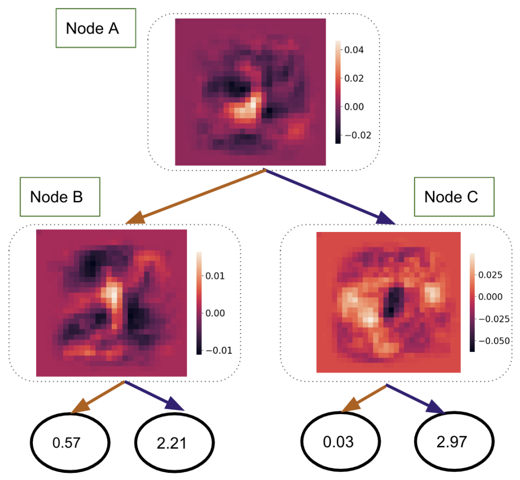


(a) Min-Max Reward Interpolation Leaf DDT

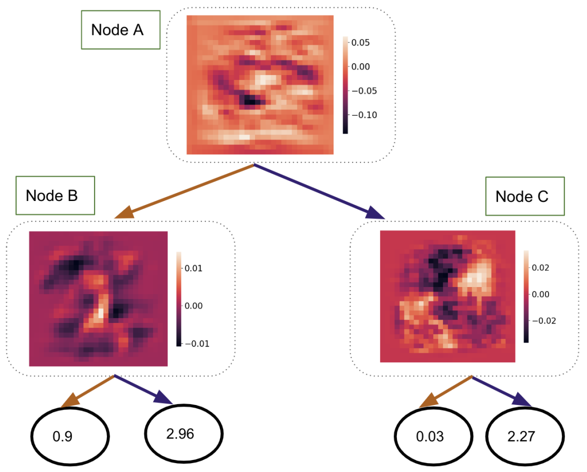


(b) Multi-Class Leaf Reward DDT

Figure 10: **Visualization of MNSIT (0-3) Reward Trees: Min-Max Reward Interpolation Leaf vs Multi-Class Leaf**



(a) Min-Max Reward Interpolation Leaf DDT with Simple Internal Nodes



(b) Min-Max Reward Interpolation Leaf DDT with Sophisticated Internal Nodes

Figure 11: **Visualization of MNSIT (0-3) Reward Trees :Simple Internal Node vs Sophisticated Internal Node**

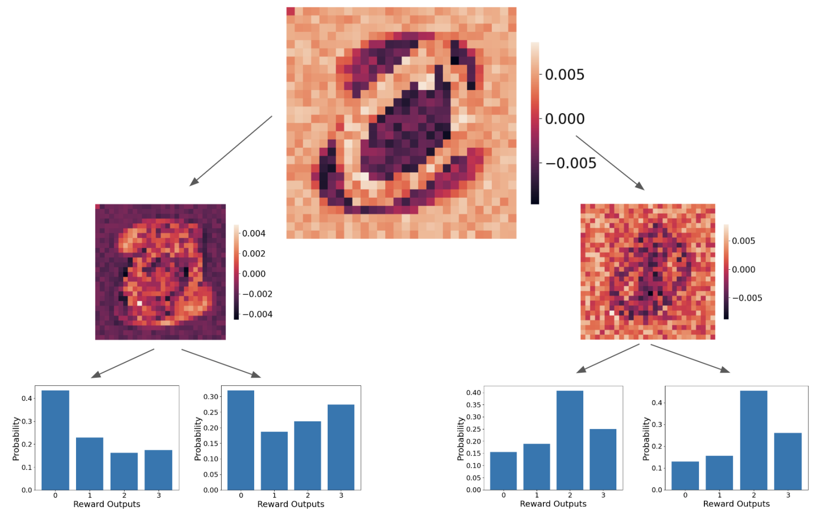
and second one with sophisticated internal nodes where a sophisticated internal node contains a single convolutional layer with filter of size 3x3 and stride 1 with Leaky ReLU as the non-linearity followed by the fully connected layer.

In Figure 11b Node A activates strongly for pixels in the middle of 1s and 3s, routing them left, while and 0s and 3s are routed right. Node B routes left for vertical pixels in the center and sends 1's left and 3's right (note the darker shadow in the middle that looks like a 3). Node C learns to distinguish between 0s and 2s, routing 0s left and 2s right. This is comparable to the activation heatmaps of the node probability distribution at each of the internal node described for reward tree(in Sec 4.2.2 of main paper).

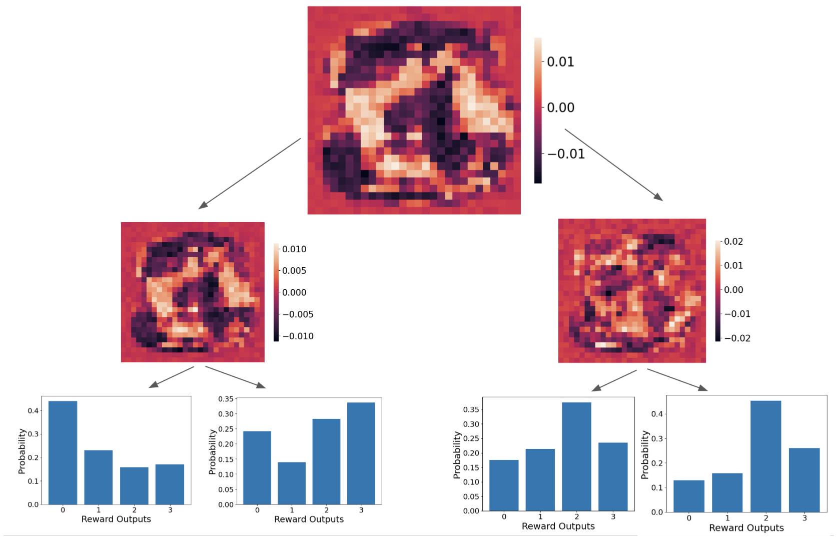
Our results depict that in a medium-complexity environment with visual inputs, both DDTs yield relatively equal interpretability but with a higher-complexity environment with larger visual input size such as Atari, the reward DDT with sophisticated node should be used as convolution layer with non-linearity are more powerful in terms of processing an input than a simple fully connected layer.

E.4 Multi-Class Reward Leaf DDT Regularization

Since the DDT with Multi-Class Reward Leaves failed to specialize, this lead us to add the penalty term to the Bradley-Terry preference loss for training the Multi-Class Reward Leaf DDT.



(a) Multi-Class Leaf Reward DDT with penalty calculated over a batch of 50 pairwise preference demonstrations where each demonstration has a single state



(b) Multi-Class Leaf Reward DDT with penalty calculated over a batch of 50 pairwise preference demonstrations where each demonstration has a single state

Figure 12: Multi-Class Leaf Reward DDT with penalty calculated over different temporal window lengths

For training the Reward DDT,we calculate penalty over batch of 50 pairwise demonstrations where each demonstration contains a single 28x28 greyscale image.To check interpretability, we plot the activation heatmaps of routing probability distributions for the internal nodes and the leaf

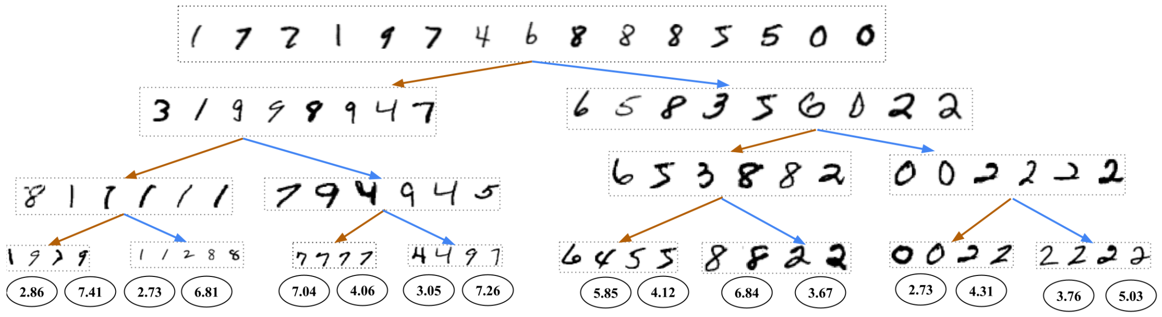


Figure 13: Synthetic traces of MNIST 0-9 IL Reward DDT of depth 4.

distribution for each leaf node in Figure 12a and the resulting plots are hugely pixelated, causing a loss in interpretability.

Following this, we increase the temporal window size for calculating penalty, as suggested in [19], and thus we calculate penalty over a pair of 50 preference demonstration where each demonstration is now 50 states long, as opposed to previous case where each demonstration contained a single state. And we again visualize the heatmaps at internal nodes and leaf distributions for each leaf node in Figure 12b. The heatmaps here are little better in contrast to Figure 12a but still have a huge loss of interpretability as compared to Figure 10b.

F Synthetic trace for MNIST 0-9 Reward DDT

We create synthetic traces Fig 13 of learned DDT with IL leaf nodes across all digits in MNIST. From the traces, we can observe that root node splits the digits based on whether they have more of vertical formulation or circular formulation. The digits with more vertical edges (such as 1,2,3,4,5,8,7,9) are routed to Node B while those with more curved edges (such as 0,2,3,5,6,8) are routed to Node C. Note some digits such as 2,3,4,8 in the actual MNIST dataset can either be more lean with straight form or can possess more rounded-curve form. The children node of Node B and C then differentiate further between each of the digits routed, as in, children of Node B learn to pick on spread of vertical edges while children of Node C distinguish between forms of curvature. These children's children then learns to pick and specialize in certain specific digits.

G Atari

The input to DDT here is a 5-dimensional tensor of size $B \times 2 \times S \times 84 \times 84 \times 4$ where B represents batch size of pairwise preference demonstrations while 2 is represents of number of demonstrations in a pairwise preference and S represents number of states in a single trajectory. We used batch size $B = 25$ and $S = 25$.The sophisticated internal node architecture here consists of a single convolution layer with kernel of size 7×7 with a stride of 2 and LeakyRelu as the non-linearity followed by the fully connected linear layer for producing the routing probability inside a tree.We used IL leaf nodes with $R_{\min} = 0$ and $R_{\max} = 1$. Note that we choose these min and max values for simplicity; though the actual numerical value of $R_{\min}$ and $R_{\max}$ can be chosen at the discretion of the user since policies are invariant to positive scaling and affine. The baseline T-REX, that we compare to has an architecture similar to Christiano et al. (2017) and consists of 4 convolutional layers of sizes 7x7, 5x5, 3x3 and 3x3 with strides 3,2,1 and 1 respectively, where each convolutional layer has 16 filters and LeakyReLU as non-linearity, followed by a fully connected layer with 64 hidden units and a single scalar output.