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

$$\alpha_i = \frac{\sum_{\mathbf{x}} P^i(\mathbf{x}) p_i(\mathbf{x})}{\sum_{\mathbf{x}} P^i(\mathbf{x})} \tag{3}$$

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 $\mathbf{x}$

$$C = -\lambda \sum_{i \in \text{ Inner Nodes}} 0.5 \log(\alpha_i) + 0.5 \log(1 - \alpha_i) \tag{4}$$

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.

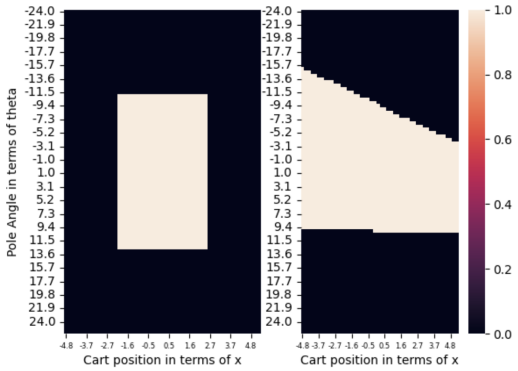


Figure 7: **Cartpole Reward DDT.** Ground Truth Reward on the left compared to Learnt Reward on the right. The reward model learnt by DDT is missing vertical boundaries visually, implying that it failed to pick up on cart position when contrasted with ground truth reward model that has both horizontal and vertical boundaries corresponding to pole angle and cart position respectively.

B Additional Cartpole Details and Analysis

For learning the reward DDT on classic control CartPole environment Fig 8, we used a learning rate and weight decay both equal to 0.001 and the Adam optimizer. The neural network we compare to, is trained on same dataset and is comprised of 2 fully connected layers with Leaky ReLU as non-linearity between the layers and the output from the last fully connected layer goes through a sigmoid activation for the final reward from the neural network.

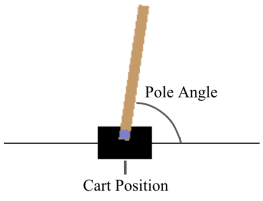


Figure 8: CartPole

To verify that our DDT reward model picks up on the misalignment in the reward function with respect to input features, particular cart position, we also compared the ground truth reward with the

output of the learned reward obtained by taking the argmax class from the leaf node with maximum routing probability. We plot these two rewards as a function of pole angle and cart position in Figure 7.

Figure 7 depicts that the learned reward is misaligned: the reward DDT has learned to approximate the preferences over pole angle, but pays much less attention to the pole angle when predicting rewards.

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.

MNIST (0/1) Gridworld Experimental Details 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 (35).

D 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 C.

D.1 MIN-MAX REWARD INTERPOLATION TREE VS CLASSIFICATION TREE

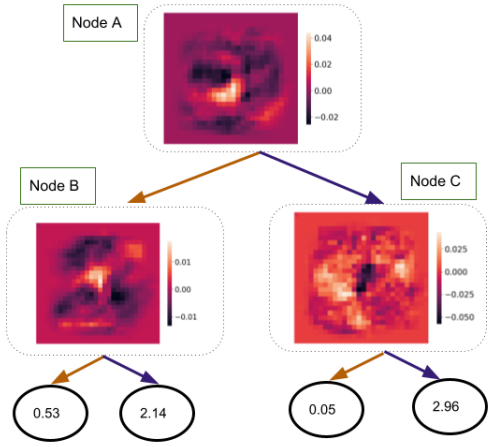
We train a DDT with explicit reward labels and a classification loss, as in, we re-produce the classification DDT from [19] and compare it to reward tree learned using preferences(refer to Sec 4.2.2 of main paper).

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.

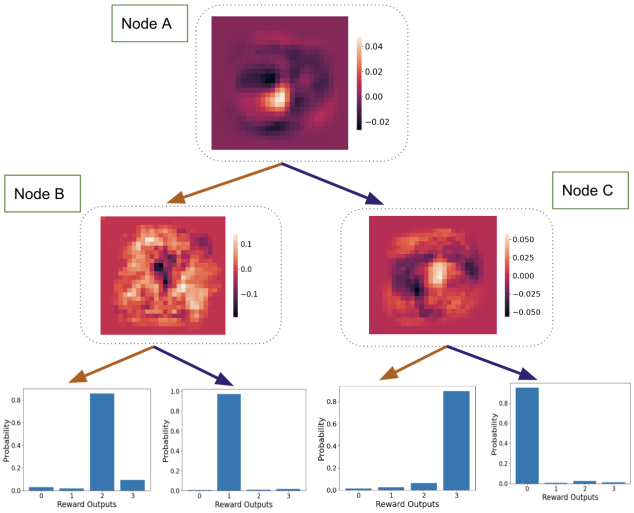
In Figure 9b Node A activates strongly for pixels in the middle of 1s and 2s, routing them left, while and 0s and 3s are routed right. Node B routes left for vertical pixels in the center and sends 2's left and 1's right (note the light shadow looks like a 2 while the darker shadow in the middle that 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).

D.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



(a) Reward Tree trained using preferences



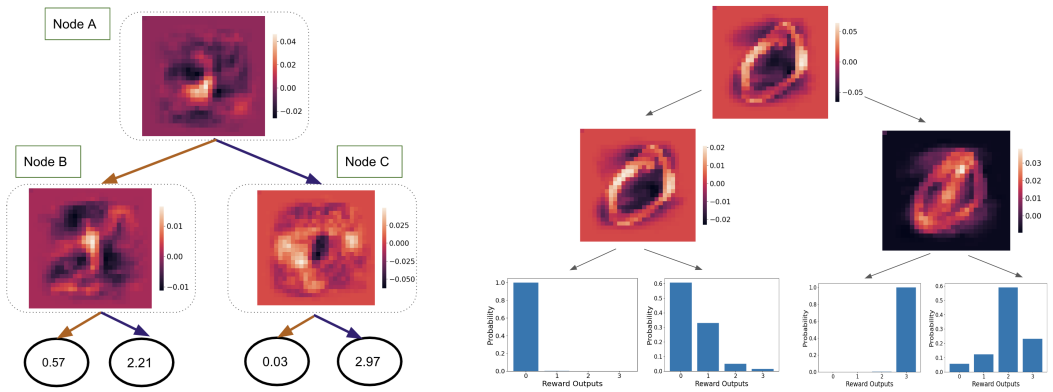
(b) Classification Tree trained on ground truth label

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

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 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.



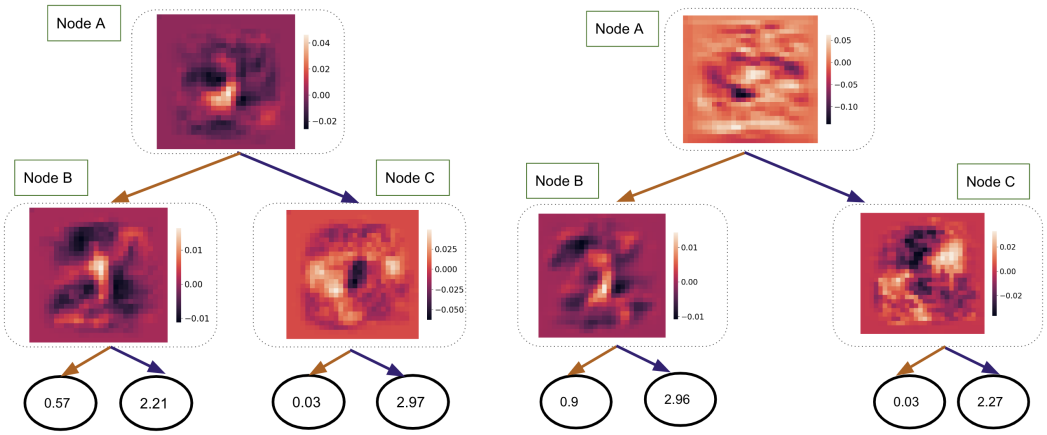
(a) Min-Max Reward Interpolation Leaf DDT

(b) Multi-Class Leaf Reward DDT

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

D.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.



(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**

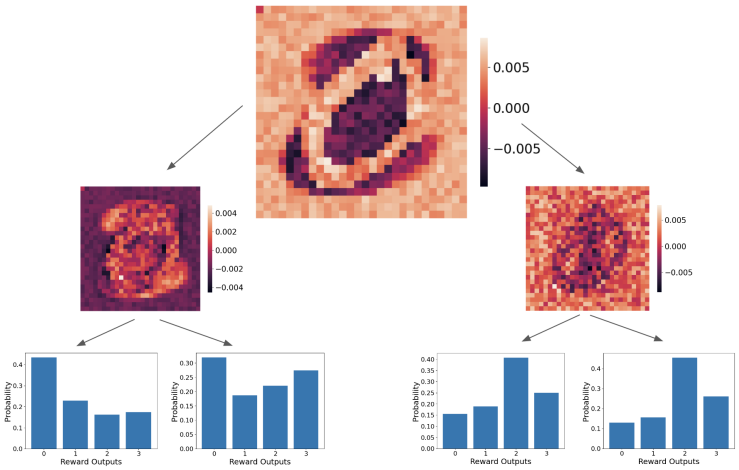
Since Min-Max Reward Interpolation Leaf DDT outperforms Multi-Class Reward Leaf DDT, hence we train two different Min-Max Reward Interpolation Leaf DDTs, first one with simple internal nodes and second one with sophisticated internal nodes where a sophisticated internal node contains a single convolutional 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).

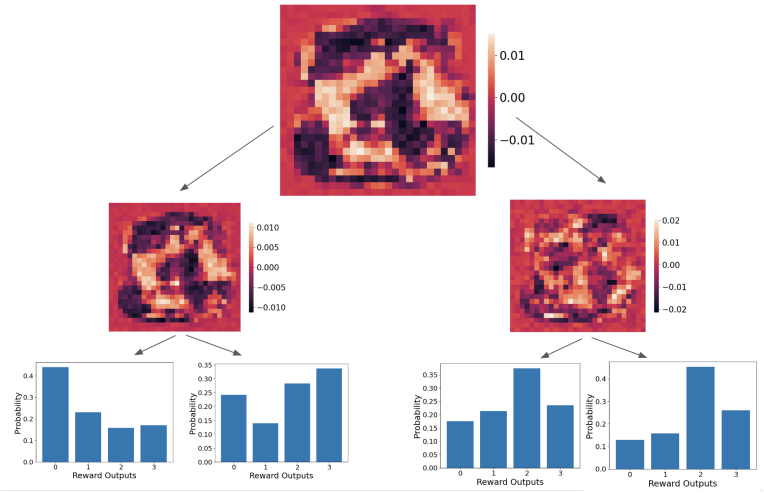
Our results depict that in a medium-complexity environment with visual inputs , both DDTs yield relatively equal interpretability but with a higher-complexity environment with larger visual input size such as Atari, the reward DDT with sophisticated node should be used as convolution layer with non-linearity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

D.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 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.

D.5 Synthetic trace for MNIST 0-3 Reward DDT

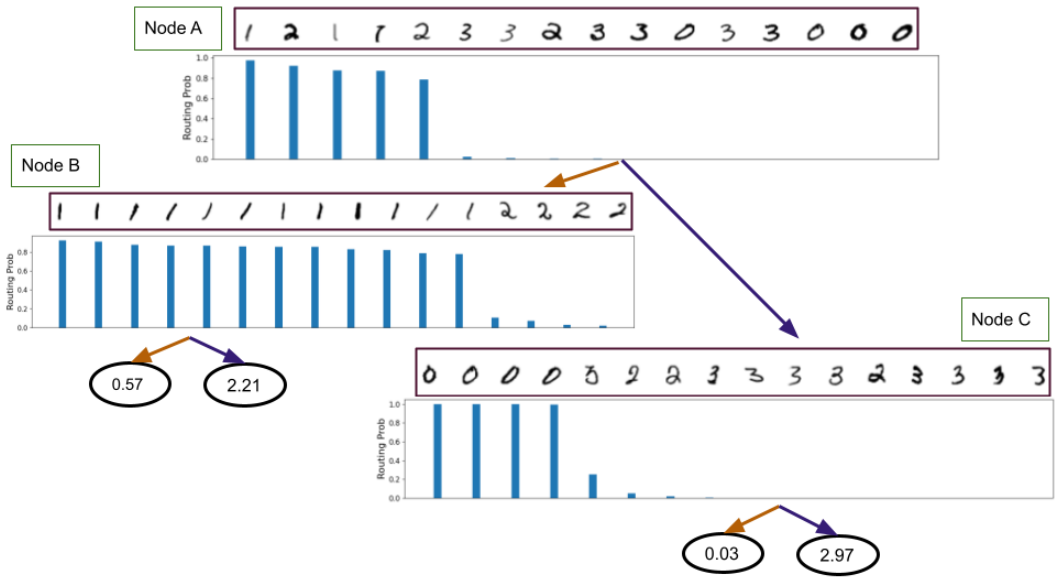


Figure 13: **Synthetic trace of MNIST(0-3) Reward DDT** We plot the trace over the entire input space at each internal node in order of decreasing routing probability from left to right in order to generate global explanations. At each internal node we depict the actual states and their respective routing probabilities. In node B images of 1 are routed to left leaf node as they have higher probability while images of 2 are routed to right leaf node. For Node C, we find that 0s get routed to left leaf node while some 2's which are visually closer to 0s get routed to left leaf node while those 2's that are closer to 3s and actual 3s get routed to right leaf node.For Node A , we find 1s and higher number of 2s are routed to Node B while 0s,3s, and some 2s , which are in terms of pixels, closer in shape to 0s and 3s are routed to Node C.

For the reward DDT trained using Min-Max Interpolation Leaf nodes in Sec 4.2.2 we visualize the synthetic traces fig 13 over the entire state space to generate global explanations. At each internal node, we also plot the respective routing probabilities of every state in the trace.To generate a trace for an internal DDT node, we sort the training data in descending order based on the routing probability assigned to each state. We then sample every Tth state to produce a trace of states that range from those routed most highly to the left child to those most highly routed to the right child.

We find that B images of 1 are routed to left leaf node as their routing probability>0.5 while images of 2 are routed to right leaf node. For Node C, our trace interestingly captures the fact that 0s get routed to left leaf node while some 2's which are visually closer to 0s get routed to left leaf node while those 2's that are closer to 3s and actual 3s get routed to right leaf node.For Node A , we find 1s and higher number of 2s are routed to Node B while 0s,3s, and some 2s , which are in terms of pixels, closer in shape to 0s and 3s are routed to Node C.

Our synthetic trace on basis can explain the reward any state in the entire state space would get in a discrete number of steps.

E 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. 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 $R_{\text{min}} = 0$ and $R_{\text{max}} = 1$. 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 (15) 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.

Beam Rider For beam rider reward ddt using a batch of $B = 10$ pairwise demonstrations where each demonstration is 25 states long and Adam optimizer, learning rate of 0.0009 with different seeds. For training Min-Max Reward Interpolation Leaf DDT with sophisticated internal nodes on Beam Rider we tried all combinations possible using the following hyper-parameter settings:

- Seed : 0, 1 and 2
- Batch size of Pairwise Preferences *B* : 1,10
- Learning Rate: 0.00005, 0.0009 for *B* equal to 1 and 10 respectively

But for all DDTs trained without the penalty, we ran into the problem of un-equal use of sub-trees.For running-RL and visualizing the synthetic traces we use the reward DDT, trained using seed 0 with B=10 and lr =0.0009.

Note: we use the same exact setting for training the reward DDT with penalty.

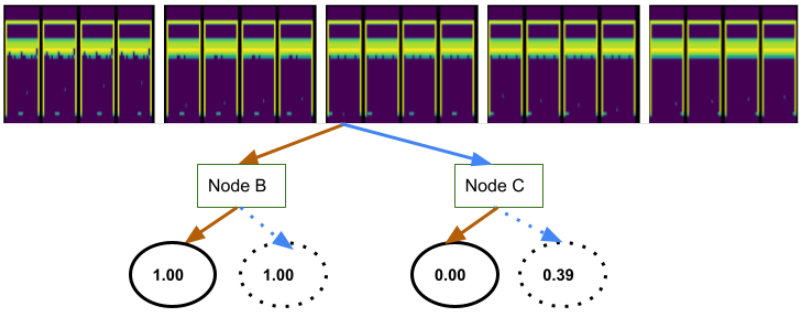


Figure 14: Interpolated DDT of depth 2 trained on Breakout without regularization

Breakout We trained 2 different Interpolated Leaf DDTs on Breakout, one without and another with Penalty added and created the synsthetic trace over all input states starting from states that are routed 100% to left and ending at states that are routed complete right (as in have 0% probability of beiung routed left).

For Breakout , we show a more rigorous trace than BeamRider,by visualizing states that are routed with 100% , 75%, 50% , 25% and 0% probability to left. And we found that, fig 14 without any penalty added both children nodes of the root node only use their respective left leaves and do not route any state to their respective right leaves. This meant that a synthetic trace could not be visualized for either Node B or Node C.

Next, we visualize the synthetic trace over the penalized DDT, fig 15 and surprisingly , we found that the regularization term added to the final loss of the tree while training was penalizing the DDT so heavily that now the routing probability at Node B and Node C was always between 0.5 and 0.499, which again lead to a failure in being able to create the synthetic trace over inputs as no states were now being routed with a 100% probability neither left nor right and also numerically the difference in routing probability was very trivial.

In fig 14 which depicts our trace for unregularized Reward DDT, we find at Node A, states that have more number of bricks missing have a higher probability(>0.5) of being routed to left, i.e. to Node B which only uses a single leaf node to give a reward of +1 while the states in Node A that have no or very few bricks missing have a lower routing probability(routing probability <0.5) and thus are routed right , i.e. to Node C which again uses a single leaf node to give a reward of 0. We can similarly interpret the trace of regularized reward DDT fig 15 for breakout.

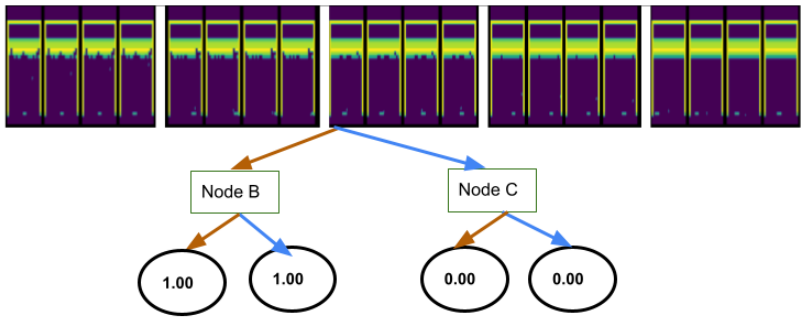


Figure 15: Interpolated DDT of depth 2 trained on Breakout with regularization